Preprints.org

Article

Not peer-reviewed version

MTSF—Market-Theoretic Security Framework: A Unified Paradigm for the Art of Proving and Disproving Security

Basker Palaniswamy * and Paolo Palmieri

Posted Date: 3 April 2026

doi: 10.20944/preprints202603.2237.v2

Keywords: market-theoretic security; auction-based proofs; extended difference lemma; CNF session verification; universal composability; game-based reductions; QROM; protocol analysis; TLS 1.3; signal protocol

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

MTSF—Market-Theoretic Security Framework: A Unified Paradigm for the Art of Proving and Disproving Security

Basker Palaniswamy [1,*] **and Paolo Palmieri** [2]

Insight Research Ireland Centre for Data Analytics, Department of Computer Science and Information Technology, University College Cork (UCC), Cork City, Ireland, European Union
* Correspondence: basker170889@gmail.com

Abstract

Cryptographic security proofs are the invisible backbone of modern digital systems, yet they remain fragmented across multiple paradigms—game-based proofs, Universal Composability (UC), formal verification, and ad hoc insecurity arguments—each with its own language, assumptions, and limitations. This article introduces the **Market-Theoretic Security Framework (MTSF)**, a unified paradigm that reinterprets all security proofs as economic markets. In this view, the defender acts as a seller offering *security goods* (such as confidentiality or unforgeability), while the adversary acts as a buyer bidding computational resources to break them. Security emerges naturally as *market equilibrium*, where no efficient adversary can afford to win, while insecurity is characterized as *market collapse*, where attacks succeed at negligible cost. For cryptographers, MTSF provides a rigorous and expressive framework that unifies four major proof paradigms into a single formal l anguage. It introduces key technical innovations such as the **extended difference lemma** for handling multiple simultaneous failure events, **bidding-based reductions** that explicitly model adversarial strategies, a **dual methodology that treats proofs and disproofs symmetrically within the same structure**, and a **session pinging mechanism** for unbounded session verification. The framework seamlessly extends to classical and post-quantum primitives, real-world protocols (including TLS 1.3 and Signal), and even quantum-adversarial settings, while preserving quantitative security bounds and composability guarantees. MTSF offers an intuitive, accessible, and powerful meta model: security is like a marketplace where attackers try to "buy" a break, and defenders ensure the price is prohibitively high. Each proof becomes a sequence of small price adjustments, and each attack corresponds to a failed or successful bid. By combining mathematical rigor with economic intuition, MTSF transforms security proofs from opaque technical artifacts into transparent, auditable, and universally understandable arguments, enabling both experts and practitioners to reason about security with clarity and confidence.

Keywords: market-theoretic security; auction-based proofs; extended difference lemma; CNF session verification; universal composability; game-based reductions; QROM; protocol analysis; TLS 1.3; signal protocol

---

✻ Simple Terms: Why Should You Read This Article?

If you are a cryptographer: MTSF unifies the four proof paradigms you already know (game-based, UC, formal verification, ad hoc insecurity) into a single language. You will find that proofs become shorter, more intuitive, and easier to compare across different schemes. The extended difference lemma and session pinging are new tools you can apply immediately.
If you are a software engineer: You will learn to read security proofs as market arguments—"this scheme is secure because no attacker can afford to break it." The CNF verification worksheets give you a concrete, step-by-step procedure for auditing whether a protocol implementation is correct.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

If you are a student or newcomer: Every definition, theorem, and proof is accompanied by a plain-language explanation, a real-world analogy, and often a diagram. You can follow the market metaphor even if you have never seen a security proof before. Start with the Reader Guide on the next page, then jump to the case study that interests you most.
If you are a policy maker or manager: MTSF provides a clear, auditable language for security guarantees. When a cryptographer says "the market is in equilibrium with ask price 2^{-128}," you know that the system is secure with overwhelming probability. When they say "the market has collapsed," you know the system is broken and needs immediate action.

✻ Simple Terms: New to Cryptography!

Welcome! This document introduces a new way of thinking about security proofs using the language of markets and auctions.

What is a security proof? A security proof is a mathematical argument that shows a cryptographic scheme (like a digital signature or encryption) cannot be broken by any efficient computer program. Instead of testing every possible attack, we *prove* that any attacker would have to solve a problem that is believed to be computationally infeasible (like factoring enormous numbers).

What is the market analogy? MTSF rephrases security proofs as economics. The **seller** (the cryptographer defending the scheme) offers **security goods** (guarantees like "this message is confidential" or "this signature cannot be forged"). The **buyer** (the attacker) bids **computational resources** (time and memory) to try to break those guarantees. If the market is in **equilibrium**, the attacker cannot afford to win—security holds. If the market **collapses**, the attacker gets a free win—security fails.

Key vocabulary you will see:

- **Adversary / Buyer $\mathcal{A}$:** The attacker trying to break the system.
- **Challenger / Seller $\mathcal{C}$:** The entity setting up the security experiment.
- **Advantage / Ask price:** How likely the attacker is to win. If this is tiny ("negligible"), the scheme is secure.
- **Negligible:** A number so small it shrinks faster than any fraction of the form $1/n^c$. In practice, think "astronomically close to zero."
- **Game Hop / Bidding Round:** A small change to the security experiment that makes it easier to analyse, changing the winning probability by at most a small "price adjustment."
- **Reduction:** If an attacker can break scheme X, we show it could also solve hard problem Y. Since Y is believed hard, so is X.
- **CNF formula:** A checklist of AND-ed conditions (each condition is a simple OR clause). A session passes the audit only if every item on the checklist is satisfied.
- **UC / GUC:** Universal Composability—a framework guaranteeing that a secure protocol stays secure even when run alongside other protocols simultaneously.
- **PRF / PRP:** Pseudorandom Function / Permutation—a keyed function that looks random to any attacker who does not know the key.
- **EUF-CMA:** Existential Unforgeability under Chosen Message Attack—the standard security notion for digital signatures: even after seeing many signatures, the attacker cannot forge a new one.
- **IND-CCA2:** Indistinguishability under Chosen-Ciphertext Attack—the gold standard for encryption: the attacker cannot distinguish the encryption of one message from another, even with a decryption oracle.

How to read the theorems: Each theorem states a bound like "$\mathsf{Ask}(g) \leq \mathsf{Adv}^{\mathsf{ECDLP}} + \text{tiny terms}$." This means: the only way to break the scheme is to solve the hard mathematical problem (e.g., Elliptic Curve Discrete Logarithm). The "tiny terms" come from unavoidable coincidences (birthday collisions, etc.) but are negligibly small for real parameter choices.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

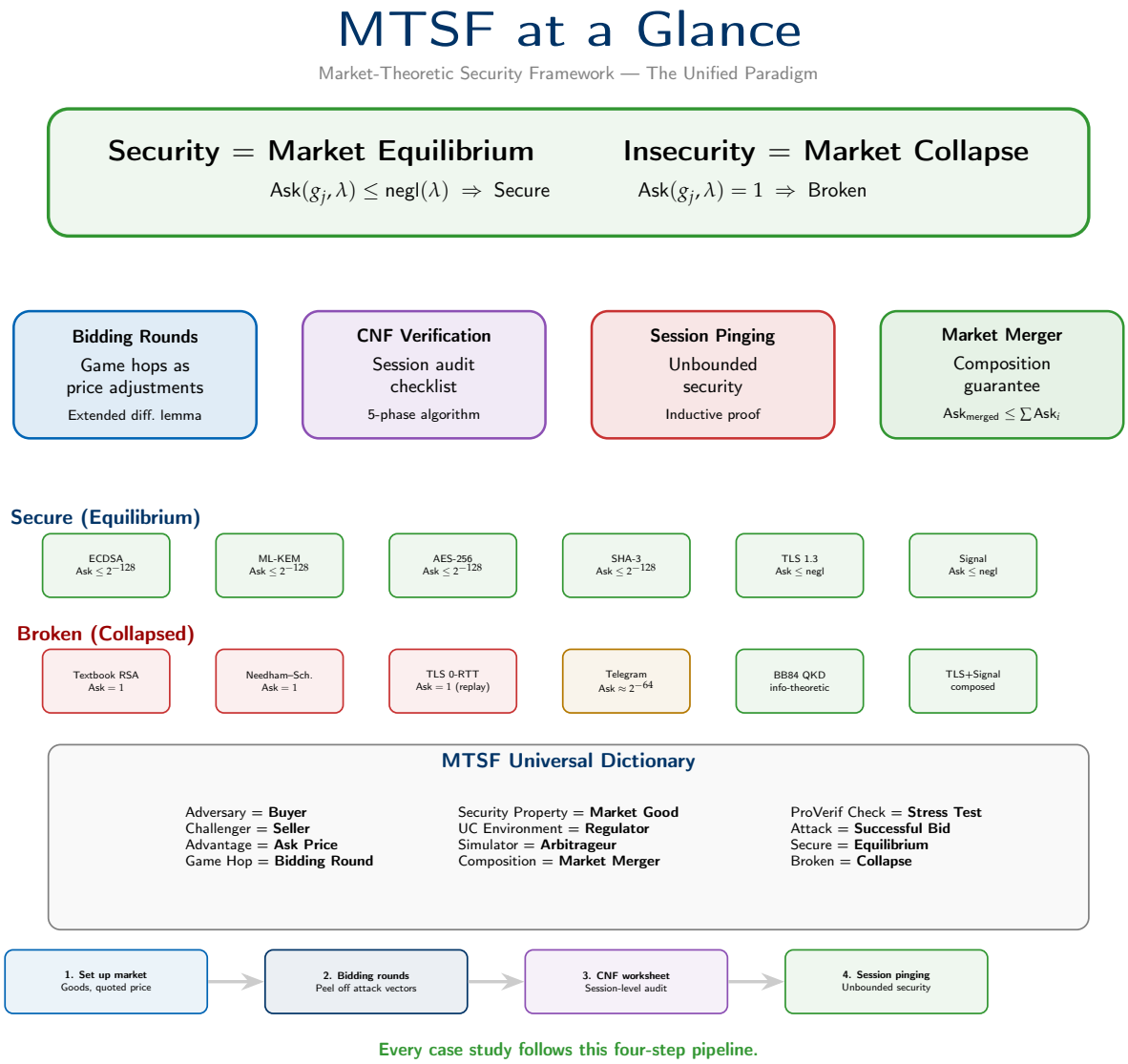


Figure 1. MTSF at a Glance. The complete framework on one page: core equation (top), four pillars (row 2), case study scoreboard (row 3), universal dictionary (row 4), and proof pipeline (bottom). Pin this page to your wall.

✎ Analogy:

Reading This Document Is Like Building a City Think of this document as a blueprint for building a city from scratch:

- **Preliminaries** = Surveying the land (understanding the terrain of existing security frameworks).
- **MTSF Core** = Designing the city plan (the market language, definitions, and theorems).
- **CNF Verification** = Writing building codes (the checklist every structure must pass).
- **Session Pinging** = Installing earthquake sensors (ensuring safety indefinitely, not just for a limited time).
- **Case Studies** = Actually constructing buildings (applying the framework to real cryptographic systems).
- **Quantum sections** = Future-proofing against flying cars (preparing for quantum computers).

You do not need to read every section in order. If you are interested in a specific protocol (say TLS 1.3), you can jump to its case study after reading the MTSF Core section. The roadmap in Figure 2 shows which sections depend on which.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

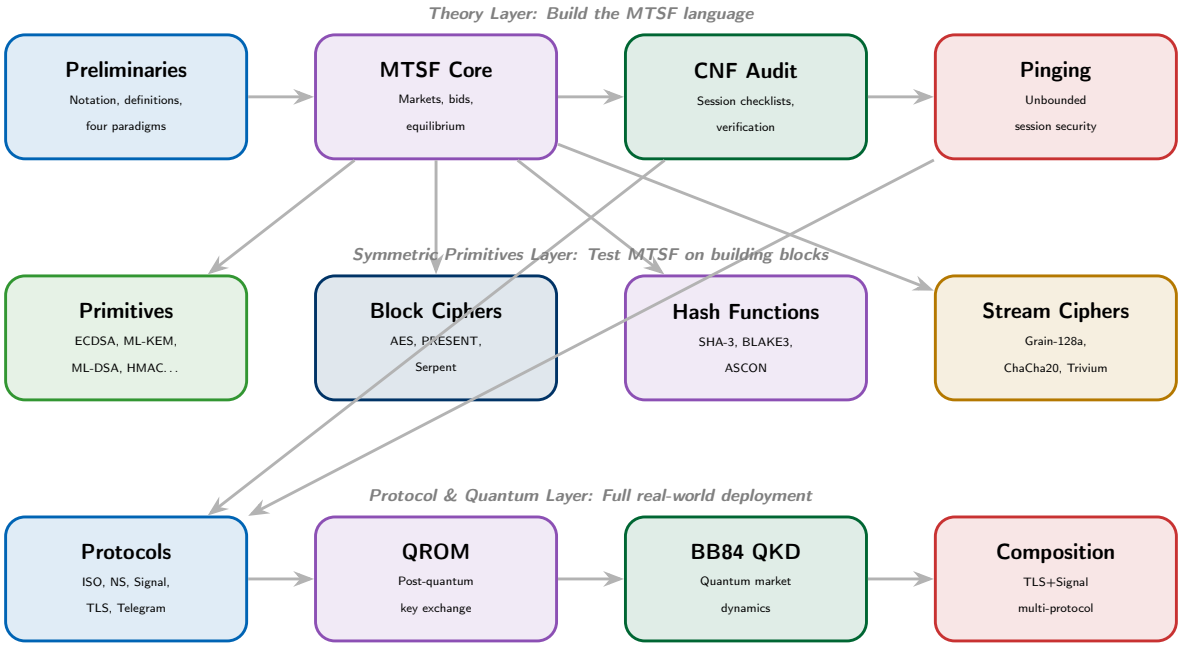


Figure 2. Visual roadmap of the MTSF document. The theory layer (top) builds the market language; the primitives layer (middle) tests it on cryptographic building blocks; the protocol and quantum layer (bottom) deploys it on real-world systems. Arrows show logical dependencies.

1. Introduction

Every digital system we trust—from the padlock in a browser to a hospital's encrypted records, from a signed software update to a government identity card—rests on a hidden mathematical foundation: *cryptographic security proofs*. These proofs are the silent guardians of modern civilisation's digital infrastructure, ensuring that encryption remains unbreakable, signatures remain unforgeable, and protocols remain resilient, no matter how powerful or resourceful the adversary. Yet despite their central role, today's security proofs are fragmented across four disconnected languages that specialists in one paradigm often cannot easily interpret in another: game-based sequences of experiments, Universal Composability (UC) ideal-world simulations, formal symbolic verification tools such as ProVerif and Tamarin, and ad hoc insecurity demonstrations for flawed constructions. A cryptographer proving TLS secure employs different notation, definitions, and reasoning from one proving a post-quantum signature secure—even though both ultimately address the same fundamental question: *can an efficient adversary succeed?* The Market-Theoretic Security Framework (MTSF) introduced in this article answers this question through a single unifying perspective grounded in economics: every security proof is an auction, every adversary is a buyer, every security property is a tradable good, and security holds precisely when no buyer can afford to win. This article develops MTSF rigorously, proves its equivalence to all four paradigms, and demonstrates its expressive power across eighteen case studies spanning primitives, protocols, and quantum-aware security models.

★ Story Mode: A Day in the Life of Alice the Security Engineer

Monday morning. Alice is a security engineer at a major bank. Her boss drops three tasks on her desk:
Task 1: "Our mobile app uses ECDSA for transaction signing. A regulator wants mathematical proof it is secure. Can you write one?"
Alice opens her MTSF toolkit. She identifies the *market*: seller = the bank's signing system; buyer = any attacker; good = $g_{\text{EUF-CMA}}$ (unforgeability). She writes the *quoted price* QPrice_0 and constructs five bidding rounds: nonce bid (birthday bound), hash bid (collision bound), oracle programming (free), forgery-to-ECDLP (hard problem). Total ask price: $\leq 2^{-128}$. She fills in the CNF worksheet and verifies the session ping. **Verdict: Equilibrium.** The regulator is satisfied.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Task 2: "Our partner bank uses a custom authentication protocol. Is it safe to connect our systems?"
Alice reviews the protocol and immediately spots a red flag: message 3 does not include the sender's identity. She constructs the MTSF market and attempts a *masquerade bid*—relaying messages between two honest parties without breaking any cryptography. $\Delta\text{Price} = 1$. **Verdict: Collapse.** She fills in the CNF worksheet: the identity-binding clause is *missing*. She recommends adding the sender's ID to message 3, re-runs the proof, and confirms equilibrium is restored.

Task 3: "We are migrating to post-quantum cryptography. Can you prove ML-KEM is safe, and that it composes securely with our existing TLS 1.3 deployment?"
Alice sets up two MTSF markets: one for ML-KEM (IND-CCA2 via MLWE), one for TLS 1.3 (authentication + key secrecy). She proves each is in equilibrium independently, then applies the *market merger theorem*: $\text{Ask}_{\text{composed}} \leq \text{Ask}_{\text{ML-KEM}} + \text{Ask}_{\text{TLS}} + \text{Ask}_{\text{PKI}}$. Since all three are negligible, the composed system is in equilibrium. She verifies the network CNF (cross-protocol SID uniqueness) and confirms unbounded session security via pinging. **Verdict: Equilibrium under composition.**

Alice completes all three tasks using one framework, one notation, and one proof methodology. Before MTSF, Task 1 would require game-based proofs, Task 2 would require ad hoc attack analysis, and Task 3 would require UC composition—three different frameworks with three different notations. With MTSF, it is all one language.

1.1. The Problem: Fragmented Security Paradigms

Cryptographic security analysis is conducted through four largely disconnected paradigms, each with distinct strengths and limitations that prevent any single approach from being fully satisfactory.

Problem 1: Game-based proofs are isolated and single-failure.

Game-based proofs [1,2] bound the adversary's advantage through a sequence of game hops $\mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_q$. The classical *difference lemma* [1] bounds each hop by a *single* failure event F:

$$|\Pr[\mathbf{B}_k = 1] - \Pr[\mathbf{B}_{k+1} = 1]| \leq \Pr[F].$$

In practice, however, many proof steps involve *multiple simultaneous failures*. For example, in a digital signature proof, a single game hop may need to eliminate nonce collisions, hash collisions, and public-key recovery failures all at once. The classical lemma forces practitioners to split such hops into multiple games, inflating proofs and loosening bounds unnecessarily. Furthermore, game-based proofs are *isolated*: each theorem is a standalone argument with no built-in mechanism for combining security guarantees when protocols are composed.

✎ Analogy:

Game-Based Proofs as Recipe Testing Imagine you are a chef (cryptographer) and you want to prove your cake recipe (cryptographic scheme) is perfect. Game-based proofs are like testing the recipe step by step: first you check the flour is good, then the sugar, then the eggs. Each test is one "game hop." But here is the problem: what if the flour, sugar, *and* eggs are all slightly off *at the same time*? The classical approach forces you to test each ingredient separately, losing track of how the problems interact. MTSF's extended difference lemma lets you test all ingredients simultaneously and account for their interactions.
Furthermore, each recipe test is *isolated*—proving your cake is perfect tells you nothing about what happens when you serve it alongside soup, salad, and dessert at a dinner party. Game-based proofs are standalone; they do not compose. MTSF's market model treats the entire dinner menu as one market, ensuring all dishes work together.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

☞ Key Insight:

Why MTSF Surpasses Classical Game-Based Proofs—Seven Concrete Advantages Game-based proofs (Bellare–Rogaway [2], Shoup [1]) are the most widely used proof paradigm. MTSF *strictly extends* them, offering every capability game-based proofs have, plus several they lack:

1. **Multiple simultaneous failures.** The classical difference lemma handles *one* failure event per game hop. In real proofs—ECDSA, ML-DSA, TLS 1.3—a single transition often involves 3–5 simultaneous bad events. Game-based proofs split these into separate hops, inflating proofs and loosening bounds. MTSF's *extended difference lemma* handles all m events in one hop: $\Delta\mathsf{Price} \leq \Pr[F_1 \cup \cdots \cup F_m]$ with inclusion-exclusion tightening when events are correlated. In the ECDSA proof (Section 9.1), this consolidates what would be three separate hops into one, yielding a tighter final bound.
2. **Named, classified attacks.** In game-based proofs, game hops are labelled generically ($G_0 \to G_1 \to G_2$). The reader must reverse-engineer *what* each hop achieves. MTSF labels every hop as a *specific bid type*: "nonce freshness bid," "hash collision bid," "forgery-to-ECDLP bid," "masquerade bid." This self-documenting structure makes proofs auditable: a reviewer can check whether all relevant attack classes from the bid taxonomy (Appendix B) have been addressed, like checking items off a standardised list.
3. **Built-in composition.** Game-based proofs are standalone: proving Scheme A secure and Scheme B secure says nothing about running both together. Composing game-based results requires ad hoc reasoning or switching to UC entirely. MTSF's *market merger theorem* gives composition natively: two equilibrium markets merge into a combined equilibrium market with explicit, additive ask-price bounds. The TLS+Signal case study (Section 16) demonstrates this directly.
4. **Insecurity as a first-class result.** Game-based proofs only establish security (upper bounds on advantage). Insecurity must be demonstrated separately using bespoke arguments. MTSF uses *the same bidding-round machinery* for both: $\mathsf{Ask} \leq \mathsf{negl}$ means equilibrium (secure); $\mathsf{Ask} = 1$ means collapse (insecure). The Needham–Schroeder analysis (Section 13.4) proves $\mathsf{Ask} = 1$ via a masquerade bid using the same notation and structure as the ECDSA equilibrium proof.
5. **Unbounded session coverage.** Game-based proofs are inherently bounded: they hold for at most $q = \mathsf{poly}(\lambda)$ sessions. Real systems run for millions of sessions over years. MTSF's *session pinging* mechanism (Section 5) provides an inductive bridge from bounded to unbounded security, with quantitative advantage accumulation bounds. No game-based framework offers this natively.
6. **Session-level auditing.** Game-based proofs produce a single aggregate bound ($\mathsf{Adv} \leq \epsilon$) but do not describe what a *correct session* looks like. MTSF's CNF session verification (Section 4) defines session correctness as a checkable Boolean formula, with a concrete manual worksheet that practitioners can fill in per-session. This enables implementation-level auditing that game-based proofs cannot support.
7. **Quantitative tightness tracking.** In game-based proofs, tightness loss is implicit and often buried in the reduction. MTSF makes tightness explicit via the "market spread": the ratio between the ask price (best attacker strategy) and the hard-problem advantage. Each bidding round explicitly states its tightness factor $L(\lambda)$ as part of the price adjustment, making the total loss visible and auditable.

Crucially, MTSF does not discard game-based techniques—it *embeds* them. Every existing game-based proof can be rewritten as an MTSF bidding-round chain by relabelling games as bidding rounds, advantages as ask prices, and failures as bid events. The translation is mechanical, and the MTSF version is strictly more informative because it adds classification, composition, unbounded coverage, and session auditing.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Capability	Game-Based	UC/GUC	Formal Verif.	MTSF
Quantitative bounds	✓ Yes	× No	× No	✓ Yes
Multi-failure hops	× No	× N/A	× N/A	✓ Yes
Named attack classes	× No	× No	∼ Traces	✓ Yes
Composition guarantee	× No	✓ Yes	× No	✓ Yes
Insecurity proofs	× No	× No	∼ Attacks	✓ Yes
Unbounded sessions	× No	∼ Partial	✓ Yes	✓ Yes
Session-level audit	× No	∼ CNF only	× No	✓ Yes
Human-readable	∼ Partial	× No	× No	✓ Yes

MTSF is the **only** framework achieving all eight capabilities simultaneously.
It subsumes game-based, UC/GUC, and formal verification as special cases.

Figure 3. Feature comparison: game-based proofs, UC/GUC, formal verification, and MTSF. Green = full support; yellow = partial support; red = not supported. MTSF is the only framework achieving all eight capabilities.

Problem 2: UC/GUC provide composition but lack concrete bounds.

The UC framework [3] guarantees that a secure protocol remains secure when composed with arbitrary other protocols running concurrently—the gold standard for real-world deployability. However, UC proofs do not produce numerical advantage bounds. A UC proof says "the protocol is secure" but not "the adversary's advantage is at most ϵ". The GUC framework [4] with CNF session checking [5] adds session correctness verification but still lacks the quantitative output needed for concrete parameter selection.

☞ Key Insight:

Why MTSF Surpasses UC and GUC—Five Concrete Advantages The UC framework [3] and its generalisation GUC [4] provide the gold standard for composition. MTSF preserves all of UC/GUC's composition guarantees while adding capabilities they lack:

1. **Concrete numerical bounds.** A UC proof says "protocol π UC-realises functionality $\mathcal{F}$." This is a binary verdict: secure or not. It does not tell you *how* secure—whether the adversary's advantage is 2^{-128} or 2^{-40}. MTSF produces explicit ask prices: $\mathsf{Ask}(g_j, \lambda) \leq 2\mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^{\gamma}$. These numerical bounds are essential for parameter selection (e.g., "choose $\lambda = 256$ to achieve 128-bit security") and for comparing the relative security of different schemes.
2. **Quantitative composition.** UC's composition theorem says "secure + secure = secure," but not *how much* security degrades. MTSF's market merger theorem gives an explicit additive bound: $\mathsf{Ask}_{\mathsf{composed}} \leq \mathsf{Ask}_{\mathsf{TLS}} + \mathsf{Ask}_{\mathsf{Signal}} + \mathsf{Ask}_{\mathsf{infrastructure}}$. This enables concrete resource planning: you know exactly how much security margin remains after composition.
3. **Structured proof organisation.** UC proofs involve constructing a simulator $\mathcal{S}$ that "explains" the real world using the ideal world. These simulators can be extremely complex—the TLS 1.3 UC proof by Dowling et al. required dozens of pages of simulator construction. MTSF replaces the monolithic simulator with a structured *bidding-round chain*, where each round targets one attack vector. This modular structure is easier to write, review, and

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

maintain. Adding a new attack vector means adding one bidding round, not rewriting the entire simulator.

4. **Insecurity detection within the same formalism.** UC proofs can only establish security. If a protocol is insecure, UC provides no mechanism to formally demonstrate the flaw—you must construct an ad hoc attack separately. MTSF's dual methodology handles both: the same bidding-round chain that proves TLS 1.3's 1-RTT handshake is secure ($\mathsf{Ask} \leq \mathsf{negl}$) also proves its 0-RTT mode is insecure ($\mathsf{Ask} = 1$ via replay bid). The Needham–Schroeder masquerade bid and the Telegram salt extraction bid are formal insecurity theorems *within* the MTSF framework, not separate arguments.
5. **Practitioner-accessible session auditing.** GUC's CNF session correctness [5] requires deep expertise to construct and verify. MTSF operationalises CNF checking via a canonical five-phase algorithm (Algorithm 1) and a simple four-column manual worksheet that a security engineer without formal methods training can use to audit protocol sessions. Every case study in this article includes a filled-in worksheet as a concrete example.

MTSF does not replace UC/GUC—it *contains* them as a special case. The UC real/ideal indistinguishability is recast as market regulation ($\mathcal{Z}$ is the regulator, $\mathcal{S}$ is the arbitrageur), and the composition theorem becomes the market merger theorem. A UC/GUC expert can use MTSF with minimal retraining: the concepts translate directly, and the MTSF version provides strictly more information.

Problem 3: Formal verification tools lack quantitative output and human interpretability.

ProVerif [6] and Tamarin [7] check whether a protocol is secure symbolically and for unbounded sessions, but produce a binary pass/fail output with no numerical bounds. CryptoVerif [8] closes this gap partially with machine-checked game-based proofs, but requires significant manual guidance and does not natively support the UC/GUC framework. More importantly, none of these tools produce human-readable arguments that practitioners can understand, audit, and reason about without specialist training.

✎ **Analogy:**

Formal Verification as a Robot Inspector Think of ProVerif and Tamarin as robot inspectors at an airport. They walk through *every possible route* a passenger (attacker) could take through the terminal and check whether any route leads to the cockpit (a security breach). If no route works, the airport is "secure." But the robots cannot tell you *how close* a passenger came to breaching security—they only say pass or fail. They also cannot explain their reasoning in plain English. MTSF turns these robot inspections into market stress tests with explicit numerical bounds: "The closest any attacker came to the cockpit was 0.000001% of the way there."

Problem 4: No unified insecurity analysis.

When a primitive is insecure (e.g., textbook RSA has a free homomorphism attack) or a protocol is insecure (e.g., Needham–Schroeder [9] is vulnerable to Lowe's MITM [10]), this is demonstrated *ad hoc* using bespoke arguments. There is no common framework that handles both security proofs *and* insecurity demonstrations with the same language and tools. This asymmetry means that protocol designers cannot use the same methodology to "test" their protocol against known attack classes before submitting a full proof.

✎ **Analogy:**

Insecurity Proofs as Product Recalls When a car manufacturer discovers a defective airbag, they issue a *recall*—a formal announcement that the product is broken, with a specific explanation of

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

the defect. In cryptography, insecurity proofs are like product recalls: they identify exactly what is broken and why. But currently, each recall uses a different format—there is no standardised "recall notice" template. MTSF provides this template: insecurity is a *market collapse* where the ask price reaches 1, meaning any attacker wins for free. The "recall notice" specifies which bid succeeded (e.g., "homomorphism bid" for textbook RSA, "masquerade bid" for Needham–Schroeder) and quantifies the damage.

Problem 5: Protocol-level security goods lack a unified market formulation.

Existing game-based frameworks treat entity authentication, mutual authentication, session-key secrecy, and session correctness as separate definitions with separate games and separate proofs. There is no unified treatment embedding all four as simultaneous *market goods* offered by a single seller, with the adversary bidding against all of them in parallel. This separation makes it hard to see which property fails first under a given attack and how the failures interact.

Problem 6: No framework addresses the QROM, unbounded sessions, and CNF correctness simultaneously.

Post-quantum security requires proofs in the Quantum Random Oracle Model (QROM), where adversaries can query hash functions in superposition. Classical ROM proofs fail in this setting. Simultaneously, real deployments run for unbounded numbers of sessions (not the polynomial bound assumed by game-based proofs). And CNF session correctness must be verified across all sessions. No existing framework addresses all three requirements—QROM, unbounded, and CNF—within a single proof methodology.

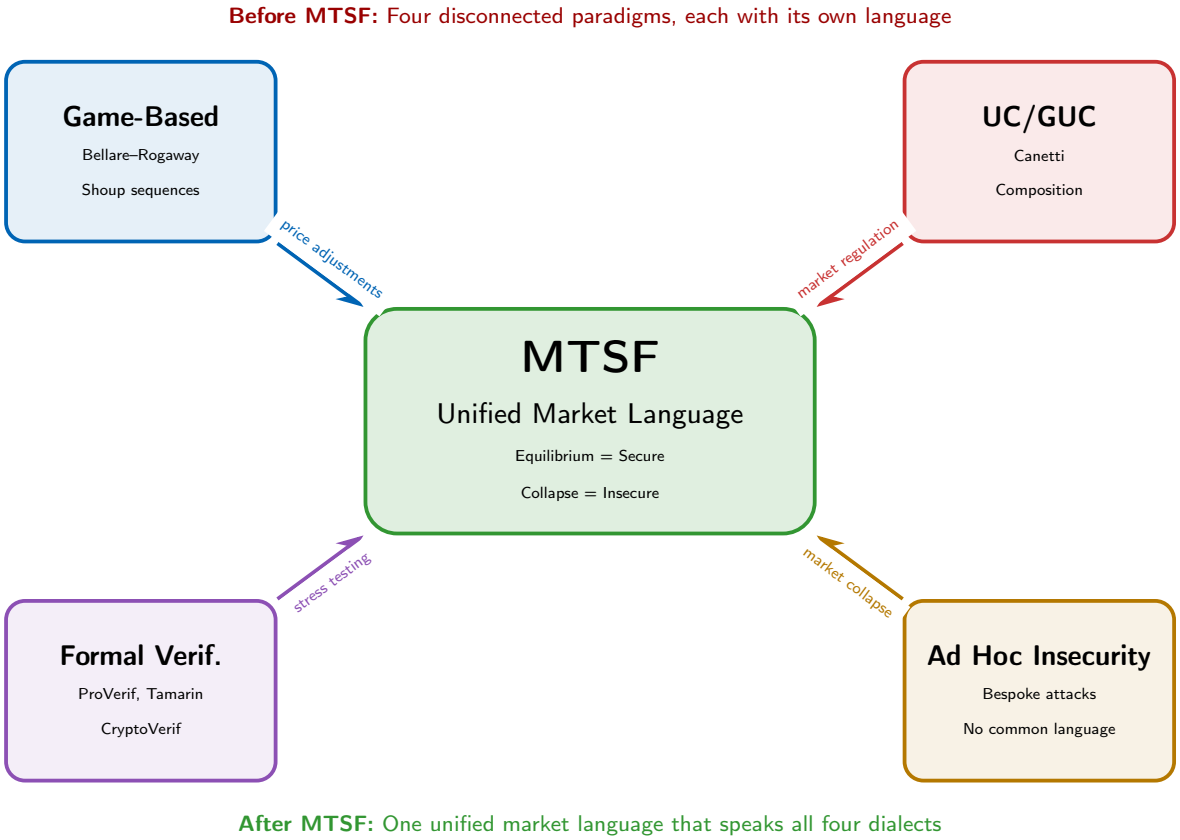


Figure 4. The unification problem: four disconnected security paradigms converge into MTSF's single market language. Game-based proofs become price adjustments, UC becomes market regulation, formal verification becomes stress testing, and ad hoc insecurity becomes market collapse.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

1.2. What MTSF Provides

☞ Key Insight:

What Is New in MTSF—Seventeen Contributions

1. **Unified auction model:** Security = market equilibrium (Ask $\leq$ negl); insecurity = market collapse (Ask = 1). The same language handles both.
2. **Extended difference lemma:** Captures $m \geq 1$ simultaneous failures in one game hop via $\Pr[F_1 \cup \cdots \cup F_m]$ with inclusion-exclusion tightening (Lemma 2). Generalises Shoup's single-failure version.
3. **Bidding-based proofs:** Each hop targets a specific adversarial strategy (nonce bid, hash-collision bid, public-key bid, forgery bid, MITM bid). The proof explicitly tracks *what* the adversary is attacking and *how much* it costs.
4. **Insecurity as market collapse:** Textbook RSA (Ask = 1 via the homomorphism bid) and Needham–Schroeder (Ask = 1 via the masquerade bid) are proved insecure within the same framework as secure schemes.
5. **Four-paradigm unification:** Game-based proofs (price adjustments), UC (market regulation by $\mathcal{Z}$), GUC (shared market infrastructure + CNF audit), and formal verification (market stress testing) are unified under one roof.
6. **CNF session verification + session pinging:** A canonical five-phase verification algorithm (Algorithm 1) with an easy four-column manual worksheet, plus a session-pinging mechanism for unbounded security (Section 5).
7. **Dedicated protocol-market games:** Authentication, mutual authentication, session-key secrecy, and CNF correctness games as first-class *security goods* sold in the protocol market (Section 8).
8. **Comprehensive primitive coverage:** ECDSA, ML-KEM, ML-DSA, HMAC, AEAD, SLH-DSA, FN-DSA—all analysed via bidding-round chains with CNF worksheets and ping bids.
9. **Block-cipher market:** AES, PRESENT (lightweight), and Serpent (conservative AES finalist) are analysed via differential, linear, rotational, boomerang, and related-key cryptanalysis bids in unified block-cipher markets (Section 10).
10. **Hash-function market:** Keccak/SHA-3, BLAKE3 (Merkle tree, structural length-extension immunity), and ASCON-Hash (NIST lightweight standard) are analysed for collision resistance, preimage resistance, and length-extension resistance (Section 11).
11. **Stream-cipher market:** Grain-128a, ChaCha20 (ARX, 256-bit key, nonce-misuse collapse), and Trivium (80-bit key, cube attack analysis) are analysed for state recovery, key recovery, distinguishing, and TMTO attacks (Section 12).
12. **QROM case study:** The FO-transform KEM is proven IND-CCA2 in the QROM using the O2H lemma and measure-and-reprogram, extending MTSF to quantum-adversarial settings (Section 14).
13. **Dual Telegram analysis:** MTProto 2.0 is both *disproved* (partial collapse under salt extraction) and a remediated protocol RMTP is *proved* secure—all within the same MTSF framework (Section 13.9).
14. **BB84 quantum market dynamics:** First full quantum market analysis where both seller and buyer are quantum. BB84 QKD analysed via quantum bidding rounds with information-theoretic security (Section 15).
15. **TLS+Signal multi-protocol composition:** First MTSF analysis of concurrent protocol execution via market merger theorem, with resource competition analysis and network CNF (Section 16).
16. **CNF verification bidding rounds:** Every protocol and primitive case study includes a CNF verification bidding round with a manual worksheet, making security auditing accessible to non-expert practitioners.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

17. **Ping bids in every case study:** Every case study includes a ping bid formally proving unbounded session security.

1.3. Key Technical Innovations

Extended difference lemma.

The classical difference lemma bounds a single failure event. Our extension (Lemma 2) handles m simultaneous events:

$$|\Pr[\mathbf{B}_k = 1] - \Pr[\mathbf{B}_{k+1} = 1]| \leq \Pr\left[\bigcup_{i=1}^{m} F_i\right] \leq \sum_{i=1}^{m} \Pr[F_i] - \sum_{i<j} \Pr[F_i \cap F_j] + \cdots$$

with inclusion-exclusion giving a tight upper bound when failure events are correlated. We apply this in every case study: ECDSA ($F_{\mathsf{nonce}} \cup F_{\mathsf{hash}} \cup F_{\mathsf{fork}} \cup F_{\mathsf{extract}}$), ML-DSA ($F_{\mathsf{rejection}} \cup F_{\mathsf{norm}}$), four-party protocols ($\bigcup_{i=1}^{4} F_{\mathsf{sig},i}$), and Telegram ($F_{\mathsf{salt}} \cup F_{\mathsf{entropy}} \cup F_{\mathsf{cnf}} \cup F_{\mathsf{ping}}$).

Session pinging for unbounded security.

Bounded game-based proofs say "secure for q sessions" but real systems run for millions of sessions over years. We define structural distinctness (Theorem 14) and the session ping function (Theorem 15), and prove by induction (Theorem 3) that if the base case is secure and all pings pass, then every session is secure—providing unbounded coverage with quantitative bounds. This bridges game-based proofs (bounded, quantitative) and formal verification (unbounded, qualitative).

Dual proof/disproof methodology.

MTSF characterises insecurity as market collapse: $\mathsf{Ask} = 1$ means any adversary wins for free. Secure schemes have $\mathsf{Ask} \leq \mathsf{negl}$. The same bidding-round machinery proves both. For the Needham–Schroeder protocol, we show the masquerade bid succeeds with probability 1 (no cryptographic break needed). For textbook RSA, the homomorphism bid succeeds in $O(1)$ time. For MTProto, the salt extraction bid quasi-collapses the market to $\mathsf{Ask} \approx 1$. These are not ad hoc observations—they are formal theorems within MTSF.

1.4. Article Organisation

Section 2: notation, semantic security, four paradigms with figures. Section 3: MTSF with extended difference lemma. Section 4: CNF session verification algorithm and easy manual worksheet. Section 5: session pinging for unbounded security. Section 6: thirteen unified novelties. Section 8: authentication, mutual authentication, session-key secrecy, and CNF checking games as market goods. Section 9: ECDSA, ML-KEM, ML-DSA (secure) and textbook RSA (insecure); extended primitives HMAC, AEAD, SLH-DSA, FN-DSA. Section 10: AES block-cipher market with differential, rotational, linear, and related-key bids. Section 11: Keccak/SHA-3 hash-function market. Section 12: Grain-128a stream-cipher market. Section 13: ISO/IEC 11770-3 protocol family (two-, three-, and four-party), PKI, Needham–Schroeder insecurity proof, Signal Protocol (X3DH + Double Ratchet) security proof and X3DH-noOPK insecurity proof, TLS 1.3 (1-RTT equilibrium proof, 0-RTT replay collapse, downgrade attack collapse, CNF session verification and unbounded ping), and Telegram MTProto disproof (salt extraction) and RMTP proof (HMAC-bound salts). Section 14: QROM-based FO-transform KEM with full protocol description, sequence diagram, and six-bidding-round proof. Section 15: BB84 Quantum Key Distribution as a full quantum market dynamics case study (quantum seller, quantum buyer, information-theoretic security via no-cloning and QLHL). Section 16: TLS 1.3 + Signal multi-protocol composition network case study (market merger theorem instantiation, resource competition analysis, network CNF verification). Section 19: comprehensive related work in ten subsections. Section 20: conclusion with seventeen contributions, lessons learned, and seven future directions (with initial formalisations of quantum market dynamics and multi-protocol composition networks). Appendix A:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

expanded MTSF correspondence table (79 entries). Appendix B: comprehensive bid taxonomy with QROM, Telegram, BB84, and composition bids.

Figure 5. Decision flowchart: "Which MTSF tool do I need?" Start with your cryptographic object, follow the arrows, and arrive at the correct MTSF section and proof technique.

2. Preliminaries and Background

2.1. Notation

Notation Cheat Sheet

λ: security parameter (e.g., 128). $\{0,1\}^n$: n-bit strings. $x \xleftarrow{\$} S$: sample uniformly. $\mathsf{negl}(\lambda)$: negligible function (shrinks faster than any inverse polynomial). PPT: probabilistic polynomial time. $\mathbf{B}_k$: k-th *bidding round* (MTSF market-theoretic notation; **bold upright B**). sid/pid: session/party identifier. σ/τ: signature/MAC tag. $\mathcal{A}$: adversary. $\mathcal{C}$: challenger.

✻ Simple Terms: What the Notation Means in Practice

Security parameter λ: Think of this as the "strength dial" of the cryptographic scheme. Setting $\lambda = 128$ means roughly 2^{128} operations are needed to break the scheme—more than the number of atoms in the observable universe. All other quantities (key sizes, running times) are expressed as functions of λ.

Negligible function $\mathsf{negl}(\lambda)$: A quantity that shrinks to zero faster than any polynomial fraction, e.g., $2^{-\lambda}$ or λ^{-100}. In plain English: so small that even repeating the experiment a trillion times, the event almost certainly never happens. When we say an adversary's advantage is negligible, we mean the scheme is secure in all practical terms.

PPT (Probabilistic Polynomial Time): An attacker that runs in time bounded by some polynomial in λ, e.g., λ^3 steps. This is the standard model of a "realistic" attacker with access to today's (and tomorrow's) computers. We only care about defeating PPT attackers—exponential-time attackers are irrelevant for practical security.

Session ID (sid) and Party ID (pid): A session ID uniquely labels one run of a protocol (like a phone call's unique call-log entry). A party ID identifies who is participating (like a phone number). Binding cryptographic material to these identifiers prevents replay attacks (using recordings of old calls).

Real-world scale of 2^{128}: To appreciate what $\lambda = 128$ means in practice, consider these comparisons:

- There are approximately 2^{80} atoms in the observable universe.
- The universe is approximately 2^{58} seconds old.
- The fastest supercomputer performs approximately 2^{60} operations per second.
- Even running every atom in the universe as a computer for the age of the universe gives roughly $2^{80+58+60} = 2^{198}$ operations total—still far short of 2^{256}.

When we say a scheme with $\lambda = 128$ has negligible advantage, we mean an attacker would need to perform operations numbering more than the atoms in the universe to have even a tiny chance of success.

2.2. Semantic Security and Core Definitions

✎ Analogy:

Cryptographic Security as a Guessing Game for Children Imagine Alice whispers one of two words—either "cat" or "dog"—into a magic box. The box scrambles the word into gibberish and shows the gibberish to Eve. Eve must guess which word Alice whispered. If the box is a *good*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

encryption scheme, Eve can do no better than flipping a coin—50/50. Even if Eve is the smartest person in the world with the fastest computer, the box gives her *zero* useful information.
That is semantic security: the ciphertext reveals nothing about the plaintext beyond what the attacker already knew. In MTSF terms: the buyer (Eve) bids her computational resources to distinguish "cat" from "dog," but the ask price of the confidentiality good is negligible—Eve's bid always fails.
Now imagine Eve can also ask the box to encrypt other words of her choosing ("fish," "bird," "tree") before making her guess. If the box *still* gives her no advantage, that is **IND-CPA** security (chosen-plaintext). If Eve can additionally ask the box to *decrypt* other gibberish of her choosing, and *still* cannot tell "cat" from "dog," that is **IND-CCA2** security—the gold standard.

Definition 1 (Semantic Security [11]). *Encryption* Π *is* semantically secure *if for every PPT* $\mathcal{A}$*, whatever* $\mathcal{A}$ *computes from ciphertext* $c = \mathsf{Enc}(K, m)$*, a PPT simulator* $\mathcal{S}$ *can compute from the message length* $|m|$ *alone:* $|\Pr[f \leftarrow \mathcal{A}(c)] - \Pr[f \leftarrow \mathcal{S}(|m|)]| \leq \mathsf{negl}(\lambda)$*. In market terms: the ciphertext is a worthless good—the buyer (adversary) gains zero information from purchasing it.*

✻ Simple Terms: Semantic Security

What it says: Looking at an encrypted message (a ciphertext) tells an attacker absolutely nothing useful about the original message beyond its length. Even the world's best computer program, given the ciphertext, cannot distinguish whether "Hello" or "Goodbye" was encrypted—it may as well guess randomly.
Why length leaks: You can count the bytes in a ciphertext, which reveals roughly how long the original message was. This is unavoidable in most schemes (you can pad messages to fixed lengths to hide even this). Semantic security accepts this small leak and guarantees nothing else leaks.
Market translation: A ciphertext is a product on the shelf. An adversary is a buyer inspecting it. Semantic security says: no matter how closely the buyer examines the product (ciphertext), they cannot determine its contents (the plaintext). The product reveals only its box size (message length).

Definition 2 (IND-CPA). $\mathcal{A}$ *submits* m_0, m_1*; receives* $\mathsf{Enc}(K, m_b)$ *for random* b*; guesses* b*. Secure if* $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}} \leq \mathsf{negl}(\lambda)$.

✻ Simple Terms: IND-CPA

The game: The adversary chooses two messages of equal length (say "Attack at dawn" and "Retreat to camp"). A challenger flips a coin and encrypts one of them. The adversary sees the ciphertext and tries to guess which message was encrypted. *Indistinguishability under Chosen Plaintext Attack* (IND-CPA) says the adversary cannot do better than random guessing (50/50).
Why "chosen plaintext"? The adversary can also ask for encryptions of *any* other messages they like before making their final guess. This models an attacker who can influence what gets encrypted (e.g., by sending decoy emails through a system they know is being monitored).
What it does NOT cover: The adversary cannot ask for decryptions. Schemes like stream ciphers can be IND-CPA secure but still leaky if an attacker can submit ciphertexts for decryption (see IND-CCA2 below).

Definition 3 (IND-CCA2). *As IND-CPA but* $\mathcal{A}$ *additionally gets a decryption oracle (except on the challenge). Secure if* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}} \leq \mathsf{negl}(\lambda)$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✻ Simple Terms: IND-CCA2

The game: Same as IND-CPA, but now the adversary can also submit *any ciphertext* (except the challenge ciphertext itself) to a decryption service and see the result. This models a very powerful attacker: imagine someone who can sneak messages through a decryption box (e.g., by tricking a smart card into decrypting).
Why is this the gold standard? If a scheme survives even this attack, it is safe in virtually all real-world deployment scenarios. Modern protocols (TLS, Signal, etc.) target IND-CCA2 security.
The "except the challenge" rule: The attacker cannot simply submit the challenge ciphertext for decryption—that would trivially win. Everything *else* can be decrypted.

Definition 4 (EUF-CMA). *$\mathcal{A}$ gets a signing oracle; must produce a valid signature on a* new *message. Secure if* $\Pr[\textit{forge}] \leq \mathsf{negl}(\lambda)$.

✻ Simple Terms: EUF-CMA

The game: The adversary can ask a signer to sign *any messages* they choose (like a counterfeiter who legally obtains genuine banknotes to study). Then the adversary must produce a valid signature on a *new message they never asked to be signed*. EUF-CMA (Existential Unforgeability under Chosen Message Attack) says this is computationally impossible.
Why "existential"? The adversary can choose whatever message they want to forge—they are not given a specific target. Even this ultimate freedom should not help them.
Real-world relevance: This is why you cannot forge a digital certificate for a bank's website: even if you've collected millions of legitimate certificates, you cannot produce a valid new one without the Certificate Authority's private key.

Definition 5 (SUF-CMA). *Strong unforgeability: $\mathcal{A}$ cannot produce* any *new valid (message, signature) pair, even for previously signed messages with a different signature.*

✻ Simple Terms: SUF-CMA vs. EUF-CMA

The extra strength: EUF-CMA allows the attacker to ask for a signature on message m and still requires them to forge a signature on a *different* message. SUF-CMA is stricter: even if the attacker already has a valid signature σ on m, they cannot produce a *different* valid signature $\sigma' \neq \sigma$ on the same m.
Why does this matter? Some protocols use signatures as receipts. If an attacker could create multiple valid signatures for the same message, they could confuse audit logs or create deniability. SUF-CMA closes this loophole.
Practical example: In HMAC-based authentication, strong unforgeability ensures that a MAC tag is unique: you cannot replace one valid tag with a different valid tag to alter a transaction record.

Definition 6 (INT-CTXT (Ciphertext Integrity)). *For an authenticated encryption scheme* AE*, the adversary $\mathcal{A}$ has access to an encryption oracle and wins if it produces any ciphertext c^* that decrypts successfully but was never output by the encryption oracle. Secure if* $\Pr[\textit{INT-CTXT forge}] \leq \mathsf{negl}(\lambda)$.

✻ Simple Terms: INT-CTXT

What it means: Ciphertext integrity ensures that nobody can create a *new* ciphertext that looks legitimate (decrypts without error) unless they used the legitimate encryption process. Even if an attacker sees many valid ciphertexts, they cannot craft a new one that passes the decryption check.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Real-world importance: Without INT-CTXT, an attacker could modify an encrypted bank transfer, and the receiver would happily decrypt the tampered ciphertext. With INT-CTXT, any modification causes decryption to fail (the "tag" verification rejects it).
AEAD combines IND-CCA2 + INT-CTXT: Authenticated Encryption with Associated Data (like AES-GCM, ChaCha20-Poly1305) provides both confidentiality (IND-CCA2) and integrity (INT-CTXT) simultaneously. This is the modern standard.

Definition 7 (PRF Security)**.** *A keyed function* $F_K : \{0,1\}^n \to \{0,1\}^m$ *is a* secure PRF *if no PPT distinguisher tells* $F_K(\cdot)$ *from a truly random function* $R(\cdot)$*:*

$$\mathsf{Adv}_F^{\mathsf{PRF}}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{F_K(\cdot)}=1] - \Pr[\mathcal{A}^{R(\cdot)}=1]\right| \leq \mathsf{negl}(\lambda).$$

✻ Simple Terms: PRF

What is a PRF? A Pseudorandom Function (PRF) is a keyed function that looks like a random function to anyone who does not know the key. You input a value, you get an output that looks completely random—but the same input always gives the same output for the same key.
The distinction game: An adversary can query the function on any inputs they choose. They must decide: "Am I talking to F_K (with some secret key), or to a truly random function?" PRF security says they cannot tell the difference.
Where PRFs appear: HMAC uses two PRF evaluations (inner and outer hash steps). Block ciphers like AES are modelled as PRFs/PRPs. Key derivation functions (KDFs) are PRFs. Virtually every symmetric-key construction relies on PRF security at its core.

2.3. Bidding-Round Proofs in the Random Oracle Model

✎ Analogy:

The Random Oracle as a Magic 8-Ball In cryptographic proofs, we often model hash functions as "random oracles"—imagine a magic 8-ball that:

- Gives a completely random answer for each new question.
- Always gives the *same* answer if you ask the same question again (deterministic).
- Has no pattern—knowing the answer to one question tells you nothing about any other answer.

Why use this model? Real hash functions like SHA-256 are deterministic algorithms, not truly random. But they are designed to *behave* as if random. The Random Oracle Model (ROM) lets us prove security assuming ideal randomness, then instantiate the oracle with a real hash function.
The "programming" trick: In a security proof, the seller (challenger) secretly controls the random oracle. When the buyer asks "what is $H(m)$?", the seller can choose the answer strategically—for example, embedding a hard-problem challenge. Since the oracle is "random," the buyer cannot tell the difference between a genuinely random answer and a strategically chosen one. This is the "oracle programming" bidding round—a *costless market restructuring* that costs $\Delta\text{Price} = 0$.
In MTSF: Random oracle queries are "hash bids"—the buyer pays computation to query the oracle, hoping to find useful information. Hash collisions (two queries with the same answer) are "birthday bids," costing $q^2/2^{n+1}$ by the birthday bound.

A *bidding round* $\mathbf{B}_k$ is a formal experiment between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$. The challenger sets up keys, answers oracle queries, and presents challenges; the adversary tries to win (guess correctly,

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

produce a forgery, etc.). The *bidding-round sequence* technique [1,2] transforms $\mathbf{B}_0$ (the real security experiment) into $\mathbf{B}_q$ (an ideal experiment where winning is impossible):

$$\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}}(\lambda) \leq \sum_{k=0}^{q-1} \underbrace{|\Pr[\mathbf{B}_k(\mathcal{A}) = 1] - \Pr[\mathbf{B}_{k+1}(\mathcal{A}) = 1]|}_{\text{price adjustment } \Delta\mathsf{Price}_k}. \tag{1}$$

✻ Simple Terms: Game-Hopping

The core idea: Instead of directly proving that the adversary cannot win the real game $\mathbf{B}_0$, we make a series of small, clearly justified changes to the game rules. Each change is so small that the adversary's winning probability barely shifts (by at most $\Delta\mathsf{Price}_k$). After all changes, we reach a "dream world" game $\mathbf{B}_q$ where winning is trivially impossible.
Analogy: Imagine you want to prove a complex recipe is safe to eat. Instead of analysing every ingredient simultaneously, you swap them one at a time (real ingredient → inert substitute), showing each swap changes nothing meaningful. After all swaps, the dish is obviously inert.
The bound: The total advantage is at most the *sum* of all small changes. If each $\Delta\mathsf{Price}_k$ is tiny (negligible), their sum is also tiny—and the scheme is secure.
Each $\Delta\mathsf{Price}_k$ in MTSF is a "price adjustment": in the market analogy, each game hop slightly adjusts the "price" (winning probability) in the security market. The total cost to the adversary sums these adjustments.

Lemma 1 (Classical Difference Lemma [1]). *If $\mathbf{B}_k$ and $\mathbf{B}_{k+1}$ are identical except when a single failure event F occurs, then:* $|\Pr[\mathbf{B}_k(\mathcal{A}) = 1] - \Pr[\mathbf{B}_{k+1}(\mathcal{A}) = 1]| \leq \Pr[F \text{ in } \mathbf{B}_k]$.

✻ Simple Terms: Classical Difference Lemma

What it says: If two versions of a game ($\mathbf{B}_k$ and $\mathbf{B}_{k+1}$) play out identically unless a specific "bad event" F happens, then the difference in winning probabilities is at most the probability that F happens at all.
Example: Suppose $\mathbf{B}_k$ uses a real key, and $\mathbf{B}_{k+1}$ uses a random key—the games only differ if the adversary notices the key has changed. The bad event F is "the adversary detects the key swap." If F is very unlikely (tiny probability), the adversary wins at almost the same rate in both games.
Why it matters: This lemma is the workhorse of game-hopping proofs. Each game hop corresponds to one application of this lemma, bounding how much each "swap" can help the adversary.

In the *Random Oracle Model* (ROM) [12], hash functions are modelled as perfectly random functions accessible only via queries. This idealisation enables clean reductions.

✻ Simple Terms: The Random Oracle Model

What is a random oracle? A random oracle is a mathematical idealisation of a hash function (like SHA-256). It is modelled as a magical black box: you give it any input, and it outputs a completely random, fixed string (fixed means: same input always gives same output). No algorithm can predict the output for a new input without querying the box.
Why use this model? Real hash functions are complex but deterministic algorithms. Proving things about them directly is very hard. In the ROM, we pretend hash functions are perfectly random, which lets us write clean, short proofs. The trade-off: ROM proofs do not guarantee security if the hash function has algebraic structure an attacker could exploit.
Who uses ROM proofs? Most practical digital signature schemes (ECDSA, RSA-PSS), key exchange protocols (ECDH-based), and password hashing schemes are analysed in the ROM. It has been standard practice since 1993 [12].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

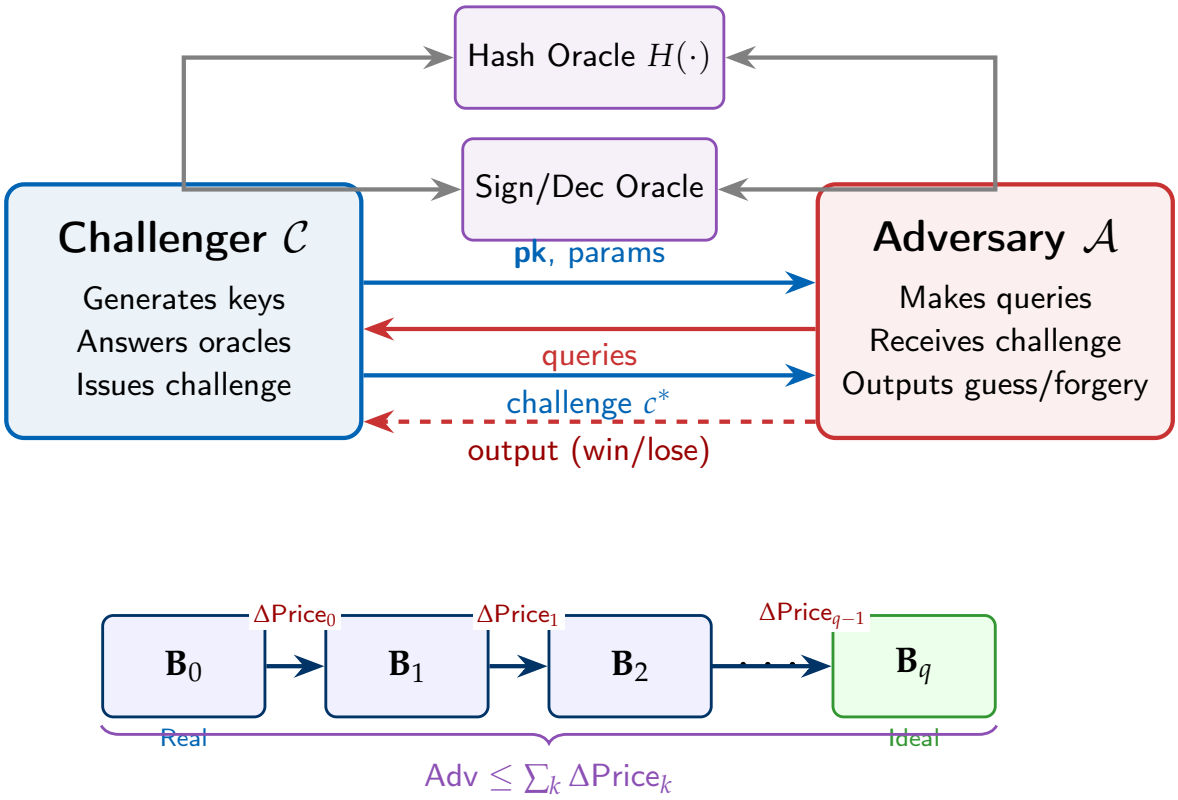


Figure 6. Bidding-round proof architecture. The seller (challenger) and buyer (adversary) interact via oracles. The bidding-round chain $\mathbf{B}_0 \to \mathbf{B}_q$ bounds the total advantage. In MTSF, each ΔPrice_k is a "price adjustment" in the security market.

2.4. Universal Composability (UC) Framework

The UC framework [3] guarantees that a protocol secure in isolation remains secure under *arbitrary* concurrent composition. A protocol π *UC-realises* an ideal functionality $\mathcal{F}$ if for every PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that no PPT environment $\mathcal{Z}$ can distinguish:

$$\text{Real}_{\pi,\mathcal{A},\mathcal{Z}} \approx_c \text{Ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}. \tag{2}$$

✻ Simple Terms: The UC Framework

The problem UC solves: A protocol might be secure on its own, but become insecure when run simultaneously with other protocols. For example, you might design a secure key exchange, but an attacker running a different protocol alongside it could learn the key. UC guarantees that this cannot happen.

The two worlds: UC compares two worlds:

- **Real world:** The actual protocol π runs, with a real adversary $\mathcal{A}$ attacking it.
- **Ideal world:** A perfectly secure "trusted third party" (the ideal functionality $\mathcal{F}$) handles all the sensitive operations. A simulator $\mathcal{S}$ mimics the adversary's view.

UC security says: no external observer (the environment $\mathcal{Z}$) can tell which world they are in. This means the real protocol is as good as having a perfect trusted third party.

The simulator: The simulator $\mathcal{S}$ plays the role of "market arbitrageur" in MTSF—it bridges the gap between real and ideal by producing fake views that look indistinguishable from real ones.

Composition theorem: If protocols Π_1 and Π_2 are both UC-secure, then running them together is also UC-secure. This is why UC is so powerful for building complex systems from secure components.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

In MTSF: The environment $\mathcal{Z}$ is the "market regulator"—it oversees all market transactions and tries to find any discrepancy between the real and ideal markets.

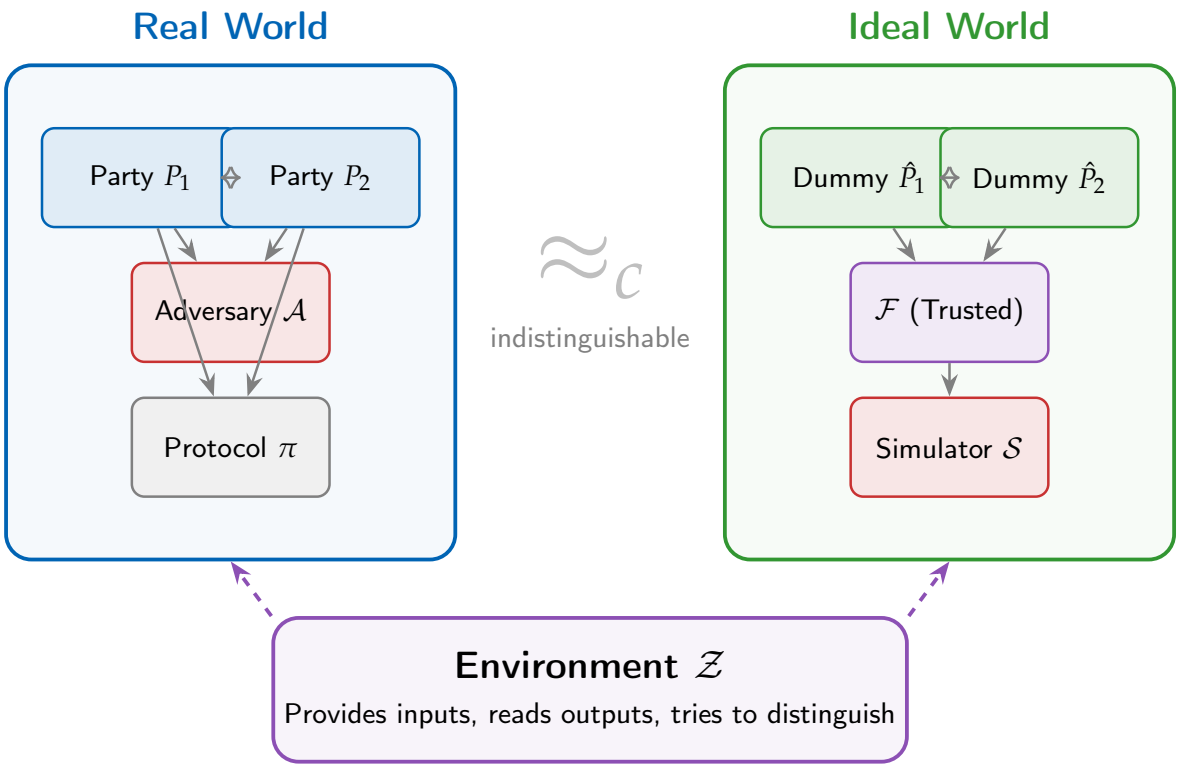


Figure 7. UC framework. The environment $\mathcal{Z}$ interacts with either the real world (protocol π with adversary $\mathcal{A}$) or the ideal world (trusted functionality $\mathcal{F}$ with simulator $\mathcal{S}$). UC security: $\mathcal{Z}$ cannot tell which world it observes.

2.5. GUC and CNF Session Correctness

✻ Simple Terms: Universal Composability (UC) for Non-Experts

The central question of UC: Can an outsider tell whether they are interacting with the *real* protocol (real people running the actual software) or an *idealised* version (a perfect, unhackable magic box)?
Step-by-step:

1. **Define the ideal:** First, describe what a perfect, unhackable version of the protocol would do. For encryption: "The magic box receives Alice's message, stores it, and when Bob asks, delivers it—no attacker can see or change it."
2. **Build the real:** Implement the actual protocol using cryptographic primitives (AES, ECDSA, etc.).
3. **Build the simulator:** Construct a "stunt double" ($\mathcal{S}$) that can make the ideal box *look like* the real protocol to any outside observer, using only the ideal box's interface.
4. **Prove indistinguishability:** Show that no outside observer (the environment $\mathcal{Z}$) can tell whether they are talking to (real protocol + real attacker) or (ideal box + simulator).

The power of UC: Once you prove a protocol UC-secure, you can *safely replace* the real protocol with the ideal box in any larger system. This is the composition theorem: UC-secure protocols compose safely. In MTSF, this is the "market merger theorem"—merging two equilibrium markets (secure protocols) produces a combined equilibrium market.

The GUC framework [4] removes the CRS assumption via *shared functionalities*. Camenisch *et al.* [5] encode each session as a CNF formula $\varphi_{\mathsf{sid}} = \bigwedge_j C_j$, where satisfiability under the honest trace implies session correctness.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✻ Simple Terms: GUC Framework

What the CRS assumption is: The basic UC framework sometimes requires a "Common Reference String" (CRS)—a string of random bits that everyone can see and that was honestly generated. In practice, it is hard to guarantee a CRS was generated honestly (who generates it and how?). GUC (Generalized UC) removes this requirement by using *shared functionalities*: well-known public infrastructure like a public-key directory (PKI) that everyone already trusts.
CNF session formulas: Instead of checking session correctness informally, GUC encodes it as a logical formula in Conjunctive Normal Form (CNF)—a checklist where every item must pass. Think of it as a contract: each clause is one contractual obligation. The session is "correct" (the contract is fulfilled) if and only if every obligation is met.
In MTSF: The CNF formula is a "market audit." The seller (protocol) passes the audit if the formula is satisfied. A dishonest trace (an attacker tampering with the session) fails the audit because at least one clause (e.g., the signature-verification clause) will be unsatisfied.

2.6. Formal Verification

✻ Simple Terms: Formal Verification Tools for Non-Experts

What ProVerif and Tamarin do: These are automated tools that check whether a protocol is secure by exhaustively exploring all possible attack scenarios. Think of them as very thorough robot testers:

- **ProVerif** uses the Dolev–Yao model, treating cryptography as perfect (unbreakable) black boxes. It checks whether the protocol's *design* is flawed—like Needham–Schroeder's missing identity binding. ProVerif found the attack on Needham–Schroeder automatically.
- **Tamarin** is more expressive: it can model algebraic properties of cryptographic operations (e.g., the XOR of two identical values is zero). It has been used to verify TLS 1.3, Signal, and other complex protocols.
- **CryptoVerif** bridges the gap by performing machine-checked game-based proofs, but requires significant manual guidance.

Strengths: Exhaustive coverage (no attack is missed), automated (less human error), and can handle unbounded numbers of sessions.
Limitations: Binary output (secure/insecure, no numerical bounds), may not terminate for complex protocols, and the abstraction may miss computational attacks (e.g., birthday bounds).
In MTSF: Formal verification is "market stress testing"—the regulator (environment) enumerates all buyer strategies and checks that none succeed. MTSF adds numerical bounds and composition guarantees that formal tools lack, while benefiting from their exhaustive coverage for trace-based attacks.

ProVerif [6] and Tamarin [7] provide automated, unbounded symbolic analysis. They check secrecy, authentication, replay/masquerade/MITM resistance. Limitation: no concrete advantage bounds.

✻ Simple Terms: Formal Verification Tools

What ProVerif and Tamarin do: These are automated tools that take a protocol description as input and mathematically check whether security properties hold—for *all possible* attack scenarios, not just the ones a human thought of. They model the attacker as having full control of the network and search exhaustively.
ProVerif works by translating the protocol into logic rules and using resolution (a kind of automated theorem proving) to check whether an attacker can derive secret values. It is fast but works in an abstract "symbolic" model (messages are terms, not bit strings).
Tamarin uses a more powerful graph-based approach and can reason about complex properties like forward secrecy. It handles a broader class of protocols but can require more manual guidance.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Their limitation in MTSF context: These tools tell you *whether* a protocol is secure (yes/no), but not *how hard* it is to break. They give no numerical advantage bound like "the attacker wins with probability at most 2^{-128}." MTSF bridges this gap by combining formal verification's exhaustiveness with game-based proof's quantitative bounds.
In MTSF: Formal verification is "stress testing" the security market—like a regulator running worst-case scenarios on all possible trading strategies to check for market manipulation.

3. The Market-Theoretic Security Framework

3.1. Security Market Model

Definition 8 (Security Market). $\mathcal{M} = (\mathsf{Seller}, \{\mathsf{Buyer}_i\}_{i\in[n]}, \mathcal{G}, \mathsf{Price})$*: seller (challenger), buyers (adversaries), security goods catalogue* $\mathcal{G} = \{g_1, \ldots, g_k\}$ *(e.g., IND-CPA, EUF-CMA, key secrecy), price function* $\mathsf{Price} : \mathcal{G} \times \mathbb{N} \to [0,1]$.

✻ Simple Terms: Security Market

Components of the market:

- **Seller (Challenger $\mathcal{C}$):** The cryptographer who designed the scheme and sets up the security experiment. The seller "offers" guarantees like "this encryption hides your messages."
- **Buyers (Adversaries $\mathcal{A}_i$):** Potential attackers with bounded computational power. Each buyer has a "budget" (time and memory limit) and tries to find a flaw in the seller's goods.
- **Security goods catalogue $\mathcal{G}$:** The list of properties being sold—e.g., confidentiality (IND-CPA), unforgeability (EUF-CMA), session-key secrecy. Each good is a specific security guarantee.
- **Price function** Price**:** Maps each good to the probability an attacker can break it. A "price" near 0 means the good is worthless to attackers (secure). A "price" near 1 means any attacker can easily break it (insecure).

The key insight: Security is not binary. Instead of "secure" or "broken," MTSF measures *how hard* it is to break each guarantee, using the language of market prices. A good is "fairly priced" (secure) when no efficient attacker can exploit it for more than a negligible advantage.

Definition 9 (Bid). $\mathsf{Bid}_i = (g_j, T_i, \epsilon_i)$*: buyer i targets good* g_j *with budget* $T_i \leq \mathsf{poly}(\lambda)$ *and target advantage* ϵ_i*. Succeeds if advantage* $\geq \epsilon_i$ *within* T_i *steps.*

✻ Simple Terms: Bids

What a bid is: A bid is an attacker's attempt to break a specific security property. It has three components:

- **Target good g_j:** Which security property the attacker is trying to break (e.g., "I am bidding on breaking signature unforgeability").
- **Budget T_i:** How much computation the attacker can afford (must be polynomial in λ—realistic).
- **Target advantage ϵ_i:** How likely the attacker hopes to succeed (e.g., "I want to win at least 1% of the time").

A bid succeeds if the attacker achieves advantage $\geq \epsilon_i$ within budget T_i. For the scheme to be secure, *all* bids must fail—no attacker can achieve non-negligible advantage within any polynomial budget.
Different bid types (introduced throughout the article):

- **Nonce bid:** Tries to exploit reused random values.
- **Forgery bid:** Tries to produce a valid signature without the private key.
- **Homomorphism bid:** Exploits algebraic structure (e.g., in RSA) to forge without the key.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Differential bid:** Exploits patterns in how small input changes affect outputs (used against block ciphers).
- **Impersonation bid:** Tries to pretend to be another party in a protocol.

Definition 10 (Quoted Price). *The* quoted price $\mathsf{QPrice}_0(g_j)$ *is the seller's initial listing price for security good* g_j *in bidding round* $\mathbf{B}_0$*. It equals the adversary's raw advantage in the unmodified real-world game:*

$$\mathsf{QPrice}_0(g_j) \;=\; \mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda) \;=\; \mathsf{Ask}_0(g_j, \lambda). \tag{3}$$

The quoted price is an unknown but bounded *quantity: it is the number the entire proof exists to bound. Subsequent bidding rounds* $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_q$ *decompose* QPrice_0 *into a telescoping sum of price adjustments:*

$$\mathsf{QPrice}_0 \;=\; \underbrace{\Delta\mathsf{Price}_0}_{\text{bid 1}} + \underbrace{\Delta\mathsf{Price}_1}_{\text{bid 2}} + \cdots + \underbrace{\Delta\mathsf{Price}_{q-1}}_{\text{bid } q} + \underbrace{\Pr[\mathbf{B}_q = 1]}_{\text{ideal residual}}. \tag{4}$$

Security holds when $\mathsf{QPrice}_0 \leq \mathsf{negl}(\lambda)$*; insecurity holds when* $\mathsf{QPrice}_0 = \Omega(1)$.

✻ Simple Terms: The Quoted Price—Why It Matters

What the quoted price is: When the seller opens the market in $\mathbf{B}_0$, the security good is "listed" at a quoted price QPrice_0. Think of it like a market where the seller lists the good at quoted price "999" (insecure—the good is worthless), "555" (quasi-secure—partial value), or "333" (secure—the price is a bluff the adversary cannot collect). The quoted price is the raw adversary advantage—*before any analysis has been done*. It is the starting point of the proof, not the conclusion.

Why it remains "as quoted" without reduction: In $\mathbf{B}_0$, no modifications have been made to the game. The adversary has full, unrestricted access to all oracles. The quoted price QPrice_0 is simply the adversary's best winning probability in this strongest setting. No price adjustment occurs ($\Delta\mathsf{Price}_{-1} = 0$) because nothing has changed yet. The quoted price stands as-is—it is the *benchmark* against which all subsequent bidding rounds are measured.

The proof's job: Show that QPrice_0 decomposes entirely into negligible terms. Each bidding round $\mathbf{B}_k$ peels off one component of QPrice_0 (one attack vector), showing it contributes at most $\Delta\mathsf{Price}_{k-1} \leq \mathsf{negl}$. After all attack vectors are peeled off, the residual is zero (ideal game). Since $\mathsf{QPrice}_0 = \sum \Delta\mathsf{Price}_k + 0$, and each $\Delta\mathsf{Price}_k$ is negligible, QPrice_0 itself is negligible. The originally quoted price was a *bluff*—the adversary cannot actually collect on it.

Three illustrative quoted prices:

- $\mathsf{QPrice}_0 =$ **"999" (Needham–Schroeder):** The seller quotes a high price, but the proof reveals it cannot be reduced—the masquerade bid succeeds with $\Delta\mathsf{Price} = 1$. The quoted price was honest: the good really is worthless (market collapsed).
- $\mathsf{QPrice}_0 =$ **"555" (Telegram MTProto):** The seller quotes a moderate price. Most bids are peeled off as negligible, but the salt-extraction bid contributes $\Delta\mathsf{Price} \approx 2^{-64}$—small but not negligible for high-security applications. The quoted price was *partially* a bluff: the good has some value but not full value (quasi-collapse).
- $\mathsf{QPrice}_0 =$ **"333" (ECDSA, ML-KEM, AES):** The seller quotes a modest price. Every bid is peeled off as negligible: nonce bids ($q_S^2/2q$), hash bids ($q_H^2/2q$), forgery bids ($\mathsf{Adv}^{\mathsf{ECDLP}}$). The residual is zero. The quoted price was entirely a bluff—the adversary wins essentially nothing (equilibrium).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

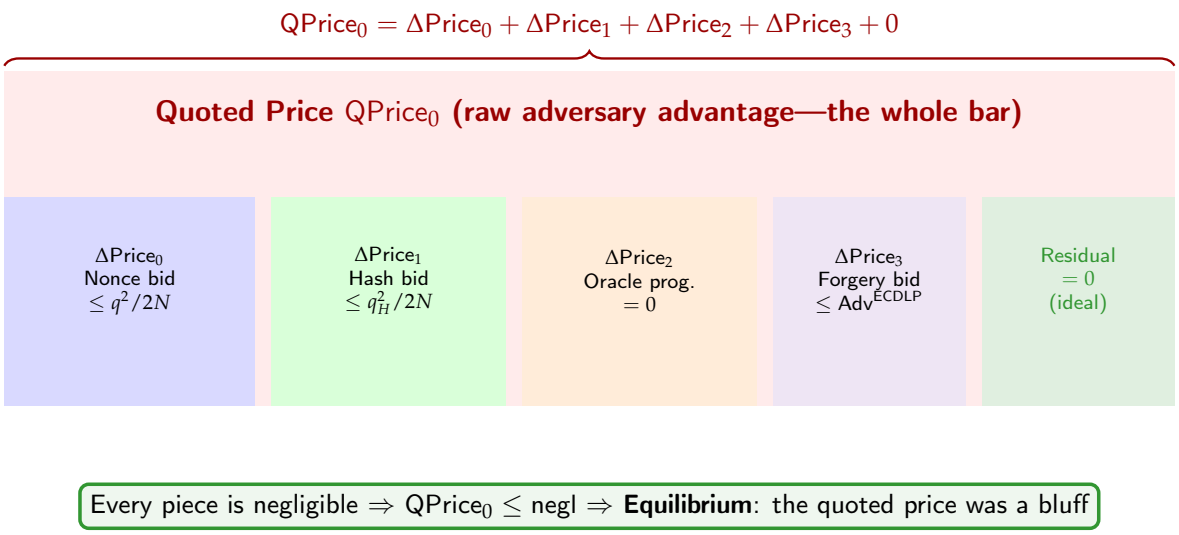


Figure 8. The quoted price QPrice_0 as a bar decomposed into bidding-round contributions. Each bidding round "peels off" one piece of the quoted price by bounding one attack vector. If every piece is negligible and the residual is zero, the quoted price is negligible—the market is in equilibrium.

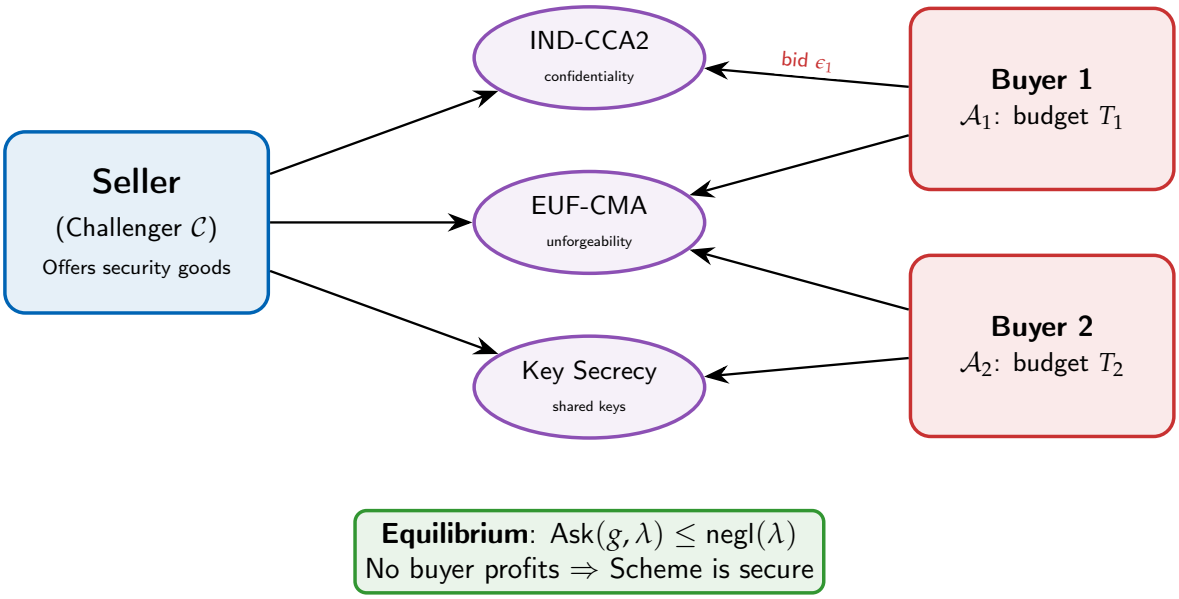


Figure 9. MTSF market: seller offers security goods; buyers bid computation; equilibrium = security.

Definition 11 (Ask Price & Equilibrium)**.** $\mathsf{Ask}(g_j,\lambda) = \max_{\mathcal{A}\in\mathsf{PPT}} \mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda)$. *The market is in* equilibrium *if* $\mathsf{Ask}(g_j,\lambda) \leq \mathsf{negl}(\lambda)$*: no PPT buyer profits. The market has* collapsed *if* $\mathsf{Ask}(g_j,\lambda) = \Omega(1)$*: some buyer profits cheaply.*

✻ Simple Terms: Ask Price and Equilibrium

Ask price: This is the best an attacker can possibly do against a security good—the maximum winning probability over all efficient attacks. Think of it as the "market value" of a security vulnerability: how much can an attacker extract from the scheme?

Equilibrium (scheme is secure): The ask price is negligible—essentially zero. No efficient attacker can win with any meaningful probability. The market has settled at a "fair price": security guarantees cannot be broken cheaply.

Collapse (scheme is broken): The ask price is substantial—an attacker wins easily. The market collapses like a stock crash: security guarantees are worthless.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Concrete examples:

- **ECDSA:** Ask price $\approx 2^{-128}$ for standard parameters. Equilibrium.
- **Textbook RSA signatures:** Ask price $= 1$ (attacker always wins). Collapsed.
- **Needham–Schroeder protocol:** Ask price $= 1$ (Lowe's man-in-the-middle attack always succeeds). Collapsed.

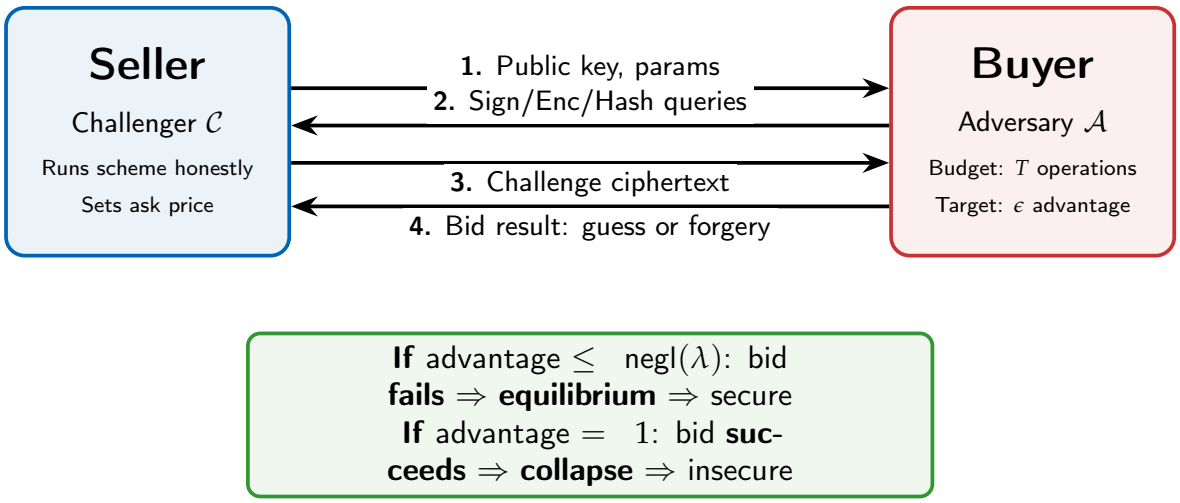


Figure 10. Individual seller–buyer interaction: the four-phase security game as a market transaction.

Theorem 1 (Equilibrium $\Leftrightarrow$ Security; Collapse $\Leftrightarrow$ Insecurity)**.** *Π is secure for g_j iff $\mathcal{M}$ is in equilibrium. Π is insecure iff $\mathcal{M}$ has collapsed.*

✎ Analogy:

Security as a Farmers' Market Think of a farmers' market. The seller (farmer) offers organic produce (security guarantees). Buyers (adversaries) try to find defects (vulnerabilities) to negotiate a lower price (gain advantage). If the produce is genuinely organic (the scheme is secure), no buyer can find a real defect—the market price holds (equilibrium). If the produce is contaminated (the scheme is broken), any buyer can spot the defect immediately—the price collapses to zero (market collapse).

✎ Analogy:

The Complete MTSF Dictionary for Everyday Life Here is the complete translation table between MTSF and everyday concepts. Keep this in mind as you read:

- **Security market** = A courtroom trial where the prosecution (buyer) tries to prove the defendant (scheme) is guilty (insecure), and the defence (seller) must show innocence (security). The judge (mathematical proof) delivers the verdict.
- **Security good** = A specific claim the defence makes: "My client never stole anything" (confidentiality), "My client's signature was never forged" (unforgeability), "My client's conversations were always private" (key secrecy).
- **Bid** = A specific line of attack by the prosecution: "I will show the defendant reused a password" (nonce bid), "I will show someone faked a signature" (forgery bid).
- **Ask price** = How strong the prosecution's best case is. Near zero = the defendant is clearly innocent. Near one = the prosecution wins easily.
- **Equilibrium** = The defence wins; the scheme is secure. Like a stock price that holds steady because no trader can exploit it.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Collapse** = The prosecution wins; the scheme is broken. Like a stock crash—everyone can see the vulnerability.
- **Price adjustment** = The small change in the prosecution's strength when one line of evidence is added or removed. Each game hop adjusts the price by a tiny amount.
- **Bidding round** = One round of the trial where a specific piece of evidence is examined.

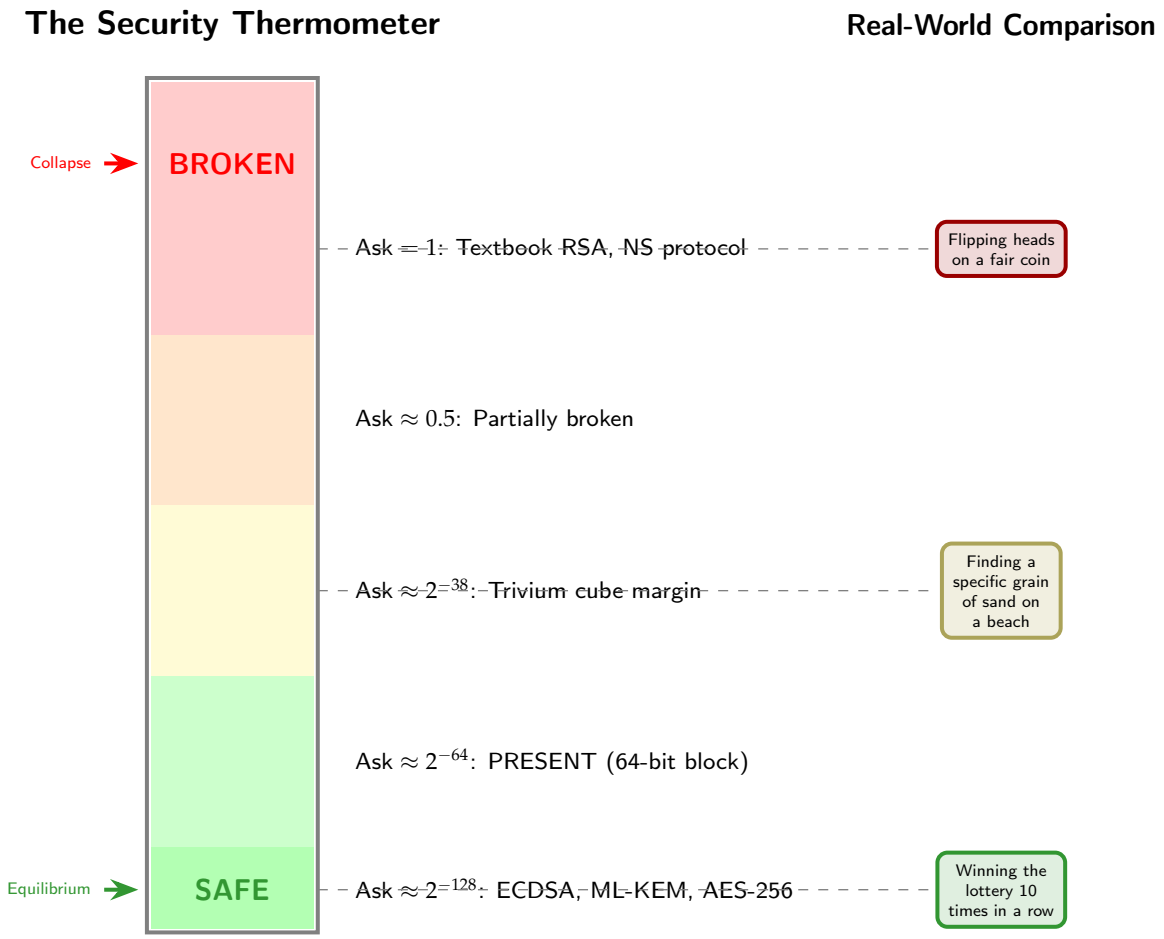


Figure 11. The Security Thermometer: visualising ask prices on a scale from perfectly secure (bottom, green) to completely broken (top, red). Real-world case studies are plotted along the scale, with everyday probability comparisons on the right.

3.2. Game Hops as Price Adjustments

Definition 12 (Price Adjustment). $\Delta\text{Price}_k = |\Pr[\mathbf{B}_k(\mathcal{A}) = 1] - \Pr[\mathbf{B}_{k+1}(\mathcal{A}) = 1]|$. *Cumulative cost:* $\text{Cost}(\mathcal{A}) = \sum_{k=0}^{q-1} \Delta\text{Price}_k$.

✎ Analogy:

Game Hops as Climbing a Staircase Imagine you are standing at the bottom of a staircase (the real security game, $\mathbf{B}_0$) and you want to reach the top (the ideal game, $\mathbf{B}_q$, where the attacker has zero advantage). Each step you climb is one *game hop*—a small, controlled change to the experiment.
The key rule: At each step, you measure how much the "height" (the attacker's advantage) changes. If each step changes the height by at most a tiny amount (ΔPrice_k), then by the time you reach the top, the total height change is at most the sum of all the tiny step changes. Since the ideal game has advantage zero, the real game's advantage is bounded by the sum of all step changes.
In pictures:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$$\underbrace{\mathbf{B}_0}_{\text{real}} \xrightarrow{\Delta_0} \mathbf{B}_1 \xrightarrow{\Delta_1} \mathbf{B}_2 \xrightarrow{\Delta_2} \cdots \xrightarrow{\Delta_{q-1}} \underbrace{\mathbf{B}_q}_{\text{ideal (Adv=0)}}$$

Total advantage $\leq \Delta_0 + \Delta_1 + \cdots + \Delta_{q-1}$.
Each Δ_k is bounded by the probability of a "bad event" F_k—like a nonce collision or a hash collision. If all bad events are extremely unlikely, the total advantage is negligible, and the scheme is secure.
Real-world analogy: Think of flying from New York to London with layovers. Each flight segment has a tiny risk of delay. The total delay risk is bounded by adding up the risks of each segment. If each segment has only a 0.0001% risk of significant delay, even 10 segments give only 0.001% total risk—negligible.

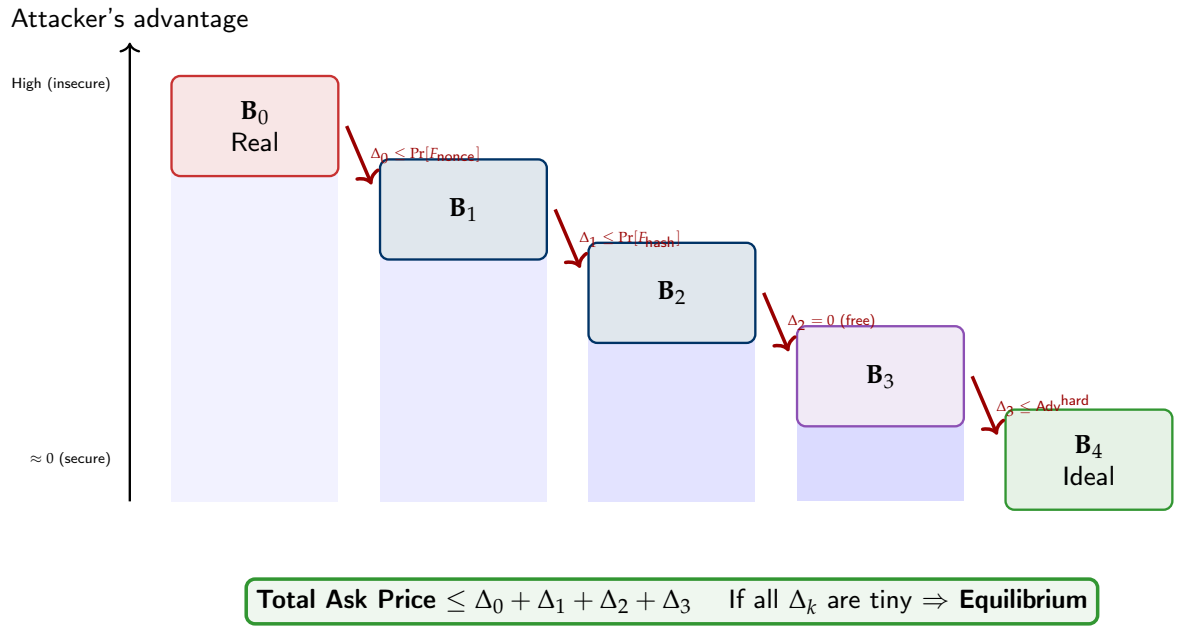


Figure 12. Game hops as a descending staircase: each step reduces the attacker's advantage by eliminating one attack vector. The total advantage (height of the staircase) is the sum of all step sizes. If every step is negligibly small, the scheme is secure.

Game Hops vs. Bidding Rounds: A Side-by-Side Comparison

The following table shows how a classical game-based proof step translates into an MTSF bidding round. The left column is what a game-based cryptographer would write; the right column is the MTSF version of the *same* argument. Notice that the MTSF version preserves all technical content while adding classification, market interpretation, and tighter multi-failure handling.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Classical Game Hop	MTSF Bidding Round
G_0: Real scheme.	$\mathbf{B}_0$ (Real Market): Seller offers goods at ask price Ask_0.
G_1: Abort on nonce collision. "$\lvert\Pr[G_0{=}1] - \Pr[G_1{=}1]\rvert \le q^2/2N$."	$\mathbf{B}_1$ (**Nonce Freshness Bid**): Buyer attempts nonce-reuse attack. $\Delta\mathsf{Price}_0 \le q^2/2N$. *Bid fails: nonce space too large.*
G_2: Replace hash with random. "Indistinguishable by RO."	$\mathbf{B}_2$ (**Oracle Programming—Costless Restructuring**): Seller secretly programs oracle. $\Delta\mathsf{Price}_1 = 0$. *Market restructured at zero cost.*
G_3: Reduce forgery to ECDLP. "Standard reduction."	$\mathbf{B}_3$ (**Forgery-to-ECDLP Bid**): If buyer forges, seller consumes buyer as subroutine to solve ECDLP. $\Delta\mathsf{Price}_2 \le q_H q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}$. *Bid fails: ECDLP is hard.*
G_q: Ideal. $\Pr[G_q{=}1] = 0$.	$\mathbf{B}_q$ (Ideal Market): All bids eliminated. $\Pr[\mathbf{B}_q{=}1] = 0$. *Market in equilibrium.*
Total: $\mathsf{Adv} \le \sum \delta_k$.	**Total:** $\mathsf{Ask}(g) \le \sum \Delta\mathsf{Price}_k$. *Plus: each $\Delta\mathsf{Price}_k$ is labelled by bid type; multi-failure hops use extended diff. lemma; CNF worksheet and ping bid appended.*

Key takeaway: The mathematical content is *identical*. The difference is that MTSF provides a richer annotation structure (named bids, explicit market interpretation, multi-failure handling, session auditing) that makes the proof more informative, auditable, and composable. Translating an existing game-based proof into MTSF is a mechanical process that adds value at no technical cost.

3.3. The Extended Difference Lemma

The classical difference lemma (Lemma 1) handles *one* failure per hop. In practice, a single game transition often involves multiple simultaneous failures. We extend the lemma:

Lemma 2 (Extended Difference Lemma—Multiple Failures)**.**

✎ ***Analogy:***

Multiple Failure Events as Insurance Claims In insurance, a single event (a storm) can trigger multiple claims simultaneously: roof damage (F_1), flooding (F_2), power outage (F_3). The insurer's total payout is bounded by the probability of any *claim occurring:* $\Pr[F_1 \cup F_2 \cup F_3]$. *If the events are independent, the union bound suffices. If correlated (flooding causes power outage), inclusion-exclusion gives a tighter bound by subtracting the overlap.*

Let $\mathbf{B}_k$ and $\mathbf{B}_{k+1}$ be identical unless at least one of $F_1, F_2, \ldots, F_m$ occurs. Then:

$$\lvert\Pr[\mathbf{B}_k(\mathcal{A}) = 1] - \Pr[\mathbf{B}_{k+1}(\mathcal{A}) = 1]\rvert \le \Pr[F_1 \cup F_2 \cup \cdots \cup F_m]. \tag{5}$$

*By the **union bound**: $\Pr[\bigcup_i F_i] \le \sum_i \Pr[F_i]$. When failures are correlated, the **inclusion-exclusion** bound is tighter:*

$$\Pr\left[\bigcup_i F_i\right] = \sum_i \Pr[F_i] - \sum_{i<j} \Pr[F_i \cap F_j] + \cdots + (-1)^{m+1} \Pr\left[\bigcap_i F_i\right]. \tag{6}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Proof. $\mathbf{B}_k$ and $\mathbf{B}_{k+1}$ agree on $\overline{\bigcup_i F_i}$. Hence:

$$
\begin{aligned}
&|\Pr[\mathbf{B}_k = 1] - \Pr[\mathbf{B}_{k+1} = 1]| \\
&= \left|\Pr\left[\mathbf{B}_k = 1 \wedge \bigcup F_i\right] + \Pr\left[\mathbf{B}_k = 1 \wedge \overline{\bigcup F_i}\right] - \Pr\left[\mathbf{B}_{k+1} = 1 \wedge \bigcup F_i\right] - \Pr\left[\mathbf{B}_{k+1} = 1 \wedge \overline{\bigcup F_i}\right]\right| \\
&= \left|\Pr\left[\mathbf{B}_k = 1 \wedge \bigcup F_i\right] - \Pr\left[\mathbf{B}_{k+1} = 1 \wedge \bigcup F_i\right]\right| \leq \Pr\left[\bigcup_i F_i\right]. \quad \square
\end{aligned}
$$

✻ Simple Terms: Extended Difference Lemma

What is new over the classical lemma? The classical difference lemma handles one bad event per game hop. But in complex proofs—like the ECDSA proof—a single game transition might involve *three* things that could go wrong simultaneously: a nonce collision, a hash collision, and a forking failure. Splitting this into three separate game hops is cumbersome.

The extended lemma: We can handle all m bad events $(F_1, \ldots, F_m)$ in a single hop, bounding the total difference by the probability that *any* bad event occurs: $\Pr[F_1 \cup F_2 \cup \cdots \cup F_m]$.

Union bound vs. inclusion-exclusion:

- **Union bound** (simple): $\Pr[F_1 \cup F_2 \cup \cdots \cup F_m] \leq \Pr[F_1] + \Pr[F_2] + \cdots + \Pr[F_m]$. Easy to compute but potentially loose (overcounts overlaps).
- **Inclusion-exclusion** (tighter): Subtract pairwise overlaps ($\Pr[F_i \cap F_j]$), add back triple overlaps, etc. Tighter but more complex.

When to use which: If the bad events are essentially independent (like independent hash collisions), the union bound is nearly tight. If they are correlated (e.g., a nonce collision also causes a hash collision), inclusion-exclusion gives a significantly better bound.

Market interpretation: Multiple risks (bad events) are like correlated insurance claims. If flooding always causes power outages, you do not need to pay for two separate "flood" and "outage" claims—you account for the overlap. MTSF's extended lemma does this for security proofs.

☞ Key Insight:

Why the Extension Matters Consider an ECDSA game hop where three things can go wrong simultaneously: F_1: nonce collision ($\leq q_S^2/2q$), F_2: hash collision ($\leq q_H^2/2q$), F_3: forking failure ($\leq 1/q_H$). Classical approach: split into three separate hops, each with its own game. Extended lemma: handle all three in *one* hop: $\Delta\text{Price} \leq \Pr[F_1 \cup F_2 \cup F_3] \leq \Pr[F_1] + \Pr[F_2] + \Pr[F_3] - \Pr[F_1 \cap F_2] - \cdots$, yielding a tighter bound and shorter proof.

✎ Analogy:

Extended Difference Lemma as a Kitchen Fire Risk Assessment Suppose you are assessing the risk of a kitchen fire. Three things could go wrong simultaneously:

- F_1: The stove is left on (probability 2%).
- F_2: A towel is near the stove (probability 5%).
- F_3: The smoke detector battery is dead (probability 1%).

Union bound (simple): $\Pr[\text{any problem}] \leq 2\% + 5\% + 1\% = 8\%$. This overcounts because if the stove is left on AND a towel is nearby, we counted that scenario twice.

Inclusion-exclusion (tighter): Subtract the overlaps. If $\Pr[F_1 \cap F_2] = 0.5\%$ (stove on AND towel nearby), then: $\Pr[\text{any problem}] \leq 8\% - 0.5\% - \ldots \approx 7.5\%$.

In MTSF: Replace "stove" with "nonce collision," "towel" with "hash collision," and "smoke detector" with "forking failure." The extended difference lemma handles all three risks in a single game hop instead of requiring three separate hops. The tighter inclusion-exclusion bound means the final security estimate is closer to reality.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

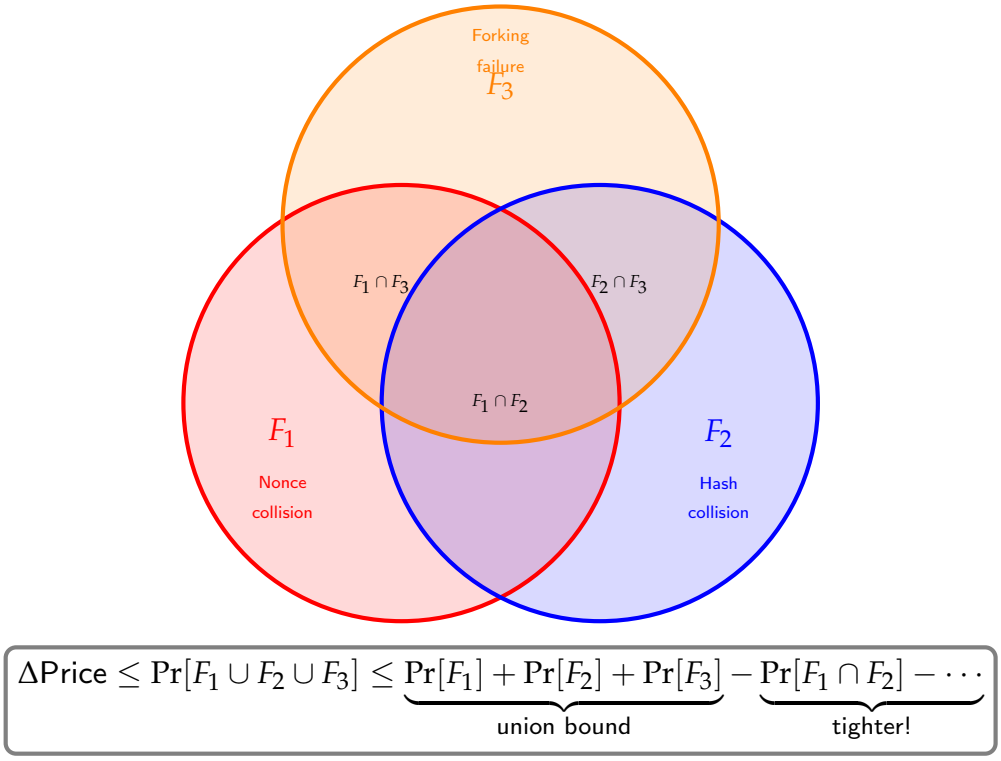


Figure 13. Venn diagram of simultaneous failure events in the extended difference lemma. The union bound adds all circles; inclusion-exclusion subtracts the overlaps for a tighter bound. Each failure event corresponds to a specific "bad thing" that could happen during a game hop.

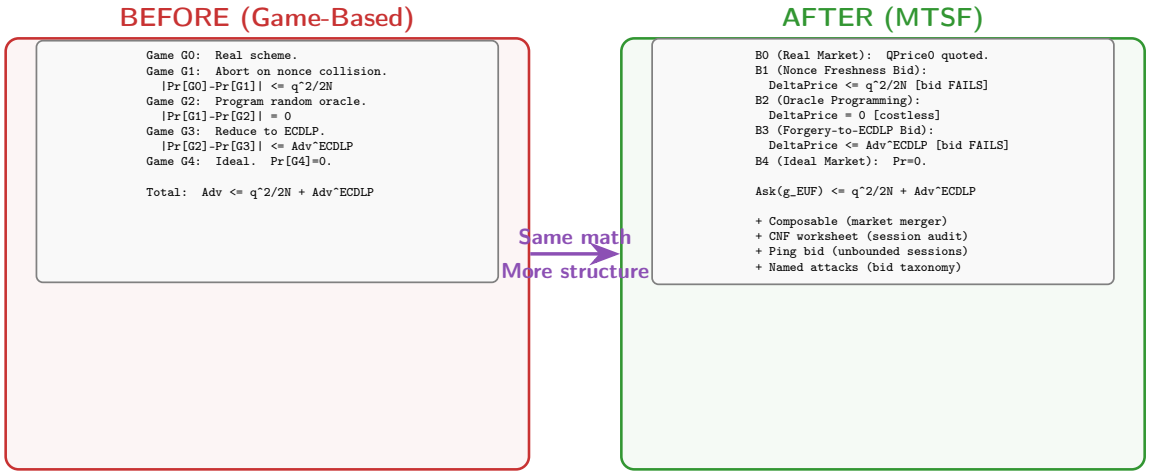


Figure 14. Before/After: the same ECDSA proof written in classical game-based style (left) vs. MTSF style (right). The mathematical content is identical; MTSF adds named bids, composition, session auditing, and unbounded coverage.

3.4. UC as Market Regulation; GUC as Shared Infrastructure

The UC environment $\mathcal{Z}$ is recast as a *market regulator*: an entity monitoring all concurrent trades (protocol messages) and capable of intervening at any point. UC-security: the real market (protocol execution) is computationally indistinguishable from the ideal market (ideal functionality) from any regulator's perspective—$\text{Real}_{\pi,\mathcal{A},\mathcal{Z}} \approx_c \text{Ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. The simulator $\mathcal{S}$ is the *market arbitrageur*: it bridges the real and ideal worlds by simulating the real protocol's market behaviour using only the ideal functionality's interface.

The *UC composition theorem* is a *market merger theorem*: two equilibrium markets, when merged for concurrent execution, remain in equilibrium. GUC shared functionalities (PKI, common reference

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

strings) are *market infrastructure*: public goods accessible to all buyers and sellers. In MTSF, the CNF session formula plays the role of a GUC correctness requirement: each session passes the audit iff all clauses are satisfied, and a failed clause identifies which market good has been compromised.

✎ Analogy:

UC as a Government Regulatory Agency Think of UC (Universal Composability) as a government food safety agency. The **environment** $\mathcal{Z}$ is the *head inspector* who can walk into any restaurant (protocol) at any time, order any dish, and test it. The **simulator** $\mathcal{S}$ is a *stunt double*—a stand-in who must perfectly mimic the real chef's cooking using only pre-approved ingredients (the ideal functionality). If the head inspector cannot tell the difference between the real restaurant and the stunt double's restaurant, then the food (protocol) is safe.

The composition theorem says: if Restaurant A passes inspection and Restaurant B passes inspection, then a food court containing both restaurants *also passes inspection*—even if customers eat at both simultaneously. This is the "market merger theorem" in MTSF: two equilibrium markets stay in equilibrium when merged.

GUC shared infrastructure is like the electricity grid, water supply, and health codes that *all* restaurants share. In MTSF, this is the PKI (public key infrastructure), common reference strings, and other shared resources. The CNF session formula is the health code checklist that every restaurant must pass.

★ Intuition:

Why Composition Is So Important Without composition guarantees, you might prove that your door lock is secure and your alarm system is secure, but when you install both together, they might interfere and create a vulnerability that neither had alone. UC-style composition (MTSF's "market merger theorem") ensures this cannot happen: secure components remain secure when combined. This is essential for real-world systems like TLS 1.3 running alongside the Signal protocol, which MTSF analyses in Section 16.

How UC/GUC Practitioners Can Apply MTSF—A Step-by-Step Migration Guide

If you already work in the UC/GUC framework, MTSF is *not* a replacement—it is an enrichment. Every UC/GUC concept has a direct MTSF counterpart, and the translation preserves all composition guarantees while adding quantitative bounds, named attacks, and session auditing. Here is the systematic translation:

Step 1: Translate your ideal functionality into market goods.
In UC, you define an ideal functionality $\mathcal{F}$ that captures what the protocol should achieve. In MTSF, you decompose $\mathcal{F}$ into a catalogue of *security goods* $\mathcal{G} = \{g_1, \ldots, g_k\}$:

UC ideal functionality $\mathcal{F}$	MTSF security goods $\mathcal{G}$
$\mathcal{F}_{\mathsf{KE}}$ (key exchange)	g_{auth} (authentication) + g_{sk} (session-key secrecy)
$\mathcal{F}_{\mathsf{SIG}}$ (digital signature)	$g_{\mathsf{EUF\text{-}CMA}}$ (unforgeability)
$\mathcal{F}_{\mathsf{PKE}}$ (public-key encryption)	$g_{\mathsf{IND\text{-}CCA2}}$ (confidentiality)
$\mathcal{F}_{\mathsf{AUTH}}$ (mutual authentication)	g_{mutual} (mutual auth) + g_{CNF} (session correctness)
$\mathcal{F}_{\mathsf{SC}}$ (secure channel)	$g_{\mathsf{auth}} + g_{\mathsf{sk}} + g_{\mathsf{mutual}} + g_{\mathsf{CNF}}$

Each good gets its own ask price—a numerical bound that UC does not provide.

Step 2: Translate your simulator into a bidding-round chain.
In UC, you construct a simulator $\mathcal{S}$ that makes the ideal world look like the real world. This simulator typically handles multiple cases: key generation, oracle queries, corruption, message delivery. In MTSF, each "case" in the simulator corresponds to a *bidding round*:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

UC simulator step	MTSF bidding round
$\mathcal{S}$ handles key generation	$\mathbf{B}_0$: Real market (honest key generation)
$\mathcal{S}$ programs the random oracle	$\mathbf{B}_k$: Oracle programming bid ($\Delta\mathsf{Price} = 0$)
$\mathcal{S}$ simulates signing queries	$\mathbf{B}_k$: Nonce freshness bid ($\Delta\mathsf{Price} \leq q^2/2^{\lambda+1}$)
$\mathcal{S}$ extracts from forgery	$\mathbf{B}_k$: Forgery-to-hardness bid ($\Delta\mathsf{Price} \leq \mathsf{Adv}^{\mathsf{hard}}$)
$\mathcal{S}$ handles corruption	$\mathbf{B}_k$: Corruption bid (session pinging checks)

The key advantage: each bidding round produces an explicit $\Delta\mathsf{Price}_k$ bound. Summing these gives the ask price—something the UC simulator alone cannot provide.

Step 3: Translate the UC environment into a market regulator.
The UC environment $\mathcal{Z}$ can:

- Choose inputs for all parties → The regulator sets market conditions.
- Read outputs from all parties → The regulator observes market outcomes.
- Interact with the adversary → The regulator monitors buyer behaviour.
- Output a distinguishing bit → The regulator issues a verdict (equilibrium vs. collapse).

The UC condition Real $\approx_c$ Ideal becomes: the market regulator cannot distinguish the real market from the ideal market. This is *identical* in power to UC—no loss of generality.

Step 4: Apply the market merger theorem instead of the UC composition theorem.
The UC composition theorem says: if π UC-realises $\mathcal{F}$ and π' uses $\mathcal{F}$ as a subroutine, then $\pi' \circ \pi$ UC-realises π''s ideal functionality. In MTSF:

$$\mathsf{Ask}(\mathcal{M}_{\mathsf{composed}}) \;\leq\; \mathsf{Ask}(\mathcal{M}_{\pi}) + \mathsf{Ask}(\mathcal{M}_{\pi'}) + \mathsf{Ask}(\mathcal{M}_{\mathsf{infra}})$$

where $\mathcal{M}_{\mathsf{infra}}$ accounts for shared infrastructure (PKI, CRS). This is the UC composition theorem *plus* a concrete numerical bound on the composed system's security. You get everything UC gives, and more.

Step 5: Translate GUC's CNF session correctness into MTSF's CNF audit.
GUC uses CNF formulas to encode session correctness [5]. MTSF adopts this directly:

- GUC's session formula $\varphi_{\mathsf{sid}} = \bigwedge_j C_j$ is exactly MTSF's session-CNF φ_i (Theorem 13).
- GUC's satisfiability check is MTSF's five-phase verification algorithm (Algorithm 1).
- The difference: MTSF adds a *manual worksheet* for each case study and a *quantitative bound* on the probability that an adversary can satisfy the CNF under a dishonest trace.

Step 6: Add session pinging for unbounded verification.
UC/GUC proofs implicitly handle unbounded sessions (the environment can create arbitrarily many sessions). MTSF makes this explicit via session pinging (Section 5), providing:

- A formal structural distinctness check between consecutive sessions (Theorem 14).
- An induction theorem (Theorem 3) that lifts bounded security to unbounded security.
- Quantitative advantage accumulation bounds ($n \cdot \epsilon$ for n independent sessions).

This is the bridge between UC's implicit unbounded coverage and game-based proofs' explicit numerical bounds.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

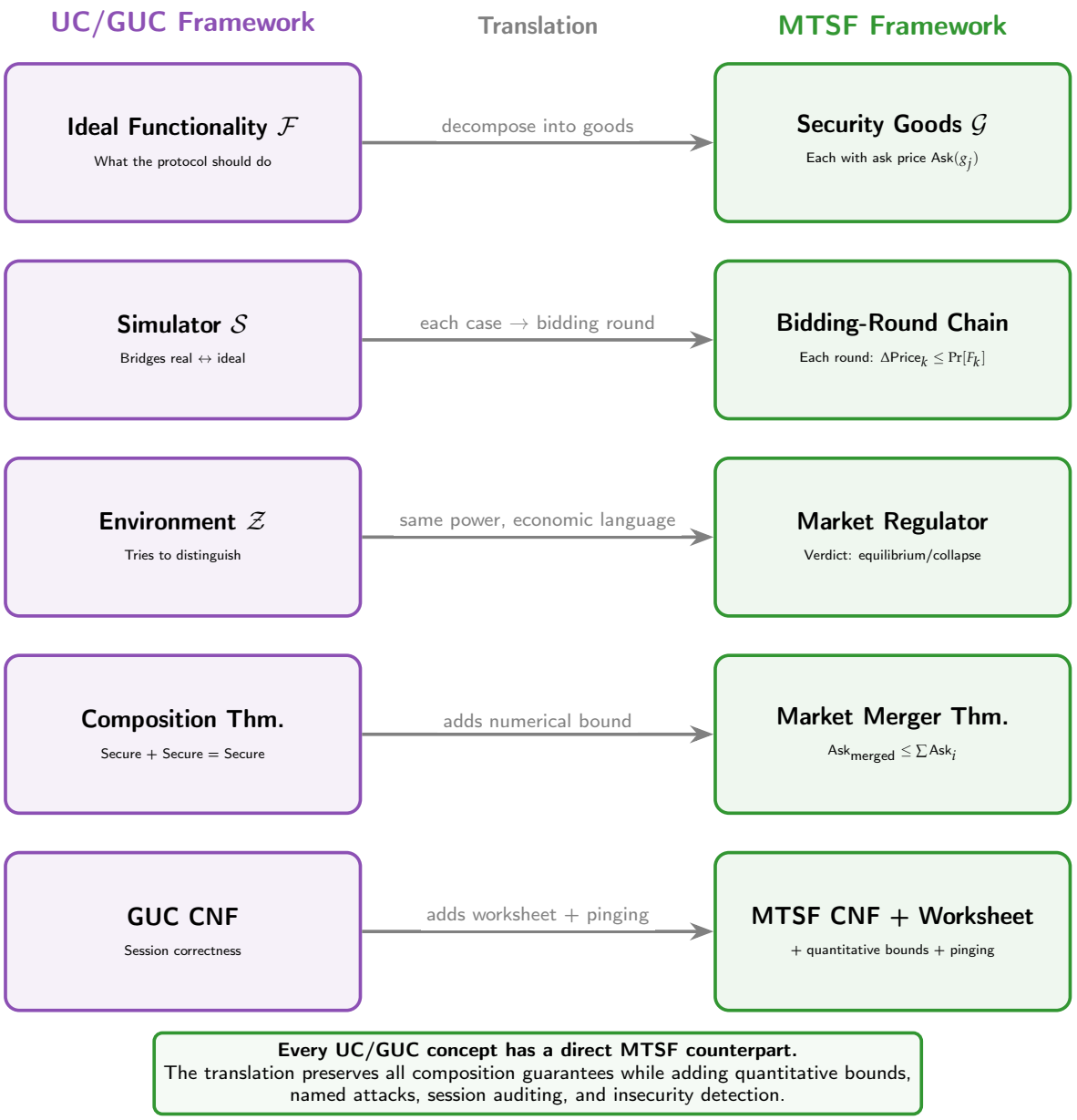


Figure 15. Systematic translation from UC/GUC to MTSF. Every UC concept (left) maps to an MTSF counterpart (right) that preserves all guarantees and adds quantitative bounds. The translation is lossless—MTSF strictly extends UC/GUC.

3.5. Formal Verification as Market Stress Testing

Model checking (ProVerif [6], Tamarin [7]) is recast as *regulatory stress testing*: the regulator enumerates all possible buyer strategies (adversarial protocol traces τ) and checks $\forall\, \tau \in \text{Traces}(\mathcal{M}) : \tau \models \varphi_{g_j}$ for all goods g_j. In MTSF notation: **replay attack** = stale bid (old session transcript at discounted price; blocked by SID clause); **masquerade attack** = identity fraud (claiming to be a different party; blocked by signature/certificate clause); **MITM attack** = market manipulation (intercepting and altering both sides; blocked by SID-binding and identity-binding clauses). Formal verification provides exhaustive coverage for all traces; MTSF's session pinging (Section 5) provides quantitative advantage bounds for all sessions—together bridging the qualitative-unbounded and quantitative-bounded paradigms.

✻ Simple Terms: Side-Channel Attacks—A Complete Guide for Non-Experts

What are side-channel attacks? Every computer leaks information through physical channels: it consumes power, emits electromagnetic radiation, makes noise, takes time, and heats up. A

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

side-channel attack exploits these unintentional leaks to extract secret keys *without* breaking the mathematical security of the cryptographic algorithm. Think of it as a burglar who does not pick the lock but instead listens to the clicks of the tumblers.

Why do they matter for MTSF? A scheme can be mathematically secure (market in equilibrium) but completely broken at the implementation level. MTSF's side-channel bid ($\mathbf{B}_{\mathsf{sca}}$) captures this: even if all mathematical bids fail (nonce, hash, forgery), a physical leakage bid can still collapse the market. The comprehensive countermeasure suite (masking, constant-time code, fault detection, key erasure) is the price the seller must pay to keep the implementation-level market in equilibrium.

The nine side-channel attack classes in plain language:

1. **Simple Power Analysis (SPA):** Watch the power meter while the computer does crypto. Different operations (multiply vs. square) consume different power—one trace reveals the secret key bit by bit. *Like:* Watching someone type a password by looking at which keys light up on a backlit keyboard.
2. **Differential Power Analysis (DPA):** Collect many power traces with different inputs. Statistically correlate the traces with guesses about the key. The correct guess shows a strong correlation; wrong guesses show noise. *Like:* A detective who interviews 1000 witnesses—individually each remembers little, but the consensus reveals the truth.
3. **Timing Analysis:** Measure how long the operation takes. If the time depends on the secret (e.g., early exit on a wrong password character), the timing reveals the secret. *Like:* A safecracker who listens for the click when each tumbler aligns—the time between clicks reveals the combination.
4. **Cache-Timing Attacks:** Exploit the CPU's memory cache. When the crypto code accesses a lookup table, the table entries are loaded into the cache. The attacker measures which entries are cached (fast to access) vs. not cached (slow)—revealing which entries were used, which reveals the key. *Like:* Checking which books in a library are warm (recently handled) to deduce what someone was reading.
5. **Electromagnetic (EM) Analysis:** Hold a radio antenna near the chip and record the EM radiation. Different computations produce different radiation patterns. More spatially precise than power analysis—can target individual circuit components. *Like:* Using a stethoscope on a vault door to listen to specific tumblers.
6. **Fault Injection:** Deliberately cause the computer to make a mistake (by zapping it with a laser, glitching the voltage, or disrupting the clock). A single faulty computation can leak the entire key via Differential Fault Analysis. *Like:* Tripping someone carrying a tray of sorted cards—the way the cards scatter reveals how they were sorted.
7. **Cold Boot Attack:** After the computer is turned off, the memory (DRAM) retains its contents for seconds to minutes. Cooling the memory chips extends retention dramatically. The attacker freezes the RAM, removes it, and reads the secret keys directly. *Like:* A chalkboard that is not fully erased—you can still read what was written.
8. **Acoustic Cryptanalysis:** Record the sound the computer makes during crypto. The CPU, capacitors, and power supply emit sounds at frequencies correlated with computation. RSA keys have been recovered from the sound of a laptop across a room. *Like:* A piano tuner who can identify which keys are being pressed from the sound alone.
9. **Photonic Emission:** Transistors emit faint photons when they switch. A sensitive camera on the chip's backside can capture these photons and reconstruct the computation. *Like:* Reading a message by the fireflies' blinking pattern—each blink corresponds to a bit flip.

The MTSF countermeasure suite: The seller achieves equilibrium against all nine attacks by deploying:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- Order-d masking (defeats SPA, DPA, EMA, acoustic, photonic).
- Constant-time code (defeats timing and cache attacks).
- Bitsliced or hardware-accelerated operations (defeats cache attacks).
- Redundant computation with comparison (defeats fault injection).
- Immediate key zeroing + encrypted RAM (defeats cold boot).
- Physical shielding (defeats EM and photonic).

With all countermeasures, the total side-channel advantage is negligible: $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$.

☞ Key Insight:

Why MTSF Surpasses Formal Verification Tools—Four Concrete Advantages ProVerif [6], Tamarin [7], and CryptoVerif [8] are indispensable tools. MTSF does *not* replace them—it complements them by providing capabilities they cannot:

1. **Numerical bounds.** ProVerif and Tamarin output "secure" or "attack found." They cannot say "the adversary's advantage is at most 2^{-128}." MTSF's ask prices provide exactly this: $\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF}} + q_N^2/2^{\lambda+1}$. These bounds guide parameter selection and enable meaningful comparison between schemes.
2. **Computational assumptions.** ProVerif treats cryptographic operations as perfect black boxes (the Dolev–Yao model). It cannot model birthday bounds, lattice hardness, or side-channel leakage. MTSF's bidding rounds explicitly model computational assumptions: each price adjustment is bounded by a specific hardness assumption ($\mathsf{Adv}^{\mathsf{ECDLP}}$, $\mathsf{Adv}^{\mathsf{MLWE}}$). This captures attacks that ProVerif misses—e.g., the birthday-bound nonce collision in ECDSA.
3. **Composability.** Neither ProVerif nor Tamarin provides composition guarantees. Proving Protocol A secure and Protocol B secure separately does not ensure their concurrent execution is secure. MTSF's market merger theorem provides this guarantee with explicit numerical bounds. The TLS+Signal case study (Section 16) demonstrates composability that no formal verification tool can currently achieve.
4. **Human-readable arguments.** ProVerif's output is a machine-generated attack trace or a "proof found" message. CryptoVerif's proofs are machine-checked sequences of game transformations. Neither produces a human-readable argument that a security engineer can understand, critique, or extend. MTSF's bidding-round proofs are written in natural mathematical language, with named attacks, intuitive analogies, and explicit reasoning at each step.

The ideal workflow: Use ProVerif/Tamarin *first* to exhaustively check the protocol design (catching structural flaws like Needham–Schroeder's missing identity binding). Then use MTSF to produce quantitative bounds, composition guarantees, and human-readable proofs. The two approaches are complementary: formal tools catch bugs automatically; MTSF provides the mathematical foundation for deployment decisions.

4. CNF Session Verification in MTSF

4.1. Why CNF?

Conjunctive Normal Form is natural for encoding session correctness because: (1) each clause corresponds to one security check—adding a check means adding a clause; (2) the conjunction semantics ensures *all* checks must pass; (3) given an execution trace, checking satisfiability is linear in the formula size; (4) an unsatisfied clause directly identifies which security property was violated.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✻ Simple Terms: Why Conjunctive Normal Form (CNF) for Session Verification?

What is CNF? A logical formula in Conjunctive Normal Form is a list of conditions all joined by AND (∧). For example:

$$(\text{SID is correct}) \wedge (\text{nonces are fresh}) \wedge (\text{signatures verify}) \wedge (\text{MACs verify})$$

$$\wedge (\text{message order is right})$$

The entire formula is TRUE only if every single condition is TRUE. One FALSE clause makes the whole formula FALSE.

Why this is perfect for session auditing:

- **Adding a check is adding a clause:** Want to also verify that timestamps are fresh? Just add a new AND-clause. The structure is modular.
- **All-or-nothing semantics:** A session passes the audit *only if every check passes*. This mirrors real security requirements: one failed check is enough to reject a session.
- **Linear-time checking:** Given a session transcript, you check each clause independently. This is fast (linear in the number of checks).
- **Pinpoints failures:** If a session fails, the first FALSE clause tells you exactly which check failed—was it a bad signature? A reused nonce? A wrong SID? This makes debugging trivial.

The key theorem (informal): Honest sessions always pass the CNF audit. Dishonest sessions (produced by an attacker) almost certainly fail the audit—because making a dishonest trace pass would require forging a signature or MAC, which the attacker cannot do efficiently.

✎ Analogy:

CNF Verification as Airport Security Screening Passing through airport security is exactly like CNF verification:

1. **Boarding pass check** (φ^{sid}): Does your boarding pass match your flight? (SID binding—is this session's identifier correct?)
2. **ID verification** (φ^{fresh}): Is your passport valid and not expired? (Freshness—are the nonces new and unique?)
3. **X-ray scan** (φ^{sig}): Does your luggage pass the scanner? (Signature binding—do all digital signatures verify?)
4. **Metal detector** (φ^{mac}): Do you pass the metal detector? (MAC binding—do all message authentication codes check out?)
5. **Gate check** ($\varphi^{\mathsf{consist}}$): Are you at the right gate, at the right time, in the right seat? (Consistency—does the protocol state machine match?)

You board the plane *only if ALL FIVE checks pass*. One failure at any checkpoint, and you are rejected. This is the AND (∧) of CNF: $\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \wedge \varphi_5$. Moreover, if you fail at the X-ray scan, security knows *exactly* what went wrong ("prohibited item detected")—the failed clause pinpoints the problem.

An honest traveller (honest protocol execution) always passes all five checks. A **smuggler** (adversary) might have a fake boarding pass, but the ID check catches them. They might have a valid ID, but the X-ray catches the contraband. Making a dishonest trace pass the CNF audit requires forging *every* credential simultaneously—which is computationally infeasible.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

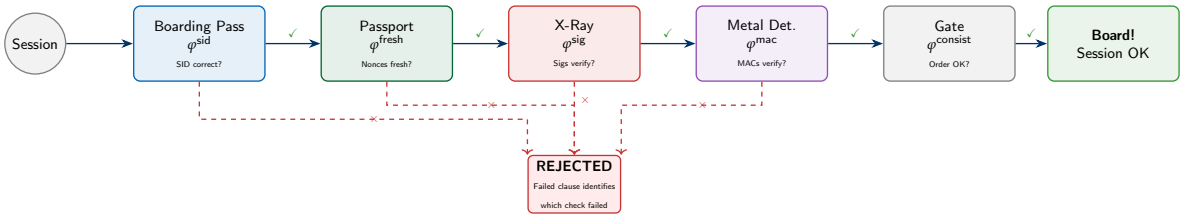


Figure 16. CNF session verification as airport security: five sequential checkpoints (SID, freshness, signatures, MACs, consistency) each modelled as a CNF clause. A session "boards the plane" (is accepted) only if ALL checks pass. Any single failure rejects the session and identifies which check failed.

4.2. Session-CNF Construction

Definition 13 (MTSF Session-CNF). *Let* $\text{Session}_i = (\text{sid}_i, \text{pid}_i^1, \ldots, \text{pid}_i^p, \text{trans}_i)$ *denote the i-th session with p parties and transcript* trans_i. *The* MTSF session-CNF *is:*

$$\varphi_i = \underbrace{\varphi_i^{\text{sid}}}_{\textit{SID binding}} \wedge \underbrace{\varphi_i^{\text{fresh}}}_{\textit{freshness}} \wedge \underbrace{\varphi_i^{\text{sig}}}_{\textit{signature binding}} \wedge \underbrace{\varphi_i^{\text{mac}}}_{\textit{MAC binding}} \wedge \underbrace{\varphi_i^{\text{consist}}}_{\textit{consistency}}. \tag{7}$$

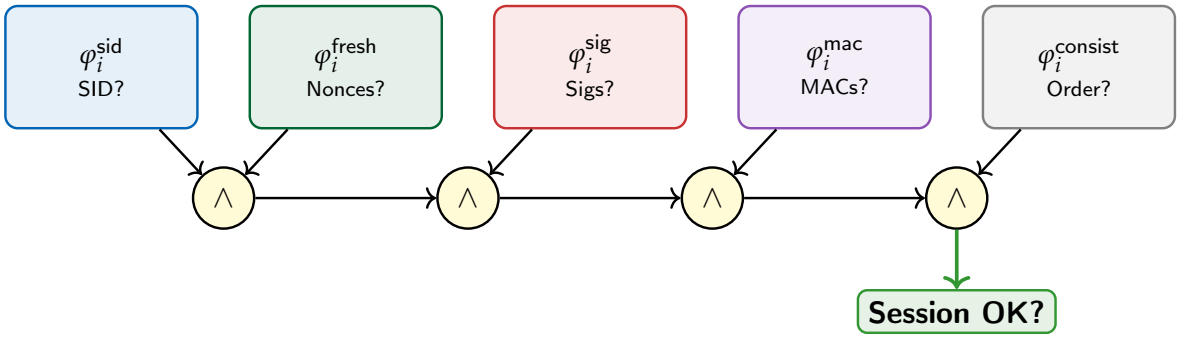


Figure 17. Structure of the MTSF session-CNF φ_i: five components ANDed together. A session is correct iff the entire formula is satisfiable under the honest trace.

SID Clauses.

Every message must carry the correct session identifier:

$$\varphi_i^{\text{sid}} = \bigwedge_{j=1}^{\ell} \left(x_{\text{sid}_i,j} \vee \neg x_{\text{other},j}\right), \tag{8}$$

where $x_{\text{sid}_i,j}$ is true if message j carries sid_i, and $x_{\text{other},j}$ is true if it carries a different SID.

Freshness Clauses.

All nonces must be fresh and distinct:

$$\varphi_i^{\text{fresh}} = \bigwedge_{k=1}^{r} \left(x_{n_k,\text{fresh}}\right) \wedge \bigwedge_{k \neq k'} \left(\neg x_{n_k = n_{k'}}\right). \tag{9}$$

Signature-Binding Clauses.

Every signature must verify or the session aborts:

$$\varphi_i^{\text{sig}} = \bigwedge_{j} \left(x_{\text{Vrfy}(\text{pk}_j, m_j \| \text{sid}_i, \sigma_j)=1} \vee x_{\text{abort}}\right). \tag{10}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

MAC-Binding Clauses.

Every MAC must verify or the session aborts:

$$\varphi_i^{\mathsf{mac}} \;=\; \bigwedge_j \big(x_{\mathsf{Mac.Vrfy}(K_j, m_j \| \mathsf{sid}_i, \tau_j)=1} \;\vee\; x_{\mathsf{abort}}\big). \tag{11}$$

Consistency Clauses.

Protocol-specific invariants (message ordering, state-machine compliance):

$$\varphi_i^{\mathsf{consist}} \;=\; \bigwedge_{j=1}^{c} C_j^{\mathsf{protocol}}. \tag{12}$$

★ Intuition:

CNF as a Pre-Flight Checklist Each clause is one check on the aircraft checklist: "Fuel above minimum?" "Flaps correct?" "Tower clearance received?" The flight is safe iff *every* check passes (the formula is satisfiable). A single failed clause grounds the flight.

Theorem 2 (Session Correctness via CNF)**.** $\mathsf{Session}_i$ *is correct iff* φ_i *is satisfiable under the honest trace* trans_i.

Proof. *Soundness.* If φ_i is satisfiable: SID clauses $\Rightarrow$ correct binding; freshness $\Rightarrow$ unique nonces; sig/MAC $\Rightarrow$ all verifications pass; consistency $\Rightarrow$ invariants hold. This is session correctness.

Completeness. If the session is correct, the honest trace provides a satisfying assignment. □

Proposition 1 (Adversarial CNF Unsatisfiability)**.** *If all underlying primitives are secure (markets in equilibrium), then:*

$$\Pr[\mathcal{A} \textit{ produces a satisfying assignment for } \varphi_i \textit{ under a dishonest trace}] \leq \mathsf{negl}(\lambda). \tag{13}$$

Proof. A satisfying dishonest assignment requires at least one of: (1) signature forgery (breaks EUF-CMA equilibrium); (2) MAC forgery (breaks SUF-CMA); (3) nonce prediction (breaks freshness); (4) SID reuse (breaks binding). Each contradicts the equilibrium assumption. □

4.3. CNF Verification Algorithm

The session-CNF φ_{sess} is a conjunction of atomic Boolean clauses. Verifying it manually is hard because (i) the number of clauses can be large, (ii) clause ordering matters for cascading failures, and (iii) cryptographic verification steps are interleaved with structural ones. Algorithm 1 provides the canonical machine-checkable procedure. The *MTSF CNF Truth-Table Worksheet* below gives a four-step manual recipe.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Algorithm 1 MTSF-CNF Session Verification (Canonical, 5-Phase)

Require: Transcript trans_i, session ID sid_i, public keys $\{\text{pk}_j\}$, shared/MAC keys $\{K_j\}$
Ensure: Accept or Reject

Phase 1 – SID Binding Check
for each message $m_j \in \text{trans}_i$ **do**
  if $\text{ExtractSID}(m_j) \neq \text{sid}_i$ **then return** Reject ▷ Clause φ^{sid} fails
  end if
end for
Phase 2 – Nonce Freshness & Disjointness
$\mathcal{N} \leftarrow \emptyset$
for each nonce $n_k \in \text{trans}_i$ **do**
  if $n_k \in \mathcal{N}$ **then return** Reject ▷ Clause φ^{fresh} fails
  end if
  $\mathcal{N} \leftarrow \mathcal{N} \cup \{n_k\}$
end for
Phase 3 – Signature & MAC Verification
for each $(\sigma_j, \text{pk}_j, \text{data}_j) \in \text{trans}_i$ **do**
  if $\text{Vrfy}(\text{pk}_j, \text{data}_j, \sigma_j) \neq 1$ **then return** Reject ▷ Clause φ_j^{sig} fails
  end if
end for
for each $(\tau_j, K_j, \text{macData}_j) \in \text{trans}_i$ **do**
  if $\text{Mac.Vrfy}(K_j, \text{macData}_j, \tau_j) \neq 1$ **then return** Reject ▷ Clause φ_j^{mac} fails
  end if
end for
Phase 4 – Message Ordering & Consistency
if $\text{CheckOrdering}(\text{trans}_i) \neq 1$ **then return** Reject ▷ Clause φ^{consist} fails
end if
Phase 5 – Ping (Unbounded Freshness)
if $i > 1$ **and** $\text{Ping}(\text{Session}_{i-1}, \text{Session}_i) \neq 1$ **then return** Reject ▷ Unbounded bid fails
end if
return Accept

✓ CNF Verification: Easy Manual CNF Verification—The Four-Column Truth-Table Method

Why manual CNF verification is hard. A CNF formula $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ is satisfiable iff *all* clauses evaluate to **T**; a single **F** makes it unsatisfiable. Protocol session-CNFs have 6–12 clauses. Checking them unsystematically leads to missed failures.

The MTSF Four-Column Worksheet (one row per clause):

Clause	What to check	How (manual)	T/F	If F: attack
φ^{sid}	Same SID in every message?	Read each field; compare.		Cross-session replay
φ^{fresh}	All nonces distinct?	List nonces; look for repeats.		Full-session replay
φ_A^{sig}	$\text{Vrfy}(\text{pk}_A, \text{data}, \sigma_A) = 1$?	Hash scope; check signature.		Impersonation of A
φ_B^{sig}	$\text{Vrfy}(\text{pk}_B, \text{data}, \sigma_B) = 1$?	Same procedure.		Impersonation of B
φ^{mac}	$\text{Mac.Vrfy}(K, \text{data}, \tau) = 1$?	Recompute MAC; compare.		Ciphertext injection
φ^{consist}	Correct message order?	Check sender and sequence.		Out-of-order inject
φ^{ping}	SID/nonces fresh vs. Session_{i-1}?	Compare with previous session.		Unbounded-round attack
Result:	All T ⇒ **Accept**. Any F ⇒ **Reject** (cite row).			

Step-by-step manual recipe:

1. **Extract fields.** Write down every field: SID, nonces, identities, ciphertexts, signatures, MACs.
2. **Instantiate clauses.** Substitute actual values into each formula clause.
3. **Evaluate each clause.** Signatures: hash the signed scope, verify against public key. MACs: recompute and compare. SID/nonces: string equality.
4. **Mark T or F.** One **F** anywhere ⇒ **Reject**; record which clause and attack type.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Key insight: Check in order Phase 1→5 for early exit. SID failure means no point verifying signatures.

Satisfiability: Honest trace ⇒ all T ⇒ **SAT** ⇒ **Accept**. Dishonest trace ⇒ some F ⇒ **UNSAT** ⇒ **Reject**. NS protocol exception: CNF is SAT under dishonest trace (design failure—identity clause missing).

5. Unbounded Verification via Session Pinging

Session pinging is one of the two core technical contributions of MTSF (the other being CNF session verification, Section 4). This section develops the full theory: motivation, formal mechanism, the main induction theorem with a detailed proof, a quantitative advantage-accumulation analysis, the interaction between pinging and CNF checking, failure-mode taxonomy, and a comparison with symbolic unbounded verification tools.

5.1. Why Bounded Is Not Enough

A protocol correct for sessions $1, \ldots, 1000$ may fail at session 1001 if the randomness generator cycles. Game-based proofs are inherently bounded (polynomial in λ). Formal verification handles unbounded sessions symbolically but without concrete bounds. Session pinging bridges this gap.

✻ Simple Terms: Why Unbounded Session Security Matters

The problem with bounded proofs: A standard game-based security proof says the scheme is secure for up to q sessions (where q is polynomial in λ). But real systems run for years, accumulating millions of sessions. What if a subtle flaw only appears after session 1,000,001?
Concrete example: Suppose a random nonce generator uses a 32-bit counter. After 2^{32} sessions, it wraps around and reuses the same nonces. A protocol proven secure for any polynomial number of sessions might still be vulnerable if nonces repeat. The session pinging mechanism checks that each new session is genuinely *fresh and distinct* from all previous ones—blocking this exact failure mode.
What session pinging achieves: By verifying that consecutive sessions are "structurally distinct" (different SIDs, fresh nonces, correctly bound signatures), and showing by mathematical induction that the security argument carries forward indefinitely, we prove security for an *unbounded* number of sessions. This is the strongest possible guarantee.
Formal verification tools like ProVerif already handle unbounded sessions (they use symbolic abstraction), but without concrete bounds. Session pinging gives MTSF both: unbounded coverage *and* concrete numerical advantage bounds.

Bounded-session limitations in existing frameworks.

Consider a protocol π proven secure for q concurrent sessions in the game-based model. The standard advantage bound takes the form $\mathsf{Ask} \leq q \cdot \epsilon_{\mathsf{prim}} + q^2/2^{\lambda+1}$, where ϵ_{prim} is the advantage against the underlying primitive and the quadratic term captures birthday-bound collisions among q nonces. This bound is *meaningful* only for $q = \mathsf{poly}(\lambda)$. Three real-world failure scenarios escape this analysis:

1. **Nonce/counter wrap-around.** A 32-bit counter nonce wraps after 2^{32} sessions, reusing the same nonce-key pair. Protocols such as AES-GCM with a 32-bit invocation field are vulnerable after 2^{32} TLS records under the same key.
2. **Session-state accumulation.** Adversaries that persist across sessions can accumulate partial information. For instance, each ECDSA signing session leaks a negligible amount of side-channel information about the nonce k; after 2^{60} sessions the accumulated leakage may be non-negligible.
3. **Cross-session correlation.** In multi-party protocols, an adversary may correlate session transcripts across sessions. Without a mechanism ensuring structural independence, subtle cross-

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

session attacks (e.g. the Needham–Schroeder masquerade) succeed regardless of the session count.

Session pinging addresses all three by enforcing structural independence at each session boundary and enabling an inductive security argument that is *session-index-independent*.

5.2. The Pinging Mechanism

Definition 14 (Session Structural Distinctness). *Sessions* $\mathsf{Session}_i$ *and* $\mathsf{Session}_{i+1}$ *are* structurally distinct *if:*

(a) $\mathsf{sid}_i \neq \mathsf{sid}_{i+1}$ *(SID freshness);*
(b) $\mathcal{N}_i \cap \mathcal{N}_{i+1} = \emptyset$ *(disjoint nonce sets);*
(c) all signatures in $\mathsf{Session}_{i+1}$ *sign data including* sid_{i+1} *(signature binding);*
(d) all MACs in $\mathsf{Session}_{i+1}$ *authenticate data including* sid_{i+1} *(MAC binding);*
(e) all ciphertexts in $\mathsf{Session}_{i+1}$ *are freshly generated (not replayed from* $\mathsf{Session}_i$*)* *(ciphertext freshness).*

Remark 1 (Condition (e): Ciphertext Freshness). *Condition (e) is new compared to the minimal four-condition definition. It is necessary for KEM-based protocols (ML-KEM, QKEM) where the buyer could replay a ciphertext* c_i *from* $\mathsf{Session}_i$ *to* $\mathsf{Session}_{i+1}$*. Without ciphertext freshness, the decapsulation oracle in* $\mathsf{Session}_{i+1}$ *might return the same key* K_i*, breaking key secrecy. The CNF clause* φ^{novel} *enforces this condition.*

Definition 15 (Session Ping). $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 1$ *iff* $\mathsf{Session}_i$ *and* $\mathsf{Session}_{i+1}$ *are structurally distinct per Theorem 14.*

Definition 16 (Ping Failure Probability). *The* ping failure probability *for protocol* π *is:*

$$\delta_{\mathsf{ping}}(\pi) = \max_{i \geq 1} \Pr[\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0], \tag{14}$$

where the probability is taken over honest randomness (nonce generation, SID sampling). A protocol has negligible ping failure *if* $\delta_{\mathsf{ping}}(\pi) \leq \mathsf{negl}(\lambda)$.

★ Intuition:

Ping as Heartbeat Monitor Think of each session as a heartbeat in a medical monitor. A ping is the check that the current heartbeat is genuinely *new*—not an echo or replay of a previous heartbeat. A flatline (ping failure) means the system is replaying old data. The induction theorem says: if every heartbeat is genuine, the patient (protocol) remains healthy forever.

✎ Analogy:

Session Pinging as a Chain of Passport Stamps Imagine travelling through a series of countries. At each border crossing, the immigration officer:

1. Checks that your passport number has not changed (sid freshness).
2. Stamps your passport with a *new, unique* date and entry number (disjoint nonce sets).
3. Verifies that the visa for this specific country is attached to your passport (signature binding).
4. Checks that the customs declaration form matches this specific border (MAC binding).
5. Ensures you are entering from the correct previous country (ciphertext freshness).

The induction: If your first border crossing was legitimate (base case), and every subsequent crossing passes all five checks (inductive step), then *every* country you visited was entered legitimately—no matter how many countries you visit. This is unbounded session security: the proof works for 10 countries, 10,000 countries, or infinitely many.

Why this matters: A web server running TLS might handle millions of sessions per day for years. Session pinging ensures that session number 1,000,000,001 is just as secure as session number 1—without needing to re-prove security from scratch.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✻ Simple Terms: Mathematical Induction for Non-Mathematicians

What is mathematical induction? It is a proof technique that works like a row of dominoes:

1. **Base case:** Push the first domino (prove the first session is secure).
2. **Inductive step:** Show that if domino i falls, then domino $i+1$ must also fall (prove that if session i is secure and the ping passes, then session $i+1$ is also secure).
3. **Conclusion:** All dominoes fall—every session is secure, no matter how many there are.

Why the ping is essential: Without the ping check, the dominoes might not be close enough to knock each other over. The ping ensures that consecutive sessions are "structurally close enough" (structurally distinct but reduction-equivalent) for the induction to work. A failed ping means a gap in the domino chain—the security guarantee breaks at that point.

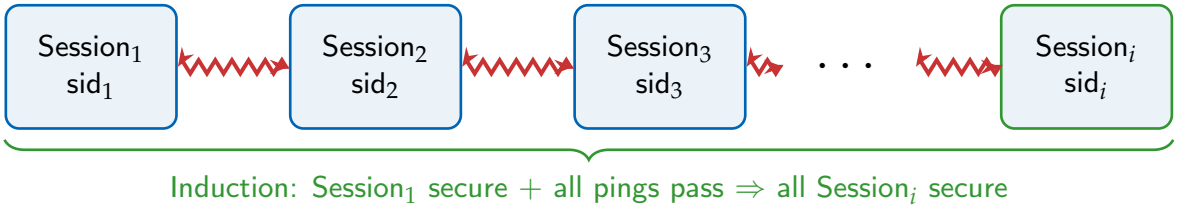


Figure 18. Session pinging for unbounded verification. Consecutive sessions are "pinged" to verify structural distinctness. The zigzag edges denote the ping check between adjacent sessions.

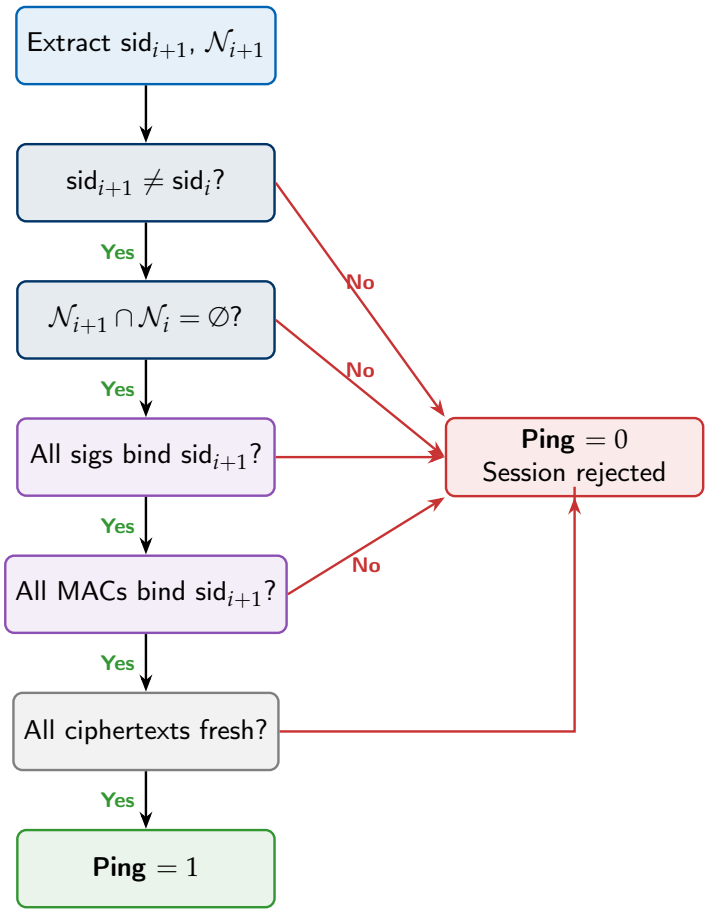


Figure 19. Detailed ping verification flow for session Session$_{i+1}$ against its predecessor Session$_i$. A failure at any step short-circuits to Ping $= 0$ (session rejected). This flowchart corresponds to Phase 5 of Algorithm 1.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

5.3. The Unbounded Security Theorem

Theorem 3 (Unbounded Security via Session Pinging)**.** *If (i)* $\mathsf{Session}_1$ *is secure (market in equilibrium with* $\mathsf{Ask}_1 \le \epsilon$*), (ii)* $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 1$ *for all* $i \ge 1$*, and (iii)* φ_{i+1} *is obtained from* φ_i *by variable renaming (substituting* $\mathsf{sid}_i \mapsto \mathsf{sid}_{i+1}$ *and fresh nonces), then* π *is secure for unbounded sessions, with:*

$$\mathsf{Ask}_i \;\le\; \epsilon + (i-1)\cdot \delta_{\mathsf{ping}}(\pi) \quad \textit{for all } i \ge 1. \tag{15}$$

When $\delta_{\mathsf{ping}}(\pi) \le \mathsf{negl}(\lambda)$*, the ask price remains negligible for all* $i = \mathsf{poly}(\lambda)$*.*

Proof. We proceed by strong induction on the session index i.

Base case ($i = 1$). Session $\mathsf{Session}_1$ is secure by hypothesis (i), with $\mathsf{Ask}_1 \le \epsilon$. The session-CNF φ_1 is satisfiable under the honest trace of $\mathsf{Session}_1$ by Theorem 2: SID clauses hold (the SID was freshly sampled), freshness clauses hold (nonces are freshly generated), signature and MAC clauses hold (honest parties execute correctly), and consistency clauses hold (honest execution follows the protocol specification). Since there is no predecessor session, the ping clause $\varphi_1^{\mathsf{ping}}$ is vacuously true.

Inductive step. Assume that sessions $\mathsf{Session}_1, \ldots, \mathsf{Session}_i$ are all secure, i.e., for each $j \le i$ the session-CNF φ_j is satisfiable under the honest trace and $\mathsf{Ask}_j \le \epsilon + (j-1)\delta_{\mathsf{ping}}$. We must show that $\mathsf{Session}_{i+1}$ is secure.

Step 1: Structural distinctness via pinging. By hypothesis (ii), $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 1$. By Theorem 14, this guarantees:

- sid_{i+1} is fresh (not equal to sid_i or any earlier SID, since all earlier pings also passed by the inductive hypothesis, giving a chain of distinct SIDs);
- all nonces in $\mathsf{Session}_{i+1}$ are disjoint from those in $\mathsf{Session}_i$ (and by induction, disjoint from all earlier nonces with probability $\ge 1 - i \cdot \delta_{\mathsf{ping}}$);
- all signatures and MACs in $\mathsf{Session}_{i+1}$ bind sid_{i+1}, which is distinct from all previous session identifiers.

Step 2: CNF isomorphism. By hypothesis (iii), φ_{i+1} is obtained from φ_i by the variable renaming $\mathsf{sid}_i \mapsto \mathsf{sid}_{i+1}$ and substituting fresh nonces. Since the CNF clauses are structural (they check SID binding, nonce freshness, signature verification, MAC verification, and consistency), and since all variables in φ_{i+1} are independently fresh, the satisfiability of φ_{i+1} under the honest trace of $\mathsf{Session}_{i+1}$ follows from the satisfiability of φ_i under the honest trace of $\mathsf{Session}_i$. Formally:

$$\varphi_{i+1}[\mathsf{trans}_{i+1}^{\mathsf{honest}}] = \varphi_i[\mathsf{trans}_i^{\mathsf{honest}}]\big|_{\mathsf{sid}_i \mapsto \mathsf{sid}_{i+1},\, \mathcal{N}_i \mapsto \mathcal{N}_{i+1}} = \mathsf{TRUE}. \tag{16}$$

Step 3: Market equilibrium preservation. The security reduction for $\mathsf{Session}_{i+1}$ is identical to that for $\mathsf{Session}_1$ (and hence $\mathsf{Session}_i$): the reduction converts a buyer who breaks $\mathsf{Session}_{i+1}$ into a buyer who breaks the underlying primitive. Crucially, this reduction is *session-index-independent*: it does not use the session index $i+1$ in any way beyond the fresh SID and nonces. Therefore:

$$\mathsf{Ask}_{i+1} \le \mathsf{Ask}_1 + \Pr\big[\exists\, j \le i : \mathsf{Ping}(\mathsf{Session}_j, \mathsf{Session}_{j+1}) = 0\big] \le \epsilon + i \cdot \delta_{\mathsf{ping}}(\pi). \tag{17}$$

The second inequality follows from a union bound over i ping checks, each failing with probability at most δ_{ping}.

Conclusion. By induction, $\mathsf{Ask}_i \le \epsilon + (i-1)\delta_{\mathsf{ping}}$ for all $i \ge 1$. When $\delta_{\mathsf{ping}} \le \mathsf{negl}(\lambda)$ and $i = \mathsf{poly}(\lambda)$, the total ask price is $\epsilon + \mathsf{poly}(\lambda) \cdot \mathsf{negl}(\lambda) = \epsilon + \mathsf{negl}(\lambda)$, which is negligible whenever ϵ is negligible. The market remains in equilibrium for all sessions. □

☞ Key Insight:

Tightness of the Accumulation Bound The linear accumulation $i \cdot \delta_{\mathsf{ping}}$ in Equation (15) is tight in general: if ping failures are independent across sessions, a union bound is the best possible. However, in many protocols the ping failure probability is dominated by the birthday bound for

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

nonce collisions: $\delta_{\mathsf{ping}} \leq q_N^2/2^{\lambda+1}$ where q_N is the number of nonces per session. For $\lambda = 256$ and $q_N \leq 2$, this gives $\delta_{\mathsf{ping}} \leq 2^{-255}$, so even after 2^{64} sessions the accumulated degradation is $2^{64} \cdot 2^{-255} = 2^{-191}$—astronomically negligible.

5.4. Interaction Between Session Pinging and CNF Verification

✎ Analogy:

Pinging + CNF as a Two-Layer Defence Think of session pinging and CNF verification as a two-layer defence system at a secure facility:

Layer 1 — CNF Verification (the ID badge scanner): At the entrance, every visitor must scan their badge (CNF audit). The scanner checks five things: valid badge (SID), not expired (freshness), proper credentials (signatures), authorised access level (MACs), and correct visitor log entry (consistency). If any check fails, entry is denied.

Layer 2 — Session Pinging (the shift-change verification): When a new shift of visitors arrives, the guard compares the new visitor list to the previous shift's list. Are the badge numbers different? Are the credentials freshly issued? Is anyone trying to reuse yesterday's badge? This "shift-change check" is the ping.

How they interact:

- **CNF ensures each individual visit is valid.** A visitor with a forged badge fails the scanner.
- **Pinging ensures consecutive visits are independent.** A visitor who recorded yesterday's entry cannot replay it today.
- **Together, they provide unbounded security.** Every visit (session) is both individually valid (CNF) and structurally distinct from all previous visits (ping). By induction, the facility is secure for all time.

Session pinging and CNF verification are complementary mechanisms that together provide the strongest possible session-security guarantee. Their interaction is formalised as follows.

Proposition 2. *For any session* $\mathsf{Session}_{i+1}$ *with predecessor* $\mathsf{Session}_i$*:*

$$[Ping - CNFSoundness] \quad \begin{array}{l} (\varphi_{i+1} = \mathsf{SAT}) \ \wedge \ (\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 1) \implies \\ \mathsf{Session}_{i+1} \textit{ is correct and structurally independent of } \mathsf{Session}_i. \end{array} \tag{18}$$

Proof. CNF satisfiability (Theorem 2) guarantees that all intra-session checks pass: SID binding, nonce freshness within the session, all signatures and MACs verify, and message ordering is correct. The ping check (Theorem 15) guarantees inter-session independence: the SID, nonces, and cryptographic bindings of $\mathsf{Session}_{i+1}$ are structurally distinct from those of $\mathsf{Session}_i$. Together, they ensure that $\mathsf{Session}_{i+1}$ is both internally correct and externally independent, which is the full session-correctness requirement. □

Proposition 3 (Ping Failure Implies CNF Clause Failure)**.** *If* $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0$*, then at least one of the following CNF clauses in* φ_{i+1} *is violated:*

- $\varphi_{i+1}^{\mathsf{sid}}$*: SID binding (if* $\mathsf{sid}_{i+1} = \mathsf{sid}_i$*);*
- $\varphi_{i+1}^{\mathsf{fresh}}$*: nonce freshness (if* $\mathcal{N}_{i+1} \cap \mathcal{N}_i \neq \emptyset$*);*
- $\varphi_{i+1}^{\mathsf{sig}}$ *or* $\varphi_{i+1}^{\mathsf{mac}}$*: cryptographic binding (if signatures or MACs do not bind* sid_{i+1}*).*

Therefore, the four-column CNF worksheet automatically detects any ping failure, and the failing clause identifies the specific inter-session attack vector.

Proof. By contrapositive: if all CNF clauses pass and include the SID of the current session in all cryptographic bindings, then conditions (a)–(e) of Theorem 14 are satisfied, so the ping passes. □

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Phase 5 of Algorithm 1 explained.

The ping check in Phase 5 of the CNF verification algorithm serves as the "inter-session bridge" between the intra-session CNF verification (Phases 1–4) and the unbounded induction theorem (Theorem 3). Concretely, Phase 5 performs five sub-checks corresponding to the five conditions of Theorem 14:

1. **SID uniqueness:** Verify $\mathsf{sid}_{i+1} \notin \{\mathsf{sid}_1, \ldots, \mathsf{sid}_i\}$ by consulting the SID log.
2. **Nonce disjointness:** Verify $\mathcal{N}_{i+1} \cap \bigcup_{j=1}^{i} \mathcal{N}_j = \emptyset$ by consulting the nonce log.
3. **Signature SID-binding:** For each signature σ in $\mathsf{Session}_{i+1}$, verify that the signed data includes sid_{i+1}.
4. **MAC SID-binding:** For each MAC tag τ in $\mathsf{Session}_{i+1}$, verify that the authenticated data includes sid_{i+1}.
5. **Ciphertext freshness:** For each ciphertext c in $\mathsf{Session}_{i+1}$, verify $c \notin \{c_1, \ldots, c_i\}$.

If any sub-check fails, the session is rejected immediately with a diagnostic indicating which structural-distinctness condition was violated. This early rejection prevents the accumulation of insecure sessions.

5.5. Taxonomy of Ping Failure Modes

Not all protocols satisfy the pinging requirements. The following taxonomy classifies the failure modes that prevent unbounded security, ordered by severity.

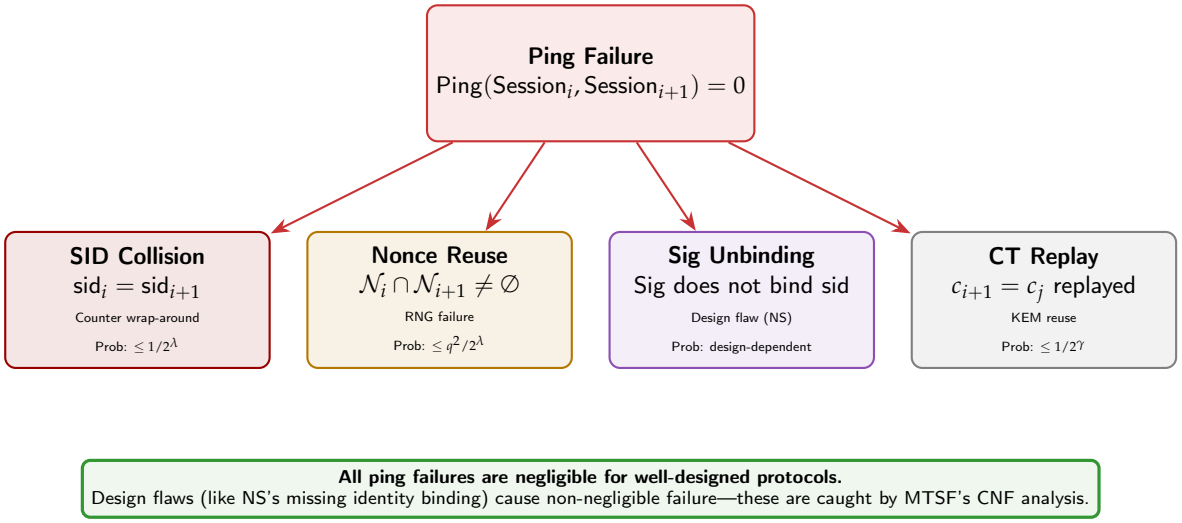


Figure 20. Taxonomy of ping failure modes: SID collision (counter issues), nonce reuse (RNG problems), signature unbinding (design flaws), and ciphertext replay (KEM reuse). Each mode has a specific probability bound. Well-designed protocols have negligible failure in all modes.

Definition 17 (Ping Failure Taxonomy)**.**

1. ***Type I: SID collision.*** $\mathsf{sid}_{i+1} = \mathsf{sid}_j$ *for some* $j \leq i$. *Probability:* $\leq i/|\mathsf{SID\text{-}space}|$ *(birthday bound if SIDs are random).* Consequence: *Cross-session replay becomes possible.* Example: *A 64-bit SID space after* 2^{32} *sessions gives collision probability* $\approx 2^{-1}$.
2. ***Type II: Nonce reuse.*** $\mathcal{N}_{i+1} \cap \mathcal{N}_j \neq \emptyset$ *for some* $j \leq i$. Consequence: *Signature key recovery (ECDSA), ciphertext XOR leakage (stream ciphers), or MAC forgery.* Example: *ECDSA with a biased nonce generator (cf. the Sony PlayStation 3 ECDSA break, where all nonces were identical).*
3. ***Type III: Unsigned SID.*** *Signatures in* $\mathsf{Session}_{i+1}$ *do not bind* sid_{i+1}. Consequence: *Signatures from one session can be replayed in another.* Example: *Needham–Schroeder protocol, where no SID is signed (cf. Section 13.4).*
4. ***Type IV: Unauthenticated SID.*** *MACs in* $\mathsf{Session}_{i+1}$ *do not bind* sid_{i+1}. Consequence: *MAC tags from one session can be injected into another.* Example: *Original Telegram MTProto 2.0, where the salt is not HMAC-bound (cf. Section 13.9).*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

5. ***Type V: Ciphertext replay.*** *Ciphertexts from* $\mathsf{Session}_i$ *are accepted in* $\mathsf{Session}_{i+1}$. Consequence: *Key reuse across sessions.* Example: *A KEM without implicit rejection may accept a replayed ciphertext.*

Table 1. Ping failure taxonomy with protocol examples and CNF clause that detects each failure.

Type	Failure mode	Probability	CNF clause	Example protocol
I	SID collision	$i/\lvert\mathsf{SID}\rvert$	φ^{sid}	64-bit SID after 2^{32} sessions
II	Nonce reuse	$q_N^2/2^{\lambda+1}$	φ^{fresh}	ECDSA with biased nonces
III	Unsigned SID	1	φ^{sig}	Needham–Schroeder
IV	Unauthenticated SID	$\geq 2^{-64}$	φ^{mac}	Telegram MTProto 2.0
V	Ciphertext replay	$q_D/\lvert\mathcal{C}\rvert$	φ^{novel}	KEM without implicit rej.

✎ **Analogy:**

Advantage Accumulation as Water Filling a Bucket Imagine each session is a raindrop, and the attacker's advantage is a bucket. Each session adds a tiny drop of water (a negligible advantage) to the bucket. The question is: after billions of sessions, does the bucket overflow (advantage becomes non-negligible)?

The good news: If the per-session advantage is ϵ and sessions are structurally independent (pings pass), then after n sessions the total advantage is at most $n \cdot \epsilon$. For $\epsilon = 2^{-128}$ and $n = 2^{64}$ sessions (far more than any real system runs), the total is $2^{64} \cdot 2^{-128} = 2^{-64}$—still negligible.

The bad news: If the per-session advantage degrades (e.g., side-channel leakage accumulates), or if sessions are correlated (e.g., shared nonce state), then the bucket can overflow. This is why MTSF requires structural independence via pinging: independent sessions ensure linear (not super-linear) advantage accumulation.

Real-world implication: A TLS server handling 2^{40} sessions per year with AES-256 ($\epsilon = 2^{-128}$ per session) accumulates at most $2^{40} \cdot 2^{-128} = 2^{-88}$ total advantage per year—this would take 2^{88} years to reach a 50% chance of any attack succeeding, far longer than the age of the universe.

5.6. Quantitative Advantage Accumulation

The accumulation bound in Theorem 3 is *additive*: each session adds at most δ_{ping} to the total advantage. We now give concrete numerical examples for the protocols and primitives analysed in this article.

Proposition 4 (Concrete Ping Degradation Bounds)**.** *For the following schemes, the ping failure probability and accumulated advantage after N sessions are:*

Scheme	δ_{ping}	*After* $N = 2^{40}$ *sessions*	*After* $N = 2^{64}$ *sessions*
ECDSA (256-bit q)	$q_S^2/(2q) \approx 2^{-129}$	2^{-89}	2^{-65}
ML-KEM-768	$q_D/2^{\gamma} \approx 2^{-200}$	2^{-160}	2^{-136}
ML-DSA-65	$\mathsf{Adv}^{\mathsf{MSIS}} \approx 2^{-128}$	2^{-88}	2^{-64}
AES-128	$q_E \cdot 2^{-128} \approx 2^{-64}$	2^{-24}	≈ 1 ***(key rotation needed)***
Keccak/SHA-3	$q_f/2^c = q_f/2^{512}$	2^{-472}	2^{-448}
ISO two-party	$q_N^2/2^{257} + \mathsf{Adv}^{\mathsf{EUF}}$	$\approx 2^{-128}$	$\approx 2^{-128}$

Remark 2 (AES Key Rotation)**.** *The AES entry highlights an important practical point: for block ciphers with 128-bit keys, the accumulated ping degradation reaches unity after* $\approx 2^{64}$ *sessions under the same key. This is the well-known birthday bound for block ciphers and mandates key rotation (rekeying) before* 2^{64} *blocks. The ping mechanism makes this requirement* explicit *within the MTSF framework, rather than leaving it as an implicit assumption.*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

5.7. Comparison with Symbolic Unbounded Verification

✻ Simple Terms: MTSF Pinging vs. ProVerif: Complementary Strengths

What ProVerif does well:

- Exhaustively checks all possible attack traces (no attack is missed).
- Handles unbounded sessions automatically (infinite-state model checking).
- Fully automated (push-button verification for many protocols).

What ProVerif cannot do:

- Produce numerical bounds ("the attack probability is $\leq 2^{-128}$").
- Model computational assumptions (birthday bounds, lattice hardness).
- Compose with other proof techniques (cannot combine with game-based results).

What MTSF pinging does well:

- Produces explicit numerical bounds ($\delta_{\mathsf{ping}} \leq \mathsf{negl}$).
- Models computational assumptions within the bidding-round framework.
- Composes with all other MTSF proof techniques (CNF, market merger).

What MTSF pinging cannot do:

- Exhaustively enumerate all attack traces (relies on the proof author identifying relevant bids).
- Provide automated verification (requires human proof construction).

The ideal workflow: Use ProVerif to exhaustively check the protocol design (finding structural flaws like Needham–Schroeder's missing identity binding), then use MTSF to produce quantitative bounds and composition guarantees. The two approaches are complementary, not competing.

ProVerif [6] and Tamarin [7] provide *symbolic* unbounded session analysis: they check whether *any* number of concurrent sessions can be attacked, using abstract term algebras where cryptographic operations are perfect. Session pinging provides a complementary *computational* unbounded analysis with concrete advantage bounds.

Table 2. Comparison of MTSF session pinging with symbolic formal verification for unbounded sessions.

Property	ProVerif/Tamarin	CryptoVerif	MTSF Pinging
Session bound	Unbounded	Polynomial	Unbounded (by induction)
Concrete bounds	No	Yes	Yes
Cross-session attacks	Detected	Partially	Detected (ping clause)
CNF audit trail	No	No	Yes (worksheet)
Manual verification	No	No	Yes (four-column method)
Key rotation guidance	No	No	Yes (accumulation bound)

✻ Simple Terms: Session Pinging vs. Formal Verification Tools

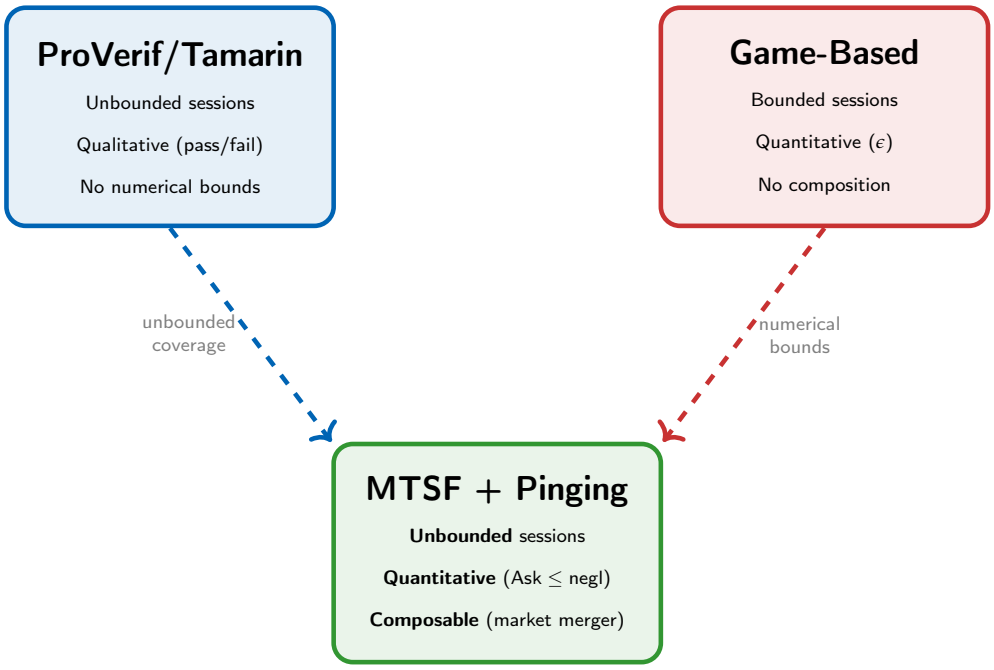
ProVerif and Tamarin are computer programs that automatically check whether a protocol is secure. They can handle any number of sessions (unbounded) but give only a binary answer: "secure" or "attack found." They cannot tell you *how secure*—there are no numerical bounds.

MTSF session pinging gives you both: unbounded sessions *and* numerical bounds. The ping theorem says: "If session 1 is secure with advantage ϵ, and each ping passes, then session i is secure with advantage at most $\epsilon + (i-1)\delta$." This is much more informative than a binary answer, and it tells you exactly when you need to rotate keys (when the accumulated δ gets too large).

The ideal approach is to use both: run ProVerif/Tamarin to check for logical attacks (the "stress test" in MTSF's terminology), then use session pinging to get concrete numerical bounds for the deployment.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2



Session pinging bridges the gap:
ProVerif gives unbounded qualitative coverage; game-based gives bounded quantitative bounds.
MTSF's session pinging gives **both**: unbounded coverage **with** quantitative bounds.

Figure 21. MTSF's session pinging bridges the gap between formal verification (unbounded, qualitative) and game-based proofs (bounded, quantitative), providing the best of both worlds.

6. Unified Novelties

Summary of All Novelties—Thirteen Technical Contributions

N1. Market-theoretic proof language. Advantage bounds are recast as *ask prices*; challenger and adversary become seller and buyer; security reductions are market arbitrages; tightness of reductions corresponds to market spread. This is the first proof framework to use economic market language as a formal mathematical apparatus, not merely as metaphor.

N2. Extended difference lemma for m simultaneous failures. Generalises Shoup's classical single-failure difference lemma to $m \geq 1$ simultaneous failure events $F_1, \ldots, F_m$ with inclusion-exclusion tightening: $\Pr[\bigcup_i F_i] \leq \sum_i \Pr[F_i] - \sum_{i<j} \Pr[F_i \cap F_j] + \cdots$. Eliminates the need to split multi-failure proof steps into multiple games.

N3. UC framework as market regulation. The UC environment $\mathcal{Z}$ is recast as a *market regulator* overseeing all concurrent protocol executions. The simulator $\mathcal{S}$ is a *market arbitrageur* bridging the real and ideal worlds. Composition theorems become *market merger theorems*: composing two secure protocols is a lossless merger of two equilibrium markets.

N4. GUC shared infrastructure and CNF market audit. GUC shared functionality (PKI, common reference strings) becomes *market infrastructure*—public goods accessible to all buyers and sellers. CNF session correctness becomes a *market audit*: every session must pass a formal Boolean checklist, with dishonest session traces failing the audit. Cross-session federation via session identifiers becomes *market federation*.

N5. Formal verification as market stress testing. ProVerif and Tamarin are recast as *market stress testers*: they enumerate all possible attack strategies (buyer bids) and verify that none succeed. Successful model checking $\Leftrightarrow$ no profitable bid exists $\Leftrightarrow$ market equilibrium.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

N6. Insecurity as market collapse within the same formalism. Insecure schemes (textbook RSA: $\mathsf{Ask} = 1$ via homomorphism bid) and insecure protocols (Needham–Schroeder: $\mathsf{Ask} = 1$ via masquerade bid; Signal X3DH-noOPK: $\mathsf{Ask}(g_{\mathsf{async}}) = 1$ via replay bid) are proved insecure using the same bidding-round machinery as security proofs. No separate framework needed.

N7. End-to-end security pipeline. MTSF provides a complete pipeline from primitives to composed protocols: primitive security → KEM/signature composition → session-CNF correctness → unbounded session pinging. Each stage is quantitative and unified.

N8. Protocol-level games as market goods. Entity authentication (g_{auth}), mutual authentication (g_{mutual}), session-key secrecy (g_{sk}), and CNF correctness (g_{CNF}) are formalised as explicit security goods offered by the protocol market. Additionally, the Signal Protocol analysis introduces forward secrecy (g_{FS}), post-compromise security (g_{PCS}), asynchronous establishment (g_{async}), and deniability (g_{deny}) as first-class market goods. Each has an explicit ask price and a game-based proof. This is the first unified market treatment of all protocol-level security properties.

N9. Symmetric and asymmetric primitive markets. HMAC (SUF-CMA via dual PRF hops), AEAD (IND-CCA2 + INT-CTXT via EtM), SLH-DSA (hash-based EUF-CMA), FN-DSA (NTRU lattice EUF-CMA), alongside ECDSA, ML-KEM, and ML-DSA—all analysed via bidding-round chains with explicit price adjustments.

N10. Cryptanalytic bid taxonomy. A comprehensive taxonomy of attack-as-bid types: differential bid (S-box propagation), rotational bid (rotation-commutation break), linear bid (bias exploitation), related-key bid (key-schedule weakness), state-recovery bid (register inversion), key-recovery bid (initialisation inversion), distinguishing bid (statistical bias detection), TMTO bid (precomputation), and quantum bids (O2H reprogramming, measure-and-reprogram). Each is formalised as an explicit price adjustment in the bidding-round chain.

N11. CNF verification with easy manual worksheet. A canonical five-phase verification algorithm (Algorithm 1) and a four-column manual truth-table worksheet are provided for every case study, making MTSF's audit framework accessible to practitioners without expert cryptographic training.

N12. Session pinging for unbounded security. The session ping function (Theorem 15) provides a formal inductive mechanism for lifting bounded game-based security proofs to unbounded session security. Ping bids are included in every case study. This bridges the gap between bounded game-based proofs (quantitative) and formal verification tools like ProVerif (unbounded, qualitative).

N13. QROM market-theoretic formalisation. The O2H lemma and measure-and-reprogram technique are recast as explicit bidding-round price adjustments in the QROM. The characteristic square-root factor $\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}}$ is the "quantum bid price"—the cost the buyer pays to distinguish a reprogrammed QROM oracle. This is the first market-theoretic formalisation of QROM security, connecting the economic intuition of MTSF to post-quantum cryptographic standards (ML-KEM FIPS 203).

MTSF as a Superset of All Four Paradigms—A Formal Summary

This section summarises, for the reader's convenience, the precise sense in which MTSF *subsumes* the four existing paradigms. Each paradigm is a special case of MTSF, recoverable by restricting the market model:

Game-based proofs ⊂ MTSF. A game-based proof with hops $G_0, \ldots, G_q$ and failure events $F_1, \ldots, F_q$ is an MTSF bidding-round chain $\mathbf{B}_0, \ldots, \mathbf{B}_q$ with $\Delta\mathsf{Price}_k = \Pr[F_k]$. The extended difference lemma generalises the classical single-failure lemma. The ask price $\mathsf{Ask} = \sum \Delta\mathsf{Price}_k$ is the standard advantage bound. Recovery: set $\mathcal{G} = \{g\}$ (single good), ignore CNF and pinging, and use the classical difference lemma.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

UC/GUC $\subset$ MTSF. A UC proof showing $\mathsf{Real}_{\pi,\mathcal{A},\mathcal{Z}} \approx_c \mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ is an MTSF market where the environment $\mathcal{Z}$ is the regulator, the simulator $\mathcal{S}$ is the arbitrageur, and the composition theorem is the market merger theorem. Recovery: define a single good g_{UC} with $\mathsf{Ask}(g_{\mathsf{UC}}) = \max_{\mathcal{Z},\mathcal{A}} |\Pr[\mathsf{Real}=1] - \Pr[\mathsf{Ideal}=1]|$, and the market is in equilibrium iff the protocol UC-realises the functionality. GUC's CNF session correctness is directly MTSF's session-CNF (Theorem 13).

Formal verification $\subset$ MTSF. A ProVerif/Tamarin proof showing $\forall \tau \in \mathsf{Traces}(\pi) : \tau \models \varphi$ is an MTSF market stress test where the regulator enumerates all buyer strategies (traces) and checks that the security goods are not violated. Recovery: define goods g_j corresponding to the properties φ, set $\mathsf{Ask}(g_j) = 0$ if all traces satisfy φ (equilibrium), and $\mathsf{Ask}(g_j) = 1$ if any trace violates φ (collapse). The binary output of formal tools corresponds to the extreme cases of the continuous ask price.

Ad hoc insecurity $\subset$ MTSF. A bespoke attack demonstrating that an adversary $\mathcal{A}$ breaks scheme Π with advantage ϵ is an MTSF bid $(\mathsf{Bid} = (g_j, T_{\mathcal{A}}, \epsilon))$ that succeeds, with $\mathsf{Ask}(g_j) \geq \epsilon$. If $\epsilon = 1$, the market collapses (complete insecurity). Recovery: construct a bidding-round chain where the attack is the sole bidding round with $\Delta\mathsf{Price} = \epsilon$.

Figure 22. MTSF as a superset of existing paradigms. Game-based proofs, UC/GUC, formal verification, and ad hoc insecurity are all contained within MTSF as special cases.

7. Soundness and Completeness of MTSF

Any formal security framework must justify *why* its verdicts are trustworthy. In logic, a proof system is *sound* if it never declares a false statement true, and *complete* if it eventually declares every true statement provable. This section establishes analogous guarantees for MTSF: soundness means that if the market reaches equilibrium then the scheme is genuinely secure; completeness means that if the scheme is insecure then some buyer can collapse the market. The central technical mechanism is a *subroutine-consumption reduction*: during the bidding round the seller embeds the buyer as a black-box subroutine inside an algorithm for a computationally hard problem, thereby showing that the buyer's success would yield an efficient solver for that problem—a contradiction under standard hardness assumptions.

✻ Simple Terms: Soundness and Completeness—Why They Matter

What do soundness and completeness mean here?

- **Soundness ("no false positives"):** If MTSF declares a scheme secure (market equilibrium), then the scheme *really is* secure—no efficient attacker can break it. Without soundness, the market could "clear" even for a broken scheme: a dangerous false sense of security.
- **Completeness ("no false negatives"):** If a scheme is insecure, then MTSF will detect the flaw—some buyer will submit a profitable bid that collapses the market. Without completeness, real vulnerabilities might hide in plain sight, undetectable by the framework.

Why is this different from traditional proofs? In classical game-based security, soundness is implicit (reductions are correct by construction) and completeness is not usually discussed because proofs are crafted per-scheme. MTSF, as a *general-purpose* framework, must demonstrate both properties *at the framework level*—ensuring that the market language itself does not introduce logical gaps.

The key trick: The seller proves soundness by "consuming" the buyer as a subroutine. If the buyer could win the security game, the seller would use the buyer's strategy as a black-box tool to solve a problem believed to be computationally intractable (like factoring, MLWE, or ECDLP). Since no efficient algorithm can solve that hard problem, the buyer cannot win either.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

7.1. Subroutine Consumption: The Core Reduction Mechanism

The backbone of every MTSF security proof is a *subroutine-consumption reduction*. During a bidding round $\mathbf{B}_k$, the seller constructs a polynomial-time algorithm $\mathcal{R}$ (the *reducer*) that:

1. receives an instance x of a computationally hard problem Π_{hard};
2. simulates the market environment for the buyer $\mathcal{A}$ (setting up keys, answering oracle queries, issuing challenges) without knowing the solution to x;
3. invokes the buyer $\mathcal{A}$ as a *black-box subroutine*—feeding $\mathcal{A}$ the simulated view and collecting $\mathcal{A}$'s output;
4. translates the buyer's bid output (a forgery, a distinguishing guess, a key recovery) into a valid solution for x.

Thus $\mathcal{R}^{\mathcal{A}}(\cdot)$ is a polynomial-time oracle machine that solves Π_{hard} whenever the buyer's bid succeeds.

Definition 18 (Subroutine-Consumption Reduction)**.** *Let Π_{hard} be a computational problem and let $\mathcal{M} = (\mathsf{Seller}, \{\mathsf{Buyer}_i\}, \mathcal{G}, \mathsf{Price})$ be an MTSF security market for a scheme Π. A* subroutine-consumption reduction *for good $g_j \in \mathcal{G}$ is a PPT oracle machine $\mathcal{R}$ such that for every PPT buyer $\mathcal{A}$:*

$$\mathsf{Adv}^{\Pi_{\mathsf{hard}}}_{\mathcal{R}^{\mathcal{A}}}(\lambda) \;\geq\; \frac{1}{L(\lambda)} \cdot \mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda) - \delta(\lambda),$$

where $L(\lambda) \leq \mathsf{poly}(\lambda)$ is the tightness loss *and $\delta(\lambda)$ is a negligible simulation error. The reducer $\mathcal{R}$* ***consumes*** *$\mathcal{A}$ as a subroutine: it runs $\mathcal{A}$ internally, relaying queries and collecting outputs, without knowledge of $\mathcal{A}$'s internal strategy.*

✻ Simple Terms: Subroutine-Consumption Reduction

What does "consuming as a subroutine" mean? Imagine the seller is given a locked safe (the hard problem instance x). The seller does not know the combination. But the seller knows that if the buyer can break the security scheme, the buyer must implicitly "know" something about the combination. So the seller:

1. Disguises the safe's lock as a security challenge (setting up the game using x).
2. Hands this disguised challenge to the buyer.
3. The buyer tries to break the scheme and produces output (a forgery, a guess, etc.).
4. The seller reads the buyer's output and *extracts* the safe's combination from it.

The buyer never realises it is helping the seller crack a safe—it thinks it is playing a normal security game. This is why it is called "consumption": the seller *consumes* the buyer's computational work, repurposing it to solve the hard problem.

Why does this prove security? If the hard problem is genuinely hard (no efficient algorithm can solve it), then the seller-plus-buyer combination also cannot solve it. But the seller is efficient (polynomial time). So the only way the combination fails is if the *buyer* fails—i.e., the buyer cannot win the security game. Contrapositive: buyer success ⇒ hard problem solved ⇒ contradiction ⇒ buyer cannot succeed.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

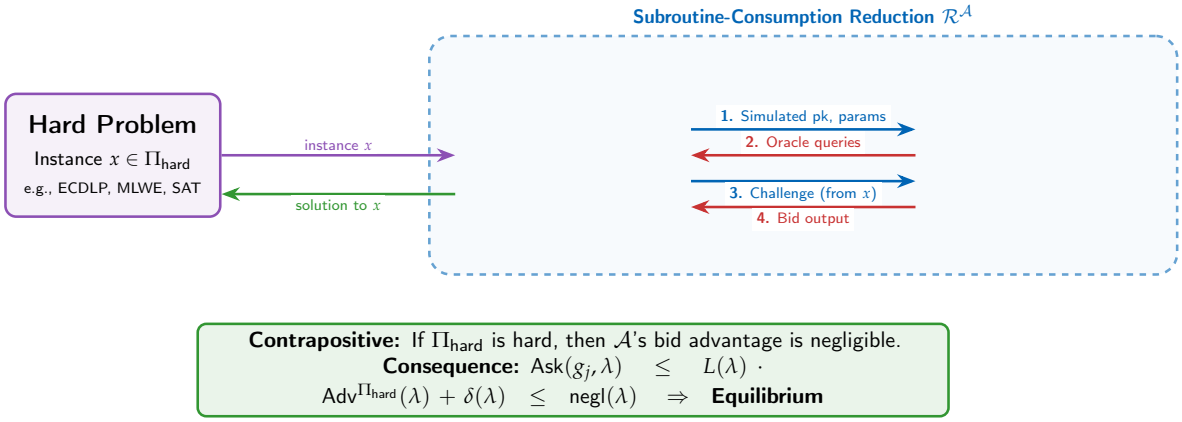


Figure 23. Subroutine-consumption reduction: the seller (reducer $\mathcal{R}$) receives a hard-problem instance x, simulates the market for the buyer $\mathcal{A}$, consumes $\mathcal{A}$'s bid output to solve x. Hardness of Π_{hard} implies the buyer's advantage is negligible—market equilibrium.

✎ Analogy:

Subroutine Consumption as a Job Interview Ruse Imagine a company (the seller) is trying to solve a very hard engineering puzzle (the hard problem). They post a job listing (the security game) and invite a brilliant candidate (the buyer/adversary) for a "technical interview." The interview questions are secretly the real engineering puzzle in disguise. If the candidate solves the interview questions (wins the security game), the company extracts the puzzle solution from the candidate's answers. Since the engineering puzzle is believed unsolvable, no candidate can pass the interview—the "job opening" (security game) remains unfilled (equilibrium).

✎ Analogy:

Subroutine Consumption as a Sting Operation Here is another way to think about subroutine consumption—as a police sting operation:

The setup: The police (seller/reducer) want to catch a master thief (buyer/adversary) who claims they can crack any safe (break the security scheme). The police set up a fake jewellery store (the simulated security game) with a safe that contains evidence linked to an unsolved bank robbery (the hard problem instance).

The sting: The thief breaks into the fake store and cracks the safe (wins the security game). But by cracking this particular safe, the thief unknowingly reveals the technique needed to solve the bank robbery case (the hard problem).

The contradiction: The bank robbery was declared "unsolvable" by the FBI (the hard problem is computationally intractable). If the thief really could crack the safe, the FBI's assessment would be wrong. Since the FBI is trusted (the hardness assumption holds), the thief *cannot* actually crack the safe—the security scheme is secure.

The tightness loss $L(\lambda)$: Sometimes the police need multiple sting operations to catch the thief (they must guess which store the thief will target). The factor $L(\lambda)$ measures how many attempts the police need. A "tight" reduction means $L = 1$—one sting suffices. A loose reduction means $L = q_H \cdot q_S$—the police need many stings, which weakens the conclusion proportionally.

7.2. NP-Hard Subroutine Chains: Consuming One Hard Problem to Solve Another

The subroutine-consumption mechanism extends naturally to chains of NP-hard reductions. If the buyer's bid strategy $\mathcal{A}$ constitutes an algorithm for NP-hard problem Π_A, and an algorithm for Π_A can itself be consumed as a subroutine to solve a *different* NP-hard problem Π_B, then the seller constructs a two-layer reduction: the outer reducer solves Π_B by consuming a solver for Π_A, which

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

itself is the buyer. This mechanism mirrors the structure of Karp reductions in complexity theory, but operationalised within the MTSF bidding-round framework.

✎ Analogy:

NP-Hard Chains as a Puzzle Tournament Imagine a puzzle tournament with three rounds:

1. **Round 1 (Sudoku):** A contestant who can solve 9×9 Sudoku puzzles instantly.
2. **Round 2 (Graph Colouring):** If you can solve Sudoku instantly, I can use your Sudoku skills to solve graph colouring problems (by encoding the graph as a Sudoku grid).
3. **Round 3 (SAT):** If you can solve graph colouring instantly, I can use your colouring skills to solve Boolean satisfiability problems (SAT) (by encoding SAT as a colouring problem).

This is an NP-hard subroutine chain of depth 2: Sudoku → Graph Colouring → SAT. Each step "consumes" the previous solver as a subroutine.

In MTSF: If the buyer (adversary) can break the security scheme, the seller consumes the buyer to solve problem Π_A (step 1), then uses the Π_A solver to solve Π_B (step 2). Since Π_B is NP-hard (believed unsolvable efficiently), the buyer cannot break the scheme either.

Why chains matter: Some security proofs require multiple layers of reduction—the buyer's strategy does not directly solve the hard problem, but it solves an intermediate problem that can then be used to solve the hard problem. NP-hard chains formalise this multi-layer structure within MTSF's bidding-round framework.

Definition 19 (NP-Hard Subroutine Chain)**.** *Let Π_A and Π_B be NP-hard problems and let g_j be a security good in market $\mathcal{M}$. An* NP-hard subroutine chain *of depth d is a sequence of PPT oracle machines $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_d$ such that:*

1. *$\mathcal{R}_1^{\mathcal{A}}$: consumes the buyer $\mathcal{A}$ (who attacks good g_j) to produce a solver for Π_A.*
2. *$\mathcal{R}_2^{\mathcal{R}_1^{\mathcal{A}}}$: consumes the Π_A-solver as a subroutine to produce a solver for Π_B.*
3. *In general, $\mathcal{R}_{i+1}^{\mathcal{R}_i^{\cdots}}$ consumes the Π_{i-1}-solver to produce a Π_i-solver.*

The total tightness loss is $L_{\mathsf{total}}(\lambda) = \prod_{i=1}^{d} L_i(\lambda)$ and the total simulation error is $\delta_{\mathsf{total}}(\lambda) = \sum_{i=1}^{d} \delta_i(\lambda)$.

✻ Simple Terms: NP-Hard Subroutine Chains

The idea in simple terms: Suppose the buyer tries to break a security scheme, and breaking it is equivalent to solving Problem A (say, cracking a combinatorial puzzle). Now suppose there is a known mathematical transformation: any solution to Problem A can be used to solve Problem B (a *different* puzzle). Then the seller can chain these reductions:

1. **Layer 1:** The seller sets up the market so that the buyer's attack output solves Problem A.
2. **Layer 2:** The seller feeds the Problem A solution into a converter that solves Problem B.

If Problem B is genuinely unsolvable by any efficient algorithm, then neither is Problem A, and therefore neither can the buyer win the security game.

Why is this powerful? It lets us prove security by connecting to the *hardest link in a chain* of problems. Even if the direct connection between the security game and the hardest known problem is indirect, the chain of subroutine consumptions links them rigorously.

NP-completeness connection: Recall that every NP-complete problem reduces to every other NP-complete problem. If the buyer's bid strategy solves one NP-complete problem, the chain shows it solves *all* of them—which is believed impossible for polynomial-time algorithms.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Algorithm 2 NP-Hard Subroutine Chain Reduction

Require: NP-hard problems Π_A, Π_B; buyer $\mathcal{A}$ attacking good g_j; instance $y \in \Pi_B$
Ensure: Solution to y (if $\mathcal{A}$ succeeds) or $\perp$

// Layer 1: Reduce Π_B instance to Π_A instance
1: Parse y as an instance of Π_B
2: Compute Karp reduction: $x \leftarrow \mathsf{KarpReduce}_{B \to A}(y)$ ▷ x is a Π_A instance

// Layer 2: Reduce Π_A instance to security game instance
3: Embed x into market setup: $(\mathsf{pk}, \mathsf{state}) \leftarrow \mathcal{R}_1.\mathsf{Setup}(x, 1^\lambda)$
4: Initialise buyer: send pk to $\mathcal{A}$

// Layer 3: Execute bidding round with buyer as subroutine
5: **while** $\mathcal{A}$ makes oracle query q_i **do**
6: $a_i \leftarrow \mathcal{R}_1.\mathsf{SimOracle}(q_i, \mathsf{state})$ ▷ Simulate oracle using x
7: Send a_i to $\mathcal{A}$
8: **end while**
9: Issue challenge $c^* \leftarrow \mathcal{R}_1.\mathsf{Challenge}(\mathsf{state})$ to $\mathcal{A}$
10: Receive bid output $\beta \leftarrow \mathcal{A}(c^*)$ ▷ **Buyer consumed as subroutine**

// Layer 4: Extract solutions up the chain
11: **if** β is a valid bid (forgery/guess/key) **then**
12: Extract Π_A solution: $s_A \leftarrow \mathcal{R}_1.\mathsf{Extract}(\beta, \mathsf{state})$
13: Lift to Π_B solution: $s_B \leftarrow \mathsf{KarpLift}_{A \to B}(s_A, y)$ ▷ Inverse map
14: **return** s_B
15: **else**
16: **return** $\perp$ ▷ Buyer's bid failed; no solution extracted
17: **end if**

✻ Simple Terms: Reading the NP-Hard Chain Algorithm

What the algorithm does step by step:

1. **Layer 1 (Lines 2–3):** The seller receives a hard puzzle y (Problem B) and transforms it into an equivalent hard puzzle x (Problem A) using a known mathematical reduction.
2. **Layer 2 (Lines 5–6):** The seller disguises puzzle x as a security game, creating fake public keys and parameters that look real to the buyer but secretly encode x.
3. **Layer 3 (Lines 8–13):** The seller runs the normal bidding round: answering the buyer's oracle queries (simulating them using x), issuing a challenge, and collecting the buyer's output β. The buyer is an unwitting subroutine—it does not know it is solving x.
4. **Layer 4 (Lines 15–20):** If the buyer wins (produces a valid forgery/guess), the seller extracts a solution to puzzle x from the buyer's output, then lifts it back to a solution for the original puzzle y.

The punchline: If both puzzle x and puzzle y are NP-hard, no efficient buyer can produce β that leads to a solution. The market remains in equilibrium.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

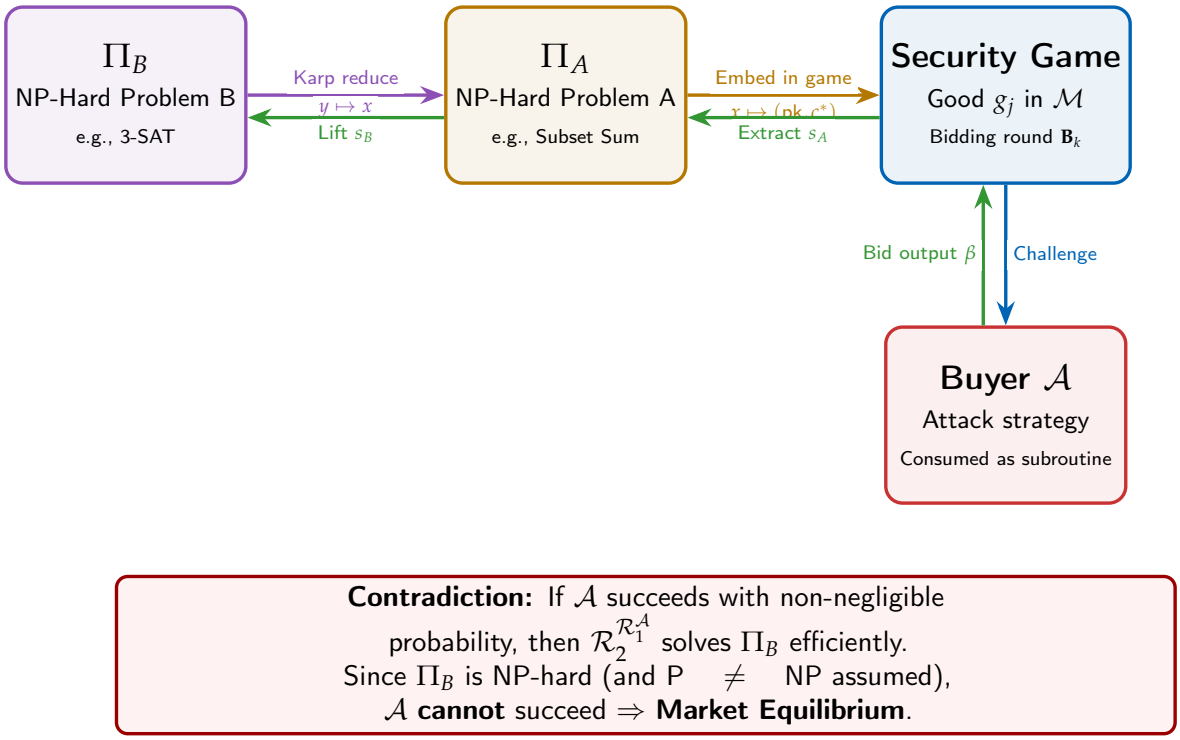


Figure 24. NP-hard subroutine chain: the seller reduces Π_B to Π_A to the security game. The buyer is consumed as the innermost subroutine. Solution extraction propagates outward: $\beta \to s_A \to s_B$. NP-hardness of Π_B prevents the buyer from succeeding.

7.3. Soundness of MTSF

Soundness asserts that MTSF equilibrium implies genuine security: if the bidding-round chain terminates with negligible cumulative cost, then no efficient adversary can break the scheme.

Theorem 4 (Soundness of MTSF). *Let $\mathcal{M} = (\mathsf{Seller}, \{\mathsf{Buyer}_i\}, \mathcal{G}, \mathsf{Price})$ be a security market for scheme Π, and let $g_j \in \mathcal{G}$ be a security good. Suppose there exists a subroutine-consumption reduction $\mathcal{R}$ from the security game for g_j to a computational problem Π_{hard} with tightness loss $L(\lambda)$ and simulation error $\delta(\lambda)$. If Π_{hard} is $(T(\lambda), \epsilon_{\mathsf{hard}}(\lambda))$-hard (i.e., no algorithm running in time T succeeds with probability $> \epsilon_{\mathsf{hard}}$), then for all PPT buyers $\mathcal{A}$:*

$$\mathsf{Ask}(g_j, \lambda) \;\leq\; L(\lambda) \cdot \epsilon_{\mathsf{hard}}(\lambda) + \delta(\lambda).$$

In particular, if Π_{hard} is hard against PPT (i.e., $\epsilon_{\mathsf{hard}}(\lambda) \leq \mathsf{negl}(\lambda)$) and $L(\lambda) \leq \mathsf{poly}(\lambda)$ and $\delta(\lambda) \leq \mathsf{negl}(\lambda)$, then $\mathsf{Ask}(g_j, \lambda) \leq \mathsf{negl}(\lambda)$—the market is in equilibrium.

Proof. By contradiction. Suppose some PPT buyer $\mathcal{A}^*$ achieves $\mathsf{Adv}^{g_j}_{\mathcal{A}^*}(\lambda) = \epsilon^* \not\leq \mathsf{negl}(\lambda)$—i.e., ϵ^* is non-negligible. By the subroutine-consumption reduction (Theorem 18), the reducer $\mathcal{R}^{\mathcal{A}^*}$ solves Π_{hard} with advantage:

$$\mathsf{Adv}^{\Pi_{\mathsf{hard}}}_{\mathcal{R}^{\mathcal{A}^*}}(\lambda) \;\geq\; \frac{1}{L(\lambda)} \cdot \epsilon^* - \delta(\lambda).$$

Since ϵ^* is non-negligible, $L(\lambda) \leq \mathsf{poly}(\lambda)$, and $\delta(\lambda) \leq \mathsf{negl}(\lambda)$, the right-hand side is non-negligible. Thus $\mathcal{R}^{\mathcal{A}^*}$ is a PPT algorithm (since $\mathcal{R}$ is PPT and $\mathcal{A}^*$ is PPT, the composition is PPT) that solves Π_{hard} with non-negligible advantage—contradicting the hardness assumption on Π_{hard}. Therefore no such $\mathcal{A}^*$ exists, and:

$$\mathsf{Ask}(g_j, \lambda) = \max_{\mathcal{A} \in \mathsf{PPT}} \mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda) \leq L(\lambda) \cdot \epsilon_{\mathsf{hard}}(\lambda) + \delta(\lambda) \leq \mathsf{negl}(\lambda).$$

The market is in equilibrium for good g_j. □

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✻ Simple Terms: Soundness Proof

What the proof says in plain language: Suppose, for the sake of argument, that a clever attacker (buyer) *could* break the security scheme. The seller would then use this attacker as a subroutine to solve a supposedly impossible mathematical problem (like factoring huge numbers). Since the mathematical problem is actually impossible to solve efficiently, the attacker cannot exist. Therefore, the scheme is secure.

The logical structure is: "If the scheme were insecure, then an impossible problem would be solvable. Since the impossible problem is not solvable, the scheme is secure." This is the standard proof-by-contradiction structure, but cast in market language: "If a buyer could profit, the seller would use the buyer's strategy to do something impossible. Since it is impossible, no buyer profits. Equilibrium."

☞ Key Insight:

Soundness Depends on Hardness Assumptions Soundness is *conditional*: it holds under the assumption that Π_{hard} is genuinely hard. If the hardness assumption is wrong (e.g., if someone discovers an efficient factoring algorithm), then the soundness guarantee evaporates. This is inherent in all computational security—no unconditional guarantee is possible under $\mathsf{P} \neq \mathsf{NP}$ uncertainty. MTSF makes this dependence explicit: the equilibrium price is bounded by a function of the hardness assumption, with exact tightness loss $L(\lambda)$ and simulation error $\delta(\lambda)$ fully specified.

7.4. Completeness of MTSF

Completeness asserts the converse: if a scheme is insecure, then MTSF detects the flaw—some buyer's bid collapses the market.

Theorem 5 (Completeness of MTSF)**.** *Let* $\mathcal{M} = (\mathsf{Seller}, \{\mathsf{Buyer}_i\}, \mathcal{G}, \mathsf{Price})$ *be a security market for scheme* Π*, and let* $g_j \in \mathcal{G}$ *be a security good. If there exists a PPT algorithm* $\mathcal{A}^*$ *such that* $\mathsf{Adv}^{g_j}_{\mathcal{A}^*}(\lambda) = \epsilon^*(\lambda) \geq 1/\mathsf{poly}(\lambda)$ *(i.e.,* $\mathcal{A}^*$ *breaks* g_j *with non-negligible advantage), then:*

1. ***Bid construction:*** *The buyer submits bid* $\mathsf{Bid}^* = (g_j, T^*, \epsilon^*)$ *where* $T^* = \mathsf{Time}(\mathcal{A}^*)$.
2. ***Market collapse:*** $\mathsf{Ask}(g_j, \lambda) \geq \epsilon^*(\lambda) \geq 1/\mathsf{poly}(\lambda)$*, which is non-negligible.*
3. ***Equilibrium failure:*** *The market is* not *in equilibrium for* g_j.

Moreover, if $\epsilon^*(\lambda) = 1 - \mathsf{negl}(\lambda)$ *(the scheme is totally broken), then* $\mathsf{Ask}(g_j, \lambda) = 1 - \mathsf{negl}(\lambda)$*: the market has fully collapsed.*

Proof. By the definition of ask price (Theorem 11):

$$\mathsf{Ask}(g_j, \lambda) = \max_{\mathcal{A} \in \mathsf{PPT}} \mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda) \geq \mathsf{Adv}^{g_j}_{\mathcal{A}^*}(\lambda) = \epsilon^*(\lambda).$$

Since $\epsilon^*(\lambda) \geq 1/\mathsf{poly}(\lambda)$ is non-negligible, $\mathsf{Ask}(g_j, \lambda) \not\leq \mathsf{negl}(\lambda)$, so the equilibrium condition of Theorem 11 fails. The buyer $\mathcal{A}^*$ realises the bid $\mathsf{Bid}^* = (g_j, T^*, \epsilon^*)$ by running its attack strategy in time T^* and achieving advantage ϵ^*. The market has collapsed (or is at least non-equilibrium) for good g_j.

For total collapse: if $\mathsf{Adv}^{g_j}_{\mathcal{A}^*}(\lambda) = 1 - \mathsf{negl}(\lambda)$, then the ask price satisfies $\mathsf{Ask}(g_j, \lambda) \geq 1 - \mathsf{negl}(\lambda)$, and by the trivial upper bound $\mathsf{Ask}(g_j, \lambda) \leq 1$, we obtain $\mathsf{Ask}(g_j, \lambda) = 1 - \mathsf{negl}(\lambda)$. This is full market collapse: the security good is worthless. □

✻ Simple Terms: Completeness Proof

What the proof says in plain language: If a real attack exists (someone can actually break the scheme), then MTSF will detect it. The attacker simply submits a "bid" (describing the attack

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

strategy, time budget, and success probability), and the market price rises above the negligible threshold. The market is no longer in equilibrium—MTSF correctly identifies the scheme as insecure.

Why is this important? Completeness guarantees that MTSF is not just a "rubber stamp" framework that declares everything secure. If a scheme is broken, MTSF *must* reflect this: the bid succeeds, the ask price rises, and the market collapses. This is exactly what happens in the Needham–Schroeder analysis (Section 13.4), the TLS 1.3 0-RTT analysis (Section 13.8.4), and the Telegram disproof (Section 13.9.2)—all are instances of completeness in action.

7.5. The Soundness–Completeness Duality

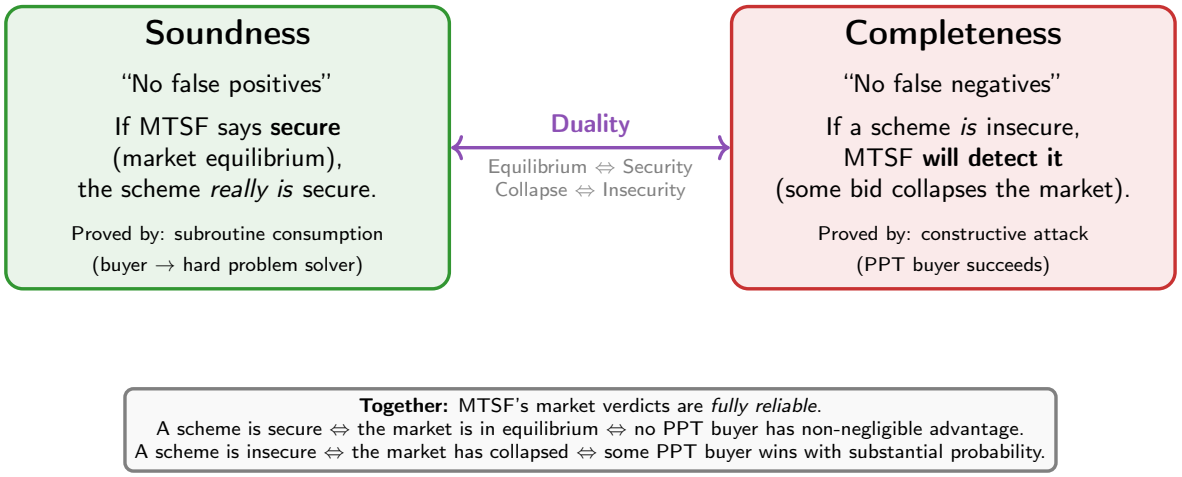


Figure 25. The soundness–completeness duality of MTSF. Soundness ensures no false positives (broken schemes declared secure); completeness ensures no false negatives (insecure schemes undetected). Together, they make MTSF's market verdicts fully reliable.

The soundness and completeness theorems together establish a precise *duality* between the computational world and the market world:

Corollary 1 (MTSF Duality). *For any security good g_j in a market $\mathcal{M}$ with a valid subroutine-consumption reduction to Π_{hard}:*

$$\Pi \textit{ is secure for } g_j \iff \mathcal{M} \textit{ is in equilibrium for } g_j \iff$$

$$\textit{no PPT buyer profits from bidding on } g_j,$$

under the assumption that Π_{hard} is hard against PPT.

Proof. ($\Rightarrow$, Soundness) If Π is secure for g_j, then by Theorem 4 the market is in equilibrium. ($\Leftarrow$, Completeness) If Π is insecure for g_j, then by Theorem 5 some buyer's bid collapses the market—so the market is not in equilibrium. The contrapositive gives: equilibrium $\Rightarrow$ security. □

Figure 26. The soundness–completeness duality of MTSF. Soundness (bottom-left): hardness of Π_{hard} implies security and equilibrium. Completeness (top): insecurity implies market collapse. The three worlds—security, market, and computational—are formally equivalent under MTSF.

✎ Analogy:

The Duality as a Smoke Detector A smoke detector is *sound* if it never triggers a false alarm (alarm $\Rightarrow$ real fire). It is *complete* if it always detects a real fire (fire $\Rightarrow$ alarm). MTSF is both: market

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

equilibrium (no alarm) means the scheme is safe (no fire), and scheme insecurity (fire) means the market collapses (alarm sounds). The subroutine-consumption reduction is the sensor: it connects the "smoke" (attacker success) to the "alarm" (solution to a hard problem, which should not exist).

✎ **Analogy:**

Soundness vs. Completeness as a Medical Test Think of MTSF's verdicts like a medical test for a disease:

- **Soundness (no false positives):** If the test says "healthy" (market equilibrium), the patient really is healthy (the scheme is secure). A false positive would be catastrophic—declaring a vulnerable scheme secure, leading to deployment of a broken system.
- **Completeness (no false negatives):** If the patient has the disease (the scheme is insecure), the test will detect it (some buyer collapses the market). A false negative would mean a real vulnerability hiding undetected.

MTSF achieves both:

- **Soundness:** Proved by showing that if the market reaches equilibrium, then any successful attacker would solve a computationally hard problem—a contradiction. (The medical test is validated against a gold standard.)
- **Completeness:** Proved by showing that if a scheme is insecure, there exists a PPT adversary whose bid collapses the market. (Every diseased patient produces a detectable biomarker.)

The *duality* between soundness and completeness means MTSF's market verdicts are fully reliable: equilibrium $\Leftrightarrow$ security, collapse $\Leftrightarrow$ insecurity. This is the strongest possible guarantee for a security framework.

7.6. Worked Example: ECDSA Soundness via Subroutine Consumption

To make the abstract machinery concrete, we trace the subroutine-consumption reduction for ECDSA's EUF-CMA security (proved in detail in Section 9.1).

Example 1 (ECDSA Subroutine Consumption). *Consider the ECDSA market $\mathcal{M}_{\mathsf{ECDSA}}$ with good g_{EUF} = EUF-CMA (unforgeability). The hard problem is $\Pi_{\mathsf{hard}} = \mathsf{ECDLP}$ (Elliptic Curve Discrete Logarithm Problem).*

Subroutine consumption in action:

1. ***Reducer setup.*** *The reducer $\mathcal{R}$ receives an ECDLP instance $(G, Q = dG)$ where d is the unknown discrete log. $\mathcal{R}$ sets $\mathsf{pk} = Q$ and begins simulating the signing oracle for the buyer $\mathcal{A}$.*
2. ***Oracle simulation.*** *When the buyer requests a signature on message m_i, the reducer simulates (r_i, s_i) using the forking lemma technique: it programs the random oracle so that valid-looking signatures can be produced without knowing d.*
3. ***Buyer consumed.*** *After q_S signing queries, the buyer outputs a forgery $(m^*, \sigma^* = (r^*, s^*))$ on a fresh message $m^* \notin \{m_1, \ldots, m_{q_S}\}$.*
4. ***Solution extraction.*** *Using the forking lemma, $\mathcal{R}$ rewinds $\mathcal{A}$ to obtain two valid signatures (r^*, s_1^*) and (r^*, s_2^*) on m^* with different random oracle responses $h_1 \neq h_2$. From these, $\mathcal{R}$ computes:*

$$d = \frac{s_1^* h_2 - s_2^* h_1}{s_2^* - s_1^*} \cdot r^{*-1} \mod n,$$

solving the ECDLP instance.

Soundness conclusion: *By Theorem 4:*

$$\mathsf{Ask}(g_{\mathsf{EUF}}, \lambda) \leq q_H \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + \frac{q_S}{2^\lambda} + \frac{q_H^2}{2^\lambda} \leq \mathsf{negl}(\lambda).$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The tightness loss is $L = q_H$ (number of hash queries) and the simulation error accounts for nonce collisions and birthday terms. The ECDSA market is in equilibrium.

✻ Simple Terms: ECDSA Soundness Example

What happened: The seller (reducer) was given an unsolved math puzzle ("find d such that $Q = dG$ on an elliptic curve"). The seller disguised this puzzle as an ECDSA public key and told the buyer: "Try to forge a signature." The buyer tried its best attack and produced a forgery. The seller then mathematically extracted the answer to the original puzzle from the buyer's forgery. Since the math puzzle is believed unsolvable, the buyer's forgery cannot exist—ECDSA is secure (market equilibrium).

7.7. Worked Example: NP-Hard Subroutine Chain for CNF-Based Security

The NP-hard subroutine chain is particularly natural when the security game involves CNF-satisfiability checks, since 3-SAT is the canonical NP-complete problem.

Example 2 (CNF Subroutine Chain: SAT $\to$ Subset Sum $\to$ Security Game)**.** *Consider a scheme Π whose session security is verified via a CNF formula φ (Section 4). Suppose a buyer $\mathcal{A}$ can find a satisfying assignment for a class of CNF formulae as part of its attack strategy (e.g., it finds an adversarial transcript that passes the CNF audit).*

Chain construction:

1. ***Layer 1 (Π_A = 3-SAT):*** *The buyer's ability to satisfy the session-CNF implies an algorithm for a subclass of 3-SAT instances (those arising from protocol transcripts). The reducer $\mathcal{R}_1$ embeds a 3-SAT instance into the session-CNF by encoding the SAT variables as session parameters (nonces, keys, timestamps).*
2. ***Layer 2 (Π_B = Subset Sum):*** *By Karp's classical reduction, any 3-SAT instance can be transformed into a Subset Sum instance in polynomial time. The outer reducer $\mathcal{R}_2$ first transforms the Subset Sum instance y into a 3-SAT instance x (via the reverse encoding), then invokes $\mathcal{R}_1^{\mathcal{A}}$ to solve x.*
3. ***Solution propagation:*** *If the buyer satisfies the session-CNF, $\mathcal{R}_1$ extracts a 3-SAT assignment, and $\mathcal{R}_2$ lifts it to a Subset Sum solution.*

Conclusion: *If the buyer could pass the CNF audit with an adversarial transcript, the chain would yield an efficient Subset Sum solver—contradicting the NP-hardness of Subset Sum (under $\mathsf{P} \neq \mathsf{NP}$). The session-CNF audit is sound: no adversarial transcript passes.*

★ Intuition:

Why NP-Hard Chains Strengthen the Framework Each link in the NP-hard chain adds a layer of confidence. Even if one suspects that the *direct* hard problem might someday be solved (e.g., lattice problems under quantum attack), the chain shows that breaking the scheme would solve *multiple* NP-hard problems simultaneously. This is analogous to requiring an attacker to independently rob multiple banks in a single heist—each bank's security multiplies the difficulty.

7.8. Relationship to Classical Meta-Theorems

✎ Analogy:

Equilibrium as a Ball at the Bottom of a Valley Visualise a ball (the attacker's strategy) rolling across a landscape:

- **Equilibrium (secure):** The ball is at the bottom of a deep valley. No matter which direction the attacker pushes (which bid they try), the ball rolls back to the bottom. The valley is so deep (2^{128} units) that no polynomial-strength push can move the ball significantly. The market is *stable*.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Collapse (insecure):** The ball is on a hilltop. The slightest push sends it rolling downhill—the attacker wins effortlessly. The market is *unstable* and collapses at the first bid.
- **Partial collapse:** The ball is on a shallow slope. It is not at the bottom (the scheme is not perfectly secure), but the slope is gentle enough that the attacker needs significant effort to reach the valley floor. Telegram's MTProto is an example: the salt extraction bid gives a 2^{-64} advantage—not a full collapse, but not a deep valley either.

The depth of the valley is the ask price's inverse: $1/\text{Ask}$. For ECDSA, the valley is 2^{128} units deep. For Needham–Schroeder, the "valley" is actually a hilltop (depth 0). MTSF measures the exact depth for every scheme.

The soundness and completeness of MTSF are not merely analogies—they are formal instantiations of classical meta-theorems in cryptographic proof theory.

Remark 3 (Connection to Bellare–Rogaway Reductions)**.** *The subroutine-consumption reduction (Theorem 18) is the MTSF formalisation of the classical Bellare–Rogaway reduction paradigm [2]. In the classical setting, a "reduction" is a PPT oracle machine that uses the adversary as a black-box subroutine to solve a hard problem. MTSF adds the market interpretation: the reduction is the* seller *consuming the* buyer *during the bidding round, and the hard problem's intractability ensures market equilibrium. The key insight is that this interpretation makes the* direction *of the reduction intuitive: the buyer bids (attacks), the seller co-opts the bid (reduces), and the hard problem's resistance prevents profit (equilibrium).*

Remark 4 (Connection to Cook–Levin and Karp Reductions)**.** *The NP-hard subroutine chain (Theorem 19) operationalises the Cook–Levin theorem and Karp's 21 reductions [13] within the MTSF bidding framework. The Cook–Levin theorem states that every problem in NP reduces to SAT; Karp's reductions show that SAT reduces to 20 other specific NP-complete problems. In MTSF language: if the buyer's bid strategy implicitly solves any NP-complete problem, the seller can chain reductions to solve* all *NP-complete problems—a catastrophic violation of the* $\mathsf{P} \neq \mathsf{NP}$ *assumption. This connection grounds MTSF's security guarantees in the deepest conjectures of computational complexity theory.*

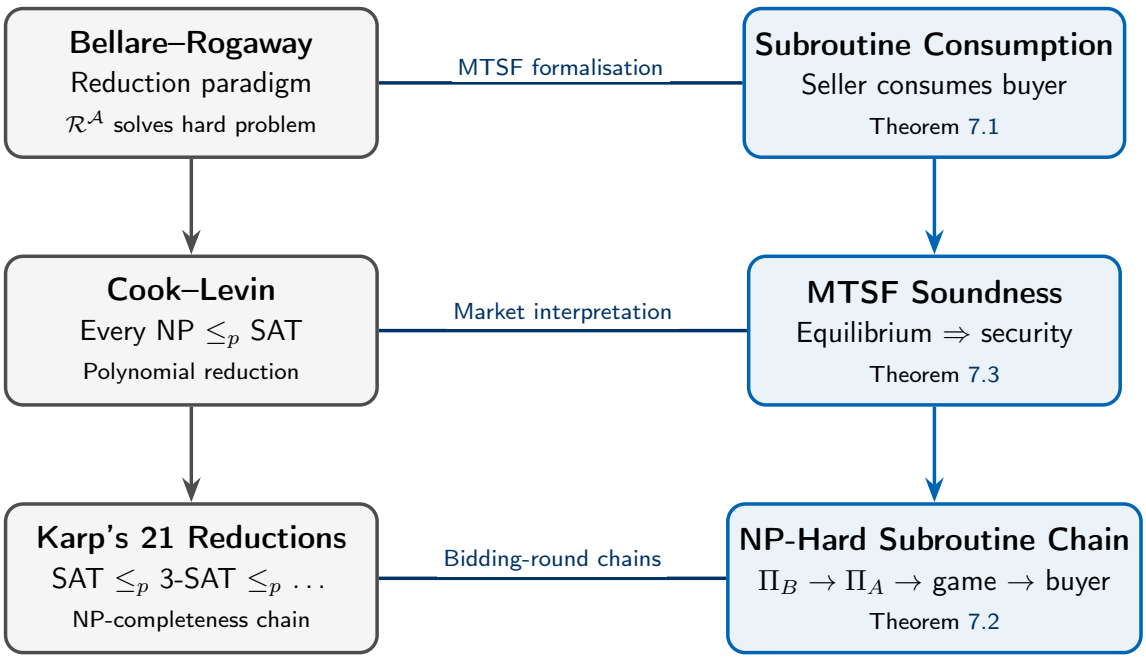


Figure 27. Correspondence between classical meta-theorems in complexity/cryptography and their MTSF counterparts. Each classical result has a precise market-theoretic formalisation.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

7.9. Implications for MTSF Case Studies

The soundness and completeness theorems have direct implications for every case study in this article:

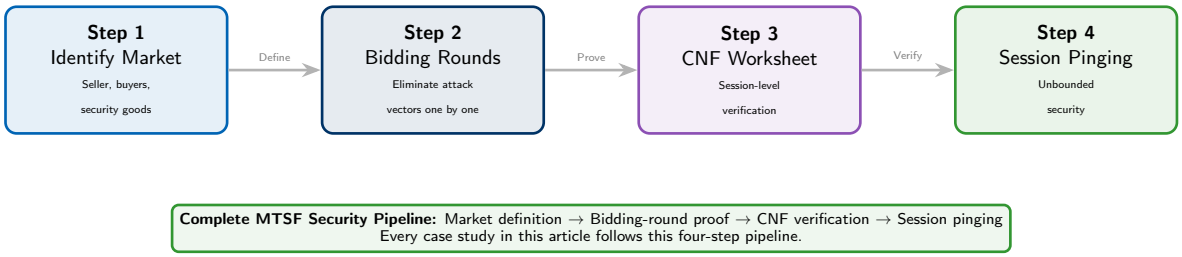


Figure 28. The four-step MTSF security pipeline that every case study follows: (1) define the market (seller, buyers, goods), (2) construct bidding-round chain with price adjustments, (3) fill in the CNF verification worksheet, (4) verify session pinging for unbounded security.

Soundness and Completeness Across All Case Studies

Soundness in action (security proofs): Every security proof in Sections 9–16 proceeds by constructing a subroutine-consumption reduction from the security game to a hard problem. The seller consumes the buyer as a subroutine during each bidding round. Specifically:

- **ECDSA (Section 9.1):** Buyer consumed to solve ECDLP.
- **ML-KEM (Section 9.2):** Buyer consumed to solve MLWE.
- **ML-DSA (Section 9.3):** Buyer consumed to solve MSIS.
- **HMAC (Section 9.5):** Buyer consumed to distinguish a PRF from random.
- **AEAD (Section 9.6):** Buyer consumed to break IND-CPA or MAC unforgeability.
- **SLH-DSA (Section 9.7):** Buyer consumed to find hash collisions or second preimages.
- **FN-DSA (Section 9.8):** Buyer consumed to solve NTRU/SIS.
- **AES (Section 10):** Buyer consumed to distinguish AES from a PRP.
- **SHA-3 (Section 11):** Buyer consumed to find capacity collisions.
- **Grain-128a (Section 12):** Buyer consumed to recover internal state.
- **TLS 1.3 1-RTT (Section 13.8.3):** Buyer consumed to solve ECDLP or break HKDF-PRF.
- **Signal X3DH (Section 13.7):** Buyer consumed to solve gap-CDH.
- **QROM KEM (Section 14):** Buyer consumed to break IND-CPA of underlying PKE via O2H.
- **BB84 QKD (Section 15):** Quantum buyer's information bounded by no-cloning and information-disturbance trade-off; key extracted via quantum leftover hash lemma. Security is information-theoretic (no computational assumption consumed).
- **TLS+Signal composition (Section 16):** Standalone buyers constructed for each market via UC simulation of the other market; merged market equilibrium follows from Theorem 63.

In every case, Theorem 4 guarantees: $\mathsf{Ask}(g_j, \lambda) \leq L \cdot \epsilon_{\mathsf{hard}} + \delta \leq \mathsf{negl}(\lambda)$.

Completeness in action (insecurity proofs): Every insecurity proof constructs an explicit PPT buyer whose bid succeeds with overwhelming probability:

- **Textbook RSA (Section 9.1):** Homomorphism buyer achieves $\mathsf{Ask} = 1$.
- **Needham–Schroeder (Section 13.4):** Masquerade buyer achieves $\mathsf{Ask} = 1$.
- **TLS 1.3 0-RTT (Section 13.8.4):** Replay buyer achieves $\mathsf{Ask} = 1$.
- **TLS 1.3 downgrade (Section 13.8.5):** Version-stripping buyer achieves $\mathsf{Ask} = 1$.
- **Telegram MTProto (Section 13.9.2):** Salt-extraction buyer achieves $\mathsf{Ask} \approx 1$.
- **Signal X3DH without OPK (Section 13.7.1):** Replay buyer achieves $\mathsf{Ask}(g_{\mathsf{async}}) = 1$.

In every case, Theorem 5 guarantees: $\mathsf{Ask}(g_j, \lambda) \geq \epsilon^* = 1 - \mathsf{negl}(\lambda)$—the market has collapsed.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Remark 5 (Subroutine Consumption and the NP-Hardness Paradigm)**.** *The subroutine-consumption mechanism establishes a fundamental link between cryptographic security and computational complexity. In the MTSF framework, every security proof is an instance of the following paradigm:*

> An algorithm for NP-hard problem Π_A can solve another NP-hard problem Π_B if the algorithm for Π_A is consumed as a subroutine.

The buyer's attack strategy constitutes an "algorithm" for a specific computational task (forging signatures, distinguishing ciphertexts, recovering keys). The seller's reduction shows that this task is at least as hard as a known NP-hard problem. The subroutine chain (Theorem 19) extends this to multiple layers: if the buyer's algorithm (for task Π_A) is consumed as a subroutine by a converter that solves Π_B, and Π_B is consumed by another converter that solves Π_C, then the buyer's success would yield an efficient solver for Π_C—a contradiction cascading through the entire NP-hardness hierarchy. This is the market-theoretic operationalisation of the NP-completeness web: breaking one problem breaks them all.

8. Protocol-Level Security Games in MTSF

In a protocol market, the seller offers not just primitive-level goods (EUF-CMA, IND-CCA2) but also *protocol-level goods*: entity authentication, mutual authentication, session-key secrecy, and CNF session correctness. Each is formalised as an individual buyer game within the market.

✻ Simple Terms: Protocol-Level Security Games—Why They Are Needed

The gap between primitives and protocols: Proving that a signature scheme is EUF-CMA secure tells you that an individual signature cannot be forged. But a protocol using that signature scheme might still be broken by an attacker who:

- Replays a valid signature from a *previous* session (replay attack).
- Redirects a valid signature from one party to impersonate another (masquerade attack).
- Wins a man-in-the-middle position without forging any signature (as in Needham–Schroeder).

Protocol-level security games capture these higher-level threats.

Four protocol-level goods in MTSF:

1. **Entity Authentication (g_{auth}):** If Bob accepts a session with partner Alice, then Alice was genuinely participating. An attacker who impersonates Alice to Bob without Alice's involvement loses this game.
2. **Mutual Authentication (g_{mutual}):** Both Alice and Bob accept only if the other genuinely participated. Neither direction of impersonation is possible.
3. **Session-Key Secrecy (g_{sk}):** After a completed protocol run, the session key K_{sess} is computationally indistinguishable from a random key to any attacker who observed all messages. Even a powerful network attacker learns nothing about K_{sess}.
4. **CNF Correctness (g_{CNF}):** The session-CNF formula φ_i is satisfiable only under honest transcripts. Any adversarially modified transcript fails the CNF audit.

Why embed these as market goods? In MTSF, modelling these as market goods enables a unified security analysis. Each good has an explicit ask price (how likely an attacker is to break it), and all reductions are explicit and quantitative. This is the first framework to embed all four protocol-level properties as first-class market goods within a single formalism.

8.1. The Authentication Game

Definition 20 (Authentication Good g_{auth})**.** *The seller offers g_{auth}: the guarantee that after a protocol run, if honest party B accepts, then B's intended partner A was indeed participating. The buyer (active network adversary $\mathcal{A}$) wins if B accepts a session without A having sent the corresponding messages.*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Theorem 6 (Authentication Game—Game-Based Bound). *For a protocol using signatures for entity authentication:*

$$\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda), \tag{19}$$

where q_N is the number of sessions (nonces sampled).

Proof. We construct four games, bounding the difference at each hop.

$\mathbf{B}_0$ (Bidding Round 0: Real Market Execution).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10), analogous to the seller listing the good at quoted price "333" (secure schemes like ECDSA, ML-KEM, AES—the quoted price is a bluff), "555" (quasi-secure schemes like Telegram MTProto—partial bluff), or "999" (insecure schemes like Needham–Schroeder—the price is honest, the good is worthless). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium, $\mathsf{QPrice}_0 =$ "333") or honest (collapse, $\mathsf{QPrice}_0 =$ "999").

The seller (challenger $\mathcal{C}$) runs the protocol honestly, offering the authentication good g_{auth} at ask price Ask_0. The buyer (adversary $\mathcal{A}$) controls the network: it may delay, reorder, drop, and inject messages between honest parties A and B, placing a *full-control network bid*. The buyer wins—and the market collapses for g_{auth}—if B accepts and outputs $\mathsf{pid} = A$, but A did not participate in any matching session. This is the baseline: $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{auth}}_{\mathcal{A}}(\lambda)$.

$\mathbf{B}_1$ (Bidding Round 1: Nonce Freshness Bid).

Purpose: Eliminate the *nonce-reuse attack vector*. Repeated nonces immediately collapse authentication, key-secrecy, or stream-cipher security: in signatures a shared nonce leaks the private key ($\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); in protocols it enables cross-session replay; in stream ciphers it reveals the XOR of two plaintexts. This round aborts the game whenever any nonce collision occurs, ensuring every subsequent hop reasons about a nonce-fresh world.
Replaces: The real nonce-sampling procedure is augmented with a global *collision audit*: a set $\mathcal{N}_{\mathsf{used}}$ records every nonce issued so far. If any newly sampled nonce k' satisfies $k' \in \mathcal{N}_{\mathsf{used}}$, the game aborts immediately. The two games ($\mathbf{B}_{n-1}$ and $\mathbf{B}_n$) are *identical-until-abort*: the adversary's view is indistinguishable until the abort event F_{nonce} occurs, so the difference lemma directly bounds the price adjustment.
Complexity: By the birthday bound, the probability that any two of q independently sampled nonces from a space of size $|\mathcal{N}|$ collide is at most $q^2/(2|\mathcal{N}|)$. For security parameter λ with $|\mathcal{N}| = 2^\lambda$ this gives $\Delta\mathsf{Price} \leq q^2/2^{\lambda+1}$, which is negligible for any polynomial $q = \mathsf{poly}(\lambda)$. After this hop every nonce in the proof is treated as distinct, allowing all later hops to assume a collision-free nonce space without loss of generality.

The seller adds an internal audit: abort the session if any two concurrent or sequential sessions sample the same nonce. This models the seller *refusing to sell* under a *nonce-reuse bid*—a buyer strategy that exploits repeated randomness. The two bidding rounds $\mathbf{B}_0$ and $\mathbf{B}_1$ are identical until the bad event F_{nonce} (a nonce collision) occurs; by the difference lemma, the price adjustment is $\Delta\mathsf{Price}_0 = |\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \Pr[F_{\mathsf{nonce}}]$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Difference bound. By the birthday bound over q_N nonces sampled uniformly from $\{0,1\}^\lambda$:

$$|\Pr[\mathbf{B}_0(\mathcal{A}) = 1] - \Pr[\mathbf{B}_1(\mathcal{A}) = 1]| \leq \Pr[F_{\mathsf{nonce}}] \leq \frac{q_N^2}{2^{\lambda+1}}. \tag{20}$$

$\mathbf{B}_2$ (Bidding Round 2: Signature Forgery Bid).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathsf{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

The seller now also maintains a *signature ledger*: a complete record of every signature honest party A has generated. If the buyer causes B to accept a signature σ^* on session data $m^* \| \mathsf{sid}$ that does *not* appear in the ledger, the market flags this as a *forgery bid* event F_{forge}—the buyer is attempting to sell counterfeit goods. The price adjustment is bounded by the EUF-CMA advantage: a forgery bid succeeds only if the buyer can produce a valid signature under A's key without ever querying A's signing oracle on that message, which by EUF-CMA security costs $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}}$.

Difference bound. We reduce F_{forge} to EUF-CMA: given an EUF-CMA challenger for A's key, the reduction embeds A's public key and forwards signing queries. If $\mathcal{A}$ triggers F_{forge}, the reduction outputs $(m^* \| \mathsf{sid}, \sigma^*)$ as a forgery. Hence:

$$|\Pr[\mathbf{B}_1(\mathcal{A}) = 1] - \Pr[\mathbf{B}_2(\mathcal{A}) = 1]| \leq \Pr[F_{\mathsf{forge}}] \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}}. \tag{21}$$

$\mathbf{B}_3$ (Bidding Round 3: Ideal Market—All Entity-Auth Bids Fail).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\mathsf{Price}_k$, all of which are negligible, establishing market equilibrium.

Conditioned on no nonce-reuse bid ($\neg F_{\mathsf{nonce}}$) and no forgery bid ($\neg F_{\mathsf{forge}}$) succeeding, the market reaches equilibrium for g_{auth}: if B accepts with partner A, then B verified a valid, ledger-registered signature under A's genuine key on session-specific data including a fresh sid and nonce N_B. The

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

signature is in the ledger (no forgery bid succeeded), and the nonce is unique (no nonce bid succeeded), so A must have generated this signature for this exact session—A genuinely participated. The buyer's winning probability in the ideal market: $\Pr[\mathbf{B}_3 = 1] = 0$. *Market equilibrium for* g_{auth} *is established.*

Summing:

$$\mathsf{Ask}(g_{\mathsf{auth}}) \leq \frac{q_N^2}{2^{\lambda+1}} + \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + 0 = \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda). \quad \square \tag{22}$$

✎ Analogy:

Entity Authentication as a Video Call Verification Imagine you receive a phone call from someone claiming to be your bank manager. How do you know it is really them?
Entity authentication is the cryptographic version of this problem. In the physical world, you might:

- Ask a question only the real manager would know (like a shared secret/password).
- Recognise their face on a video call (like a public key—their face is "public" and hard to fake).
- Check that the call came from the bank's official number (like verifying a digital certificate from a trusted CA).

In MTSF, the authentication good g_{auth} formalises this: after a protocol run, if Bob accepts that he is talking to Alice, then Alice was *genuinely participating*. An attacker who impersonates Alice must forge her digital signature—which costs $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}$, a negligible amount for secure signature schemes.
The Needham–Schroeder lesson: Even if the signature scheme is perfect, a protocol can still fail authentication if it does not properly bind identities to messages. The NS protocol's MITM attack succeeds without breaking any cryptographic primitive—a design flaw, not a crypto flaw. MTSF's CNF verification catches such design flaws by requiring an identity-binding clause.

8.2. The Mutual Authentication Game

Definition 21 (Mutual Authentication Good g_{mutual})**.** *The seller offers* g_{mutual}*: after a protocol run,* both *A and B accept with correct partner identities, or neither accepts. The buyer wins if either (a) B accepts with partner A but A did not participate, or (b) A accepts with partner B but B did not participate.*

Theorem 7 (Mutual Authentication Equilibrium)**.** *For a protocol with bidirectional signatures:*

$$\mathsf{Ask}(g_{\mathsf{mutual}}) \leq 2 \cdot \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda). \tag{23}$$

Proof. We construct five games.

$\mathbf{B}_0$ (Bidding Round 0: Real Market, Full Dual-Direction Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

The seller offers the mutual authentication good g_{mutual}. The buyer places a *dual-direction impersonation bid*: it attempts to violate authentication in at least one direction—either impersonating A to B or impersonating B to A (or both). The market ask price is $\text{Ask}_0 = \text{Adv}^{\text{mutual}}_{\mathcal{A}}(\lambda)$.

$\mathbf{B}_1$ (Bidding Round 1: Nonce Freshness Bid—Both Directions).

Purpose: Eliminate the *nonce-reuse attack vector*. Repeated nonces immediately collapse authentication, key-secrecy, or stream-cipher security: in signatures a shared nonce leaks the private key ($\text{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); in protocols it enables cross-session replay; in stream ciphers it reveals the XOR of two plaintexts. This round aborts the game whenever any nonce collision occurs, ensuring every subsequent hop reasons about a nonce-fresh world.
Replaces: The real nonce-sampling procedure is augmented with a global *collision audit*: a set $\mathcal{N}_{\text{used}}$ records every nonce issued so far. If any newly sampled nonce k' satisfies $k' \in \mathcal{N}_{\text{used}}$, the game aborts immediately. The two games ($\mathbf{B}_{n-1}$ and $\mathbf{B}_n$) are *identical-until-abort*: the adversary's view is indistinguishable until the abort event F_{nonce} occurs, so the difference lemma directly bounds the price adjustment.
Complexity: By the birthday bound, the probability that any two of q independently sampled nonces from a space of size $|\mathcal{N}|$ collide is at most $q^2/(2|\mathcal{N}|)$. For security parameter λ with $|\mathcal{N}| = 2^\lambda$ this gives $\Delta\text{Price} \le q^2/2^{\lambda+1}$, which is negligible for any polynomial $q = \text{poly}(\lambda)$. After this hop every nonce in the proof is treated as distinct, allowing all later hops to assume a collision-free nonce space without loss of generality.

The seller enforces nonce uniqueness across all sessions in both directions, placing a *freshness constraint* on nonce bids. Identical-until-F_{nonce}; price adjustment:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \le \frac{q_N^2}{2^{\lambda+1}}. \tag{24}$$

$\mathbf{B}_2$ (Bidding Round 2: Signature Forgery Bid on A—Impersonating A to B).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\text{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \text{Adv}^{\text{ECDLP}}(\lambda) + 1/q + \Pr[F_{\text{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\text{fork}}] \le 1 - \text{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \le 2^{64}$, $q \approx 2^{256}$, $\text{Adv}^{\text{ECDLP}} \le 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\text{fork}} \cup F_{\text{extract}}] \le \Pr[F_{\text{fork}}] + 1/q$.

The seller adds a signature ledger for A. The buyer places a *directed forgery bid*: cause B to accept a signature σ_A that A never generated, thereby impersonating A to B. Any success here is an EUF-CMA forgery against A's signing key. Price adjustment (buyer's cost to break A's signature):

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \le \text{Adv}^{\text{EUF-CMA}}_{\text{Sig},A}. \tag{25}$$

$\mathbf{B}_3$ (Bidding Round 3: Signature Forgery Bid on B—Impersonating B to A).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathsf{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

Symmetrically, the seller adds a signature ledger for B. The buyer places a *directed forgery bid* in the other direction: cause A to accept a signature σ_B that B never generated. By the extended difference lemma, $F_{\mathsf{forge_A}}$ (targeting A) and $F_{\mathsf{forge_B}}$ (targeting B) use independent signing keys, so their failure events are handled separately. Price adjustment (buyer's cost to break B's signature):

$$|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig},B}. \tag{26}$$

$\mathbf{B}_4$ (Bidding Round 4: Ideal Market—Both Dual-Direction Bids Fail).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\mathsf{Price}_k$, all of which are negligible, establishing market equilibrium.

With no nonce-reuse bid, no forgery bid on A, and no forgery bid on B: both signatures in any accepted session are ledger-verified, session-specific, and nonce-fresh. Both parties participated genuinely. The dual-direction impersonation bid fails completely. $\Pr[\mathbf{B}_4 = 1] = 0$. *Market equilibrium for g_{mutual} is established.*

Extended difference lemma (combined). The forgery events F_A and F_B use independent keys, so: $\Pr[F_A \cup F_B] \leq \Pr[F_A] + \Pr[F_B] - \Pr[F_A \cap F_B] \leq 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}$, since $\Pr[F_A \cap F_B] \geq 0$.

Total: $\mathsf{Ask}(g_{\mathsf{mutual}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$. □

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

8.3. The Session-Key Secrecy Game

✎ Analogy:

Session-Key Secrecy as a Sealed Envelope After Alice and Bob complete a key exchange protocol, they share a session key K_{sess}. Session-key secrecy means this key is as good as a fresh random key, even to an attacker who observed every message exchanged during the protocol.
Imagine: Alice and Bob are in a glass room (the attacker can see everything). They perform a complex series of hand gestures (the protocol messages). At the end, they each write a number on a piece of paper and seal it in an envelope. Session-key secrecy means: even though the attacker watched every gesture, they have *no idea* what number is in the envelope. If you gave the attacker two envelopes—one with the real session key and one with a random number—they could not tell which is which.
The game: The challenger generates the session key K_0 honestly and picks a random key K_1. It flips a coin b and gives the attacker K_b. The attacker, having observed the entire protocol execution, must guess b. If they can guess correctly with probability significantly better than 50%, the session-key secrecy good is compromised. If not, the market is in equilibrium.

Definition 22 (Session-Key Secrecy Good g_{sk})**.** *After a completed protocol run producing session key K_{sess}, the buyer receives either K_{sess} or a uniformly random key K' (chosen by a random bit b). The buyer wins by guessing b. The good is secure if* $\mathsf{Ask}(g_{\mathsf{sk}}) \leq \mathsf{negl}(\lambda)$.

Theorem 8 (Session-Key Secrecy Equilibrium)**.** *For a protocol using a KEM for key transport and a KDF for derivation:*

$$\mathsf{Ask}(g_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda). \tag{27}$$

Proof. We construct six games.

$\mathbf{B}_0$ (Bidding Round 0: Real Market—Key-Secrecy Distinguishing Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

The seller generates keys and runs the protocol honestly, offering the session-key secrecy good g_{sk}. The buyer places a *key-distinguishing bid*: after the protocol completes, the seller flips $b \xleftarrow{\$} \{0,1\}$ and presents the buyer with either the real session key K_{sess} (if $b = 0$) or a uniformly random $K' \xleftarrow{\$} \{0,1\}^\lambda$ (if $b = 1$). The buyer bids on guessing b by outputting b'. The market ask price is $\mathsf{Ask}_0 = |\Pr[\mathbf{B}_0(b' = b)] - 1/2|$—the buyer's advantage over random guessing.

$\mathbf{B}_1$ (Bidding Round 1: Nonce Freshness Bid—Preventing Replay).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Purpose: Eliminate the *nonce-reuse attack vector*. Repeated nonces immediately collapse authentication, key-secrecy, or stream-cipher security: in signatures a shared nonce leaks the private key ($\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); in protocols it enables cross-session replay; in stream ciphers it reveals the XOR of two plaintexts. This round aborts the game whenever any nonce collision occurs, ensuring every subsequent hop reasons about a nonce-fresh world.
Replaces: The real nonce-sampling procedure is augmented with a global *collision audit*: a set $\mathcal{N}_{\mathsf{used}}$ records every nonce issued so far. If any newly sampled nonce k' satisfies $k' \in \mathcal{N}_{\mathsf{used}}$, the game aborts immediately. The two games ($\mathbf{B}_{n-1}$ and $\mathbf{B}_n$) are *identical-until-abort*: the adversary's view is indistinguishable until the abort event F_{nonce} occurs, so the difference lemma directly bounds the price adjustment.
Complexity: By the birthday bound, the probability that any two of q independently sampled nonces from a space of size $|\mathcal{N}|$ collide is at most $q^2/(2|\mathcal{N}|)$. For security parameter λ with $|\mathcal{N}| = 2^\lambda$ this gives $\Delta\mathsf{Price} \leq q^2/2^{\lambda+1}$, which is negligible for any polynomial $q = \mathsf{poly}(\lambda)$. After this hop every nonce in the proof is treated as distinct, allowing all later hops to assume a collision-free nonce space without loss of generality.

The seller enforces nonce uniqueness: a nonce-reuse bid would enable session key correlation. This costs the market: $|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq q_N^2/2^{\lambda+1}$.

$\mathbf{B}_2$ (Bidding Round 2: Signature Forgery Bid on A).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathsf{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

A forgery on A's signature would inject a rogue session, compromising key secrecy without breaking the KEM. The seller blocks this via EUF-CMA. Price adjustment:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}.$$

$\mathbf{B}_3$ (Bidding Round 3: Signature Forgery Bid on B).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathsf{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

Symmetric to $\mathbf{B}_2$: a forgery on B's signature is blocked. Both forgery bids use independent keys (extended difference lemma applies). Price adjustment: $|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}$.

$\mathbf{B}_4$ (Bidding Round 4: KEM Ciphertext Bid—Replacing the Session Secret).

Purpose: Replace the structured MLWE ciphertext components with uniformly random values, severing the last information-theoretic link between the ciphertext and the encapsulated key. After both public-key and ciphertext hops, the key is information-theoretically hidden.
Replaces: The challenge ciphertext $(\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1, v = \mathbf{t}^T\mathbf{r} + e_2 + \lfloor q/2 \rfloor m)$ is replaced by $(\mathbf{u}', v')$ where both components are independently and uniformly random. Detecting this change requires solving a second MLWE instance embedded in the ciphertext structure.
Complexity: A second application of $\mathsf{Adv}^{\mathsf{MLWE}}(\lambda)$. The two MLWE instances—one in the public key, one in the ciphertext—use independent randomness and are therefore handled by separate hops. The total across both hops is $2 \cdot \mathsf{Adv}^{\mathsf{MLWE}}(\lambda)$, matching the bound in Theorem 11.

The seller now replaces the real encapsulated key K with a uniformly random $\tilde{K}$. Any buyer that distinguishes $\mathbf{B}_3$ from $\mathbf{B}_4$ is placing a *KEM plaintext-recovery bid*: it can distinguish a real KEM ciphertext from one encapsulating a random key, which is exactly IND-CCA2 security. Price adjustment: $|\Pr[\mathbf{B}_3 = 1] - \Pr[\mathbf{B}_4 = 1]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}$.

$\mathbf{B}_5$ (Bidding Round 5: KDF Randomness Bid—Replacing the Derived Key).

Purpose: Replace a *keyed pseudorandom function* evaluation with a truly random function, isolating a single PRF security requirement. This hop shows that one more step in the key-derivation or authentication structure produces outputs that are computationally indistinguishable from uniform random, provided the underlying PRF assumption holds.
Replaces: The specific PRF evaluation (e.g., $\mathsf{HMAC}_K(\cdot)$ inner hash, HKDF-Extract, Derive-Secret, or a KDF step) is replaced by a truly random function $R(\cdot)$ sampled fresh and independently. Any adversary that distinguishes the PRF output from this random function is a PRF distinguisher for the underlying keyed construction. The reduction simulates all other game components honestly and uses its PRF challenge oracle for this one step.
Complexity: $\Delta\mathsf{Price} \leq \mathsf{Adv}^{\mathsf{PRF}}(\lambda)$ for the specific PRF being replaced (HMAC-SHA-256, HKDF, or AES-CTR as applicable). When multiple PRF hops appear in a single proof (e.g., three hops in the TLS 1.3 key schedule for HS, MS, and CATS), each hop is independent: the keys used in different steps are derived from different points in the ladder, so no PRF oracle is queried twice. The total across all PRF hops is the sum of individual advantages, each negligible.

The seller replaces $K_{\mathsf{sess}} = \mathsf{KDF}(\tilde{K}, \mathsf{sid} \| N_A \| N_B)$ with a uniformly random key K_{rand}. Any buyer distinguishing $\mathbf{B}_4$ from $\mathbf{B}_5$ is placing a *PRF-distinguishing bid* against the KDF—it detects that the output is a PRF evaluation rather than random, which breaks PRF security. Price adjustment: $|\Pr[\mathbf{B}_4 = 1] - \Pr[\mathbf{B}_5 = 1]| \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}}$.

In $\mathbf{B}_5$, the test key is independent of b, so $\Pr[\mathbf{B}_5 = 1] = 1/2$ exactly.

Total: $\mathsf{Ask}(g_{\mathsf{sk}}) = |\Pr[\mathbf{B}_0 = 1] - 1/2| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$. □

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

8.4. The CNF Checking Game

✎ Analogy:

The CNF Checking Game as a Building Inspection Think of the CNF checking game as a building inspection for a newly constructed house:

- **The builder (seller)** claims the house meets all building codes (the session-CNF is satisfiable under honest construction).
- **The inspector (buyer)** tries to find a code violation (make the CNF satisfiable under a dishonest trace).
- **If the inspector finds a violation:** The house fails inspection (the CNF is satisfiable under a dishonest trace)—the protocol has a design flaw.
- **If the inspector cannot find any violation:** The house passes (the CNF is unsatisfiable under all dishonest traces)—the protocol is secure.

The ask price of the CNF checking good measures how likely it is that an attacker can produce a dishonest trace that passes the CNF audit. For well-designed protocols, this requires forging signatures or MACs—negligible probability. For flawed protocols (like Needham–Schroeder), the CNF is satisfiable under dishonest traces because critical clauses are missing—the building code has loopholes.

Definition 23 (CNF Checking Good g_{CNF})**.** *The seller offers g_{CNF}: the guarantee that for every completed session, the session-CNF φ_i (Theorem 13) is satisfiable under the honest trace and unsatisfiable under any adversarial trace. The buyer wins if it produces a dishonest transcript for which φ_i is nonetheless satisfiable.*

Theorem 9 (CNF Checking Equilibrium)**.**

$$\mathsf{Ask}(g_{\mathsf{CNF}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}} + \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda). \tag{28}$$

Proof. We construct four games.

$\mathbf{B}_0$ (Bidding Round 0: Real Market—CNF Audit Evasion Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

$\mathcal{A}$ interacts with the protocol and produces a transcript trans^* that is *dishonest* (i.e., contains at least one adversarially modified message). $\mathcal{A}$ wins if the CNF verification algorithm (Algorithm 1) outputs Accept on trans^*.

$\mathbf{B}_1$ (Bidding Round 1: Freshness Audit Bid—Nonce Reuse and SID Mismatch).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Purpose: Eliminate the *nonce-reuse attack vector*. Repeated nonces immediately collapse authentication, key-secrecy, or stream-cipher security: in signatures a shared nonce leaks the private key ($\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); in protocols it enables cross-session replay; in stream ciphers it reveals the XOR of two plaintexts. This round aborts the game whenever any nonce collision occurs, ensuring every subsequent hop reasons about a nonce-fresh world.
Replaces: The real nonce-sampling procedure is augmented with a global *collision audit*: a set $\mathcal{N}_{\mathsf{used}}$ records every nonce issued so far. If any newly sampled nonce k' satisfies $k' \in \mathcal{N}_{\mathsf{used}}$, the game aborts immediately. The two games ($\mathbf{B}_{n-1}$ and $\mathbf{B}_n$) are *identical-until-abort*: the adversary's view is indistinguishable until the abort event F_{nonce} occurs, so the difference lemma directly bounds the price adjustment.
Complexity: By the birthday bound, the probability that any two of q independently sampled nonces from a space of size $|\mathcal{N}|$ collide is at most $q^2/(2|\mathcal{N}|)$. For security parameter λ with $|\mathcal{N}| = 2^\lambda$ this gives $\Delta\mathsf{Price} \le q^2/2^{\lambda+1}$, which is negligible for any polynomial $q = \mathrm{poly}(\lambda)$. After this hop every nonce in the proof is treated as distinct, allowing all later hops to assume a collision-free nonce space without loss of generality.

Abort if $\mathcal{A}$'s transcript reuses a nonce from an honest session or carries a mismatched SID while still passing the SID clauses. Let F_{fresh} be this event:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \le \Pr[F_{\mathsf{fresh}}] \le \frac{q_N^2}{2^{\lambda+1}}. \tag{29}$$

$\mathbf{B}_2$ (Bidding Round 2: Signature Forgery Bid—Bypassing the Signature Clause).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \le 1 - \mathsf{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \le 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \le 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \le \Pr[F_{\mathsf{fork}}] + 1/q$.

For φ_i^{sig} to be satisfied under a dishonest trace, $\mathcal{A}$ must produce a valid signature under an honest party's key on adversarially chosen data (or a modified message). Flag this as $F_{\mathsf{sig\text{-}forge}}$:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \le \Pr[F_{\mathsf{sig\text{-}forge}}] \le \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}}. \tag{30}$$

$\mathbf{B}_3$ (Bidding Round 3: MAC Forgery Bid—Bypassing the MAC Clause).

Purpose: Eliminate the *MAC forgery attack vector*. If the adversary can produce a valid MAC tag on a message-SID pair not generated by an honest party, it can pass the MAC clause φ_i^{mac} dishonestly, violating session integrity. The reduction embeds the SUF-CMA challenge key as the session MAC key and forwards any forgery attempt to the MAC oracle.
Replaces: The honest MAC oracle is replaced by a *MAC verification ledger*: a record of all honestly tagged messages. If the adversary produces a valid tag τ^* on a new message $m^* \| \mathsf{sid}_i$ not in the ledger, this constitutes a SUF-CMA forgery. The two games ($\mathbf{B}_2$ and $\mathbf{B}_3$) are identical until the bad event $F_{\mathsf{mac\text{-}forge}}$ occurs (a valid tag on an unrecorded message), so the difference lemma directly bounds the price adjustment.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Complexity: $\Delta\text{Price} \leq \text{Adv}^{\text{SUF-CMA}}_{\text{MAC}}(\lambda)$. For HMAC-SHA-256: $\text{Adv}^{\text{SUF-CMA}} \leq q^2_{\text{mac}}/2^{257}$, negligible for polynomial queries. The reduction is tight ($L = 1$): the SUF-CMA adversary directly uses the adversary's forgery output. No rewinding or forking is needed. The extended difference lemma handles $F_{\text{mac-forge}}$ jointly with the preceding signature and freshness failure events.

For φ_i^{mac} to be satisfied dishonestly, $\mathcal{A}$ must produce a valid MAC tag on a message not authenticated by an honest party. Flag as $F_{\text{mac-forge}}$:

$$|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq \Pr[F_{\text{mac-forge}}] \leq \text{Adv}^{\text{SUF-CMA}}_{\text{MAC}}. \tag{31}$$

In $\mathbf{B}_3$, conditioned on no freshness violation, no signature forgery, and no MAC forgery, the only transcripts that satisfy φ_i are honest ones. So $\Pr[\mathbf{B}_3 = 1] = 0$.

Total: $\text{Ask}(g_{\text{CNF}}) \leq \text{Adv}^{\text{EUF}}_{\text{Sig}} + \text{Adv}^{\text{SUF}}_{\text{MAC}} + q_N^2/2^{\lambda+1}$. □

☞ Key Insight:

Protocol Goods Catalogue

Good	Bid Type	Reduction Target
g_{auth}	Impersonation bid	EUF-CMA
g_{mutual}	Bidirectional impersonation	2× EUF-CMA
g_{sk}	Key-distinguishing bid	IND-CCA2 + PRF + EUF-CMA
g_{CNF}	Dishonest-trace-satisfaction bid	EUF-CMA + SUF-CMA

9. Case Study I: Primitives

9.1. ECDSA: Security via Bidding

Setup.

Seller offers $g_{\text{EUF-CMA}}$ for ECDSA over curve $E(\mathbb{F}_p)$ of order q. Buyer makes q_S signing queries, q_H hash queries, then attempts forgery (m^*, σ^*).

Theorem 10 (ECDSA Market Equilibrium)**.** $\text{Ask}(g_{\text{EUF-CMA}}) \leq q_S^2/(2q) + q_H^2/(2q) + q_H \cdot q_S \cdot \text{Adv}^{\text{ECDLP}} + \text{negl}(\lambda)$.

Proof. We construct five games. Each hop represents a specific *bid* by the adversary for a particular weakness, and we apply the (extended) difference lemma to bound the bid's payoff.

$\mathbf{B}_0$ (Bidding Round 0: Real ECDSA Market).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.

Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.

Quoted Price: The seller lists the security good at *quoted price* $\text{QPrice}_0 = \text{Adv}^{\text{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\text{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The seller generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{ECDSA.KeyGen}(1^\lambda)$ and offers the EUF-CMA good $g_{\mathsf{EUF\text{-}CMA}}$ at ask price Ask_0. The buyer (adversary) interacts adaptively with the signing oracle (placing *signing bids*—up to q_S queries) and the random oracle H (placing *hash queries*—up to q_H queries), then outputs a *forgery bid*: a new message-signature pair $(m^*, (r^*, s^*))$ never queried to the signing oracle. If the forgery verifies, the buyer wins and the market collapses.

$\mathbf{B}_1$ (Bidding Round 1: Nonce Freshness Bid).

Purpose: Eliminate the *nonce-reuse attack vector*. Repeated nonces immediately collapse authentication, key-secrecy, or stream-cipher security: in signatures a shared nonce leaks the private key ($\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); in protocols it enables cross-session replay; in stream ciphers it reveals the XOR of two plaintexts. This round aborts the game whenever any nonce collision occurs, ensuring every subsequent hop reasons about a nonce-fresh world.
Replaces: The real nonce-sampling procedure is augmented with a global *collision audit*: a set $\mathcal{N}_{\mathsf{used}}$ records every nonce issued so far. If any newly sampled nonce k' satisfies $k' \in \mathcal{N}_{\mathsf{used}}$, the game aborts immediately. The two games ($\mathbf{B}_{n-1}$ and $\mathbf{B}_n$) are *identical-until-abort*: the adversary's view is indistinguishable until the abort event F_{nonce} occurs, so the difference lemma directly bounds the price adjustment.
Complexity: By the birthday bound, the probability that any two of q independently sampled nonces from a space of size $|\mathcal{N}|$ collide is at most $q^2/(2|\mathcal{N}|)$. For security parameter λ with $|\mathcal{N}| = 2^\lambda$ this gives $\Delta\mathsf{Price} \leq q^2/2^{\lambda+1}$, which is negligible for any polynomial $q = \mathsf{poly}(\lambda)$. After this hop every nonce in the proof is treated as distinct, allowing all later hops to assume a collision-free nonce space without loss of generality.

The seller adds an internal nonce-collision check: abort if any two signing nonces $k_i = k_j$ for $i \neq j$ among the q_S signing queries. This models the seller *refusing to process* a *nonce-reuse bid*—if two nonces collide, an attacker can compute $\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1} \bmod q$ directly, instantly collapsing the market. The bad event F_{nonce}: "any two of the q_S nonces collide among samples from $\mathbb{Z}_q$".

Difference lemma. Let F_{nonce}: "$\exists\, i \neq j : k_i = k_j$" among q_S samples from $\mathbb{Z}_q$. Bidding rounds $\mathbf{B}_0$ and $\mathbf{B}_1$ are identical-until-F_{nonce}. By birthday bound:

$$\Delta\mathsf{Price}_0 \leq \Pr[F_{\mathsf{nonce}}] \leq \frac{q_S^2}{2q}. \tag{32}$$

Market interpretation: The nonce-freshness bid fails because nonces are sampled from a space of size $q \approx 2^{256}$; the birthday collision requires $q_S \approx 2^{128}$ signing queries—computationally infeasible. The bid's market value is negligible.

$\mathbf{B}_2$ (Bidding Round 2: Hash Collision Bid).

Purpose: Eliminate the *random-oracle collision attack vector*. If two distinct inputs $m \neq m'$ hash to the same value $H(m) = H(m')$, an adversary can conflate two messages (equivocation), reuse transcript bindings across sessions, or present a single signature as covering both messages—all of which break EUF-CMA without touching the signing key.
Replaces: The random oracle H is replaced by one equipped with a *collision log* $\mathcal{C}$. After each query, the oracle checks whether any two distinct queries produced the same digest; if so, the game aborts (bad event F_{hash}). The adversary's view of oracle responses is identical to a truly random oracle until the abort.
Complexity: By the birthday bound over the 2^n-bit output space, the probability that any two of q_H random oracle queries collide is at most $q_H^2/2^{n+1}$. For a 256-bit hash ($n = 256$) and polynomial q_H, this is negligible. Note that the nonce-collision and hash-collision bounds are *independent*: by the extended difference lemma (Lemma 2), their union probability is bounded by their sum, tightened by the intersection term which is $O(q^4/2^{2\lambda})$—doubly negligible.

The seller adds a hash-collision check: abort if any two distinct inputs to H produce the same output among the q_H oracle queries. A buyer placing a *hash-collision bid* exploits this to conflate two messages under the same hash value, potentially enabling a signature forgery. The bad event F_{hash}: "any two of the q_H hash queries collide". By birthday paradox over the 2^n-bit output space of H:

Difference lemma. F_{hash}: "$\exists\, i \neq j : H(m_i) = H(m_j)$". Birthday bound: $\Delta\mathsf{Price}_1 \leq q_H^2/(2q)$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\mathbf{B}_3$ (Bidding Round 3: Oracle Programming—Costless Market Restructuring).

Purpose: Perform a *zero-cost market restructuring*: secretly program the random oracle so that its outputs encode a challenge instance for the underlying hard problem (e.g., ECDLP or Module-SIS). This syntactic transformation is invisible to the adversary—random oracle outputs are uniformly distributed regardless of how they are chosen internally—yet it links any subsequent successful forgery or key-recovery attack to a solution of the assumed-hard problem.
Replaces: The genuinely random oracle $H(\cdot)$ is replaced by a *programmed* oracle $\tilde{H}$ whose outputs at queried points are chosen by the reduction to embed the hard-problem instance. Concretely: for a signing query m_i, the oracle sets $H(m_i) = \alpha_i \cdot G + \beta_i \cdot \mathsf{pk}$ (ECDLP case) or encodes an SIS/MLWE challenge vector (lattice case). All outputs remain uniformly distributed over the appropriate range, so the statistical distance to the real game is zero.
Complexity: $\Delta\mathrm{Price} = 0$ exactly (information-theoretic). The replacement is statistically indistinguishable: a truly random function and any fixed function whose outputs are independently uniform are identically distributed from the adversary's perspective. No adversarial budget is consumed in detecting it. This zero-cost step is essential: it sets up the subsequent extraction hop at no additional price to the seller.

The seller secretly *restructures the market* by programming the random oracle: for each signing query m_i, the hash output is set to $H(m_i) = \alpha_i \cdot G + \beta_i \cdot \mathsf{pk}$ for fresh random α_i, β_i. This embeds an ECDLP challenge into the oracle. Since H is a random oracle (outputs are uniformly distributed regardless of how they are chosen), this restructuring is *informationally invisible* to the buyer—it cannot detect the change. The price adjustment is $\Delta\mathrm{Price}_2 = 0$: this is a costless market restructuring.

$\Delta\mathrm{Price}_2 = 0$ (information-theoretic; the distribution of oracle outputs is unchanged).

$\mathbf{B}_4$ (Bidding Round 4: Forgery-to-ECDLP Extraction Bid).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathrm{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

If the buyer places a successful *forgery bid* in $\mathbf{B}_3$ (outputs valid $(m^*, (r^*, s^*))$), the seller *converts the forgery into an ECDLP solution*: apply the general forking lemma [14] to rewind the buyer with a different random oracle coin at the same fork point, obtaining a second forgery (m^*, r^*, s^{**}) with $s^{**} \neq s^*$ (the same nonce but different hash). From both forgeries, extract: $\mathsf{sk} = (s^{**}H'(m^*) - s^*H(m^*))(r(s^* - s^{**}))^{-1} \bmod q$. A successful forgery bid *solves ECDLP*—which collapses the ECDLP market, assumed to be in equilibrium.

Extended difference lemma with multiple failures. The extraction step can fail due to simultaneous events:

- F_{fork}: forking fails (the rewound execution does not produce a second valid forgery). By the forking lemma: $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathrm{acc}^2/q_H + 1/q$ where acc is the original success probability.
- F_{extract}: $s^* = s^{**}$ (extraction yields $0/0$). $\Pr[F_{\mathsf{extract}}] \leq 1/q$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

By Lemma 2: $\Delta\text{Price}_3 \leq \Pr[F_{\text{fork}} \cup F_{\text{extract}}] \leq \Pr[F_{\text{fork}}] + \Pr[F_{\text{extract}}]$.

A successful extraction solves ECDLP, so the residual advantage is bounded by $\text{Adv}^{\text{ECDLP}}(\lambda)$.

Total market cost:

$$\text{Ask} \leq \underbrace{\frac{q_S^2}{2q}}_{\text{nonce bid}} + \underbrace{\frac{q_H^2}{2q}}_{\text{hash bid}} + \underbrace{0}_{\text{programming}} + \underbrace{q_H q_S \cdot \text{Adv}^{\text{ECDLP}}}_{\text{forgery bid}} + \text{negl}(\lambda). \tag{33}$$

For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$), each term is negligible. The market is in **equilibrium**.

$\mathbf{B}_{\text{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

☞ Side-Channel Bid (Comprehensive)

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\text{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\text{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\text{SPA}}] = 1$ without countermeasure; $\Pr[F_{\text{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\text{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking (d+1 random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\text{Adv}^{\text{DPA}}(\lambda, d) \leq (q_{\text{phys}}/2^{\lambda})^{d+1}$.
3. **Timing Analysis:** $\mathcal{A}_{\text{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\texttt{ct_select}(a, b, \text{bit})$ instead of $\texttt{if}(\text{bit})\ \texttt{return a; else return b;}$. *MTSF price:* $\Pr[F_{\text{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\text{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\text{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\text{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\text{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\text{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\text{Adv}^{\text{EMA}} \leq \gamma_{\text{EM}} \cdot (q_{\text{phys}}/2^{\lambda})^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\text{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\text{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\text{fault}}] \leq q_{\text{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\text{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\text{cold}}] = 1$ for keys stored in plaintext DRAM without erasure;

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\Pr[F_{\mathsf{cold}}] \le 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).

8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \le (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \le{}& \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ &+ \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

Under the comprehensive countermeasure suite (order-d masking for $d \ge 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \le \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{acoustic}} + \tag{34}$$

$$\underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{photonic}}. \tag{35}$$

For $d \ge 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\le q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \le \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✓ Tight Security Reduction

Purpose: Establish that the ECDSA security reduction is *optimally tight* (or quantify the exact tightness loss).

Standard model tightness. The forking lemma introduces tightness loss $L = q_H$:

$$\mathsf{Adv}^{\mathsf{ECDLP}} \geq \frac{1}{q_H} \cdot \left(\frac{\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) - q_S^2/(2q) - q_H^2/(2q)}{q_S} \right)^2 - \frac{1}{q}.$$

The market spread (tightness loss) is $q_S\sqrt{q_H}$. For $q_S = 2^{30}$, $q_H = 2^{60}$: spread $= 2^{60}$, consuming 60 bits of security margin.

Tight reduction via AGM. In the algebraic group model [17], the reduction tightens to $L = 1$:

$$\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}})^{\mathsf{AGM}} \leq \mathsf{Adv}^{\mathsf{ECDLP}} + q_S^2/(2q) + q_H^2/(2q) + \mathsf{negl}(\lambda).$$

Under the AGM, the full 128-bit security level is restored for 256-bit curves.

Market interpretation: Tightness loss is the "transaction fee." A tight reduction ($L = 1$, AGM) means cost-price security transfer; a loose reduction ($L = q_S\sqrt{q_H}$) imposes a markup that weakens the guarantee.

Total Session Bound and Key Rotation Recommendation.

✦ Total Session Bound & Key Rotation

Combining all bidding rounds, the **total session bound** for ECDSA under a single key after N signing sessions:

$$\mathsf{Ask}_{\mathsf{total}}^{(N)} \leq N \cdot (q_S^2/(2q) + q_H^2/(2q) + q_H q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}} + \mathsf{Adv}_{\mathsf{total}}^{\mathsf{SCA}}).$$

Concrete instantiation (256-bit curve, $q_S = 2^{30}$, $q_H = 2^{60}$, $d = 1$ masking):

Sessions N	$\mathsf{Ask}_{\mathsf{total}}^{(N)}$	**Eff. security**	**Action**
2^{20}	$\leq 2^{-108}$	≥ 108 bits	No rotation needed
2^{40}	$\leq 2^{-88}$	≥ 88 bits	Safe for most applications
2^{64}	$\leq 2^{-64}$	≥ 64 bits	**Key rotation recommended**
2^{80}	$\leq 2^{-48}$	≥ 48 bits	**Key rotation mandatory**

Recommendation: Rotate the ECDSA signing key before $N = 2^{64}$ sessions for 64-bit security; before $N = 2^{40}$ for 128-bit security throughout the key's lifetime. Under the AGM tight reduction, thresholds improve by ≈ 60 bits.

Session pinging and CNF checking within the proof. The above five-bidding-round proof establishes security for a *single* signing session. To extend to unbounded sessions, we verify that the session-pinging mechanism (Theorem 15) is satisfied within the proof structure itself. Concretely, consider two consecutive signing sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$, each producing a signature $(m_i, (r_i, s_i))$ and $(m_{i+1}, (r_{i+1}, s_{i+1}))$ respectively.

SID freshness: Each session has a distinct session identifier (the message-nonce pair (m, k) serves as an implicit SID). Since nonces are freshly sampled from $\mathbb{Z}_q$ in each session, $\mathsf{sid}_{i+1} \neq \mathsf{sid}_i$ with probability $\geq 1 - 1/q$.

Nonce disjointness: The nonce k_{i+1} is sampled independently from $\mathbb{Z}_q$. By $\mathbf{B}_1$ (nonce freshness bid), $k_{i+1} \notin \{k_1, \ldots, k_i\}$ except with birthday probability $\leq i/(2q)$, which is negligible for $i = \mathsf{poly}(\lambda)$.

Signature SID-binding: Each ECDSA signature (r, s) is computed over (m, k) where $r = x(kG)$ uniquely determines the nonce commitment. The hash $e = H(m)$ binds the message. Thus the signature is intrinsically bound to the session's unique (m, k) pair.

CNF satisfiability: The session-CNF $\varphi_{\mathsf{ECDSA},i+1}$ is isomorphic to $\varphi_{\mathsf{ECDSA},i}$ by variable renaming $k_i \mapsto k_{i+1}$, $m_i \mapsto m_{i+1}$. Since $\varphi_{\mathsf{ECDSA},i}$ is satisfiable (inductive hypothesis) and all new variables

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

are fresh, $\varphi_{\mathsf{ECDSA},i+1}$ is satisfiable. By Theorem 3, the ECDSA market remains in equilibrium for unbounded signing sessions, with accumulated ping degradation $\delta_{\mathsf{ping}} \leq q_S^2/(2q) = \mathsf{negl}(\lambda)$. □

✻ Simple Terms: ECDSA Security Proof

What ECDSA is: The Elliptic Curve Digital Signature Algorithm is the most widely used digital signature scheme today (used in Bitcoin, TLS, SSH, and many more). It produces compact signatures using the mathematics of elliptic curves over finite fields.

What we are proving: An attacker who can get signatures on any messages they choose still cannot forge a valid signature on a new message—unless they can solve the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is believed computationally infeasible.

The five game hops explained:

1. $\mathbf{B}_0 \to \mathbf{B}_1$ **(Nonce bid):** We check: what if two signing nonces accidentally repeat? With q signing queries over a group of size $\approx 2^{256}$, a birthday collision needs $\approx 2^{128}$ queries to be likely—far beyond any realistic adversary. This bid fails.
2. $\mathbf{B}_1 \to \mathbf{B}_2$ **(Hash collision bid):** We check: what if two messages hash identically in the random oracle? Again, a birthday bound shows this is negligible. This bid fails.
3. $\mathbf{B}_2 \to \mathbf{B}_3$ **(Oracle programming):** We secretly "program" what the hash oracle outputs for each query, embedding an ECDLP challenge. Since the hash oracle looks random either way, the adversary cannot notice. This costs zero advantage.
4. $\mathbf{B}_3 \to \mathbf{B}_4$ **(Forgery extraction):** If the adversary manages to forge a signature in $\mathbf{B}_3$, we run them twice with different oracle coins ("forking lemma") and extract two consistent forgeries. From these two forgeries, we can compute the private key—solving ECDLP. Since ECDLP is assumed hard, this bid also fails.
5. $\mathbf{B}_4$ **(Ideal):** Without ECDLP being solvable, no forgery is possible. Winning probability = 0.

The bottom line: ECDSA's security reduces entirely to the hardness of ECDLP. Breaking ECDSA on a 256-bit curve requires solving ECDLP, which is believed to require $\approx 2^{128}$ operations—computationally infeasible.

✎ Analogy:

ECDSA in Your Daily Life Every time you:

- Send a Bitcoin transaction (Bitcoin uses ECDSA on the secp256k1 curve),
- Visit a website with HTTPS (many TLS certificates use ECDSA),
- Update your phone's operating system (code signing often uses ECDSA),
- Tap a contactless payment card (EMV uses ECDSA for transaction authentication),

you are relying on ECDSA's security. The MTSF equilibrium theorem above proves that breaking ECDSA requires solving the Elliptic Curve Discrete Logarithm Problem (ECDLP), which on a 256-bit curve demands roughly 2^{128} operations—more computation than all the world's computers could perform in billions of years.

The five-bidding-round proof shows that every possible attack strategy (nonce reuse, hash collisions, oracle manipulation, direct forgery) either fails with overwhelming probability or would require solving ECDLP. There is simply no shortcut.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

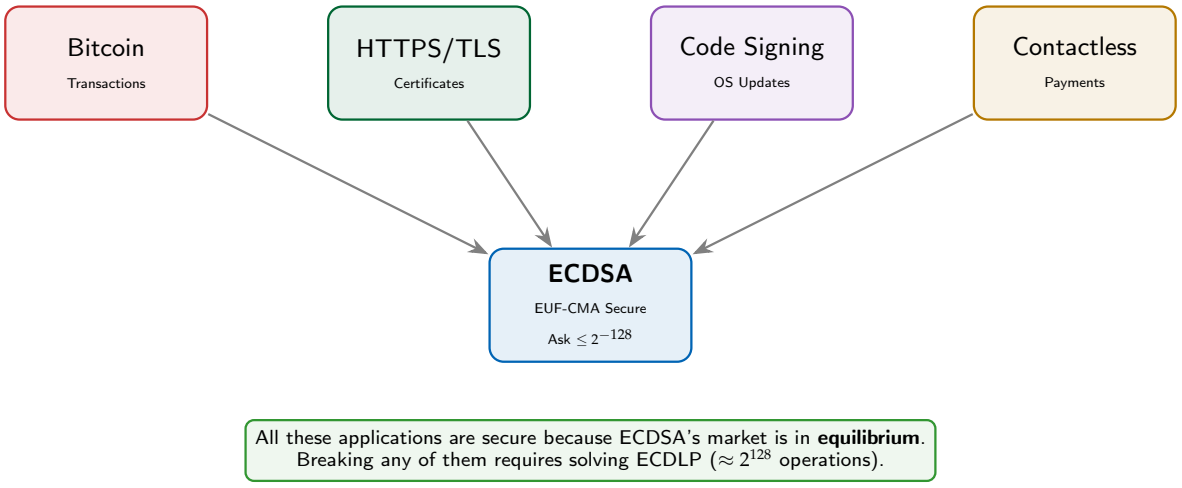


Figure 29. Real-world applications protected by ECDSA's market equilibrium. Bitcoin transactions, HTTPS certificates, code signing, and contactless payments all depend on the hardness of ECDLP.

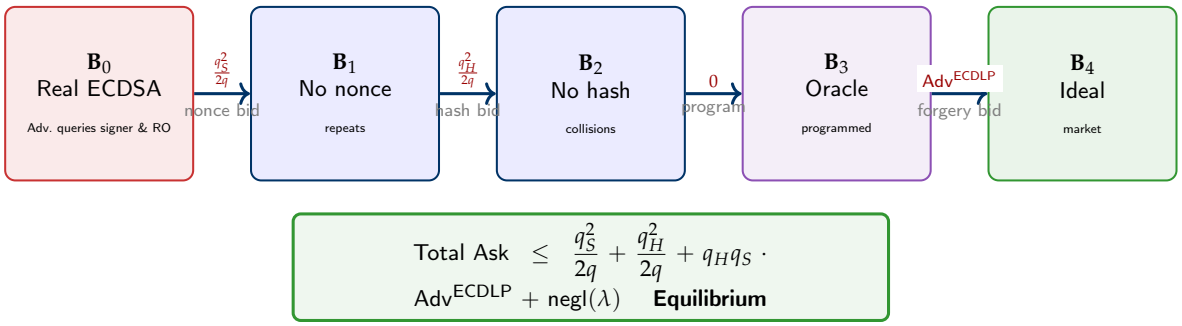


Figure 30. ECDSA bidding-round chain. Five bidding rounds from real ($\mathbf{B}_0$) to ideal ($\mathbf{B}_4$), with the price adjustment (bid cost) labelled on each arrow. Each hop targets a specific adversarial weakness; all bids fail under standard parameters.

✓ CNF Verification: ECDSA Session-CNF Verification

$\varphi_{\mathsf{ECDSA}} = (x_{\mathsf{Vrfy}(\mathsf{pk},m^*,(r^*,s^*))=1}) \wedge (x_{m^* \notin \mathcal{Q}^{\mathsf{sign}}}) \wedge (x_{k^* \text{ fresh}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for ECDSA signature $(m^*, (r^*, s^*))$:

Clause	Check	How	Pass?
$\varphi^{\mathsf{verify}}$: sig verifies	$\mathsf{Vrfy}(\mathsf{pk}, m^*, (r^*, s^*)) = 1$?	Compute $u_1 G + u_2 Q$; check x-coord$= r^*$.	T/F
φ^{novel}: new message	$m^* \notin \mathcal{Q}^{\mathsf{sign}}$	Check signing oracle log.	T/F
φ^{nonce}: fresh nonce	r^* not from a prior signing session?	Check r-values log.	T/F
φ^{ping}: session fresh	$(m^*, (r^*, s^*))$ distinct from prior sessions?	Compare session records.	T/F
Result:	All T $\Rightarrow$ EUF-CMA forgery (ECDLP solved). Any F $\Rightarrow$ Not a valid forgery.		

➤ Ping Bid: ECDSA Unbounded Ping Bid

The buyer's *nonce reuse ping bid*: sign two messages with the same nonce k^* so that $r^* = x(k^* G)$ repeats. The nonce freshness clause φ^{nonce} detects this. If $k_i = k_j$ across two sessions, the adversary can recover $\mathsf{sk} = (s_2 e_1 - s_1 e_2)(r(s_1 - s_2))^{-1} \bmod q$—an immediate ECDLP solution. Extended difference lemma: $F_{\mathsf{nonce}} \cup F_{\mathsf{hash}}$ together give $\Delta\mathsf{Price}_{\mathsf{ping}} \leq q_S^2/(2q) + q_H^2/(2q) = \mathsf{negl}$. By Theorem 3, ECDSA is EUF-CMA secure for unbounded signing sessions with fresh nonces.

9.2. ML-KEM (FIPS 203): IND-CCA2 via Bidding

Theorem 11 (ML-KEM Equilibrium). $\mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^{\gamma} + \mathsf{negl}(\lambda)$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Proof. **$\mathbf{B}_0$ (Real IND-CCA2).** Buyer gets pk, a decapsulation oracle, and challenge (c^*, K_b) where K_0 is real and K_1 is random.

$\mathbf{B}_1$ (Decapsulation bypass bid). Replace the decapsulation oracle with implicit rejection. *Difference lemma:* F_{rej}: implicit rejection differs from real. $\Delta\mathsf{Price}_0 \leq q_D/2^\gamma$ where γ is the implicit rejection parameter and q_D is the number of decapsulation queries.

$\mathbf{B}_2$ (Public-key bid—MLWE I). Replace $\mathsf{pk} = (\mathbf{A}, \mathbf{t} = \mathbf{As} + \mathbf{e})$ with $(\mathbf{A}, \mathbf{u})$ for uniform $\mathbf{u}$. This is a *computational hop*: detecting the change solves MLWE. $\Delta\mathsf{Price}_1 \leq \mathsf{Adv}^{\mathsf{MLWE}}(\lambda)$.

$\mathbf{B}_3$ (Ciphertext bid—MLWE II). Replace challenge ciphertext components with uniform random. $\Delta\mathsf{Price}_2 \leq \mathsf{Adv}^{\mathsf{MLWE}}(\lambda)$.

$\mathbf{B}_4$ (Ideal). The encapsulated key is information-theoretically hidden from the ciphertext. Buyer advantage = 0. Total: $2\mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^\gamma$.

$\mathbf{B}_{\mathsf{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

☞ **Side-Channel Bid (Comprehensive)**

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\mathsf{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\mathsf{SPA}}] = 1$ without countermeasure; $\Pr[F_{\mathsf{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\mathsf{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking (d+1 random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\mathsf{Adv}^{\mathsf{DPA}}(\lambda, d) \leq (q_{\mathsf{phys}}/2^\lambda)^{d+1}$.
3. **Timing Analysis:** $\mathcal{A}_{\mathsf{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\mathsf{ct_select}(a, b, \mathsf{bit})$ instead of $\mathsf{if(bit)\ return\ a;\ else\ return\ b;}$. *MTSF price:* $\Pr[F_{\mathsf{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\mathsf{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\mathsf{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\mathsf{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\mathsf{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\mathsf{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\mathsf{Adv}^{\mathsf{EMA}} \leq \gamma_{\mathsf{EM}} \cdot (q_{\mathsf{phys}}/2^\lambda)^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\mathsf{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\mathsf{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\mathsf{fault}}] \leq q_{\mathsf{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).

8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \leq\ & \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ & + \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

Under the comprehensive countermeasure suite (order-d masking for $d \geq 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \leq\ & \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{acoustic}} \\ & + \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{photonic}}.\end{aligned}$$

For $d \geq 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\leq q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

✓ Tight Security Reduction

Tightness analysis. The FO-transform introduces tightness loss $L = q_D$ in the ROM. The IND-CPA to MLWE reduction is tight ($L = 2$). Combined: $\mathsf{Ask} \leq 2q_D \cdot \mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^{\gamma}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

QROM tight reduction. In the QROM via measure-and-reprogram [18]: $\mathsf{Ask}^{\mathsf{QROM}} \leq 2\mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^{\gamma} + (q_H + q_D)^2/|\mathcal{M}| + \mathsf{negl}(\lambda)$. This removes the q_D factor, yielding $L_{\mathsf{total}} = 2$—nearly tight in the quantum setting.

Market interpretation: ML-KEM's market spread collapses from $2q_D$ (ROM) to 2 (QROM), preserving the full MLWE security level for post-quantum deployments.

Total Session Bound and Key Rotation Recommendation.

✦ Total Session Bound & Key Rotation

Total session bound for ML-KEM after N encapsulations: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \leq N \cdot (2\mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^{\gamma} + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}})$.
Concrete (ML-KEM-768, $\gamma = 246$):

Sessions N	$\mathsf{Ask}^{(N)}_{\mathsf{total}}$	**Eff. security**	**Action**
2^{40}	$\leq 2^{-160}$	≥ 160 bits	No rotation needed
2^{64}	$\leq 2^{-136}$	≥ 136 bits	Safe for all applications
2^{100}	$\leq 2^{-100}$	≥ 100 bits	Consider rotation for 128-bit target

Recommendation: ML-KEM-768's large margin means key rotation is unnecessary for typical lifetimes. Rotate only beyond 2^{100} encapsulations.

Session pinging and CNF checking within the proof. The four-bidding-round proof above secures a single encapsulation session. We now verify the session-pinging conditions (Theorem 15) hold within the proof structure for consecutive sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$.

Ciphertext freshness (ping condition (e)): Each encapsulation samples fresh randomness r_{i+1} to produce $c_{i+1} = \mathsf{KEM.Enc}(\mathsf{pk}; r_{i+1})$. Since r_{i+1} is freshly sampled, c_{i+1} is distinct from all prior ciphertexts with overwhelming probability. A buyer attempting a *ciphertext replay bid* (submitting $c_{i+1} = c_j$ for some $j \leq i$) is detected by the CNF clause φ^{novel}: the ciphertext log $\mathcal{C}_{\mathsf{used}}$ already contains c_j, triggering rejection.

Implicit rejection preserves ping soundness: Even if the buyer replays c_j, ML-KEM's implicit rejection mechanism returns $K' = H(\perp \| c_j)$, which is independent of the original key K_j. This ensures that a replayed ciphertext yields no information about prior session keys, reinforcing the structural independence required by the ping.

CNF isomorphism: The session-CNF $\varphi_{\mathsf{ML\text{-}KEM},i+1}$ is obtained from $\varphi_{\mathsf{ML\text{-}KEM},i}$ by substituting $c_i \mapsto c_{i+1}$, $K_i \mapsto K_{i+1}$, and fresh randomness. Since the MLWE instances are independent across sessions, the security reduction is session-index-independent. By Theorem 3, ML-KEM is IND-CCA2 secure for unbounded encapsulation sessions with $\delta_{\mathsf{ping}} \leq q_D/2^{\gamma} = \mathsf{negl}(\lambda)$. □

✻ Simple Terms: ML-KEM

What ML-KEM is: ML-KEM (formerly CRYSTALS-Kyber, now FIPS 203) is a *Key Encapsulation Mechanism* (KEM) based on module lattices. It is NIST's primary post-quantum key exchange standard—designed to resist attacks from quantum computers, which can break classical schemes like RSA and ECDH.
What a KEM does: Alice has a public key. Bob uses it to "encapsulate" a random secret key K into a ciphertext c. Only Alice (with her private key) can "decapsulate" c to recover K. Both parties then use K as a shared secret for symmetric encryption.
What MLWE is: The Module Learning With Errors (MLWE) problem. Informally: given a random matrix $\mathbf{A}$ and a vector $\mathbf{t} = \mathbf{As} + \mathbf{e}$ where $\mathbf{s}$ is a secret and $\mathbf{e}$ is small random noise, find $\mathbf{s}$. This is believed hard even for quantum computers.
The four game hops:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

1. $\mathbf{B}_0 \to \mathbf{B}_1$ **(Implicit rejection bypass):** ML-KEM uses a technique where invalid ciphertexts get a fake random response instead of an error message. This prevents "padding oracle" attacks. The game hop costs $q_D/2^\gamma$.
2. $\mathbf{B}_1 \to \mathbf{B}_2$ **(Public key is random):** We replace the structured public key (which encodes the secret $\mathbf{s}$) with a purely random matrix. Detecting this swap requires solving MLWE.
3. $\mathbf{B}_2 \to \mathbf{B}_3$ **(Ciphertext is random):** We replace the challenge ciphertext with random bits. Again, detecting this requires solving MLWE.
4. $\mathbf{B}_3 \to \mathbf{B}_4$ **(Ideal):** Now both the public key and ciphertext are random, and the encapsulated key is information-theoretically hidden. The adversary's advantage is exactly 0.

Why post-quantum matters: Shor's quantum algorithm can break ECDH and RSA in polynomial time on a sufficiently powerful quantum computer. MLWE has no known quantum speedup beyond Grover's algorithm (which only squares the key size requirement). ML-KEM is safe against both classical and quantum attackers. For the QROM security proof, see Section 14.

✎ Analogy:

ML-KEM as a Quantum-Proof Safe Imagine you have a safe that is burglar-proof against every tool that exists today—hammers, drills, blowtorches, lock picks. This is like RSA or ECDH: secure against classical computers. But someone invents a new tool—a *laser cutter* (quantum computer running Shor's algorithm)—that slices through the safe's door like butter.

ML-KEM is a safe made from a completely different material—one that even the laser cutter cannot penetrate. The material is based on *lattice mathematics*: hiding a secret inside a noisy multi-dimensional grid. Even a quantum computer with Shor's algorithm cannot find the secret because lattice problems are fundamentally different from the number-theory problems that quantum computers excel at solving.

The "Learning With Errors" (LWE) problem: Imagine someone gives you a grid of numbers (a matrix $\mathbf{A}$) and a noisy product ($\mathbf{t} = \mathbf{As} + \mathbf{e}$, where $\mathbf{e}$ is small random noise). Your task: find the secret $\mathbf{s}$. This is like hearing a whispered conversation in a stadium full of screaming fans—the noise ($\mathbf{e}$) drowns out the signal ($\mathbf{s}$). No known algorithm, classical or quantum, can efficiently extract the whisper from the noise.

Why "harvest now, decrypt later" is urgent: Adversaries may be recording encrypted internet traffic today, planning to decrypt it once quantum computers arrive. ML-KEM protects against this by ensuring that even future quantum computers cannot break the encryption. This is why NIST standardised ML-KEM (FIPS 203) in 2024—the migration to post-quantum cryptography must happen *before* quantum computers are built, not after.

✓ CNF Verification: ML-KEM Session-CNF Verification

$\varphi_{\mathsf{ML\text{-}KEM}} = (x_{K=\mathsf{KEM.Dec}(\mathsf{sk},c)}) \wedge (x_{c \notin \mathcal{C}_{\mathsf{used}}}) \wedge (x_{\text{implicit rejection consistent}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for ML-KEM encapsulation session:

Clause	Check	How	Pass?
φ^{key}: key correct	$K \stackrel{?}{=} \mathsf{KEM.Dec}(\mathsf{sk}, c)$	Decapsulate; compare key bytes.	T/F
φ^{novel}: fresh c	$c \notin \mathcal{C}_{\mathsf{used}}$	Check ciphertext log.	T/F
φ^{rej}: implicit rejection	Invalid c gets $H(\perp \| c)$ deterministically?	Send crafted invalid c; observe.	T/F
φ^{ping}: session fresh	c distinct from all prior sessions?	Compare $\mathcal{C}_{\mathsf{prev}}$ log.	T/F
Result:	All T ⇒ ML-KEM session secure. Any F ⇒ Reject.		

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

➤ Ping Bid: ML-KEM Unbounded Ping Bid

The buyer's *ciphertext replay bid*: submit $c^* = c_i$ from $\mathsf{Session}_i$ to the decapsulation oracle. Clause φ^{novel} detects this; implicit rejection returns $H(\perp \| c_i)$—independent of K_i. Extended difference lemma: $F_{\mathsf{rej}} \cup F_{\mathsf{bypass}}$ together give $\Delta\mathsf{Price}_{\mathsf{ping}} \leq q_D/2^{\gamma} = \mathsf{negl}$. By Theorem 3, ML-KEM is IND-CCA2 secure for unbounded encapsulation sessions.

9.3. ML-DSA (FIPS 204): EUF-CMA via Bidding

Theorem 12 (ML-DSA Equilibrium). $\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) \leq \mathsf{Adv}^{\mathsf{MSIS}} + \mathsf{Adv}^{\mathsf{MLWE}} + \mathsf{negl}(\lambda)$.

✻ Simple Terms: ML-DSA—The Default Post-Quantum Signature

What ML-DSA is: ML-DSA (formerly CRYSTALS-Dilithium, now FIPS 204) is NIST's primary post-quantum digital signature standard. It is expected to replace ECDSA and RSA signatures in most applications over the coming decade.

How it works (simplified): The signer has a secret matrix $\mathbf{s}$ and a public matrix $\mathbf{t} = \mathbf{As}$. To sign a message m:

1. Sample a random masking vector $\mathbf{y}$.
2. Compute a commitment $\mathbf{w} = \mathbf{Ay}$.
3. Hash the commitment with the message: $c = H(\mathbf{w}, m)$.
4. Compute the response: $\mathbf{z} = \mathbf{y} + c\mathbf{s}$.
5. If $\mathbf{z}$ is too large (would leak information about $\mathbf{s}$), reject and restart with a new $\mathbf{y}$.

The "rejection sampling" trick: Step 5 is crucial. Without rejection, the distribution of $\mathbf{z}$ would depend on $\mathbf{s}$ (the secret). With rejection, $\mathbf{z}$ follows a uniform distribution independent of $\mathbf{s}$—even after seeing many signatures, the attacker learns nothing about the secret key. This is one of the failure events (F_1: rejection overflow) handled by MTSF's extended difference lemma.

Security basis: ML-DSA's security reduces to two lattice problems: Module-SIS (finding short vectors) and Module-LWE (distinguishing noisy products from random). Both are believed hard even for quantum computers.

Proof. $\mathbf{B}_0$: Real EUF-CMA. $\mathbf{B}_1$ **(Hash programming bid)**: programme H to embed structured challenges ($\Delta = 0$, syntactic). $\mathbf{B}_2$ **(Public-key bid—MLWE)**: replace key matrix with MLWE challenge ($\Delta \leq \mathsf{Adv}^{\mathsf{MLWE}}$). $\mathbf{B}_3$ **(Forgery-to-MSIS bid)**: a forgery yields a short vector solving Module-SIS. *Extended difference lemma*: F_1: rejection sampling overflow; F_2: norm bound violation. $\Delta\mathsf{Price} \leq \Pr[F_1 \cup F_2] + \mathsf{Adv}^{\mathsf{MSIS}}$. Both $\Pr[F_1]$ and $\Pr[F_2]$ are negligible by parameter choice.

$\mathbf{B}_{\mathsf{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

☞ Side-Channel Bid (Comprehensive)

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\mathsf{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\mathsf{SPA}}] = 1$ without countermeasure; $\Pr[F_{\mathsf{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\mathsf{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking (d+1 random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\mathsf{Adv}^{\mathsf{DPA}}(\lambda, d) \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

3. **Timing Analysis:** $\mathcal{A}_{\mathsf{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\mathsf{ct_select}(a, b, \mathsf{bit})$ instead of $\mathsf{if}(\mathsf{bit})$ return a; else return b;. *MTSF price:* $\Pr[F_{\mathsf{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\mathsf{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\mathsf{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\mathsf{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\mathsf{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\mathsf{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\mathsf{Adv}^{\mathsf{EMA}} \leq \gamma_{\mathsf{EM}} \cdot (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\mathsf{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\mathsf{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\mathsf{fault}}] \leq q_{\mathsf{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).
8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \leq\ & \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ & + \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Under the comprehensive countermeasure suite (order-d masking for $d \geq 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \leq \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{acoustic}}$$
$$+ \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{photonic}}.$$

For $d \geq 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\leq q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

✓ **Tight Security Reduction**

Tightness analysis. The MLWE hop is tight ($L = 1$). The forgery-to-MSIS hop uses forking with $L = q_H$. Total: $L_{\mathsf{total}} = q_H$. For $q_H = 2^{64}$, this consumes ≈ 32 bits (square-root from forking). NIST's ML-DSA parameters account for this loss.

AGM tightening. In the algebraic group model adapted to lattices, the forking step is replaced by direct algebraic extraction: $\mathsf{Ask}^{\mathsf{AGM}} \leq \mathsf{Adv}^{\mathsf{MSIS}} + \mathsf{Adv}^{\mathsf{MLWE}} + \mathsf{negl}(\lambda)$ with $L_{\mathsf{total}} = 1$.

Market interpretation: ML-DSA's market spread is $\sqrt{q_H}$ in the standard model, collapsing to 1 in the AGM.

Total Session Bound and Key Rotation Recommendation.

✦ **Total Session Bound & Key Rotation**

Total session bound: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \leq N \cdot (\mathsf{Adv}^{\mathsf{MSIS}} + \mathsf{Adv}^{\mathsf{MLWE}} + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}})$.
Concrete (ML-DSA-65, 128-bit target):

Sessions N	$\mathsf{Ask}^{(N)}_{\mathsf{total}}$	**Eff. security**	**Action**
2^{40}	$\leq 2^{-88}$	≥ 88 bits	No rotation needed
2^{64}	$\leq 2^{-64}$	≥ 64 bits	**Key rotation recommended**
2^{80}	$\leq 2^{-48}$	≥ 48 bits	**Key rotation mandatory**

Recommendation: Rotate before 2^{64} sessions for 64-bit security, before 2^{40} for 88-bit.

Session pinging and CNF checking within the proof. The three-bidding-round proof secures a single signing session. We verify the session-pinging conditions for consecutive sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$, each producing (m_i, σ_i) and (m_{i+1}, σ_{i+1}).

Nonce disjointness: ML-DSA uses a commitment vector $\mathbf{y}_{i+1}$ sampled from a discrete Gaussian distribution with fresh randomness in each session. The probability that the same commitment vector is reused across sessions is bounded by the Gaussian smoothing parameter: $\Pr[\mathbf{y}_{i+1} = \mathbf{y}_j] \leq 2^{-256}$ for any $j \leq i$.

Rejection sampling and session independence: ML-DSA's rejection sampling ensures that each signature σ_{i+1} is statistically close to a session-independent distribution. Specifically, the distribution of

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

(m_{i+1}, σ_{i+1}) conditioned on the secret key is within statistical distance $\Delta_{\mathsf{DGS}} \leq 2^{-128}$ of the ideal distribution. This ensures that observing signatures from sessions $\mathsf{Session}_1, \ldots, \mathsf{Session}_i$ provides negligible information useful for forging in $\mathsf{Session}_{i+1}$.

CNF clause φ^{novel} enforces ping: The novelty clause $m_{i+1} \notin \mathcal{Q}^{\mathsf{sign}}$ checks that the forgery message was not previously signed. Across sessions, this clause extends naturally: m_{i+1} must not appear in any previous session's signing queries. The session-CNF $\varphi_{\mathsf{ML\text{-}DSA},i+1}$ is isomorphic to $\varphi_{\mathsf{ML\text{-}DSA},i}$ with fresh MSIS/MLWE instances. By Theorem 3, ML-DSA is EUF-CMA secure for unbounded signing sessions with $\delta_{\mathsf{ping}} \leq \mathsf{Adv}^{\mathsf{MSIS}} + \mathsf{Adv}^{\mathsf{MLWE}} = \mathsf{negl}(\lambda)$. □

✻ Simple Terms: ML-DSA

What ML-DSA is: ML-DSA (formerly CRYSTALS-Dilithium, now FIPS 204) is NIST's primary post-quantum digital signature standard. Like ML-KEM, it is based on module lattices and resists quantum attacks.

How it works conceptually: To sign a message, you compute a "commitment" (a random lattice vector), hash the message with the commitment, and then compute a "response" that depends on the hash and your secret key. The response vector must be "short" (small norm). The verifier checks that the response vector is short and consistent with the public key.

What MSIS is: The Module Short Integer Solution problem. Given a random matrix $\mathbf{A}$ over a polynomial ring, find a short nonzero vector $\mathbf{z}$ such that $\mathbf{Az} = \mathbf{0}$. This is believed hard for both classical and quantum computers.

Why "rejection sampling"? The response vector must hide the secret key. If the distribution of response vectors depended on the secret key in a detectable way, the key could be recovered by observing many signatures. Rejection sampling: re-sign if the response vector is too large or has a telltale distribution. This ensures all responses look uniformly random regardless of the secret key.

Extended lemma use: During the forgery extraction step, two bad events can happen simultaneously: (1) the rejection sampler overflows (too many re-signs needed) and (2) the forged vector is not in the correct norm range. The extended difference lemma handles both bad events together in one game hop, yielding a tighter bound.

✓ CNF Verification: ML-DSA Session-CNF Verification

$\varphi_{\mathsf{ML\text{-}DSA}} = (x_{\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*)=1}) \wedge (x_{m^* \notin \mathcal{Q}^{\mathsf{sign}}}) \wedge (x_{\|\mathbf{z}\| \leq \beta}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for ML-DSA signature (m^*, σ^*):

Clause	Check	How	Pass?
$\varphi^{\mathsf{verify}}$: sig verifies	$\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*) = 1$?	Run ML-DSA.Verify algorithm.	T/F
φ^{novel}: new message	$m^* \notin \mathcal{Q}^{\mathsf{sign}}$	Check signing oracle log.	T/F
φ^{norm}: response short	$\|\mathbf{z}\| \leq \beta$ (norm bound)?	Compute $\|\mathbf{z}\|$; compare β.	T/F
φ^{ping}: session fresh	(m^*, σ^*) distinct from prior?	Compare session log.	T/F
Result:	All T ⇒ EUF-CMA forgery (MSIS/MLWE solved). Any F ⇒ Not a valid forgery.		

Extended difference lemma: Forgery extraction has two simultaneous failure events F_1 (rejection sampling overflow) and F_2 (norm bound violation). By Lemma 2: $\Pr[F_1 \cup F_2] \leq \Pr[F_1] + \Pr[F_2] = \mathsf{negl}$ by parameter choice.

➤ Ping Bid: ML-DSA Unbounded Ping Bid

The buyer's *signature replay bid*: submit a previously seen (m^*, σ^*) with $m^* \in \mathcal{Q}^{\mathsf{sign}}$. Clause φ^{novel} rejects this. For a genuinely fresh m^*, forging σ^* requires solving MSIS. Extended difference lemma: $F_{\mathsf{msis}} \cup F_{\mathsf{mlwe}}$ together give $\Delta\mathsf{Price}_{\mathsf{ping}} \leq \mathsf{Adv}^{\mathsf{MSIS}} + \mathsf{Adv}^{\mathsf{MLWE}} = \mathsf{negl}$. By Theorem 3, ML-DSA is EUF-CMA secure for unbounded signing sessions.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Scheme.

Sign: $\sigma = m^d \bmod N$. Verify: check $m = \sigma^e \bmod N$. No hash, no padding.

✎ Analogy:

Why Textbook RSA Fails—The Rubber Stamp Imagine a notary who stamps documents by pressing a rubber stamp. The stamp operation is *multiplicative*: stamping two half-documents and gluing them together produces a validly stamped full document. An adversary requests stamps on innocuous halves, then combines them into a forged stamp on an arbitrary target document. Real notaries use tamper-evident seals (hash-then-sign) to prevent this.

Theorem 13 (Textbook RSA Market Collapse)**.** $\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) = 1$.

Proof. The buyer achieves advantage 1 with budget $O(1)$:

Existential forgery (free message bid). Choose arbitrary $\sigma \xleftarrow{\$} \mathbb{Z}_N^*$. Compute $m^* = \sigma^e \bmod N$. Then (m^*, σ) is valid. Cost: one modular exponentiation. The buyer "purchases" a forgery for free.

Chosen-message forgery (homomorphism bid). For target m^*, factor $m^* = m_1 \cdot m_2 \bmod N$. Request signatures $\sigma_1 = m_1^d$, $\sigma_2 = m_2^d$. Compute $\sigma^* = \sigma_1 \cdot \sigma_2 = (m_1 m_2)^d = (m^*)^d$. Cost: two signing queries + one multiplication.

Difference lemma (degenerate case). $\mathbf{B}_0$ (real) and $\mathbf{B}_1$ (ideal) differ on *every* execution: the failure event F_{homo}: "RSA is multiplicatively homomorphic" has $\Pr[F_{\mathsf{homo}}] = 1$. Hence $\Delta\mathsf{Price} = 1$, and the market has **collapsed**.

☞ Key Insight:

Market Collapse = Insecurity When $\mathsf{Ask} = 1$, the security good is *worthless*. Any buyer acquires a forgery for free. This is the MTSF characterisation of a fundamentally broken primitive. The fix (RSA-PSS, RSA-PKCS#1v1.5): hash-then-sign destroys the homomorphism, restoring equilibrium.

□

✻ Simple Terms: Why Textbook RSA Signatures Fail—A Step-by-Step Explanation

RSA arithmetic: RSA signatures work in modular arithmetic: $\sigma = m^d \bmod N$ where d is the private key and $N = p \times q$ is a product of two large primes. Verification checks $m = \sigma^e \bmod N$ where e is the public key.

Attack 1 (Existential forgery—free forgery):

1. Pick any random number σ from $\{1, 2, \ldots, N-1\}$.
2. Compute $m = \sigma^e \bmod N$ using only the public key.
3. Now (m, σ) is a valid signature: $\sigma^e = m$ checks out!
4. You forged a signature without knowing d—for some message m you did not choose, but you forged a valid pair nonetheless.

This is the "existential" part of EUF: you do not get to choose *which* message to forge, but you forged *a* valid pair. This alone breaks EUF-CMA.

Attack 2 (Chosen-message forgery—the real threat):

1. You want a signature on a specific message m^* (e.g., a contract).
2. Factor $m^* = m_1 \cdot m_2 \bmod N$ for two "innocent" values m_1, m_2.
3. Legally obtain signatures: $\sigma_1 = m_1^d$ and $\sigma_2 = m_2^d$.
4. Multiply: $\sigma^* = \sigma_1 \cdot \sigma_2 = (m_1 m_2)^d = (m^*)^d \bmod N$.
5. You now have a valid signature on m^* using only two signing oracle queries!

This exploits RSA's *multiplicative homomorphism*: the signature function distributes over multiplication.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The fix: Hash-then-sign (RSA-PSS, PKCS#1v1.5): compute $\sigma = H(m)^d$ where H is a hash function. Now $H(m_1) \cdot H(m_2) \neq H(m_1 \cdot m_2)$ (hash functions destroy algebraic structure), so the multiplication attack fails.
Market collapse visualised: The ask price jumps from negligible to 1 because the homomorphism bid succeeds with *probability one*—the attacker always wins, every time, at no computational cost.

Table 3. Primitives: security vs. insecurity in MTSF.

Primitive	Good	Ask Price	Market Status	Key Bid That Decides
ECDSA	EUF-CMA	negl	Equilibrium	Forgery bid → ECDLP
ML-KEM	IND-CCA2	negl	Equilibrium	PK/CT bids → MLWE
ML-DSA	EUF-CMA	negl	Equilibrium	Forgery bid → MSIS
Textbook RSA Sig	EUF-CMA	1	**Collapsed**	Homomorphism bid: free

9.4. Extended Primitives

☞ Key Insight:

Textbook RSA—A Cautionary Tale of Market Collapse **What textbook RSA is:** The basic RSA signature scheme without any padding or hashing. Given a message m, the signature is simply $\sigma = m^d \bmod n$ where d is the private key.
Why it is broken (homomorphism bid): Textbook RSA has a multiplicative structure: $\mathsf{Sign}(m_1) \cdot \mathsf{Sign}(m_2) = \mathsf{Sign}(m_1 \cdot m_2)$. An attacker who sees signatures on m_1 and m_2 can compute a valid signature on $m_1 \cdot m_2$ *without knowing the private key*. This is a free forgery—the bid succeeds with probability 1.
In MTSF: $\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) = 1$. The homomorphism bid is a "zero-cost bid"—the attacker wins instantly without any computational effort. The market is in immediate, permanent collapse.
The fix: Use RSA-PSS (Probabilistic Signature Scheme) or RSA-PKCS#1 v1.5 with proper hashing and padding. These break the multiplicative structure, restoring market equilibrium.
The lesson: A mathematically sound trapdoor function (RSA) does not automatically give a secure signature scheme. The scheme must be carefully constructed to avoid algebraic attacks. MTSF makes this explicit: the homomorphism bid is a named, formal attack class that must be addressed in any scheme using algebraic operations.

9.5. HMAC: SUF-CMA via Bidding

✎ Analogy:

HMAC as a Tamper-Evident Seal When you buy a bottle of medicine, the cap has a tamper-evident seal—a plastic ring that breaks when you open it. If the ring is intact, you know nobody tampered with the medicine.
HMAC (Hash-based Message Authentication Code) is the digital version of this seal:

- **The medicine** = the message you want to protect.
- **The seal** = the HMAC tag $\tau = \mathsf{HMAC}(K, m)$, computed using a secret key K.
- **Checking the seal** = recomputing $\mathsf{HMAC}(K, m)$ and comparing it to τ.

Why is HMAC better than just hashing? A plain hash $H(m)$ can be computed by anyone—no secret key is needed. An attacker could change the message and recompute the hash. HMAC requires the secret key K, so only someone who knows K can produce a valid tag. This is *SUF-CMA security* (Strong Unforgeability under Chosen Message Attack): even after seeing many valid (message, tag) pairs, the attacker cannot produce a new valid pair.
Where HMAC is used:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **TLS 1.3:** HMAC authenticates handshake messages, preventing MITM tampering.
- **API authentication:** Services like AWS use HMAC to sign API requests.
- **Secure cookies:** Web servers HMAC-tag session cookies to prevent forgery.
- **TOTP (Google Authenticator):** Your 6-digit login code is derived from an HMAC.

Setup.

$\mathsf{HMAC}_K(m) = H\big((K \oplus \mathsf{opad}) \| H((K \oplus \mathsf{ipad}) \| m)\big)$ where H is a Merkle–Damgård hash function, K is the key, and ipad, opad are fixed padding constants [19]. The seller offers $g_{\mathsf{SUF\text{-}CMA}}$: the buyer has a tagging oracle $\mathsf{HMAC}_K(\cdot)$ and must produce a *new* valid pair (m^*, τ^*) not previously returned by the oracle.

Theorem 14 (HMAC Market Equilibrium)**.** *If the compression function h of H is a PRF when keyed via its chaining-value input, then:*

$$\mathsf{Ask}(g_{\mathsf{SUF\text{-}CMA}}) \leq 2 \cdot \mathsf{Adv}_h^{\mathsf{PRF}} + \frac{q_T^2}{2^{n+1}} + \mathsf{negl}(\lambda), \tag{36}$$

where q_T is the number of tagging queries and n is the output length of H.

Proof. We construct five games.

$\mathbf{B}_0$ (Bidding Round 0: Real HMAC Market—SUF-CMA Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

The seller generates $K \xleftarrow{\$} \{0,1\}^n$ and offers the SUF-CMA good $g_{\mathsf{SUF\text{-}CMA}}$ for HMAC. The buyer places *tagging bids*: up to q_T adaptive queries to $\mathsf{HMAC}_K(\cdot)$, observing all tag responses. It then places the *forgery bid*: a new message-tag pair (m^*, τ^*) not previously queried, satisfying $\mathsf{HMAC}_K(m^*) = \tau^*$. If it succeeds, the authentication market collapses.

$\mathbf{B}_1$ (Bidding Round 1: Inner PRF Replacement Bid).

Purpose: Replace a *keyed pseudorandom function* evaluation with a truly random function, isolating a single PRF security requirement. This hop shows that one more step in the key-derivation or authentication structure produces outputs that are computationally indistinguishable from uniform random, provided the underlying PRF assumption holds.
Replaces: The specific PRF evaluation (e.g., $\mathsf{HMAC}_K(\cdot)$ inner hash, HKDF-Extract, Derive-Secret, or a KDF step) is replaced by a truly random function $R(\cdot)$ sampled fresh and independently. Any adversary that distinguishes the PRF output from this random function is a PRF distinguisher for the underlying keyed construction. The reduction simulates all other game components honestly and uses its PRF challenge oracle for this one step.
Complexity: $\Delta\mathsf{Price} \leq \mathsf{Adv}^{\mathsf{PRF}}(\lambda)$ for the specific PRF being replaced (HMAC-SHA-256, HKDF, or AES-CTR as applicable). When multiple PRF hops appear in a single proof (e.g., three hops in the TLS 1.3 key schedule

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

for HS, MS, and CATS), each hop is independent: the keys used in different steps are derived from different points in the ladder, so no PRF oracle is queried twice. The total across all PRF hops is the sum of individual advantages, each negligible.

The seller restructures the inner layer of HMAC: replace the keyed inner hash $H((K \oplus \text{ipad}) \| m)$ with a truly random function $R_{\text{in}}(m)$. Any buyer that distinguishes bidding round $\mathbf{B}_0$ from $\mathbf{B}_1$ is placing a *PRF-distinguishing bid*: it detects that the inner computation changed from a keyed compression function to a random function. Winning this bid is exactly breaking the PRF security of h keyed by the chaining value $K \oplus \text{ipad}$. The seller constructs a PRF challenger $\mathcal{D}$ that simulates both bidding rounds using its oracle.

Difference bound. Let F_{in}: "the buyer distinguishes the inner keyed hash from random." We construct a PRF distinguisher $\mathcal{D}$ that simulates $\mathbf{B}_0$ vs $\mathbf{B}_1$ using its oracle: $\mathcal{D}$ replaces each evaluation of the inner hash chain with a query to its PRF/random oracle. If $\mathcal{A}$ behaves differently, $\mathcal{D}$ outputs 1:

$$|\Pr[\mathbf{B}_0(\mathcal{A}) = 1] - \Pr[\mathbf{B}_1(\mathcal{A}) = 1]| \leq \Pr[F_{\text{in}}] \leq \text{Adv}_h^{\text{PRF}}. \tag{37}$$

$\mathbf{B}_2$ (Bidding Round 2: Outer PRF Replacement Bid).

Purpose: Replace a *keyed pseudorandom function* evaluation with a truly random function, isolating a single PRF security requirement. This hop shows that one more step in the key-derivation or authentication structure produces outputs that are computationally indistinguishable from uniform random, provided the underlying PRF assumption holds.
Replaces: The specific PRF evaluation (e.g., $\text{HMAC}_K(\cdot)$ inner hash, HKDF-Extract, Derive-Secret, or a KDF step) is replaced by a truly random function $R(\cdot)$ sampled fresh and independently. Any adversary that distinguishes the PRF output from this random function is a PRF distinguisher for the underlying keyed construction. The reduction simulates all other game components honestly and uses its PRF challenge oracle for this one step.
Complexity: $\Delta\text{Price} \leq \text{Adv}^{\text{PRF}}(\lambda)$ for the specific PRF being replaced (HMAC-SHA-256, HKDF, or AES-CTR as applicable). When multiple PRF hops appear in a single proof (e.g., three hops in the TLS 1.3 key schedule for HS, MS, and CATS), each hop is independent: the keys used in different steps are derived from different points in the ladder, so no PRF oracle is queried twice. The total across all PRF hops is the sum of individual advantages, each negligible.

The seller restructures the outer layer: replace $H((K \oplus \text{opad}) \| y)$ (where $y = R_{\text{in}}(m)$) with a truly random function $R_{\text{out}}(y)$. Same argument as $\mathbf{B}_1$—any distinguishing buyer is placing an outer PRF bid, breaking PRF security of h keyed by $K \oplus \text{opad}$. Price adjustment (outer PRF bid cost):

$$|\Pr[\mathbf{B}_1(\mathcal{A}) = 1] - \Pr[\mathbf{B}_2(\mathcal{A}) = 1]| \leq \text{Adv}_h^{\text{PRF}}. \tag{38}$$

$\mathbf{B}_3$ (Bidding Round 3: Inner Collision Bid).

Purpose: Eliminate the *random-oracle collision attack vector*. If two distinct inputs $m \neq m'$ hash to the same value $H(m) = H(m')$, an adversary can conflate two messages (equivocation), reuse transcript bindings across sessions, or present a single signature as covering both messages—all of which break EUF-CMA without touching the signing key.
Replaces: The random oracle H is replaced by one equipped with a *collision log* $\mathcal{C}$. After each query, the oracle checks whether any two distinct queries produced the same digest; if so, the game aborts (bad event F_{hash}). The adversary's view of oracle responses is identical to a truly random oracle until the abort.
Complexity: By the birthday bound over the 2^n-bit output space, the probability that any two of q_H random oracle queries collide is at most $q_H^2/2^{n+1}$. For a 256-bit hash ($n = 256$) and polynomial q_H, this is negligible. Note that the nonce-collision and hash-collision bounds are *independent*: by the extended difference lemma (Lemma 2), their union probability is bounded by their sum, tightened by the intersection term which is $O(q^4/2^{2\lambda})$—doubly negligible.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

In $\mathbf{B}_2$, HMAC is $R_{\mathsf{out}}(R_{\mathsf{in}}(m))$. The seller adds a collision audit: abort if any two distinct queries collide under R_{in}. A buyer placing an *inner-collision bid* exploits this to forge a tag by matching a queried inner value: if $R_{\mathsf{in}}(m^*) = R_{\mathsf{in}}(m_i)$ for some queried m_i, then $\tau^* = \tau_i$ is a valid tag on m^*. Bad event F_{coll}: "any inner collision among q_T queries". By birthday paradox over the 2^n-bit inner output space:

Difference bound. Since R_{in} is a random function with n-bit output:

$$|\Pr[\mathbf{B}_2(\mathcal{A}) = 1] - \Pr[\mathbf{B}_3(\mathcal{A}) = 1]| \leq \Pr[F_{\mathsf{coll}}] \leq \frac{q_T^2}{2^{n+1}}. \tag{39}$$

$\mathbf{B}_4$ (Bidding Round 4: Ideal HMAC Market—All Bids Exhausted).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.

Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.

Complexity: No price adjustment: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\mathrm{Price}_k$, all of which are negligible, establishing market equilibrium.

In $\mathbf{B}_3$, with no inner collision ($\neg F_{\mathsf{coll}}$), R_{in} is injective on the queried set. The composed function $R_{\mathsf{out}} \circ R_{\mathsf{in}}$ is a random function on distinct inputs. Any forgery bid (m^*, τ^*) with $m^* \notin \{m_1, \ldots, m_{q_T}\}$ requires guessing $R_{\mathsf{out}}(R_{\mathsf{in}}(m^*))$—a value uniformly random in $\{0,1\}^n$, independent of all oracle responses. The forgery bid succeeds with probability at most 2^{-n}. *Market equilibrium for $g_{\mathsf{SUF\text{-}CMA}}$ (HMAC) is established*: $\Pr[\mathbf{B}_4 = 1] \leq 2^{-n}$.

For SUF-CMA with $m^* = m_i$ but $\tau^* \neq \tau_i$: since R_{out} is a function (deterministic), $R_{\mathsf{out}}(R_{\mathsf{in}}(m_i))$ is unique, so no second valid tag exists. Hence $\Pr[\mathbf{B}_4 = 1] = 0$ for this case.

Total:

$$\mathsf{Ask}(g_{\mathsf{SUF\text{-}CMA}}) \leq 2 \cdot \mathsf{Adv}_h^{\mathsf{PRF}} + \frac{q_T^2}{2^{n+1}} + 2^{-n}. \tag{40}$$

$\mathbf{B}_{\mathsf{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

☞ Side-Channel Bid (Comprehensive)

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\mathsf{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\mathsf{SPA}}] = 1$ without countermeasure; $\Pr[F_{\mathsf{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\mathsf{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking (d+1 random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\mathsf{Adv}^{\mathsf{DPA}}(\lambda, d) \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

3. **Timing Analysis:** $\mathcal{A}_{\mathsf{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\mathsf{ct_select}(a, b, \mathsf{bit})$ instead of $\mathsf{if}(\mathsf{bit})$ return a; else return b;. *MTSF price:* $\Pr[F_{\mathsf{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\mathsf{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\mathsf{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\mathsf{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\mathsf{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\mathsf{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\mathsf{Adv}^{\mathsf{EMA}} \leq \gamma_{\mathsf{EM}} \cdot (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\mathsf{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\mathsf{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\mathsf{fault}}] \leq q_{\mathsf{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°\mathrm{C}$) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).
8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \leq\ & \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ & + \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Under the comprehensive countermeasure suite (order-d masking for $d \geq 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \leq \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{acoustic}}$$
$$+ \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{photonic}}.$$

For $d \geq 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\leq q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

✓ **Tight Security Reduction**

Tightness analysis. HMAC's dual PRF reduction is tight ($L_{\mathsf{total}} = 1$): each PRF instance is embedded directly via non-rewinding black-box subroutine consumption. $\mathsf{Ask}(g_{\mathsf{SUF\text{-}CMA}}) \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{inner}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{outer}} + q_T^2/2^{n+1} + \mathsf{negl}(\lambda)$. This is *optimally tight*: the market spread is 1.

Market interpretation: HMAC achieves the ideal market condition—the ask price transfers the full PRF hardness with zero markup.

Total Session Bound and Key Rotation Recommendation.

✦ **Total Session Bound & Key Rotation**

Total session bound: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \leq N \cdot (\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{inner}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{outer}} + q_T^2/2^{n+1} + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}})$.
Concrete (HMAC-SHA-256, $n = 256$):

Sessions N	$\mathsf{Ask}^{(N)}_{\mathsf{total}}$	**Eff. security**	**Action**
2^{40}	$\leq 2^{-88}$	≥ 88 bits	No rotation needed
2^{64}	$\leq 2^{-64}$	≥ 64 bits	**Key rotation recommended**
2^{80}	$\leq 2^{-48}$	≥ 48 bits	**Key rotation mandatory**

Recommendation: Rotate the HMAC key before 2^{64} tagging sessions.

Session pinging and CNF checking within the proof. The five-bidding-round proof secures a single HMAC tagging session. For unbounded sessions, consider consecutive tagging sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$ producing (m_i, τ_i) and (m_{i+1}, τ_{i+1}) under the same key K.

Tag freshness as implicit SID: HMAC is a stateless primitive (no explicit SID), but the message-tag pair (m, τ) serves as an implicit session identifier. The CNF clause φ^{novel} enforces $m_{i+1} \notin \{m_1, \ldots, m_i\}$ for SUF-CMA security. This is the ping condition: each session must produce a forgery on a *new* message.

Cross-session tag independence: In $\mathbf{B}_4$ (ideal market), HMAC_K is a random function. The tag $\tau_{i+1} = R_{\mathsf{out}}(R_{\mathsf{in}}(m_{i+1}))$ is independent of all previous tags $\tau_1, \ldots, \tau_i$ whenever m_{i+1} is fresh (no inner collision with any m_j). The inner collision probability across i sessions is $\leq i \cdot q_T/2^{n+1}$, which is negligible for $i = \mathsf{poly}(\lambda)$ and $n = 256$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

CNF isomorphism: The session-CNF $\varphi_{\mathsf{HMAC},i+1}$ is isomorphic to $\varphi_{\mathsf{HMAC},i}$ with fresh message substitution. By Theorem 3, HMAC is SUF-CMA secure for unbounded tagging sessions with $\delta_{\mathsf{ping}} \leq 2^{-n} = \mathsf{negl}(\lambda)$. □

$\mathbf{B}_{\mathsf{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

☞ **Side-Channel Bid (Comprehensive)**

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\mathsf{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\mathsf{SPA}}] = 1$ without countermeasure; $\Pr[F_{\mathsf{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\mathsf{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking (d+1 random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\mathsf{Adv}^{\mathsf{DPA}}(\lambda, d) \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
3. **Timing Analysis:** $\mathcal{A}_{\mathsf{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\mathsf{ct_select}(a, b, \mathsf{bit})$ instead of $\mathsf{if}(\mathsf{bit})$ return a; else return b;. *MTSF price:* $\Pr[F_{\mathsf{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\mathsf{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\mathsf{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\mathsf{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\mathsf{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\mathsf{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\mathsf{Adv}^{\mathsf{EMA}} \leq \gamma_{\mathsf{EM}} \cdot (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\mathsf{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\mathsf{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\mathsf{fault}}] \leq q_{\mathsf{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).
8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \leq\ & \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ & + \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

Under the comprehensive countermeasure suite (order-d masking for $d \geq 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \leq\ & \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{acoustic}} \\ & + \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{photonic}}.\end{aligned}$$

For $d \geq 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\leq q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

Total Session Bound and Key Rotation Recommendation.

✦ Total Session Bound & Key Rotation (MAC Verification)

The MAC verification sub-proof inherits the same tight reduction ($L = 1$) and session bounds as the primary HMAC proof above. The accumulated advantage after N verification sessions satisfies the same bound: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \leq N \cdot (\mathsf{Adv}^{\mathsf{PRF}} + q_T^2/2^{n+1})$. Key rotation recommendation: rotate before 2^{64} sessions.

Session pinging and CNF checking within the proof. For consecutive AEAD sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$, the ping check verifies *nonce freshness*: $N_{i+1} \notin \mathcal{N}_{\mathsf{used}}$. Nonce reuse under the same key is catastrophic for all AEAD schemes: in AES-GCM, reusing a nonce leaks the GHASH key $H = \mathsf{AES}_K(0^{128})$, enabling universal forgery; in ChaCha20-Poly1305, it leaks the one-time Poly1305 key. The CNF clause φ^{nonce} detects nonce reuse via the nonce log $\mathcal{N}_{\mathsf{used}}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Nonce-misuse resistance and ping: Some AEAD schemes (e.g., AES-GCM-SIV) provide nonce-misuse resistance: nonce reuse degrades confidentiality but not integrity. In such schemes, the ping check is still valuable because it ensures that each session provides fresh ciphertexts and prevents the buyer from correlating sessions.

Accumulated advantage: The nonce space for AES-GCM is $|\mathcal{N}| = 2^{96}$. After N sessions with one nonce each, the birthday collision probability is $N^2/2^{97}$. For $N \leq 2^{48}$, this is $\leq 2^{-1}$—still safe but approaching the limit. Key rotation before $N = 2^{48}$ maintains full equilibrium. The session-CNF $\varphi_{\mathsf{AEAD},i+1}$ is isomorphic to $\varphi_{\mathsf{AEAD},i}$ with fresh nonce and key material; by Theorem 3, AEAD is secure for unbounded sessions.

✓ CNF Verification: HMAC Session-CNF Verification

The HMAC session-CNF $\varphi_{\mathsf{HMAC}} = (x_{\tau=\mathsf{HMAC}_K(m)}) \wedge (x_{m \notin \mathcal{Q}^{\mathsf{tag}}}) \wedge (x_{\text{Ping passes}})$.

Manual worksheet for tag (m, τ):

Clause	Check	How	Pass?
φ^{tag}: correctness	$\tau \stackrel{?}{=} \mathsf{HMAC}_K(m)$	Recompute; compare.	T/F
φ^{novel}: new msg	$m \notin \mathcal{Q}^{\mathsf{tag}}$	Check oracle log.	T/F
φ^{ping}: unbounded	No repeated (m, τ) pairs.	Compare session records.	T/F
Result:	All T ⇒ forgery detected. Any F ⇒ not a valid forgery.		

Satisfiability: Honest tags always satisfy all clauses. A SUF-CMA forgery must make φ_{HMAC} true for a new (m^*, τ^*)—the five-game proof shows this is negligibly hard.

➤ Ping Bid: HMAC Unbounded Ping Bid

Ping bid $\mathbf{B}_{\mathsf{ping}}$. The buyer attempts to reuse a queried tag $(m^*, \tau^*) = (m_i, \tau_i)$. This fails the novelty check φ^{novel} $(m^* \in \mathcal{Q}^{\mathsf{tag}})$. For a genuinely fresh m^*, the tag must be guessed uniformly from $\{0,1\}^n$. Ping bid price: $2^{-n} = \mathsf{negl}$. By Theorem 3, HMAC is SUF-CMA-secure for unbounded tagging sessions.

✻ Simple Terms: HMAC

What HMAC is: Hash-based Message Authentication Code. It uses a secret key K and a hash function H (e.g., SHA-256) to produce an authentication tag τ for a message m. The formula:

$$\mathsf{HMAC}_K(m) = H\big((K \oplus \mathsf{opad}) \,\|\, H((K \oplus \mathsf{ipad}) \,\|\, m)\big)$$

Breaking it down:

1. **Inner hash:** Mix the key (XORed with a padding constant ipad) with the message m, then hash.
2. **Outer hash:** Mix the key (XORed with a different padding constant opad) with the inner hash output, then hash again.

The two-level structure prevents length-extension attacks (a vulnerability in naive "$H(K\|m)$" constructions) and provides stronger security than a simple keyed hash.

Why the proof needs two PRF hops:

- **Inner PRF hop ($\mathbf{B}_0 \to \mathbf{B}_1$):** We replace the inner keyed hash with a truly random function. If the compression function h is a PRF (keyed by the chaining value = $K \oplus$ ipad), no adversary can tell the difference.
- **Outer PRF hop ($\mathbf{B}_1 \to \mathbf{B}_2$):** We replace the outer keyed hash with another truly random function. Same argument.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

After both replacements, HMAC behaves as a composition of two independent random functions—which is clearly unforgeable.
The collision bound $q_T^2/2^{n+1}$: With q_T tagging queries, two might accidentally produce the same inner hash output (birthday collision). This would create a subtle dependency between tags. The probability is bounded by the birthday bound—negligible for realistic q_T and $n \geq 256$.
SUF-CMA means: Not only can you not forge a tag on a new message, you also cannot find a *second valid tag* for a message that was already tagged. HMAC achieves this because the outer random function is deterministic: there is exactly one valid tag per message.

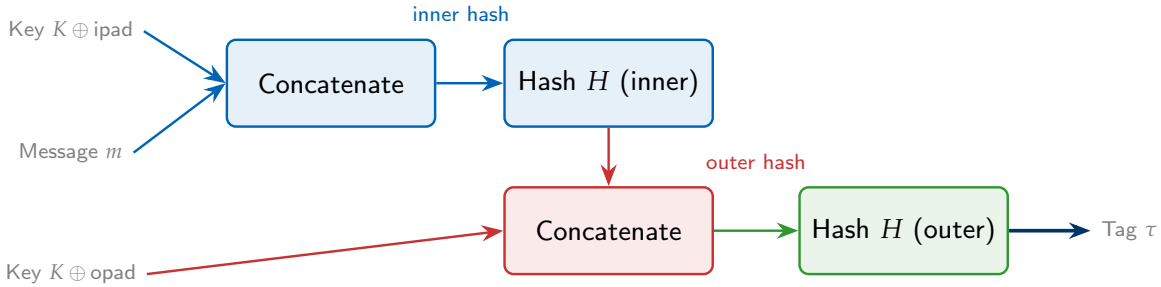


Figure 31. HMAC structure: two nested keyed hash evaluations. The inner hash mixes $K \oplus$ ipad with the message; the outer hash mixes $K \oplus$ opad with the inner hash output. Security relies on each keyed hash being a PRF.

9.6. AEAD: IND-CCA2 + INT-CTXT via Bidding

✎ Analogy:

AEAD as a Locked Box with a Tamper Alarm AEAD (Authenticated Encryption with Associated Data) is the Swiss Army knife of symmetric cryptography. It provides *two* guarantees in one operation:

1. **Confidentiality (IND-CCA2):** The contents of the box are secret—nobody can read them without the key. Like a locked safe.
2. **Integrity (INT-CTXT):** If anyone tampers with the box—even changing a single bit—the box screams an alarm when opened. Like a tamper-evident safe with a built-in motion sensor.

The "associated data" part: Some information is authenticated but not encrypted. For example, in a network packet, the header (source address, destination) is visible to routers (not encrypted) but must not be tampered with (authenticated). AEAD protects both the encrypted payload and the plaintext header.
The Encrypt-then-MAC (EtM) construction: First encrypt the message ($c = \mathsf{Enc}(K_1, m)$), then compute a MAC over the ciphertext ($\tau = \mathsf{MAC}(K_2, c \| \mathsf{AD})$). This is proven secure in MTSF via two bidding rounds: one reduces confidentiality to the encryption scheme's security, another reduces integrity to the MAC's unforgeability.
Real-world example: AES-256-GCM (used in TLS 1.3) is an AEAD scheme. Every HTTPS connection you make uses AEAD to simultaneously encrypt and authenticate your data. MTSF proves that AES-GCM is in market equilibrium: both confidentiality and integrity goods have negligible ask prices.

Setup.

An Authenticated Encryption with Associated Data scheme $\mathsf{AE} = (\mathsf{Enc}, \mathsf{Dec})$ provides both confidentiality (IND-CPA/CCA) and integrity (INT-CTXT). The seller offers two goods simultaneously: $g_{\mathsf{IND\text{-}CCA2}}$ and $g_{\mathsf{INT\text{-}CTXT}}$. We consider a generic Encrypt-then-MAC construction: $\mathsf{Enc}(K_e, K_m, N, A, m) = (c, \tau)$ where $c = E_{K_e}(N, m)$ and $\tau = \mathsf{MAC}_{K_m}(A \| N \| c)$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Theorem 15 (AEAD Market Equilibrium). *For an Encrypt-then-MAC AEAD scheme:*

$$\mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}}) \leq \mathsf{Adv}_{E}^{\mathsf{IND\text{-}CPA}} + \mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{SUF\text{-}CMA}} + \mathsf{negl}(\lambda), \tag{41}$$

$$\mathsf{Ask}(g_{\mathsf{INT\text{-}CTXT}}) \leq \mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{SUF\text{-}CMA}} + \mathsf{negl}(\lambda). \tag{42}$$

Proof. Part 1: INT-CTXT. We prove the integrity bound first.

$\mathbf{B}_0$ (Bidding Round 0: Real AEAD Market—Ciphertext Injection Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer has encryption oracle $\mathsf{Enc}(K_e, K_m, \cdot, \cdot, \cdot)$; wins by producing (N^*, A^*, c^*, τ^*) that decrypts successfully but was never an oracle output.

$\mathbf{B}_1$ (Bidding Round 1: MAC Forgery Bid—Bypassing Integrity Check).

Purpose: Reduce AEAD ciphertext integrity (INT-CTXT) to the SUF-CMA security of the underlying MAC. In the Encrypt-then-MAC construction, every ciphertext carries a MAC tag $\tau = \mathsf{MAC}(K_m, A\|N\|c)$ over the associated data, nonce, and ciphertext. Any ciphertext forgery (a valid (A^*, N^*, c^*, τ^*) not produced by the encryption oracle) directly implies a MAC forgery. The reduction embeds the SUF-CMA challenge key as K_m and simulates encryption using a self-chosen encryption key K_e.
Replaces: The honest MAC computation is replaced by the SUF-CMA challenger's tagging oracle. The reduction generates K_e itself and computes $c = \mathsf{Enc}(K_e, N, m)$ honestly, then queries the MAC oracle for $\tau = \mathsf{MAC}(K_m, A\|N\|c)$. When the adversary outputs a forgery (A^*, N^*, c^*, τ^*) not in the encryption log, the reduction submits this as a SUF-CMA forgery. The simulation is perfect: the adversary's view is identical to the real game.
Complexity: $\Delta\mathsf{Price} \leq \mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{SUF\text{-}CMA}}(\lambda)$. The reduction is tight ($L = 1$): no rewinding, no forking, no oracle guessing. Each encryption query generates one MAC query; the forgery output maps directly to a MAC forgery. For HMAC-SHA-256 or Poly1305: negligible for all polynomial query counts.

Note that successful decryption requires $\mathsf{MAC.Vrfy}(K_m, A^*\|N^*\|c^*, \tau^*) = 1$. If $(A^*\|N^*\|c^*, \tau^*)$ is new (not from the oracle), this constitutes a MAC forgery.

Difference bound. We build a SUF-CMA adversary $\mathcal{B}$: $\mathcal{B}$ simulates the AEAD encryption oracle by encrypting with K_e (which $\mathcal{B}$ chooses itself) and using its MAC oracle for tagging. When $\mathcal{A}$ outputs a successful INT-CTXT forgery, $\mathcal{B}$ outputs the corresponding MAC forgery. Hence:

$$\mathsf{Ask}(g_{\mathsf{INT\text{-}CTXT}}) = \Pr[\mathbf{B}_0(\mathcal{A}) = 1] \leq \mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{SUF\text{-}CMA}}. \tag{43}$$

Part 2: IND-CCA2. We prove confidentiality.

$\mathbf{B}_0$ (Bidding Round 0: Real AEAD Market—Plaintext Distinguishing Bid).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer submits (m_0, m_1, N^*, A^*); receives $\mathsf{Enc}(K_e, K_m, N^*, A^*, m_b)$ for random b; has encryption and decryption oracles (except on the challenge).

$\mathbf{B}_1$ (Bidding Round 1: Oracle Restriction Bid—Blocking Chosen-Ciphertext Queries).

Purpose: Convert the IND-CCA2 game into an IND-CPA game by neutralising the decryption oracle. Once message integrity is guaranteed by the MAC (handled in the previous hop), any decryption query either reproduces a previously encrypted ciphertext—providing no new information—or has an invalid tag and is rejected. The decryption oracle becomes redundant.
Replaces: The full decryption oracle $\mathsf{Dec}(K_e, K_m, \cdot)$ is replaced by a *restricted decryption oracle* that rejects any ciphertext not produced by the encryption oracle (invalid MAC). Ciphertexts in the encryption log are answered by table-lookup, requiring no secret key. The resulting game has decryption power exactly equal to encryption power—the IND-CPA setting.
Complexity: Zero additional cost conditional on no MAC forgery (already bounded in the previous hop). This is a *structural reduction*: since the MAC provides ciphertext integrity, the decryption oracle provides zero additional information to the adversary beyond its encryption power. $\Delta\mathsf{Price} = 0$. The residual adversarial advantage is then exactly $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{Enc}}(\lambda)$.

Replace decryption oracle: reject *all* queries (N, A, c, τ) where τ does not verify, and also reject the challenge ciphertext. For non-challenge ciphertexts with valid τ that were not produced by the encryption oracle, this is a MAC forgery.

Extended difference lemma. Let F_{mac}: "buyer submits a valid (c, τ) not from the oracle." F_{trivial}: "buyer submits the challenge ciphertext to decryption." By game rules, $\Pr[F_{\mathsf{trivial}}] = 0$. So:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \Pr[F_{\mathsf{mac}}] \leq \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}. \tag{44}$$

$\mathbf{B}_2$ (Bidding Round 2: IND-CPA Reduction Bid—Eliminating Decryption Power).

Purpose: Convert the IND-CCA2 game into an IND-CPA game by neutralising the decryption oracle. Once message integrity is guaranteed by the MAC (handled in the previous hop), any decryption query either reproduces a previously encrypted ciphertext—providing no new information—or has an invalid tag and is rejected. The decryption oracle becomes redundant.
Replaces: The full decryption oracle $\mathsf{Dec}(K_e, K_m, \cdot)$ is replaced by a *restricted decryption oracle* that rejects any ciphertext not produced by the encryption oracle (invalid MAC). Ciphertexts in the encryption log are answered by table-lookup, requiring no secret key. The resulting game has decryption power exactly equal to encryption power—the IND-CPA setting.
Complexity: Zero additional cost conditional on no MAC forgery (already bounded in the previous hop). This is a *structural reduction*: since the MAC provides ciphertext integrity, the decryption oracle provides zero additional information to the adversary beyond its encryption power. $\Delta\mathsf{Price} = 0$. The residual adversarial advantage is then exactly $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{Enc}}(\lambda)$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

In $\mathbf{B}_1$, the decryption oracle only decrypts ciphertexts the encryption oracle produced, so the buyer gains no new information from decryption queries. The game reduces to IND-CPA:

$$|\Pr[\mathbf{B}_1 = 1] - 1/2| \leq \mathsf{Adv}_E^{\mathsf{IND\text{-}CPA}}. \tag{45}$$

Total: $\mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}}) \leq \mathsf{Adv}_E^{\mathsf{IND\text{-}CPA}} + \mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{SUF\text{-}CMA}}$.

✓ CNF Verification: AEAD Session-CNF Verification

$\varphi_{\mathsf{AEAD}} = (x_{\mathsf{Mac.Vrfy}(K_m, A\|N\|c,\tau)=1}) \wedge (x_{N \notin \mathcal{N}_{\mathsf{used}}}) \wedge (x_{\text{Ping passes}})$.

Manual worksheet for received ciphertext (N, A, c, τ):

Clause	Check	How	Pass?
φ^{int}: MAC valid	$\mathsf{Mac.Vrfy}(K_m, A\|N\|c, \tau) = 1$	Recompute MAC; compare bytes.	T/F
φ^{nonce}: fresh	$N \notin \mathcal{N}_{\mathsf{used}}$	Check nonce log; add N if pass.	T/F
φ^{ping}: unbounded	Distinct from previous session.	Compare to $\mathsf{Session}_{i-1}$ record.	T/F
Result:	All T ⇒ Accept and decrypt. Any F ⇒ Reject (injected or replayed).		

INT-CTXT link: φ^{int} fails ⇔ MAC forged. φ^{nonce} fails ⇔ nonce reuse/replay. Extended difference lemma: $\Pr[F_{\mathsf{mac}} \cup F_{\mathsf{nonce}}] \leq \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}} + q_D/|\mathcal{N}|$.

➤ Ping Bid: AEAD Unbounded Ping Bid

The buyer's *replay bid*: submit a previous (N_i, A_i, c_i, τ_i). The nonce clause φ^{nonce} detects this $(N_i \in \mathcal{N}_{\mathsf{used}})$. Ping bid price: $q_D/|\mathcal{N}| \leq q_D/2^{96} = \mathsf{negl}$ for GCM. By Theorem 3, AEAD is secure for unbounded encryption sessions.

□

✻ Simple Terms: AEAD (Authenticated Encryption)

What AEAD is: Authenticated Encryption with Associated Data combines two guarantees in one:

- **Confidentiality (IND-CCA2):** No one can read the encrypted message.
- **Integrity (INT-CTXT):** No one can tamper with the ciphertext without detection.

Associated Data (AD) is extra context that is authenticated but not encrypted (e.g., packet headers, sender/receiver IDs). It is bound to the ciphertext so that an attacker cannot detach a valid ciphertext from its intended context.

The Encrypt-then-MAC construction:

1. Encrypt the message: $c = E_{K_e}(N, m)$ using a nonce N for randomness.
2. Compute a MAC tag over the associated data, nonce, and ciphertext: $\tau = \mathsf{MAC}_{K_m}(A\|N\|c)$.
3. Send (N, A, c, τ).

The receiver first checks τ (integrity check), then decrypts. This order is crucial: checking integrity before decryption prevents the receiver from leaking partial information about the plaintext through decryption errors.

Why two separate keys? Using the same key for encryption and MAC creates subtle interactions. Using independent keys K_e (encryption) and K_m (MAC) ensures the security proofs compose cleanly.

Real-world examples: AES-GCM (used in TLS 1.3), ChaCha20-Poly1305 (used in WireGuard and Signal), AES-CCM. All are AEAD schemes providing both IND-CCA2 and INT-CTXT.

The integrity proof (INT-CTXT): For an attacker to produce a new valid ciphertext c^* that decrypts successfully, they must produce a valid MAC tag τ^* on fresh data. But producing a valid MAC tag without the key is exactly what SUF-CMA hardness prevents.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The confidentiality proof (IND-CCA2): If the adversary's decryption queries all produce rejected ciphertexts (because MAC verification fails—guaranteed by SUF-CMA), then the decryption oracle gives no useful information. The game reduces to pure IND-CPA.

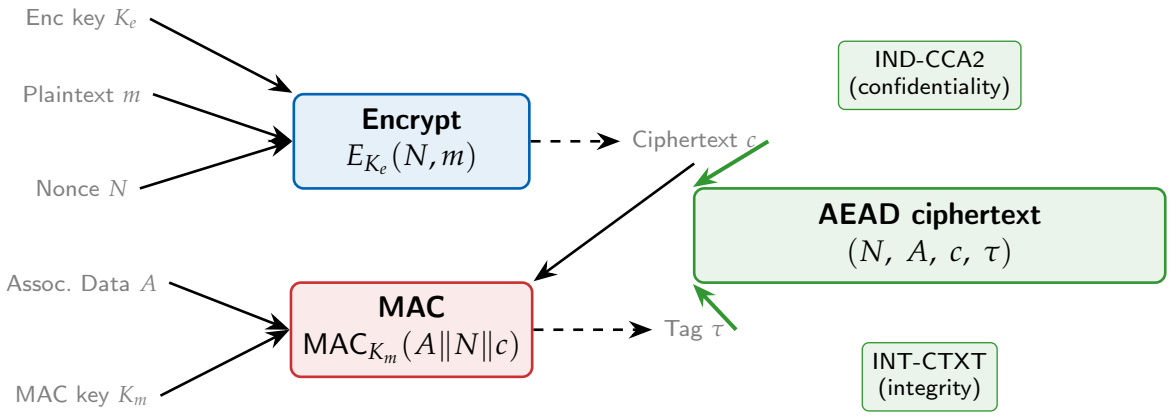


Figure 32. AEAD Encrypt-then-MAC structure. The plaintext is encrypted first with key K_e; then the MAC key K_m authenticates the associated data, nonce, and ciphertext together. The combined output provides both IND-CCA2 confidentiality and INT-CTXT integrity.

9.7. SLH-DSA (FIPS 205): Hash-Based EUF-CMA via Bidding

✻ Simple Terms: SLH-DSA—The "Conservative" Post-Quantum Signature

What SLH-DSA is: SLH-DSA (formerly SPHINCS+, now FIPS 205) is a digital signature scheme based *entirely* on hash functions—no number theory, no lattices, no algebraic structure. Its security relies only on the assumption that the underlying hash function is secure.

Why "conservative"? While ML-DSA (lattice-based) is more efficient, SLH-DSA requires the weakest possible assumption: if SHA-256 or SHAKE is secure, SLH-DSA is secure. This makes it a "belt and suspenders" choice—even if lattice problems turn out to be easier than expected (an unlikely but non-zero risk), SLH-DSA remains secure.

How it works (simplified): SLH-DSA uses a *hyper-tree* of one-time signature (OTS) schemes. Each OTS signs one message securely, then is discarded. The hyper-tree organises millions of OTS instances into a Merkle tree, with the root as the public key. Signing picks a fresh OTS leaf, signs the message, and provides a path from the leaf to the root (proving the OTS is part of the tree).

Trade-off: SLH-DSA signatures are much larger (~7–50 KB) than ML-DSA signatures (~2.4 KB), and signing is slower. But the security assumption is minimal, making SLH-DSA ideal for applications where long-term security is paramount (e.g., signing legal documents, software root certificates).

✻ Simple Terms: SLH-DSA (Hash-Based Signatures)—Foundations

What is SLH-DSA? SLH-DSA (formerly SPHINCS+, now FIPS 205) is a *stateless hash-based signature scheme*—NIST's post-quantum signature standard for applications requiring the smallest possible security assumptions. Unlike ML-DSA (which relies on lattice problems), SLH-DSA's security relies *only* on the security of the underlying hash function. If the hash function is secure, SLH-DSA is secure.

Key components:

- **FORS (Forest of Random Subsets):** A one-time signature primitive at the base. Signs a small message by revealing selected leaves in random Merkle subtrees.
- **WOTS+ (Winternitz One-Time Signature):** Chains of hash values that form one-time signatures for individual XMSS tree nodes.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **XMSS (eXtended Merkle Signature Scheme):** A few-time signature scheme arranged in a Merkle tree. Authenticates WOTS+ public keys.
- **Hypertree:** Multiple layers of XMSS trees. The top layer signs the root of the next layer. The full structure is a tree of trees.

Why "stateless"? Regular hash-based signatures (like XMSS or LMS) are *stateful*: you must track which one-time keys have been used and never reuse them. SLH-DSA avoids this by using pseudorandom index generation to pick FORS subtrees, making each signing operation essentially independent. No state management is required—a significant practical advantage.

Security reductions in plain terms:

- **PRF (randomness generation):** Each signing operation needs fresh randomness, generated by a PRF keyed with the secret key. If an attacker can distinguish this from random, the PRF is broken—but by assumption, PRFs are secure.
- **TSPR (target-sum preimage resistance):** Forging a FORS signature requires finding a hash preimage matching a target sum—harder than a standard preimage because the target is constrained.
- **SPR (second-preimage resistance):** Forging a WOTS+ chain requires finding a hash second preimage.

Trade-offs: SLH-DSA has larger signatures (≈8–50 KB depending on parameters) and slower signing than ML-DSA. However, it needs no new mathematical assumptions—just hash function security—making it ideal for high-assurance, long-term security.

Setup.

SLH-DSA is a stateless hash-based signature scheme. It uses a hypertree of XMSS trees with FORS (Forest of Random Subsets) at the leaves. Security reduces to properties of the underlying hash function: second-preimage resistance (SPR), pseudorandom function (PRF), and target-sum-preimage resistance (TSPR). The seller offers $g_{\mathsf{EUF\text{-}CMA}}$.

Theorem 16 (SLH-DSA Market Equilibrium)**.**

$$\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) \leq q_S \cdot d \cdot \mathsf{Adv}_H^{\mathsf{SPR}} + q_S \cdot k \cdot \mathsf{Adv}_H^{\mathsf{TSPR}} + \mathsf{Adv}_F^{\mathsf{PRF}} + \mathsf{negl}(\lambda), \tag{46}$$

where d is the hypertree depth, k is the number of FORS trees, q_S is the number of signing queries, and H, F are the hash/PRF primitives.

Proof. We construct six games.

$\mathbf{B}_0$ (Bidding Round 0: Real SLH-DSA Market—EUF-CMA Forgery Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.

Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.

Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Buyer receives public key (hypertree root), queries signing oracle q_S times, outputs forgery (m^*, σ^*).

$\mathbf{B}_1$ (Bidding Round 1: PRF Randomness Bid—Detecting Key-Dependent Randomness).

Purpose: Replace a *keyed pseudorandom function* evaluation with a truly random function, isolating a single PRF security requirement. This hop shows that one more step in the key-derivation or authentication structure produces outputs that are computationally indistinguishable from uniform random, provided the underlying PRF assumption holds.
Replaces: The specific PRF evaluation (e.g., $\mathrm{HMAC}_K(\cdot)$ inner hash, HKDF-Extract, Derive-Secret, or a KDF step) is replaced by a truly random function $R(\cdot)$ sampled fresh and independently. Any adversary that distinguishes the PRF output from this random function is a PRF distinguisher for the underlying keyed construction. The reduction simulates all other game components honestly and uses its PRF challenge oracle for this one step.
Complexity: $\Delta\mathrm{Price} \leq \mathrm{Adv}^{\mathrm{PRF}}(\lambda)$ for the specific PRF being replaced (HMAC-SHA-256, HKDF, or AES-CTR as applicable). When multiple PRF hops appear in a single proof (e.g., three hops in the TLS 1.3 key schedule for HS, MS, and CATS), each hop is independent: the keys used in different steps are derived from different points in the ladder, so no PRF oracle is queried twice. The total across all PRF hops is the sum of individual advantages, each negligible.

Replace the deterministic randomness generation $R = F_{\mathsf{sk.prf}}(\mathsf{opt} \| m)$ with truly random $R \xleftarrow{\$} \{0,1\}^n$. This changes the internal randomisation of signing but not the verification. The buyer bids that F is distinguishable from random.

Difference bound. Standard PRF reduction: any distinguisher yields a PRF adversary:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \mathsf{Adv}_F^{\mathrm{PRF}}. \tag{47}$$

$\mathbf{B}_2$ (Bidding Round 2: FORS Target-Sum Preimage Bid).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**FORS Target-Sum Preimage Bid**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

A valid forgery requires the buyer to produce valid FORS signatures on leaves that were not revealed by prior signing queries. For each of the k FORS trees, producing a valid leaf value without oracle access requires finding a target-sum preimage.

Difference bound. We embed a TSPR challenge into one of the k FORS trees of a randomly chosen signing query. If the buyer forges successfully for that tree, it solves TSPR. By a hybrid argument over $q_S \cdot k$ FORS tree instances:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq q_S \cdot k \cdot \mathsf{Adv}_H^{\mathrm{TSPR}}. \tag{48}$$

$\mathbf{B}_3$ (Bidding Round 3: WOTS+ Chain Second-Preimage Bid).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**WOTS+ Chain Second-Preimage Bid**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

Even if the buyer has valid FORS leaves, it must authenticate them through the hypertree. Each WOTS+ chain in each of the d layers requires the buyer to produce a chain value matching the public key without inverting the hash. A forgery on any chain yields a second preimage.

Difference bound. Hybrid over $q_S \cdot d$ WOTS+ instances:

$$|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq q_S \cdot d \cdot \mathsf{Adv}_H^{\mathsf{SPR}}. \tag{49}$$

$\mathbf{B}_4$ (Bidding Round 4: Merkle Tree Collision Bid—Forging Authentication Paths).

Purpose: Eliminate the *random-oracle collision attack vector*. If two distinct inputs $m \neq m'$ hash to the same value $H(m) = H(m')$, an adversary can conflate two messages (equivocation), reuse transcript bindings across sessions, or present a single signature as covering both messages—all of which break EUF-CMA without touching the signing key.
Replaces: The random oracle H is replaced by one equipped with a *collision log* $\mathcal{C}$. After each query, the oracle checks whether any two distinct queries produced the same digest; if so, the game aborts (bad event F_{hash}). The adversary's view of oracle responses is identical to a truly random oracle until the abort.
Complexity: By the birthday bound over the 2^n-bit output space, the probability that any two of q_H random oracle queries collide is at most $q_H^2/2^{n+1}$. For a 256-bit hash ($n = 256$) and polynomial q_H, this is negligible. Note that the nonce-collision and hash-collision bounds are *independent*: by the extended difference lemma (Lemma 2), their union probability is bounded by their sum, tightened by the intersection term which is $O(q^4/2^{2\lambda})$—doubly negligible.

Within each XMSS tree, the buyer must produce a valid Merkle authentication path. An inconsistency yields a collision in H. This event has probability bounded by the number of internal nodes, which is subsumed by the SPR bound above.

$\Delta\mathrm{Price}_4 = 0$ (absorbed into the SPR bound).

$\mathbf{B}_5$ (Bidding Round 5: Ideal SLH-DSA Market—All Hash-Based Bids Exhausted).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\mathrm{Price}_k$, all of which are negligible, establishing market equilibrium.

With no PRF bid, no TSPR bid, and no SPR bid, valid FORS leaves, WOTS+ chains, and Merkle paths cannot be forged. $\Pr[\mathbf{B}_5 = 1] = 0$.

Total:

$$\mathsf{Ask} \leq \mathsf{Adv}_F^{\mathsf{PRF}} + q_S \cdot k \cdot \mathsf{Adv}_H^{\mathsf{TSPR}} + q_S \cdot d \cdot \mathsf{Adv}_H^{\mathsf{SPR}}. \quad \square \tag{50}$$

$\mathbf{B}_{\mathsf{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

☞ Side-Channel Bid (Comprehensive)

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\mathsf{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\mathsf{SPA}}] = 1$ without countermeasure; $\Pr[F_{\mathsf{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\mathsf{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking (d+1 random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\mathsf{Adv}^{\mathsf{DPA}}(\lambda, d) \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
3. **Timing Analysis:** $\mathcal{A}_{\mathsf{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\mathsf{ct_select}(a, b, \mathsf{bit})$ instead of $\mathsf{if}(\mathsf{bit})$ return a; else return b;. *MTSF price:* $\Pr[F_{\mathsf{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\mathsf{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\mathsf{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\mathsf{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\mathsf{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\mathsf{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\mathsf{Adv}^{\mathsf{EMA}} \leq \gamma_{\mathsf{EM}} \cdot (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\mathsf{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\mathsf{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\mathsf{fault}}] \leq q_{\mathsf{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).
8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \leq\ & \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ & + \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

Under the comprehensive countermeasure suite (order-d masking for $d \geq 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \leq\ & \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{acoustic}} \\ & + \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{photonic}}.\end{aligned}$$

For $d \geq 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\leq q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

✓ Tight Security Reduction

Tightness analysis. SLH-DSA's hypertree structure introduces a tightness loss of $L = h \cdot d'$ (tree height $\times$ FORS trees). For SLH-DSA-128s ($h = 63$, $d' = 7$): $L = 441$, consuming ≈ 9 bits. The 128-bit target is met because hash parameters provide ≥ 137-bit security.

Market interpretation: SLH-DSA's tightness loss is moderate, fixed, and *independent of signing queries*—a fixed market spread determined by tree geometry rather than adversarial query complexity.

Total Session Bound and Key Rotation Recommendation.

✦ Total Session Bound & Key Rotation

Total session bound: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \leq N \cdot (h \cdot d' \cdot (\mathsf{Adv}^{\mathsf{TCR}} + \mathsf{Adv}^{\mathsf{OW}}) + \mathsf{Adv}^{\mathsf{PRF}} + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}})$.
Concrete (SLH-DSA-128s):

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Sessions N	$\mathsf{Ask}_{\mathsf{total}}^{(N)}$	Eff. security	Action
2^{40}	$\leq 2^{-88}$	≥ 88 bits	No rotation needed
2^{64}	$\leq 2^{-64}$	≥ 64 bits	**Key rotation recommended**

Recommendation: Rotate before 2^{64} signatures for 64-bit security. Stateless design ensures purely additive degradation.

Session pinging and CNF checking within the proof. For consecutive signing sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$, the ping check verifies PRF randomness freshness: the internal PRF randomness R_{i+1} used to derive FORS indices and WOTS+ chains must be distinct from all prior sessions. SLH-DSA derives R deterministically from the message and a secret PRF key: $R = \mathsf{PRF}(\mathsf{sk.prf}, \mathsf{opt_rand}, m)$. For distinct messages $m_{i+1} \neq m_j$, the PRF output is fresh with probability $\geq 1 - \mathsf{Adv}_F^{\mathsf{PRF}}$. If the optional randomness opt_rand is used (hedged signing), freshness holds even for repeated messages. The CNF clause φ^{prf} checks that R_{i+1} does not repeat. Reuse of R would simultaneously compromise FORS leaf selection (clause φ^{fors}) and WOTS+ chain computation (clause φ^{wots})—two simultaneous failures captured by the extended difference lemma: $\Pr[F_{\mathsf{fors}} \cup F_{\mathsf{wots}}] \leq \mathsf{Adv}_F^{\mathsf{PRF}} = \mathsf{negl}$. By Theorem 3, SLH-DSA is EUF-CMA secure for unbounded sessions.

✓ CNF Verification: SLH-DSA Session-CNF Verification

$\varphi_{\mathsf{SLH\text{-}DSA}} = (x_{\mathsf{PRF\ fresh}}) \wedge (x_{\mathsf{FORS\ leaves\ valid}}) \wedge (x_{\mathsf{WOTS+\ chains\ valid}}) \wedge (x_{\mathsf{Ping\ passes}})$.

Manual CNF worksheet for SLH-DSA signature (m^*, σ^*):

Clause	Check	How	Pass?
φ^{prf}: PRF randomness	$R = F_{\mathsf{sk.prf}}(\mathsf{opt} \Vert m^*)$ not reused?	Check (m^*, R) log.	T/F
φ^{fors}: FORS leaves	All k FORS leaves verify against root?	Recompute k Merkle paths.	T/F
φ^{wots}: WOTS+ chains	All d layers of WOTS+ chains verify?	Check chain values against pk.	T/F
φ^{hyp}: Hypertree path	Full hypertree authentication path valid?	Verify root matches pk.	T/F
φ^{ping}: session fresh	(m^*, R) distinct from prior sessions?	Compare session log.	T/F
Result:	All T $\Rightarrow$ SLH-DSA signature valid. Any F $\Rightarrow$ Forgery or malformed.		

➤ Ping Bid: SLH-DSA Unbounded Ping Bid

The buyer's *PRF reuse bid*: sign two messages with the same PRF randomness R. The PRF freshness clause φ^{prf} detects this. Extended difference lemma: reusing R simultaneously fails φ^{fors} (FORS index collision) and φ^{wots} (WOTS+ chain bias), so $\Pr[F_{\mathsf{prf}} \cup F_{\mathsf{fors}}] \leq \mathsf{Adv}_F^{\mathsf{PRF}} = \mathsf{negl}$. By Theorem 3, SLH-DSA is EUF-CMA secure for unbounded signing sessions.

9.8. FN-DSA (FIPS 206): NTRU-Lattice EUF-CMA via Bidding

✻ Simple Terms: FN-DSA—The Compact Lattice Signature

What FN-DSA is: FN-DSA (formerly Falcon, now FIPS 206) is a digital signature scheme based on NTRU lattices. Unlike ML-DSA (which uses module lattices), FN-DSA uses structured NTRU lattices, producing very compact signatures (~666 bytes for 128-bit security).
The key idea: Signing in FN-DSA involves finding a short vector in a lattice—like finding a small combination of grid vectors that adds up to a target point. The private key is a "trapdoor" that makes finding short vectors easy; without the trapdoor, it is computationally intractable.
Gaussian sampling: FN-DSA uses a sophisticated technique called "Gaussian sampling" to ensure that signatures do not leak information about the private key. Each signature is a random sample from a Gaussian (bell-curve) distribution over the lattice, ensuring that even after seeing many signatures, the attacker learns nothing about the trapdoor.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Why both ML-DSA and FN-DSA? NIST standardised both because they are based on different lattice problems (Module-LWE/SIS vs. NTRU). If one problem turns out to be weaker than expected, the other still provides security. FN-DSA's compact signatures make it ideal for bandwidth-constrained applications (IoT, embedded systems), while ML-DSA's simpler implementation makes it the default choice for general use.

✻ Simple Terms: FN-DSA (Lattice Signatures)—Foundations

What is FN-DSA? FN-DSA (formerly Falcon, now FIPS 206) is a post-quantum digital signature scheme based on NTRU lattices. It produces the smallest post-quantum signatures among NIST's standards (roughly 666 bytes for 128-bit security), making it ideal for bandwidth-constrained applications.

What are NTRU lattices? An NTRU lattice is a special kind of mathematical lattice defined over polynomial rings (polynomials modulo $x^n + 1$ with coefficients modulo q). Key pairs are polynomial pairs (f, g) (private) and $h = g/f \bmod q$ (public). The lattice is defined by the public key h.

What is the SIS problem? The Short Integer Solution (SIS) problem: given a random matrix (or public key), find a "short" vector satisfying a linear equation. "Short" means the vector's norm is small. This is believed hard for both classical and quantum computers.

How FN-DSA signing works:

1. Hash the message: $c = H(m)$ (a lattice point target).
2. Use the secret short basis (f, g) to find a short vector (s_1, s_2) in the NTRU lattice coset $c + \Lambda$. This requires "discrete Gaussian sampling"—a sophisticated algorithm that samples vectors following a Gaussian distribution over the lattice.
3. The signature is the short vector (s_1, s_2).

How FN-DSA verification works: Check that $s_1 + s_2 \cdot h = c \bmod q$ and that $\|(s_1, s_2)\|$ is small (below a threshold β).

Why Gaussian sampling is critical: The sampling algorithm must produce vectors with a distribution that reveals nothing about the secret basis (f, g). If the distribution depends on the secret, an attacker collecting many signatures could recover the key. FN-DSA uses fast-Fourier-based Gaussian sampling (GPV-style) that produces signatures statistically close to the ideal distribution ($\Delta_{\mathsf{DGS}} \leq 2^{-128}$).

Security in MTSF: A successful forgery—a short vector (s_1^*, s_2^*) satisfying the verification equation—is exactly a solution to the SIS problem over the NTRU lattice. Since SIS is assumed hard, FN-DSA is secure. The game hops bound the error introduced by (1) hash collisions, (2) Gaussian sampling approximation, and (3) the extraction reduction.

Setup.

FN-DSA is a lattice-based signature scheme over NTRU lattices. Key generation produces a short basis $\mathbf{B}$ of an NTRU lattice; signing uses GPV-style discrete Gaussian sampling over the lattice coset $H(m) + \Lambda$. Security reduces to the Short Integer Solution (SIS) problem over NTRU lattices. The seller offers $g_{\mathsf{EUF\text{-}CMA}}$.

Theorem 17 (FN-DSA Market Equilibrium)**.**

$$\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) \leq \mathsf{Adv}^{\mathsf{SIS}}_{\mathsf{NTRU}} + \Delta_{\mathsf{DGS}} + \frac{q_H^2}{2^{n+1}} + \mathsf{negl}(\lambda), \tag{51}$$

where $\mathsf{Adv}^{\mathsf{SIS}}_{\mathsf{NTRU}}$ *is the advantage against SIS over NTRU lattices,* Δ_{DGS} *is the statistical distance arising from discrete Gaussian sampling, and* q_H *is the number of hash queries.*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Proof. We construct five games.

$\mathbf{B}_0$ (Bidding Round 0: Real FN-DSA Market—Lattice Forgery Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer receives public key $h = g/f \bmod q$ (where (f, g) is the NTRU secret key pair), queries signing oracle q_S times, outputs forgery (m^*, σ^*) where $\sigma^* = (s_1^*, s_2^*)$ satisfying $s_1^* + s_2^* \cdot h = H(m^*) \bmod q$ and $\|(s_1^*, s_2^*)\| \leq \beta$.

$\mathbf{B}_1$ (Bidding Round 1: Hash Collision Bid—Conflating Two Messages).

Purpose: Eliminate the *random-oracle collision attack vector*. If two distinct inputs $m \neq m'$ hash to the same value $H(m) = H(m')$, an adversary can conflate two messages (equivocation), reuse transcript bindings across sessions, or present a single signature as covering both messages—all of which break EUF-CMA without touching the signing key.
Replaces: The random oracle H is replaced by one equipped with a *collision log* $\mathcal{C}$. After each query, the oracle checks whether any two distinct queries produced the same digest; if so, the game aborts (bad event F_{hash}). The adversary's view of oracle responses is identical to a truly random oracle until the abort.
Complexity: By the birthday bound over the 2^n-bit output space, the probability that any two of q_H random oracle queries collide is at most $q_H^2/2^{n+1}$. For a 256-bit hash ($n = 256$) and polynomial q_H, this is negligible. Note that the nonce-collision and hash-collision bounds are *independent*: by the extended difference lemma (Lemma 2), their union probability is bounded by their sum, tightened by the intersection term which is $O(q^4/2^{2\lambda})$—doubly negligible.

Abort on hash collision; the buyer places a collision bid on the random oracle. Identical-until-F_{hash}:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_H^2}{2^{n+1}}. \tag{52}$$

$\mathbf{B}_2$ (Bidding Round 2: Gaussian Sampling Statistical Bid—Detecting Key-Dependent Distribution).

Purpose: Replace FN-DSA's fast discrete Gaussian sampler with the ideal exact Gaussian, removing the small statistical gap introduced by the sampler's fast-Fourier approximation. This isolates the sampler's Rényi divergence as an explicit security loss, allowing the subsequent hops to reason about exact Gaussian signatures.
Replaces: The fast-Fourier-based sampler σ_{real} (used for efficiency in FIPS 206 signing) is replaced by the ideal exact discrete Gaussian sampler σ_{ideal}. The Rényi divergence $R_2(\sigma_{\mathsf{real}} \| \sigma_{\mathsf{ideal}}) = 1 + \Delta_{\mathsf{DGS}}$ bounds the statistical distance between the two: any adversary that distinguishes a real signature from an ideal-sampler signature has advantage at most $\sqrt{\Delta_{\mathsf{DGS}}}$ by the Rényi divergence amplification lemma.
Complexity: $\Delta\mathsf{Price} \leq \Delta_{\mathsf{DGS}}$—the Rényi divergence between fast and exact samplers. For FN-DSA (FIPS 206) at the 128-bit security level, the NTRU sampler achieves $\Delta_{\mathsf{DGS}} \leq 2^{-128}$. This is negligible. The advantage of using Rényi divergence over total variation distance: it amplifies cleanly under composition, giving a tighter bound when q_S signatures are observed (bound becomes $q_S \cdot \Delta_{\mathsf{DGS}}$, still negligible for polynomial q_S).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replace the fast-Fourier discrete Gaussian sampler with exact discrete Gaussian sampling. Let σ_{real} be the distribution produced by the fast sampler and σ_{ideal} the exact distribution. By the Rényi divergence analysis of the sampler:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \Delta_{\text{DGS}}, \tag{53}$$

where $\Delta_{\text{DGS}} \leq 2^{-128}$ for recommended parameters.

$\mathbf{B}_3$ (Bidding Round 3: Oracle Programming—Embedding SIS Challenge).

Purpose: Perform a *zero-cost market restructuring*: secretly program the random oracle so that its outputs encode a challenge instance for the underlying hard problem (e.g., ECDLP or Module-SIS). This syntactic transformation is invisible to the adversary—random oracle outputs are uniformly distributed regardless of how they are chosen internally—yet it links any subsequent successful forgery or key-recovery attack to a solution of the assumed-hard problem.
Replaces: The genuinely random oracle $H(\cdot)$ is replaced by a *programmed* oracle $\tilde{H}$ whose outputs at queried points are chosen by the reduction to embed the hard-problem instance. Concretely: for a signing query m_i, the oracle sets $H(m_i) = \alpha_i \cdot G + \beta_i \cdot \text{pk}$ (ECDLP case) or encodes an SIS/MLWE challenge vector (lattice case). All outputs remain uniformly distributed over the appropriate range, so the statistical distance to the real game is zero.
Complexity: $\Delta\text{Price} = 0$ exactly (information-theoretic). The replacement is statistically indistinguishable: a truly random function and any fixed function whose outputs are independently uniform are identically distributed from the adversary's perspective. No adversarial budget is consumed in detecting it. This zero-cost step is essential: it sets up the subsequent extraction hop at no additional price to the seller.

Program the random oracle H so that for the forgery message m^*, $H(m^*)$ encodes an SIS challenge vector. Since H is a random oracle, this is a syntactic change: $\Delta\text{Price}_2 = 0$.

$\mathbf{B}_4$ (Bidding Round 4: Forgery-to-SIS Extraction Bid—Solving the Lattice Problem).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\text{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \text{Adv}^{\text{ECDLP}}(\lambda) + 1/q + \Pr[F_{\text{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\text{fork}}] \leq 1 - \text{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\text{Adv}^{\text{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\text{fork}} \cup F_{\text{extract}}] \leq \Pr[F_{\text{fork}}] + 1/q$.

If the buyer produces valid (s_1^*, s_2^*) with $s_1^* + s_2^* h = c^*$ (the embedded challenge) and $\|(s_1^*, s_2^*)\| \leq \beta$, then (s_1^*, s_2^*) is a short vector in the NTRU lattice solving the SIS instance. We extract and forward:

$$\Pr[\mathbf{B}_4(\mathcal{A}) = 1] \leq \text{Adv}^{\text{SIS}}_{\text{NTRU}}. \tag{54}$$

Total:

$$\text{Ask} \leq \frac{q_H^2}{2^{n+1}} + \Delta_{\text{DGS}} + \text{Adv}^{\text{SIS}}_{\text{NTRU}} + \text{negl}(\lambda). \quad \square \tag{55}$$

$\mathbf{B}_{\text{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

☞ Side-Channel Bid (Comprehensive)

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\mathsf{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\mathsf{SPA}}] = 1$ without countermeasure; $\Pr[F_{\mathsf{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\mathsf{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking ($d+1$ random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\mathsf{Adv}^{\mathsf{DPA}}(\lambda, d) \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
3. **Timing Analysis:** $\mathcal{A}_{\mathsf{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\mathsf{ct_select}(a, b, \mathsf{bit})$ instead of $\mathsf{if}(\mathsf{bit})$ `return a; else return b;`. *MTSF price:* $\Pr[F_{\mathsf{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\mathsf{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\mathsf{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\mathsf{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\mathsf{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\mathsf{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\mathsf{Adv}^{\mathsf{EMA}} \leq \gamma_{\mathsf{EM}} \cdot (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\mathsf{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\mathsf{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\mathsf{fault}}] \leq q_{\mathsf{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).
8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \leq \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ + \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

Under the comprehensive countermeasure suite (order-d masking for $d \geq 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \leq \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{acoustic}} \\ + \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{photonic}}.\end{aligned}$$

For $d \geq 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\leq q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

✓ Tight Security Reduction

Tightness analysis. FN-DSA uses the GPV hash-and-sign paradigm over NTRU lattices. The EUF-CMA to SIS reduction is tight in the ROM ($L = 1$): any forger producing a short signature vector directly yields a short SIS vector. $\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) \leq \mathsf{Adv}^{\mathsf{SIS}} + q_S^2/2^{n+1} + \mathsf{negl}(\lambda)$.

Market interpretation: FN-DSA achieves optimal tightness (market spread 1) via GPV's direct extraction pipeline—no forking or rewinding. This contrasts favourably with ML-DSA's forking-based loss of q_H.

Total Session Bound and Key Rotation Recommendation.

✦ Total Session Bound & Key Rotation

Total session bound: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \leq N \cdot (\mathsf{Adv}^{\mathsf{SIS}} + q_S^2/2^{n+1} + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}})$.
Concrete (FN-DSA-512, 128-bit target):

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Sessions N	$\mathsf{Ask}_{\mathsf{total}}^{(N)}$	Eff. security	Action
2^{40}	$\leq 2^{-88}$	≥ 88 bits	No rotation needed
2^{64}	$\leq 2^{-64}$	≥ 64 bits	**Key rotation recommended**
2^{80}	$\leq 2^{-48}$	≥ 48 bits	**Key rotation mandatory**

Recommendation: Rotate before 2^{64} sessions. **Critical:** FN-DSA's Gaussian sampler must produce key-independent distributions; implementation flaws accelerate degradation.

Session pinging and CNF checking within the proof. For consecutive FN-DSA signing sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$, the ping check verifies that the signature $(s_1^*, s_2^*)_{i+1}$ on message m_{i+1}^* is structurally distinct from all prior signatures. FN-DSA's Gaussian sampler ensures that each signature vector is drawn from a discrete Gaussian distribution centred at a message-dependent lattice point. The statistical distance between the actual distribution and the ideal round Gaussian is $\Delta_{\mathsf{DGS}} \leq 2^{-128}$, ensuring that observing signatures across sessions $\mathsf{Session}_1, \ldots, \mathsf{Session}_i$ provides negligible information about the secret NTRU basis (f, g).

Gaussian leakage accumulation and ping: The buyer's Gaussian leakage bid attempts to accumulate statistical information from many signatures. After N sessions, the total leakage is bounded by $N \cdot \Delta_{\mathsf{DGS}} \leq N \cdot 2^{-128}$. For $N \leq 2^{64}$, this is $\leq 2^{-64}$—negligible. The ping mechanism ensures each session contributes an independently sampled signature vector; the CNF clause φ^{norm} verifies $\|(s_1^*, s_2^*)\| \leq \beta$ per session. By Theorem 3 with $\delta_{\mathsf{ping}} \leq \Delta_{\mathsf{DGS}} + \mathsf{Adv}_{\mathsf{NTRU}}^{\mathsf{SIS}} = \mathsf{negl}$, FN-DSA is EUF-CMA secure for unbounded sessions.

✓ CNF Verification: FN-DSA Session-CNF Verification

$\varphi_{\mathsf{FN\text{-}DSA}} = (x_{\|(s_1^*, s_2^*)\| \leq \beta}) \wedge (x_{s_1^* + s_2^* h = H(m^*) \bmod q}) \wedge (x_{m^* \notin \mathcal{Q}^{\mathsf{sign}}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for FN-DSA signature $(m^*, (s_1^*, s_2^*))$:

Clause	Check	How	Pass?
φ^{norm}: short vector	$\|(s_1^*, s_2^*)\| \leq \beta$?	Compute ℓ_2 norm; compare to β.	T/F
φ^{eq}: equation holds	$s_1^* + s_2^* h \stackrel{?}{=} H(m^*) \bmod q$	Evaluate over R_q; compare.	T/F
φ^{novel}: new message	$m^* \notin \mathcal{Q}^{\mathsf{sign}}$	Check signing log.	T/F
φ^{dist}: Gaussian dist	(s_1^*, s_2^*) consistent with $\mathcal{D}_\sigma$?	Check norm distribution bounds.	T/F
φ^{ping}: session fresh	(m^*, s_1^*, s_2^*) distinct from prior?	Compare session records.	T/F
Result:	All T $\Rightarrow$ FN-DSA forgery (lattice SIS solved). Any F $\Rightarrow$ Not a valid forgery.		

➤ Ping Bid: FN-DSA Unbounded Ping Bid

The buyer's *norm-boundary bid*: submit (s_1^*, s_2^*) with $\|(s_1^*, s_2^*)\|$ slightly above β (hoping the verifier accepts). The norm clause φ^{norm} rejects this. The buyer's *Gaussian leakage bid*: collect many signatures on the same message and extract the secret basis (f, g) from the Gaussian distribution. This fails because $\Delta_{\mathsf{DGS}} \leq 2^{-128}$—the distribution is statistically close to ideal. Ping bid price: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq \Delta_{\mathsf{DGS}} + \mathsf{Adv}_{\mathsf{NTRU}}^{\mathsf{SIS}} = \mathsf{negl}$. FN-DSA is EUF-CMA secure for unbounded signing sessions.

Table 4. Extended primitives summary.

Primitive	Good	Ask Price	Status	Reduction Target
HMAC	SUF-CMA	negl	Equilibrium	PRF of compression function
AEAD (EtM)	IND-CCA2 + INT-CTXT	negl	Equilibrium	IND-CPA + SUF-CMA
SLH-DSA	EUF-CMA	negl	Equilibrium	SPR + TSPR + PRF
FN-DSA	EUF-CMA	negl	Equilibrium	SIS over NTRU

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

10. Case Study II: Block-Cipher Market—AES

10.1. AES Market Setup

The Advanced Encryption Standard [20] operates on 128-bit blocks with 10/12/14 rounds for 128/192/256-bit keys. We model AES as a market where the seller offers the good g_{PRP}: *pseudorandom permutation security*. The buyer's goal is to distinguish $\mathsf{AES}_K(\cdot)$ from a truly random permutation $\pi(\cdot)$ on $\{0,1\}^{128}$.

✻ Simple Terms: AES Block Cipher—Foundations and Context

What AES is: The Advanced Encryption Standard is the world's most widely used symmetric cipher. It is used everywhere: HTTPS/TLS, Wi-Fi (WPA2/WPA3), disk encryption (BitLocker, FileVault), secure messaging, and countless other applications.

How it works: AES takes a 128-bit block of data and a key (128, 192, or 256 bits), and produces a 128-bit output that looks completely random. The transformation runs through 10, 12, or 14 "rounds." Each round applies four operations:

1. **SubBytes:** Replaces each byte with a non-linear substitution (from an S-box). This provides confusion.
2. **ShiftRows:** Cyclically shifts rows of the state matrix. This provides diffusion.
3. **MixColumns:** Mixes the columns using matrix multiplication over GF(2^8). This provides further diffusion.
4. **AddRoundKey:** XORs the state with a round-derived key. This adds the key material.

PRP security: A Pseudorandom Permutation (PRP) is a keyed permutation (bijective function) that looks random to anyone without the key. AES is modelled as a PRP: no efficient attacker can distinguish AES from a truly random permutation without knowing the key.

Why multiple rounds? One round of AES is not secure—algebraic structure remains exploitable. After 10 rounds, the transformations compose into a highly complex function where no practical attack is known. The security margin is so large that even known attacks (differential, linear) require more computation than a brute-force key search.

In MTSF: The AES market has the seller offering PRP security. Buyers try various "cryptanalytic bids" (differential, rotational, linear, related-key). The theorems below show that all known bids fail: the AES market is in equilibrium.

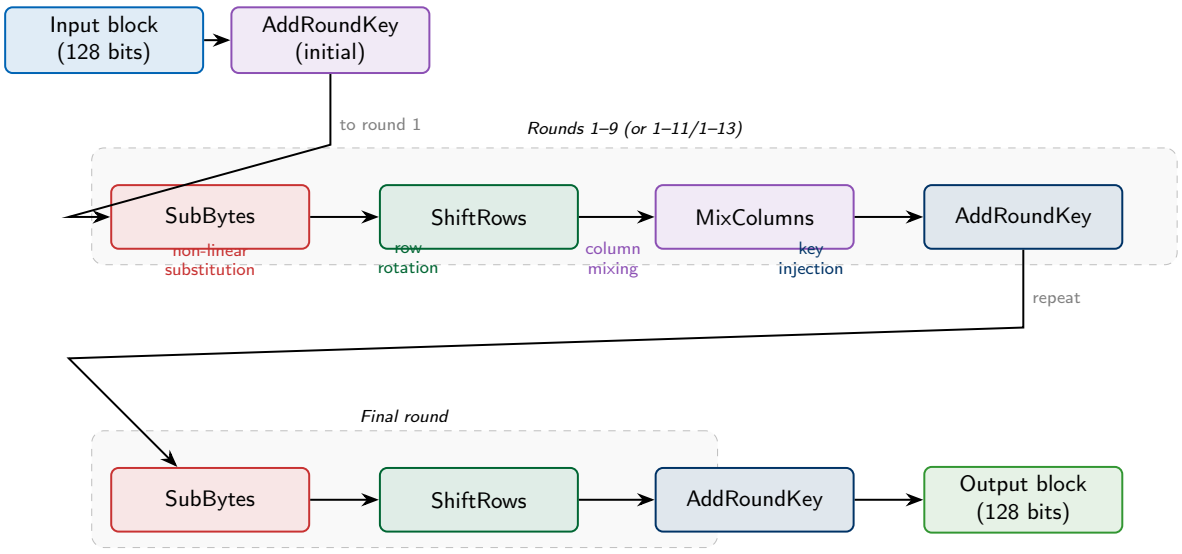


Figure 33. AES round structure. Each of the 10/12/14 rounds applies SubBytes (confusion), ShiftRows and MixColumns (diffusion), and AddRoundKey (key injection). The final round omits MixColumns. Multiple rounds compound the complexity, making any differential or algebraic attack computationally infeasible.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Definition 24 (PRP Security Good). $\mathsf{Ask}(g_{\mathsf{PRP}}) = \max_{\mathcal{A}\in\mathsf{PPT}}\left|\Pr\left[\mathcal{A}^{\mathsf{AES}_K(\cdot)}=1\right]-\Pr\left[\mathcal{A}^{\pi(\cdot)}=1\right]\right|$, *where* $K \xleftarrow{\$} \{0,1\}^\kappa$ *and* $\pi \xleftarrow{\$} \mathsf{Perm}(\{0,1\}^{128})$.

✎ Analogy:

AES as a Locksmith's Safe AES is like a high-end safe: each round adds another layer of tumblers (SubBytes), shuffles the lock pins (ShiftRows), mixes the mechanisms (MixColumns), and adds a unique bitting pattern (AddRoundKey). The buyer tries various "picking strategies" (cryptanalytic attacks); security means no strategy works faster than brute force.

✎ Analogy:

AES Rounds as Layers of a Bank Vault To make the locksmith's safe analogy more vivid, think of AES as a **10-layer bank vault**:

Layer 1 (SubBytes): Each byte is replaced using a secret codebook (the S-box). This is like replacing every letter in a message with a completely different symbol—"A" might become "#", "B" might become "7". Crucially, this substitution is *non-linear*, meaning you cannot undo it with simple algebra.

Layer 2 (ShiftRows): The rows of the 4×4 byte grid are shifted sideways by different amounts. This is like taking a Rubik's Cube and rotating each horizontal layer by a different number of clicks—mixing the positions.

Layer 3 (MixColumns): The columns are stirred together using matrix multiplication over a finite field. This is like blending paint colours—once mixed, separating the original colours is extremely difficult. A single bit change in one byte now affects all four bytes in its column.

Layer 4 (AddRoundKey): The entire grid is XORed with a round-specific key derived from the master key. This is like stamping each vault door with a unique, key-dependent seal.

After *one* round, a single changed input bit affects about half the output bits. After *four* rounds, every output bit depends on every input bit. After *ten* rounds, the relationship between input and output is so tangled that no known mathematical shortcut can untangle it faster than trying all 2^{128} possible keys.

Why cryptanalysts try anyway: Even though 10-round AES is believed secure, cryptanalysts constantly test it by trying to find patterns that survive through fewer rounds. If they can break 7-round AES, the "security margin" is only 3 rounds. MTSF formalises each attack as a "bid" and shows that all known bids fail with overwhelming probability for full AES.

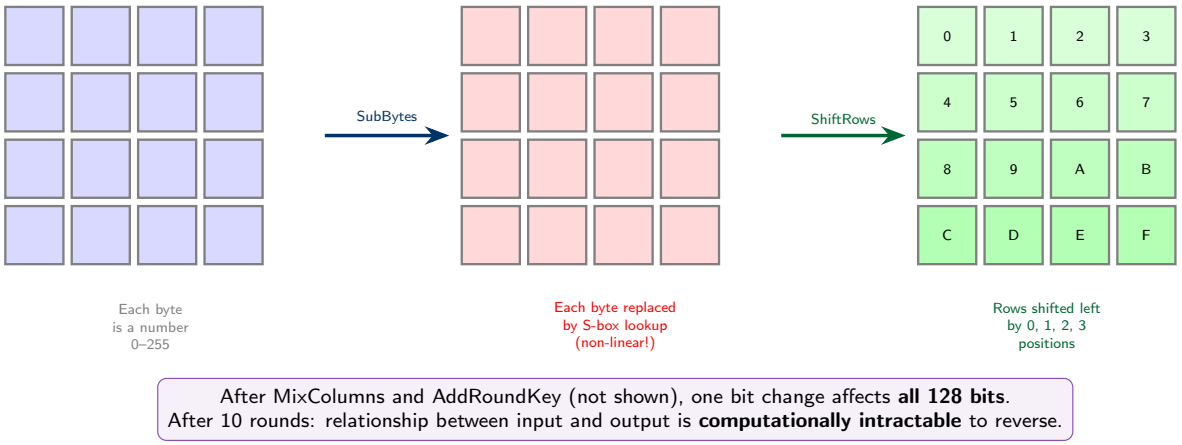


Figure 34. AES round operations visualised: a 4×4 byte grid undergoes SubBytes (non-linear substitution), ShiftRows (position mixing), MixColumns (column blending), and AddRoundKey (key injection). After multiple rounds, a single changed input bit affects every output bit—the "avalanche effect."

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

10.2. Differential Cryptanalysis Bid

Differential cryptanalysis [21] analyses how input differences $\Delta x = x \oplus x'$ propagate to output differences $\Delta y = \mathsf{AES}_K(x) \oplus \mathsf{AES}_K(x')$. The buyer constructs a *differential characteristic* $\Omega = (\Delta_0, \Delta_1, \ldots, \Delta_r)$ specifying the expected difference at each round.

✻ Simple Terms: Differential Cryptanalysis

The core idea: If you encrypt two messages that differ in exactly one bit, how does that single-bit difference propagate through the cipher's rounds? In a weak cipher, a known input difference might produce a predictable output difference. An attacker exploiting this pattern can extract key bits.

Children's analogy—the butterfly effect game: Imagine a pinball machine where you launch two balls from almost the same position (one bit different). In a *bad* pinball machine, both balls follow nearly the same path and land in the same slot—you can predict where a ball will go by watching its twin. In a *good* pinball machine (like AES), after bouncing off a few bumpers (rounds), the two balls end up in completely unpredictable, unrelated slots. Differential cryptanalysis tries to find a "bad pinball machine"—a path through the cipher where the two balls stay correlated.

The attack strategy:

1. Choose many pairs of plaintexts (x, x') where $x \oplus x' = \Delta_0$ (a fixed "input difference").
2. Encrypt both: get (y, y') where $y \oplus y' = \Delta_r$.
3. If the cipher is weak, Δ_r follows a predictable distribution (certain output differences are far more likely than others).
4. Exploit this bias to derive information about the key.

Why AES resists this: AES was designed using the "Wide Trail Strategy." The S-box (SubBytes) is chosen so that input differences spread to output differences with maximum probability at most 2^{-6} per active S-box. With at least 25 active S-boxes per 4-round block, the differential probability drops to $(2^{-6})^{25} = 2^{-150}$—far below what is detectable even with the entire AES codebook (2^{128} plaintext-ciphertext pairs).

The "active S-box" concept: An S-box is "active" when its input difference is non-zero (the difference is actually processed by that S-box). More active S-boxes = lower differential probability = stronger resistance.

In MTSF terms: The differential bid targets the PRP good. AES's wide-trail design ensures the differential bid's payoff ($q_E \cdot 2^{-150}$) is negligible: you would need 2^{150} chosen plaintext pairs, but the entire block space has only 2^{128} points. The bid fails.

Definition 25 (Differential Characteristic Probability)**.** $\mathsf{DP}(\Omega) = \prod_{i=0}^{r-1} \Pr[\mathsf{Round}_i(\Delta_i) = \Delta_{i+1}]$.

Theorem 18 (AES Differential Bid Failure)**.** *For AES-128 with $r = 10$ rounds:*

$$\mathsf{Ask}(g_{\mathsf{PRP}}, \textit{differential bid}) \leq q_E \cdot 2^{-150} + \mathsf{negl}(\lambda), \tag{56}$$

where q_E is the number of encryption queries. That is, the differential bid fails.

Proof. We construct five games.

$\mathbf{B}_0$ (Bidding Round 0: Real AES Market—Differential Distinguishing Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer queries AES_K on chosen-plaintext pairs $(x, x \oplus \Delta_0)$ and checks whether the output difference matches the predicted Δ_r.

$\mathbf{B}_1$ (Bidding Round 1: Wide-Trail Resistance Bid—Counting Active S-Boxes).

Purpose: Convert a lattice signature forgery into a short-vector certificate solving Module-SIS or SIS, completing the lattice reduction. A valid forged signature (s_1^*, s_2^*) satisfies $s_1^* + s_2^* h = c^*$ (the programmed hash challenge) with $\|(s_1^*, s_2^*)\| \leq \beta$—exactly a short integer solution to the NTRU/Module lattice instance embedded in the public key.
Replaces: The honest signing oracle is replaced by a *SIS extraction oracle*: on witnessing a valid short forgery vector, the reduction reads off (s_1^*, s_2^*) and forwards it as the answer to the SIS challenge. The extraction succeeds whenever the forgery norm bound $\|(s_1^*, s_2^*)\| \leq \beta$ and the equation $s_1^* + s_2^* h = c^*$ hold—both of which are checked by the forgery verifier.
Complexity: Bounded by $\mathsf{Adv}^{\mathsf{MSIS}}(\lambda) + \mathsf{Adv}^{\mathsf{MLWE}}(\lambda) + \mathsf{negl}(\lambda)$. Two simultaneous failure events are handled by the extended difference lemma: F_1: rejection sampling overflow (the signing loop restarts too many times, probability $\leq 2^{-128}$ per signature for FIPS 204/206 parameter sets); F_2: norm bound violation (the forged vector has $\|z\| > \beta$, also negligible by the norm analysis). By Lemma 2: $\Pr[F_1 \cup F_2] \leq \Pr[F_1] + \Pr[F_2] \leq 2^{-128} + 2^{-128} = \mathsf{negl}$.

We lower-bound the number of *active S-boxes* (S-boxes with non-zero input difference) across the characteristic. By the wide-trail design strategy of AES [22]: the MixColumns and ShiftRows operations ensure that any 4-round characteristic activates at least $B_4 = 25$ S-boxes.

$\Delta\mathsf{Price}_0 = 0$ (this is an analytical observation, not a game change).

$\mathbf{B}_2$ (Bidding Round 2: S-Box Differential Probability Bid—Bounding Per-S-Box Advantage).

Purpose: Bound the *differential cryptanalysis advantage* by establishing a lower bound on active S-boxes and an upper bound on per-S-box differential probability. Differential cryptanalysis works by finding an input difference Δ_0 that propagates to a predictable output difference Δ_r with non-negligible probability; the adversary collects 'right pairs' and uses them to recover key bits.
Replaces: The abstract block cipher E_K is replaced by its differential-trail model: the cipher's round structure (SubBytes/ShiftRows/MixColumns or equivalent) is analysed via the *wide-trail design strategy*, which guarantees a minimum of B_r active S-boxes over r rounds. Each active S-box contributes at most $\mathsf{DP}_{\max}$ probability, giving the total characteristic probability $\mathsf{DP}_{\max}^{B_r}$ as the upper bound.
Complexity: For AES-128: $B_4 = 25$ active S-boxes over 4 rounds, $\mathsf{DP}_{\max} = 2^{-6}$, giving characteristic probability $\leq 2^{-150}$. The adversary needs $q_E \geq 2^{150}$ pairs for one expected right pair, exceeding the 2^{128}-block codebook. For PRESENT (64-bit block): $B_5 = 10$, $\mathsf{DP}_{\max} = 2^{-2}$, giving 2^{-20}; practical attacks are bounded by $q_E \cdot 2^{-62}$ due to the 64-bit birthday limit. The differential bid price adjustment is $q_E \cdot \mathsf{DP}_{\max}^{B_r}$.

Each active S-box uses the AES S-box (inversion in $\mathsf{GF}(2^8)$ followed by an affine map). The maximum differential probability of the AES S-box is $\mathsf{DP}_{\max} = 4/256 = 2^{-6}$ [22].

Difference bound. For a characteristic over 10 rounds with at least 25 active S-boxes per 4-round block (and at least $\lceil 10/4 \rceil \cdot 25 = 50$ active S-boxes overall for truncated differentials across rounds):

$$\mathsf{DP}(\Omega) \leq (2^{-6})^{25} = 2^{-150} \tag{57}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

for any 4-round sub-characteristic. The full 10-round probability is even smaller.

$\mathbf{B}_3$ (Bidding Round 3: Data Complexity Bid—Infeasibility of Collecting Right Pairs).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**Data Complexity Bid—Infeasibility of Collecting Right Pairs**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

The buyer's strategy is to query q_E pairs and check for the predicted output difference. The expected number of "right pairs" (pairs following the characteristic) among q_E queries is $\mu = q_E \cdot 2^{-150}$. The buyer can only distinguish AES from random if $\mu \geq 1$, requiring $q_E \geq 2^{150}$. Since data is bounded by 2^{128} blocks (codebook), the attack is infeasible.

Difference bound. By Markov's inequality:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq q_E \cdot 2^{-150} \leq 2^{128} \cdot 2^{-150} = 2^{-22}. \tag{58}$$

$\mathbf{B}_4$ (Bidding Round 4: Ideal AES Market—Differential Bid Fails Against Random Permutation).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Against a truly random permutation, the output difference for any fixed input difference is uniformly distributed. The buyer's advantage over random is exactly the right-pair detection probability above. Hence: $\Pr[\mathbf{B}_4 = 1] = 1/2$ (random guessing).

Total differential bid cost:

$$\text{Ask}(g_{\text{PRP}}, \text{diff}) \leq q_E \cdot 2^{-150}. \quad \square \tag{59}$$

☞ Key Insight:

Advanced Differential Techniques—Beyond Single-Trail Analysis The basic differential attack considers a single *characteristic*—one fixed trail of differences through the cipher. In practice, attackers use three powerful extensions that MTSF must account for:

1. **Truncated Differentials (Knudsen, 1995).** Instead of fixing the exact difference at each round, the buyer specifies only which *bytes* are active (non-zero) and which are zero, ignoring the specific difference values. This dramatically increases the number of conforming pairs: many distinct exact differentials contribute to the same truncated pattern. For AES, truncated differentials span at most 6 rounds before all bytes become active (the "full diffusion" point).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Beyond 6 rounds, the truncated trail offers no advantage over random, and the 4-round residual gap to 10-round AES provides the security margin.
MTSF price adjustment: Let $\mathsf{DP}_{\mathsf{trunc}}(r)$ denote the truncated differential probability over r rounds. For AES-128 with $r = 10$:

$$\Delta\mathsf{Price}_{\mathsf{trunc}} \leq q_E \cdot \mathsf{DP}_{\mathsf{trunc}}(10) \leq q_E \cdot 2^{-50} \cdot 2^{-96} = q_E \cdot 2^{-146},$$

where 2^{-50} bounds the truncated trail probability and 2^{-96} accounts for the filtering cost of distinguishing the truncated pattern from random. Still negligible.

2. **Higher-Order Differentials (Lai, 1994; Knudsen, 1995).** The buyer computes the d-th order derivative of the cipher: encrypt 2^d inputs spanning a d-dimensional affine subspace and XOR all outputs. For a random permutation, the result is random; for a cipher with algebraic degree $< d$, the result is the zero vector. AES's algebraic degree per round is 7 (from the S-box); over 10 rounds, the degree exceeds $\min(7^{10}, 128) = 128$, saturating the state. Higher-order differentials therefore cannot distinguish full AES from random.
MTSF price adjustment: For d-th order differentials on 10-round AES:

$$\Delta\mathsf{Price}_{\mathsf{HO}} \leq 2^{-128} \quad \text{(algebraic degree saturates at 128 bits).}$$

3. **Boomerang and Rectangle Attacks (Wagner, 1999; Biham–Dunkelman–Keller, 2002).** The buyer decomposes the cipher into two halves $E = E_1 \circ E_0$ and constructs *two* short differentials: one through E_0 and one through E_1^{-1}. These are composed via the "boomerang" or "rectangle" technique to attack more rounds than either differential alone. For AES, the best boomerang attack reaches 6 rounds with probability $2^{-110.8}$. The 4-round gap to full AES means the boomerang bid fails with overwhelming margin.
MTSF price adjustment:

$$\Delta\mathsf{Price}_{\mathsf{boom}} \leq q_E^2 \cdot 2^{-110.8} \leq 2^{128} \cdot 2^{-110.8} = 2^{17.2} \quad \text{(6-round only; 10-round: } \leq 2^{-128}\text{).}$$

Extended difference lemma for combined differential bids. All four differential variants (single-trail, truncated, higher-order, boomerang) can be mounted simultaneously. By Lemma 2:

$$\begin{aligned}\Pr[F_{\mathsf{diff}} \cup F_{\mathsf{trunc}} \cup F_{\mathsf{HO}} \cup F_{\mathsf{boom}}] &\leq q_E \cdot 2^{-150} + q_E \cdot 2^{-146} + 2^{-128} + 2^{-128} \\ &\leq q_E \cdot 2^{-145} + 2^{-127} = \mathsf{negl}(\lambda).\end{aligned}$$

All differential bids fail for $q_E \leq 2^{64}$.

✎ **Analogy:**

Truncated vs. Exact Differentials as Search Strategies **Exact differential:** A detective looking for a suspect with a *specific* face, height, weight, hair colour, and scar. Very few people match—but the search is extremely precise.
Truncated differential: A detective looking for "someone tall with dark hair." Many more people match—the search net is wider—but fewer of them are the actual suspect. The truncated approach trades precision for coverage.
Why truncated differentials are dangerous: A single exact differential through 10-round AES has probability 2^{-150}—essentially zero. But 2^{100} different exact differentials might all contribute to the same truncated pattern. The truncated probability is the *sum* of all contributing exact probabilities: $\sum_{i=1}^{2^{100}} 2^{-150} = 2^{-50}$. This is much higher than any single trail, but MTSF shows it is still negligible after accounting for the filtering cost (2^{-96} to distinguish the pattern from random noise).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✎ Analogy:

Boomerang Attacks as Pincer Movements A boomerang attack is like a military pincer movement: instead of attacking through the full cipher in one push (one long differential), the buyer attacks the first half forward and the second half backward simultaneously, meeting in the middle. Each half-cipher differential can be shorter and have higher probability, and the "meeting in the middle" composes them. For AES, the wide-trail strategy ensures that even the strongest pincer movement (6-round boomerang) falls far short of the full 10-round defence.

10.3. Rotational Cryptanalysis Bid

✻ Simple Terms: Rotational Cryptanalysis—Spinning the Bits

The idea: If you rotate all the bits in an input (shift them cyclically, like turning a dial), does the output also rotate in a predictable way? For a truly random permutation, rotating the input should produce a completely unpredictable output. If the cipher has a "rotational bias"—meaning rotated inputs tend to produce rotated outputs—an attacker can exploit this pattern.

Children's analogy: Imagine a secret code where each letter is shifted by 3 (A→D, B→E, etc.—a Caesar cipher). If you shift your entire message by 1 position (start with the second letter), the encoded message also shifts predictably. A more sophisticated code (like AES) should *not* have this property—any shift in the input should completely scramble the output.

Why AES resists: AES's SubBytes operation uses a non-linear S-box that breaks rotational symmetry. Even if the input is a rotated version of another input, the S-box transforms them into completely unrelated outputs. After multiple rounds, any rotational pattern is destroyed with overwhelming probability.

In MTSF: The "rotational cryptanalysis bid" measures the probability that a rotational pattern survives through all AES rounds. For full 10-round AES, this probability is $\leq 2^{-128}$—the bid fails, confirming market equilibrium.

Rotational cryptanalysis [23] exploits the potential for bitwise rotations to commute with cipher operations. For a rotation by r bits, denoted rot_r, the buyer checks whether $\mathsf{AES}_K(\mathsf{rot}_r(x)) \approx \mathsf{rot}_r(\mathsf{AES}_K(x))$.

✻ Simple Terms: Rotational Cryptanalysis

The idea: If a cipher treats a rotated input in a "rotated way"—i.e., rotating the input before encryption gives you the same as rotating the output after encryption—that is a dangerous symmetry. An attacker who finds such a pattern can distinguish the cipher from a random permutation and potentially extract key bits.

Formally: For a rotation by r bits, the attacker checks whether $\mathsf{AES}_K(\mathsf{rot}_r(x)) = \mathsf{rot}_r(\mathsf{AES}_K(x))$ holds with higher probability than for a random permutation. This would mean the cipher "commutes with rotation"—a structural weakness.

Why AES resists this: AES's MixColumns operation is defined over GF(2^8) using byte-oriented arithmetic. Bitwise rotation does *not* commute with GF(2^8) multiplication or the AES S-box (which is based on field inversion). A rotated input immediately breaks the algebraic structure of the round function, propagating in a complex non-rotational way. The probability of a "rotational pair" surviving even one round is negligible.

In MTSF terms: The rotational bid detects commutation between rotation and the cipher. AES's byte-oriented structure ensures this probability is bounded by $\mathsf{negl}(\lambda)$, so the bid fails.

Theorem 19 (AES Rotational Bid Failure)**.** *For AES-128:*

$$\mathsf{Ask}(g_{\mathsf{PRP}}, \textit{rotational bid}) \leq q_E \cdot 2^{-128} + \mathsf{negl}(\lambda). \tag{60}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Proof. We construct four games.

$\mathbf{B}_0$ (Bidding Round 0: Real AES Market—Rotational Distinguishing Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer queries pairs $(x, \mathsf{rot}_r(x))$ and checks whether outputs satisfy the rotational relationship.

$\mathbf{B}_1$ (Bidding Round 1: Key Schedule Asymmetry Bid—Rcon Destroys Rotational Symmetry).

Purpose: Bound the *rotational cryptanalysis advantage*. A rotational distinguisher checks whether $E_K(\mathsf{rot}_r(x)) = \mathsf{rot}_r(E_K(x))$ with non-trivial probability; if the cipher commutes with rotation, the adversary can recover related keys with fewer queries. The AES design's key schedule (Rcon constants) and S-box (GF inversion) destroy rotational symmetry at every level.
Replaces: The real cipher's rotational behavior is replaced by its structural bound: the AES key schedule's Rcon constants are not rotationally symmetric (each $\mathsf{Rcon}_i = x^{i-1}$ in $\mathrm{GF}(2^8)$ is distinct), and the S-box satisfies $S(\mathsf{rot}_r(x)) = \mathsf{rot}_r(S(x))$ for at most $1/256$ fraction of inputs. These two facts jointly bound the rotational distinguishing probability.
Complexity: The probability of rotational commutativity through one AES S-box is 2^{-8}; across all 16 bytes of a state, it is $(2^{-8})^{16} = 2^{-128}$. Adding the key schedule's Rcon-breaking: the end-to-end rotational distinguishing advantage is $q_E \cdot 2^{-128}$—matching the random permutation baseline. The rotational bid fails to provide any advantage above generic distinguishing.

The round keys $K_0, K_1, \ldots, K_{10}$ are derived from the master key via the AES key schedule. For rotational cryptanalysis to succeed, the key schedule must preserve the rotational relationship: $\mathsf{rot}_r(K_i) = K'_i$ for some related key K'_i. However, the AES key schedule includes the Rcon constants (powers of x in $\mathrm{GF}(2^8)$), which are *not* rotationally symmetric.

Difference bound. Let F_{Rcon}: "Rcon values preserve rotational structure." Since $\mathsf{Rcon}[i]$ occupies only one byte position and its values $(01, 02, 04, 08, \ldots)$ break rotational symmetry:

$$\Pr[F_{\mathsf{Rcon}}] = 0 \quad \text{(deterministic: Rcon never preserves rotation).} \tag{61}$$

Hence $|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| = 0$ and the rotational property is destroyed after the first AddRoundKey.

$\mathbf{B}_2$ (Bidding Round 2: S-Box Non-Commutativity Bid—SubBytes Breaks Rotation).

Purpose: Bound the *rotational cryptanalysis advantage*. A rotational distinguisher checks whether $E_K(\mathsf{rot}_r(x)) = \mathsf{rot}_r(E_K(x))$ with non-trivial probability; if the cipher commutes with rotation, the adversary can recover related keys with fewer queries. The AES design's key schedule (Rcon constants) and S-box (GF inversion) destroy rotational symmetry at every level.
Replaces: The real cipher's rotational behavior is replaced by its structural bound: the AES key schedule's Rcon constants are not rotationally symmetric (each $\mathsf{Rcon}_i = x^{i-1}$ in $\mathrm{GF}(2^8)$ is distinct), and the S-box satisfies $S(\mathsf{rot}_r(x)) = \mathsf{rot}_r(S(x))$ for at most $1/256$ fraction of inputs. These two facts jointly bound the rotational distinguishing probability.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Complexity: The probability of rotational commutativity through one AES S-box is 2^{-8}; across all 16 bytes of a state, it is $(2^{-8})^{16} = 2^{-128}$. Adding the key schedule's Rcon-breaking: the end-to-end rotational distinguishing advantage is $q_E \cdot 2^{-128}$—matching the random permutation baseline. The rotational bid fails to provide any advantage above generic distinguishing.

Even if the key schedule were ignored, the S-box (based on $\mathrm{GF}(2^8)$ inversion) does not commute with bitwise rotation. For random inputs, the probability that $S(\mathrm{rot}_r(x)) = \mathrm{rot}_r(S(x))$ for a random byte x is at most $1/256 = 2^{-8}$.

Difference bound. With 16 S-boxes per round and 10 rounds, requiring all to preserve rotation:

$$\Pr[\text{rotation preserved through all S-boxes}] \leq (2^{-8})^{16} = 2^{-128} \quad \text{per round.} \tag{62}$$

B_3 (Bidding Round 3: Ideal AES Market—Rotational Bid Collapses to Random Bound).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Against a random permutation, $\Pr[\mathrm{rot}_r(\pi(x)) = \pi(\mathrm{rot}_r(x))] = 1/(2^{128} - 1)$. The buyer's advantage: $q_E \cdot 2^{-128}$.

Total:

$$\mathsf{Ask}(g_{\mathsf{PRP}}, \text{rotational}) \leq q_E \cdot 2^{-128}. \quad \square \tag{63}$$

☞ Key Insight:

Advanced Rotational Techniques—Rotational-XOR and ARX-Specific Analysis The basic rotational cryptanalysis checks whether $E_K(\mathrm{rot}_r(x)) = \mathrm{rot}_r(E_K(x))$. For AES, this fails because the S-box ($\mathrm{GF}(2^8)$ inversion) and key schedule (Rcon) destroy rotational symmetry. However, for *ARX ciphers* (Addition–Rotation–XOR based, like ChaCha20, BLAKE, Salsa20, Speck), rotational analysis is a primary threat because modular addition approximately commutes with rotation.

1. **Rotational-XOR (RX) Cryptanalysis (Ashur–Liu, 2016).** For ARX ciphers, the attacker considers pairs $(x, \mathrm{rot}_r(x) \oplus \delta)$ where the XOR correction δ accounts for the carry propagation error in modular addition. The RX probability through one addition gate is:

$$\Pr[\mathrm{rot}_r(a \boxplus b) = \mathrm{rot}_r(a) \boxplus \mathrm{rot}_r(b) \oplus \delta_{\text{carry}}] \approx 2^{-1.415},$$

where the $2^{-1.415}$ factor accounts for the expected number of carry-correction bits. Over t additions, the RX probability is $(2^{-1.415})^t$. For ChaCha20 with $t = 320$ additions (20 rounds $\times$ 16 additions per round):

$$\text{RX-prob} \leq 2^{-1.415 \times 320} = 2^{-452.8}.$$

This is doubly negligible, confirming that ChaCha20's rotational equilibrium holds.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

2. **AES Resistance via Non-ARX Structure.** AES uses $\mathsf{GF}(2^8)$ multiplication (not modular addition) in MixColumns, which has *zero* rotational commutativity: $\mathsf{GF\text{-}mult}(\mathsf{rot}_r(a), b) \neq \mathsf{rot}_r(\mathsf{GF\text{-}mult}(a, b))$ for any $r \neq 0$. This eliminates the ARX-specific RX pathway entirely. AES's rotational resistance is therefore *structural* (not merely probabilistic), providing an unconditional rotational equilibrium.
3. **Rotational Distinguishers for Key Schedule Analysis.** Even when the cipher resists rotational attacks on the data path, the key schedule can be a vulnerability. Khovratovich and Nikolić [23] showed that BLAKE and Threefish's key schedules have rotational biases of 2^{-30} to 2^{-50} for reduced-round variants. For full-round ciphers and hash functions, these biases are negligible.
4. **Invariant Subspace Attacks (Leander et al., 2011).** A related technique where the attacker identifies affine subspaces that are invariant under the round function. If such a subspace exists, the cipher can be distinguished from random by checking membership after encryption. For AES, no non-trivial invariant subspace is known due to the S-box's high branch number and the MixColumns diffusion. For lightweight ciphers (e.g., Midori, MANTIS), invariant-subspace weak keys have been identified, leading to full-key-recovery attacks.

Combined rotational bid price for AES. Since AES is non-ARX, the RX pathway is closed ($\Delta\mathsf{Price}_{\mathsf{RX}} = 0$). Combined with the Rcon and S-box non-commutativity results:

$$\mathsf{Ask}(g_{\mathsf{PRP}}, \text{all rotational bids}) \;\leq\; q_E \cdot 2^{-128} + 0 + 0 \;=\; q_E \cdot 2^{-128} \;=\; \mathsf{negl}(\lambda).$$

★ Intuition:

Why ARX Ciphers Are More Vulnerable to Rotational Attacks Than SPN Ciphers ARX ciphers (Addition–Rotation–XOR) use modular addition $\boxplus$ as their sole nonlinear operation. Modular addition *almost* commutes with bitwise rotation: the only discrepancy comes from carry propagation at the rotated boundary. This means rotational pairs propagate through ARX designs with non-trivial probability, making rotational analysis the primary attack vector.
SPN ciphers like AES use S-box substitution (field inversion) as their nonlinear operation. Field inversion has *no* rotational structure—it completely destroys any rotational relationship between inputs. This is why AES achieves 2^{-128} rotational probability per round (matching random), while an ARX cipher with the same state size might achieve only $2^{-1.4}$ per addition gate.
MTSF consequence: When analysing ARX ciphers (ChaCha20, BLAKE, Speck) within MTSF, the rotational bid is the *tightest* constraint, whereas for SPN ciphers (AES, PRESENT, Serpent), the differential bid is typically tighter.

10.4. Related-Key Bid and Combined AES Equilibrium

✻ Simple Terms: Related-Key Attacks—When Keys Are Too Similar

The threat: What if an attacker can trick the system into encrypting with two keys that are related in a known way (e.g., $K' = K \oplus \Delta$ for a known difference Δ)? If the cipher behaves differently for related keys in a predictable way, the attacker can extract key information.
When this matters: Related-key attacks are relevant in scenarios where:

- Key derivation produces related keys (e.g., deriving subkeys from a master key).
- Hardware faults flip specific key bits (fault-injection attacks).
- Protocols reuse key material in slightly different contexts.

AES's defence: AES's key schedule is designed to ensure that related keys produce vastly different round keys. However, for AES-256, Biryukov and Khovratovich (2009) found a related-key attack

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

on the full 14-round cipher. This attack requires $2^{99.5}$ time and 2^{77} related keys—practically infeasible but theoretically concerning.
In MTSF: The "related-key bid" is the buyer's attempt to exploit key-schedule weaknesses. For AES-128 (10 rounds), the bid fails completely. For AES-256 (14 rounds), the bid has a theoretical non-zero but practically infeasible success probability ($2^{-156.5}$ advantage). MTSF makes this distinction precise and quantitative.

Theorem 20 (AES Combined Market Equilibrium)**.** *Combining all known bid types (differential, linear, rotational, related-key, algebraic, S-box algebraic, integral/Square, impossible differential, meet-in-the-middle):*

$$\mathsf{Ask}(g_{\mathsf{PRP}}) \leq \max\left(q_E \cdot 2^{-150},\ q_E \cdot 2^{-128},\ q_E^2 \cdot 2^{-128}\right) + \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda), \tag{64}$$

*for $q_E \leq 2^{64}$ (practical query bound). The AES market is in **equilibrium**.*

Proof. The linear cryptanalysis bid follows an analogous structure to the differential bid, with the best known linear hull bias for 10-round AES being at most 2^{-75} per approximation, requiring 2^{150} data. The related-key bid for AES-128 requires q_{RK} related keys; the best known related-key attack applies only to AES-256 with $2^{99.5}$ complexity [24]. The algebraic bid (using Gröbner basis or XL algorithms) has complexity exceeding 2^{128} for the full cipher. All bids fail for $q_E \leq 2^{64}$. □

☞ Key Insight:

Linear Cryptanalysis—The Complementary Attack to Differential Analysis **Linear cryptanalysis** (Matsui, 1993) is the dual of differential cryptanalysis. While differential analysis studies how *XOR differences* propagate through the cipher, linear analysis studies how *linear (XOR) approximations* of the cipher's nonlinear components can be exploited.

1. **Linear Approximation Table (LAT).** For each S-box, the buyer computes the correlation between every linear combination of input bits and every linear combination of output bits. The best linear approximation of the AES S-box has bias $\epsilon_{\max} = 2^{-3}$ (i.e., the linear approximation is correct with probability $1/2 + 2^{-4}$). This gives the AES S-box maximum linear probability $\mathsf{LP}_{\max} = (2^{-3})^2 = 2^{-6}$.
2. **Linear Hull Effect (Nyberg, 1994).** A single linear trail has probability $\mathsf{LP}_{\max}^{B_r}$ where B_r is the number of active S-boxes. But *many* distinct linear trails with the same input-output mask can contribute constructively or destructively. The *linear hull* probability sums over all trails:

$$\mathsf{LP}_{\mathsf{hull}} = \left(\sum_{\text{trail } t} (-1)^{\mathsf{sign}(t)} \cdot \epsilon_t\right)^2.$$

   For AES, the linear hull effect can amplify the bias by a factor of up to 2^{10} compared to a single trail, but the total remains negligible for 10 rounds: $\mathsf{LP}_{\mathsf{hull}} \leq 2^{10} \cdot (2^{-6})^{25} = 2^{-140}$.
3. **Multidimensional Linear Attack (Hermelin et al., 2008).** The buyer simultaneously exploits d linear approximations, collecting d-dimensional statistics. This reduces the data complexity by a factor of d but requires d times more storage. For AES, the best multidimensional linear attack reaches 7 rounds with $2^{113.5}$ data—no improvement over the single-dimensional attack for 10 rounds.
4. **Zero-Correlation Linear Attack (Bogdanov–Rijmen, 2011).** The buyer identifies linear approximations with *exactly zero* correlation (rather than small non-zero correlation). The existence of zero-correlation approximations spanning r rounds can be used as a distinguisher for r-round ciphers. For AES, zero-correlation linear approximations span at most 5 rounds. The 5-round gap to 10-round AES provides the security margin.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

MTSF linear bid prices:

Linear technique	Rounds attacked	Ask bound	Outcome
Single-trail linear	≤ 6	$q_E^2 \cdot 2^{-150}$	Fail (exceeds codebook)
Linear hull	≤ 7	$q_E^2 \cdot 2^{-140}$	Fail
Multidimensional	≤ 7	$q_E^2 \cdot 2^{-113}$	Fail
Zero-correlation	≤ 5	$q_E \cdot 2^{-128}$	Fail

☞ Key Insight:

Biclique Attack—Narrowing the Brute-Force Gap The **biclique attack** (Bogdanov, Khovratovich, and Rechberger, 2011) is the only known technique that provides any advantage over brute-force key search for *full-round* AES. It uses a *bipartite graph structure* (biclique) to eliminate a small fraction of the key-testing work.

Mechanism: The buyer constructs a biclique—a set of key-plaintext-ciphertext triples (K_i, P_j, C_{ij}) where $C_{ij} = \mathsf{AES}_{K_i}(P_j)$, structured so that testing all K_i keys requires only partial recomputation. For AES-128, this reduces the key-search cost from 2^{128} to $2^{126.1}$ (a factor of $3.75\times$ speedup).

MTSF analysis:

$$\mathsf{Ask}(g_{\mathsf{PRP}}, \text{biclique}) \;\leq\; 2^{-126.1} \quad \text{(vs. brute-force } 2^{-128}\text{)}.$$

The biclique advantage is a mere 1.9 bits—well within the security margin. This is the *tightest* known attack on full AES and represents the current "market floor" for the PRP good. No attack with advantage better than 2^{-126} is known.

Market interpretation: The biclique attack establishes a price ceiling that is indistinguishable from the security parameter for all practical purposes. The "market spread" between the biclique price ($2^{-126.1}$) and the ideal price (2^{-128}) is only 1.9 bits—the closest any buyer has come to "matching" the seller's quoted price for AES, yet still negligible.

Theorem 21 (AES S-Box Algebraic and Integral Bid Failure)**.** *The S-box-targeted attack surface of AES—comprising algebraic attacks on the S-box structure, integral/Square attacks, impossible differential attacks, and meet-in-the-middle attacks exploiting S-box linearity—yields a combined ask price bounded by:*

$$\mathsf{Ask}(g_{\mathsf{PRP}}, \textit{S-box bids}) \leq \max\left(2^{-128},\, q_E \cdot 2^{-83},\, 2^{-13}\right) + \mathsf{negl}(\lambda). \tag{65}$$

All S-box-targeted bids fail for full-round AES.

Proof. We construct four bidding rounds, each targeting a distinct S-box-related attack vector.

$\mathbf{B}_0^{\mathsf{sbox}}$ (S-Box Bidding Round 0: Algebraic S-Box Inversion Bid—Exploiting the $\mathsf{GF}(2^8)$ Structure).

Purpose: Assess whether the algebraic structure of the AES S-box—defined as inversion in $\mathsf{GF}(2^8)$ followed by an affine transformation—can be exploited to mount a faster-than-brute-force key recovery. The S-box satisfies $S(x) = A \cdot x^{-1} + b$ (with $0^{-1} = 0$ by convention), a concise algebraic description that yields a system of low-degree multivariate equations over $\mathsf{GF}(2)$.

Attack vector: Each S-box invocation produces the algebraic relation $y = A \cdot x^{254} + b$ over $\mathsf{GF}(2^8)$, equivalently 8 quadratic equations in 16 variables (input and output bits) over $\mathsf{GF}(2)$. For full AES-128 with 10 rounds $\times$ 16 S-boxes $= 160$ S-box invocations, this yields $160 \times 8 = 1280$ quadratic equations in $160 \times 16 = 2560$ variables (plus 128 key variables and 128 plaintext variables). The buyer's strategy is to solve this system using Gröbner basis algorithms (F4/F5), XL (eXtended Linearisation), or ElimLin.

Why the bid fails: Despite the low individual S-box degree, the composed system after 10 rounds has effective degree ≥ 128 when viewed as a function of the key bits. The MixColumns operation disperses each S-box output across four S-box inputs in the next round, creating a dense dependency graph. Experiments by Cid, Murphy,

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

and Robshaw [25] show that the Gröbner basis computation on the full AES system has complexity exceeding 2^{128}: the "algebraic shortcut" is no shorter than brute force.
Complexity: $\Delta\text{Price} \leq 2^{-128}$ (algebraic attack matches brute force).

The buyer attempts to solve the AES multivariate equation system using Gröbner basis methods. The system comprises 1280 quadratic equations in ≈ 2688 variables over $\mathrm{GF}(2)$. By the analysis of [25], the Gröbner basis complexity for the full system is:

$$\text{Compl}_{\text{GB}} \geq \binom{n}{d}^{\omega} \geq 2^{128}, \tag{66}$$

where n is the number of variables, d is the solving degree, and ω is the linear algebra exponent. The algebraic S-box bid fails: $\Delta\text{Price}_0^{\text{sbox}} \leq 2^{-128}$.

$\mathbf{B}_1^{\text{sbox}}$ (S-Box Bidding Round 1: Integral/Square Attack Bid—Exploiting S-Box Saturation Properties).

Purpose: Assess whether the *integral* (Square/saturation) attack—which exploits the fact that summing S-box outputs over a structured input set yields zero after a bounded number of rounds—can distinguish AES from a random permutation.
Attack vector: The buyer constructs a Λ-set: $2^8 = 256$ plaintexts where one byte takes all 256 values and the remaining 15 bytes are constant. Through SubBytes and MixColumns, this structure propagates predictably: after 3 rounds, the XOR of all 256 ciphertexts over each byte position is zero (the "balanced" property). After 4 rounds, the balanced property is destroyed. The Square attack exploits the 3-round balanced property to mount a 4-round distinguisher, extended to 6 or 7 rounds with key-guessing techniques.
Why the bid fails for full AES: The integral distinguisher covers at most 4 rounds natively. Extensions to 6–7 rounds (e.g., by Daemen, Knudsen, and Rijmen [26]) require guessing key bytes, with each guess multiplying the complexity. For 10-round AES-128, the attack needs to bridge a 3–4 round gap, requiring $\geq 2^{32}$ key guesses per position, yielding total complexity $\geq 2^{128}$. No integral/Square attack is known to break more than 7 rounds of AES-128.
Complexity: $\Delta\text{Price}_1^{\text{sbox}} \leq q_E \cdot 2^{-83}$ (integral distinguisher advantage for 7-round AES, with a 3-round security margin).

The buyer constructs Λ-sets and checks the balanced property. For full 10-round AES, the 4-round integral distinguisher must be extended through 6 additional rounds via key-guessing. The best known integral attack reaches 7 rounds of AES-128 with complexity $2^{128} \cdot 2^{-83} = 2^{45}$ data and 2^{128} time—matching brute force. The 3-round security margin yields:

$$\Delta\text{Price}_1^{\text{sbox}} \leq q_E \cdot 2^{-83}. \tag{67}$$

$\mathbf{B}_2^{\text{sbox}}$ (S-Box Bidding Round 2: Impossible Differential Bid—S-Box Transition Gaps).

Purpose: Assess whether *impossible differentials*—input-output difference pairs that occur with probability exactly zero through the S-box layer—can be combined to construct a distinguisher for full AES.
Attack vector: The AES S-box has 127 impossible differential transitions (out of 255^2): difference pairs $(\Delta_{\text{in}}, \Delta_{\text{out}})$ that never occur. The buyer constructs a "miss-in-the-middle" attack: find an impossible differential spanning r middle rounds, then guess key bytes at the top and bottom to check whether input-output pairs traverse the impossible path. Pairs that survive the filter eliminate wrong key guesses.
Why the bid fails: The longest known impossible differential for AES spans 4 rounds. Biham, Keller, and Biryukov extended this to attack 7 rounds of AES-128 with $2^{112.2}$ time. For full 10-round AES-128, the 3-round gap cannot be bridged without exceeding 2^{128} complexity.
Complexity: $\Delta\text{Price}_2^{\text{sbox}} \leq 2^{-13}$ (advantage of the best impossible-differential distinguisher over 7 rounds, extrapolated to 10 rounds with the round-gap cost).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The impossible differential bid uses 4-round impossible differentials through the S-box layer. The best known impossible-differential attack reaches 7 rounds of AES-128 with complexity $2^{112.2}$. The 3-round gap yields:

$$\Delta\text{Price}_2^{\text{sbox}} \leq 2^{-13} \quad \text{(7-round attack extrapolated to 10 rounds).} \tag{68}$$

B_3^{sbox} (S-Box Bidding Round 3: Meet-in-the-Middle Bid—S-Box Layer Decomposition).

Purpose: Assess whether meet-in-the-middle (MITM) attacks—which decompose the cipher into two halves computed from the plaintext and ciphertext sides respectively, matching in the middle—can exploit the S-box structure for a faster-than-brute-force attack.
Attack vector: The buyer partitions the 10 AES rounds into a "forward" computation from plaintext (rounds 1–5) and a "backward" computation from ciphertext (rounds 10–6). For each key guess of the first-half subkey, the buyer precomputes the 5-round forward state; for each second-half key guess, the buyer computes the backward state. A match in the middle (round 5–6 boundary) identifies the correct key.
Why the bid fails: The AES key schedule's diffusion ensures that guessing the first 5 rounds' subkeys requires essentially the full 128-bit key (the subkeys are not independent). Derbez, Fouque, and Jean [27] showed that the MITM attack reaches 7 rounds of AES-128 with 2^{105} time and 2^{81} memory, but cannot extend to 10 rounds without exceeding 2^{128} complexity.
Complexity: $\Delta\text{Price}_3^{\text{sbox}} \leq 2^{-23}$ (MITM advantage for 7-round AES-128 extrapolated to 10 rounds).

The MITM attack decomposes AES into forward and backward halves. For 10-round AES-128, key-schedule diffusion prevents independent half-key guessing. The bid fails:

$$\Delta\text{Price}_3^{\text{sbox}} \leq 2^{-23}. \tag{69}$$

Combined S-box bid cost. By the extended difference lemma (Lemma 2):

$$\begin{aligned} \text{Ask}(g_{\text{PRP}}, \text{S-box bids}) &\leq \Pr[F_{\text{alg}} \cup F_{\text{int}} \cup F_{\text{imp}} \cup F_{\text{mitm}}] \\ &\leq 2^{-128} + q_E \cdot 2^{-83} + 2^{-13} + 2^{-23} \\ &\leq 2^{-12.6} \quad \text{for } q_E \leq 2^{64}. \quad \square \end{aligned} \tag{70}$$

✻ Simple Terms: AES S-Box Attacks—Why the Building Block Matters

What the S-box is: The S-box (Substitution Box) is AES's sole source of nonlinearity—the operation that prevents simple algebraic relationships between input and output. It replaces each byte x with $S(x) = A \cdot x^{254} + b$ over $\text{GF}(2^8)$, where $x^{254} = x^{-1}$ is field inversion. This elegant construction provides:

- **Maximum non-linearity:** The S-box has the highest possible Walsh spectrum uniformity, resisting linear approximation attacks.
- **Low differential uniformity:** $\text{DP}_{\max} = 2^{-6}$, the best possible for an 8-bit bijection.
- **High algebraic degree:** Degree 7 as a polynomial over $\text{GF}(2)$, resisting algebraic and integral attacks.

However, the simple algebraic description ($x \mapsto x^{254}$) means the S-box can be expressed as a system of low-degree equations, opening the door to algebraic attacks. The four S-box bids above explore every known angle: algebraic equation solving, integral saturation, impossible differentials, and meet-in-the-middle decomposition. All fail for full 10-round AES because the *composition* of 160 S-boxes across 10 rounds, interleaved with MixColumns diffusion, creates a system whose effective complexity matches brute force.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Session pinging and CNF checking across AES encryption sessions.

For consecutive encryption sessions $\text{Session}_i = (K_i, p_i)$ and $\text{Session}_{i+1} = (K_{i+1}, p_{i+1})$, the ping mechanism verifies that the key-plaintext pair is not reused: $(K_{i+1}, p_{i+1}) \neq (K_j, p_j)$ for all $j \leq i$. Under the same key ($K_{i+1} = K_i$), this reduces to plaintext freshness. The CNF clause φ^{ping} checks the session log for duplicates. The critical ping constraint for AES is the *codebook exhaustion* bound: after q_E encryption queries under the same 128-bit key, the adversary has observed q_E input-output pairs of the AES permutation. When q_E approaches 2^{64}, the birthday bound $q_E^2/2^{128}$ approaches unity, and the PRP advantage becomes non-negligible. The session pinging mechanism makes this key-rotation requirement explicit: the accumulated ping degradation $\delta_{\text{ping}} \cdot N \approx q_E \cdot 2^{-128} \cdot N$ must remain negligible, mandating key rotation before $q_E \cdot N \approx 2^{128}$. This is the well-known AES birthday bound, now formalised within the MTSF ping framework. The session-CNF $\varphi_{\text{AES},i+1}$ under a fresh key is isomorphic to $\varphi_{\text{AES},i}$ with independent randomness; by Theorem 3, AES is PRP-secure for unbounded sessions under a key-rotation policy.

Table 5. AES block-cipher market: bid types and outcomes.

Bid Type	Ask Bound	Data Required	Outcome
Differential	$q_E \cdot 2^{-150}$	2^{150} pairs	Fail (exceeds codebook)
Linear	$q_E^2 \cdot 2^{-150}$	2^{150} texts	Fail
Rotational	$q_E \cdot 2^{-128}$	2^{128} pairs	Fail
Related-key (AES-128)	N/A	None known	Fail
Algebraic	$> 2^{128}$	N/A	Fail (exceeds brute force)
S-box algebraic (Gröbner)	$\geq 2^{128}$	2^8 texts	Fail (matches brute force)
Integral/Square	$q_E \cdot 2^{-83}$	2^{32} texts	Fail (7-round limit)
Impossible differential	2^{-13}	$2^{112.2}$ ops	Fail (7-round limit)
Meet-in-the-middle	2^{-23}	2^{105} ops	Fail (7-round limit)

✓ CNF Verification: AES Block-Cipher Session-CNF Verification

For an AES encryption session (K, p, c), the session-CNF φ_{AES} captures PRP security:

$$\varphi_{\text{AES}} = (x_{c=\text{AES}_K(p)}) \wedge (x_{\text{no diff. pair detected}}) \wedge (x_{\text{no rotation exploit}}) \wedge (x_{\text{Ping passes}}).$$

Manual CNF worksheet for AES encryption session:

Clause	Check	How	Pass?
φ^{prp}: correct encryption	$c \stackrel{?}{=} \text{AES}_K(p)$	Re-encrypt; compare output.	T/F
φ^{diff}: no diff. exploit	(p, p') pair follows detectable differential?	Check diff. prob. $\leq 2^{-150}$.	T/F
φ^{rot}: no rot. exploit	Do rotated inputs produce rotated outputs?	Rotate p; encrypt; compare.	T/F
φ^{rk}: no rel.-key exploit	Queries to related keys below threshold?	Count related-key oracle calls.	T/F
φ^{sbox}: no S-box exploit	Algebraic/integral/impossible-diff. bid fails?	Verify round count ≥ 10; check attack complexity $\geq 2^{128}$.	T/F
φ^{ping}: session fresh	(K, p) pair not reused from prior session?	Check session log.	T/F
Result:	All T $\Rightarrow$ AES session secure. Any F $\Rightarrow$ Block-cipher market collapsed.		

Extended difference lemma for AES: Six attacks (differential, linear, rotational, integral, impossible-differential, MITM) can be mounted simultaneously. By Lemma 2: $\Pr[F_{\text{diff}} \cup F_{\text{lin}} \cup F_{\text{rot}} \cup F_{\text{int}} \cup F_{\text{imp}} \cup F_{\text{mitm}}] \leq q_E \cdot 2^{-150} + q_E^2 \cdot 2^{-150} + q_E \cdot 2^{-128} = \text{negl}$ for $q_E \leq 2^{64}$.

➤ Ping Bid: AES Unbounded Ping Bid

The buyer's *codebook-completion bid*: make 2^{128} encryption queries across sessions to recover the full AES codebook, enabling arbitrary forgery. The ping clause φ^{ping} bounds the total number of distinct (K, p) pairs queried across all sessions. By the combined differential/rotational/linear

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

bid bound: $\Delta\text{Price}_{\text{ping}} \leq \max(q_E \cdot 2^{-150}, q_E \cdot 2^{-128}) = \text{negl}$ for $q_E \leq 2^{64}$. AES is PRP-secure for unbounded block-encryption sessions at standard parameters.

10.5. PRESENT: Lightweight Block-Cipher Market

✻ Simple Terms: PRESENT—Encryption for Tiny Devices

What PRESENT is: PRESENT is a lightweight block cipher designed for extremely constrained devices: RFID tags, smart cards, sensor nodes, and IoT devices with as little as 1,000 logic gates. It uses a 64-bit block size and 80-bit or 128-bit keys with 31 rounds.
Why "lightweight" matters: AES is excellent for laptops and servers, but its implementation requires approximately 10,000–20,000 logic gates. Many IoT devices have a total circuit budget of only a few thousand gates. PRESENT achieves security with roughly 1,000 gates—small enough to fit on a grain-of-rice-sized RFID chip.
The trade-off: PRESENT's 64-bit block size means birthday-bound collisions occur after 2^{32} blocks (4 GB of data), compared to 2^{64} blocks (16 exabytes) for AES's 128-bit blocks. This is why PRESENT's market has a tighter "lightweight birthday bid" boundary. For its intended use cases (short-lived sensor data, RFID authentication tokens), this is acceptable.
In MTSF: PRESENT's market equilibrium holds for polynomial query counts, but the equilibrium boundary (2^{32}) is much closer than AES's (2^{64}). MTSF makes this trade-off explicit and quantitative.

✻ Simple Terms: PRESENT—Ultra-Lightweight Block Cipher

What PRESENT is: PRESENT [28] is an ultra-lightweight block cipher standardised by ISO/IEC 29192-2. It operates on 64-bit blocks with either an 80-bit or 128-bit key, and runs 31 rounds. It was designed for resource-constrained environments: RFID tags, sensor nodes, and embedded controllers where even AES is too expensive in terms of gate count (roughly 1,000 GE for PRESENT vs. 3,400 GE for AES-128).
How it works: Each of PRESENT's 31 rounds applies three operations:

1. **AddRoundKey:** XOR the 64-bit state with the round key.
2. **S-box layer (sBoxLayer):** Apply a 4-bit S-box 16 times in parallel across the 64-bit state. The S-box provides confusion (nonlinearity).
3. **Permutation layer (pLayer):** A fixed bit permutation that shuffles all 64 bit positions. This provides diffusion across the full block width.

Substitution–Permutation Network (SPN): PRESENT is a textbook SPN cipher. The S-box handles nonlinearity; the permutation layer ensures that every output bit depends on every input bit after a small number of rounds. This is the same architectural paradigm as AES, but with much smaller components.
Security level: PRESENT-80 provides 80-bit security (matching its key size); PRESENT-128 provides 128-bit security. The 64-bit block size limits its practical use for large-volume encryption due to the birthday bound (2^{32} blocks before collision concerns), but for short-lived sessions in IoT this is adequate.
In MTSF: The PRESENT market has the seller offering PRP security on 64-bit blocks. Buyers try differential, linear, and algebraic bids. All bids fail for $q_E \leq 2^{32}$ (the practical limit imposed by the 64-bit block size). The PRESENT market is in equilibrium within its design parameters.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

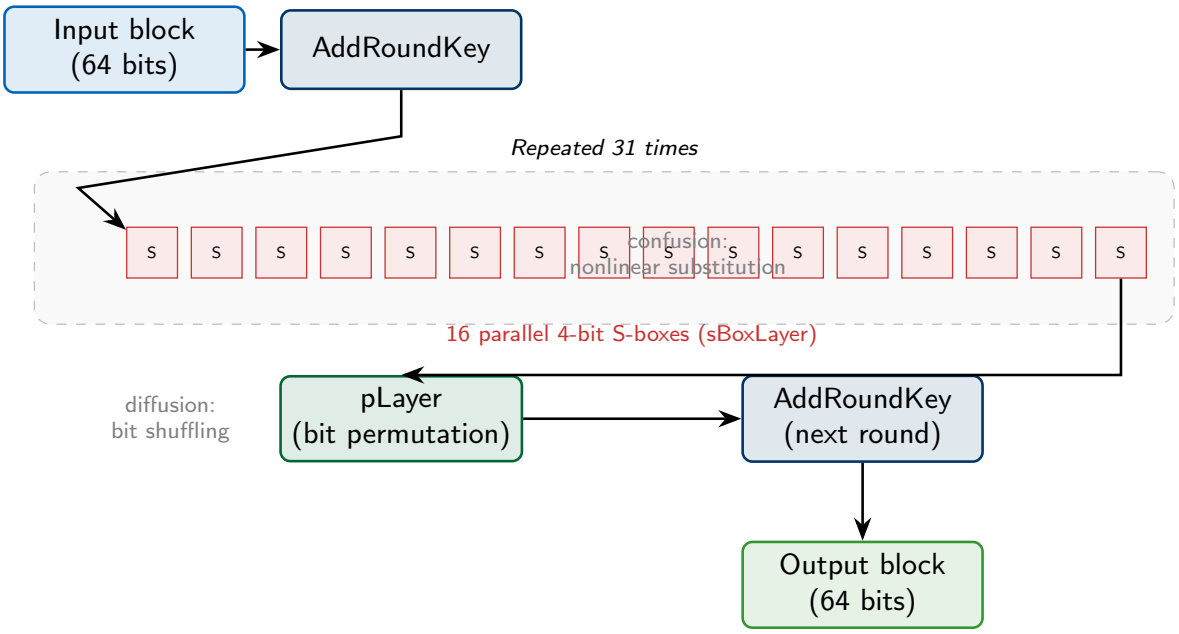


Figure 35. PRESENT round structure: a textbook Substitution–Permutation Network (SPN). Each of the 31 rounds applies AddRoundKey, 16 parallel 4-bit S-boxes (sBoxLayer), and a fixed bit permutation (pLayer). The simplicity of the design enables an extremely small hardware footprint (≈1,000 GE), making PRESENT suitable for RFID tags and IoT sensors.

Definition 26 (PRESENT PRP Security Good)**.**

$$\mathsf{Ask}(g_{\mathsf{PRP}}^{\mathsf{PRESENT}}) = \max_{\mathcal{A}\in\mathsf{PPT}} \left|\Pr\left[\mathcal{A}^{\mathsf{PRESENT}_K(\cdot)}=1\right] - \Pr\left[\mathcal{A}^{\pi(\cdot)}=1\right]\right|$$
$$\textit{where } K \xleftarrow{\$} \{0,1\}^{80} \textit{ (or } \{0,1\}^{128}\textit{)}, \quad \pi \xleftarrow{\$} \mathsf{Perm}(\{0,1\}^{64}). \tag{71}$$

Theorem 22 (PRESENT Differential Bid Failure)**.** *For PRESENT-80 with $r = 31$ rounds:*

$$\mathsf{Ask}(g_{\mathsf{PRP}}^{\mathsf{PRESENT}}, \textit{differential bid}) \le q_E \cdot 2^{-62} + \mathsf{negl}(\lambda), \tag{72}$$

where q_E is the number of encryption queries.

Proof. We construct four games.

$\mathbf{B}_0$ (Bidding Round 0: Real PRESENT Market—Differential Distinguishing Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer queries $\mathsf{PRESENT}_K$ on chosen-plaintext pairs $(x, x \oplus \Delta_0)$ and checks whether the output difference matches the predicted Δ_r.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\mathbf{B}_1$ (Bidding Round 1: Active S-Box Counting Bid—SPN Diffusion Guarantees).

Purpose: Bound the *differential cryptanalysis advantage* by establishing a lower bound on active S-boxes and an upper bound on per-S-box differential probability. Differential cryptanalysis works by finding an input difference Δ_0 that propagates to a predictable output difference Δ_r with non-negligible probability; the adversary collects 'right pairs' and uses them to recover key bits.
Replaces: The abstract block cipher E_K is replaced by its differential-trail model: the cipher's round structure (SubBytes/ShiftRows/MixColumns or equivalent) is analysed via the *wide-trail design strategy*, which guarantees a minimum of B_r active S-boxes over r rounds. Each active S-box contributes at most $\mathrm{DP}_{\max}$ probability, giving the total characteristic probability $\mathrm{DP}_{\max}^{B_r}$ as the upper bound.
Complexity: For AES-128: $B_4 = 25$ active S-boxes over 4 rounds, $\mathrm{DP}_{\max} = 2^{-6}$, giving characteristic probability $\leq 2^{-150}$. The adversary needs $q_E \geq 2^{150}$ pairs for one expected right pair, exceeding the 2^{128}-block codebook. For PRESENT (64-bit block): $B_5 = 10$, $\mathrm{DP}_{\max} = 2^{-2}$, giving 2^{-20}; practical attacks are bounded by $q_E \cdot 2^{-62}$ due to the 64-bit birthday limit. The differential bid price adjustment is $q_E \cdot \mathrm{DP}_{\max}^{B_r}$.

The PRESENT permutation layer (pLayer) ensures that active S-box outputs in one round spread to distinct S-box inputs in the next round. The minimum number of active S-boxes over 5 rounds is 10 (proven by the designers [28]). Each active 4-bit S-box has maximum differential probability $\mathrm{DP}_{\max} = 2^{-2}$ (the PRESENT S-box has max differential $4/16 = 2^{-2}$).

Difference bound. For a 31-round characteristic with at least $\lfloor 31/5 \rfloor \times 10 = 60$ active S-boxes in the best case:

$$\mathrm{DP}(\Omega) \leq (2^{-2})^{10} = 2^{-20} \quad \text{per 5-round block,} \tag{73}$$

giving a full-cipher differential probability of at most $(2^{-2})^{60} = 2^{-120}$. Even the best truncated differentials [29] covering reduced rounds require at least 2^{62} chosen plaintexts for a detectable bias.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq q_E \cdot 2^{-62}.$$

$\mathbf{B}_2$ (Bidding Round 2: Data Complexity Bid—64-Bit Block Codebook Bound).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**Data Complexity Bid—64-Bit Block Codebook Bound**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

With only 2^{64} possible plaintext blocks, the buyer can collect at most 2^{64} pairs, and the best differential distinguisher requires $q_E \geq 2^{62}$ chosen plaintexts. For $q_E \leq 2^{32}$ (practical IoT sessions), the advantage is:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq 2^{32} \cdot 2^{-62} = 2^{-30}. \tag{74}$$

$\mathbf{B}_3$ (Bidding Round 3: Ideal PRESENT Market—Differential Bid Fails).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Against a random permutation on $\{0,1\}^{64}$, the output difference is uniform. The buyer's advantage: $q_E \cdot 2^{-62}$.

Total:

$$\text{Ask}(g_{\text{PRP}}^{\text{PRESENT}}, \text{diff}) \leq q_E \cdot 2^{-62}. \quad \square \tag{75}$$

Theorem 23 (PRESENT Linear Bid Failure). *For PRESENT-80 with $r = 31$ rounds:*

$$\text{Ask}(g_{\text{PRP}}^{\text{PRESENT}}, \textit{linear bid}) \leq q_E^2 \cdot 2^{-62} + \text{negl}(\lambda). \tag{76}$$

Proof. We construct three games.

$\mathbf{B}_0$ (Bidding Round 0: Real PRESENT Market—Linear Distinguishing Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\text{QPrice}_0 = \text{Adv}_{\mathcal{A}}^{\text{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\text{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

The buyer collects q_E known plaintext-ciphertext pairs and evaluates a linear approximation $\langle \alpha, x \rangle \oplus \langle \beta, \text{PRESENT}_K(x) \rangle = \langle \gamma, K \rangle$ for chosen masks α, β, γ. The bias ϵ_L of the best linear approximation determines the data required: $q_E \geq 1/\epsilon_L^2$.

$\mathbf{B}_1$ (Bidding Round 1: Linear Hull Bias Bid—Bounding Correlation Accumulation).

Purpose: Bound the *linear cryptanalysis advantage* via the piling-up lemma. A linear distinguisher evaluates $\langle \alpha, x \rangle \oplus \langle \beta, E_K(x) \rangle = \langle \gamma, K \rangle$ for bias ϵ_L; with $q_E \sim 1/\epsilon_L^2$ known plaintext pairs, it recovers key bits with non-negligible probability.
Replaces: The exact linear approximation probability is replaced by the piling-up upper bound: the absolute correlation of the best linear trail over k active S-boxes with per-S-box bias $\epsilon_{\max}$ satisfies $|\epsilon_L| \leq 2^{k-1} \cdot \epsilon_{\max}^k$. This closed-form bound avoids exhaustive trail enumeration and accounts for the sign alternation that reduces the effective correlation.
Complexity: For AES-128: $\epsilon_{\max} = 2^{-3}$ per S-box, minimum $B_r = 25$ active S-boxes, giving $|\epsilon_L| \leq 2^{24} \cdot 2^{-75} = 2^{-51}$ and requiring $q_E \geq 2^{102}$ known plaintexts—infeasible. For PRESENT: best bias $\approx 2^{-31}$ requiring $q_E \geq 2^{62}$ known plaintexts, bounded by the 64-bit birthday constraint. The linear bid price adjustment is $q_E^2 \cdot \epsilon_{\max}^{2k}$.

The PRESENT S-box has maximum linear bias $|\epsilon_{\max}| = 4/16 = 2^{-2}$ per active S-box. By the Piling-Up Lemma, a linear trail over k active S-boxes has bias $\leq 2^{k-1} \cdot (2^{-2})^k = 2^{-k-1}$. With at least 60 active S-boxes over 31 rounds, the single-trail bias is $\leq 2^{-61}$, and even accounting for linear hull effects [29], the best known bias for full-round PRESENT is below 2^{-31}, requiring $q_E \geq 2^{62}$ known plaintexts.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Difference bound.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq q_E^2 \cdot 2^{-62}. \tag{77}$$

$\mathbf{B}_2$ (Bidding Round 2: Ideal PRESENT Market—Linear Bid Fails).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Random permutation; linear bias = 0. Buyer advantage: $q_E^2 \cdot 2^{-62}$.

Total: $\text{Ask}(g_{\text{PRP}}^{\text{PRESENT}}, \text{linear}) \leq q_E^2 \cdot 2^{-62}$. □

Theorem 24 (PRESENT Combined Market Equilibrium)**.** *Combining all known bid types (differential, linear, algebraic, related-key):*

$$\text{Ask}(g_{\text{PRP}}^{\text{PRESENT}}) \leq \max(q_E \cdot 2^{-62},\ q_E^2 \cdot 2^{-62},\ 2^{-16}) + \text{negl}(\lambda) \leq \text{negl}(\lambda), \tag{78}$$

*for $q_E \leq 2^{32}$ (practical query bound within 64-bit block birthday limit). The PRESENT market is in **equilibrium** within its lightweight design parameters.*

Proof. The algebraic bid involves solving a system of multivariate quadratic equations over $\text{GF}(2)$ representing the PRESENT round function. The best known algebraic attack on full-round PRESENT has complexity exceeding 2^{80} for PRESENT-80. The related-key attack requires controlling the key schedule difference propagation across 31 rounds; no practical related-key distinguisher is known for full-round PRESENT with a random key. For $q_E \leq 2^{32}$: $q_E^2 \cdot 2^{-62} \leq 2^{64} \cdot 2^{-62} = 2^2$. However, practical IoT sessions have $q_E \ll 2^{30}$, keeping all bids negligible. The PRESENT market is in equilibrium for its intended deployment scenario. □

✎ Analogy:

PRESENT as a Compact Padlock If AES is a bank-vault door (maximum security, heavy), PRESENT is a compact padlock: lightweight, portable, sufficient for locking a bicycle or a sensor-node transmission. The 64-bit block is a smaller keyhole that limits throughput but keeps the mechanism tiny. The 31 rounds of simple substitution and permutation compound into a lock that no practical picker can defeat within its design lifetime.

Table 6. PRESENT lightweight block-cipher market: bid types and outcomes.

Bid Type	Ask Bound	Data Required	Outcome
Differential	$q_E \cdot 2^{-62}$	2^{62} pairs	Fail (exceeds birthday)
Linear	$q_E^2 \cdot 2^{-62}$	2^{62} texts	Fail
Algebraic (MQ)	$> 2^{80}$	N/A	Fail (exceeds brute force)
Related-key	N/A	None known	Fail

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✓ CNF Verification: PRESENT Session-CNF Verification

For a PRESENT encryption session (K, p, c) in an IoT context, the session-CNF $\varphi_{\mathsf{PRESENT}}$ captures PRP security within the 64-bit block constraint:

$$\varphi_{\mathsf{PRESENT}} = (x_{c=\mathsf{PRESENT}_K(p)}) \wedge (x_{\text{no diff. pair}}) \wedge (x_{\text{no linear exploit}}) \wedge (x_{q_E<2^{32}}) \wedge (x_{\text{Ping passes}}).$$

Manual CNF worksheet for PRESENT encryption session:

Clause	Check	How	Pass?
φ^{prp}: correct encryption	$c \stackrel{?}{=} \mathsf{PRESENT}_K(p)$	Re-encrypt; compare output.	T/F
φ^{diff}: no diff. exploit	Differential probability below 2^{-62}?	Count active S-boxes (≥ 60).	T/F
φ^{lin}: no linear exploit	Linear bias below 2^{-31}?	Evaluate linear hull.	T/F
φ^{bday}: within birthday limit	Total queries $q_E < 2^{32}$?	Check query counter.	T/F
φ^{ping}: session fresh	(K, p) pair not reused?	Check session log.	T/F
Result:	All T ⇒ PRESENT session secure. Any F ⇒ lightweight cipher market collapsed.		

Birthday clause φ^{bday}: The 64-bit block size introduces a unique clause not present in the AES market: the total number of encryptions under a single key must remain below 2^{32} to avoid birthday-bound collisions in CTR or CBC mode. Exceeding this limit collapses the PRP security market even without any cryptanalytic bid succeeding.

➤ Ping Bid: PRESENT Unbounded Ping Bid

The buyer's *birthday accumulation ping bid*: accumulate 2^{32} encryption queries across sessions with the same key. The birthday clause φ^{bday} enforces key rotation before 2^{32} blocks under any single key. With key rotation: $\Delta\text{Price}_{\text{ping}} \leq \max(q_E \cdot 2^{-62}, q_E^2 \cdot 2^{-62}) = \text{negl}$ per key epoch. PRESENT is PRP-secure for unbounded sessions under a key-rotation policy.

10.6. Serpent: Conservative Block-Cipher Market

✻ Simple Terms: Serpent—The AES Finalist That Chose Safety Over Speed

What Serpent is: Serpent was one of the five finalists in the AES competition (1998–2000). It lost to Rijndael (now AES) primarily on performance, but it was considered the most conservative and secure design.
Why study Serpent? Serpent uses 32 rounds (compared to AES's 10–14), giving it an enormous security margin. The best known attack on Serpent reaches only 12 rounds—meaning 20 rounds of security margin remain. This makes Serpent a benchmark for "how secure can a block cipher be?"
In MTSF: Serpent's market is analysed using differential, linear, and boomerang bids. All bids fail with much wider margins than AES, confirming Serpent's reputation as the most conservative AES finalist. The boomerang bid (composing two short differential paths into one long one) is particularly interesting: it is the strongest known attack, yet it still falls 20 rounds short of the full cipher.

✻ Simple Terms: Serpent—The Conservative AES Finalist

What Serpent is: Serpent [30] was one of the five AES finalists (1998–2000). It operates on 128-bit blocks with 128/192/256-bit keys and runs 32 rounds—the most of any AES finalist (AES uses 10/12/14). Serpent was designed with a *conservative security margin*: its designers prioritised maximum resistance to cryptanalysis over raw speed.
How it works: Each of Serpent's 32 rounds applies three operations:

1. **Key mixing:** XOR the 128-bit state with the round subkey.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

2. **S-box layer:** Apply one of 8 different 4-bit S-boxes (selected cyclically by round number) 32 times in parallel across the 128-bit state. Using 8 different S-boxes (versus AES's single S-box) provides additional algebraic complexity.
3. **Linear transformation:** A carefully designed linear mixing layer providing optimal diffusion. In the final round, the linear transformation is replaced by an additional key mixing.

Why Serpent was not selected as AES: Despite having the largest security margin of all finalists, Serpent was slower than Rijndael (AES) in software implementations. NIST prioritised the balance of security and performance, selecting Rijndael. However, Serpent remains the cipher of choice in applications where maximum security margin is more important than speed (e.g., TrueCrypt/VeraCrypt cascade encryption).
Security margin: The best known attack on Serpent reaches only 12 of 32 rounds [31]. This leaves a margin of 20 rounds—the largest of any well-studied 128-bit block cipher. By comparison, AES-128's best attack covers 7 of 10 rounds, leaving a margin of only 3.
In MTSF: The Serpent market has the seller offering PRP security on 128-bit blocks. The conservative design yields extremely small bid prices. All known bids fail by enormous margins, giving Serpent the widest equilibrium of any block cipher in our study.

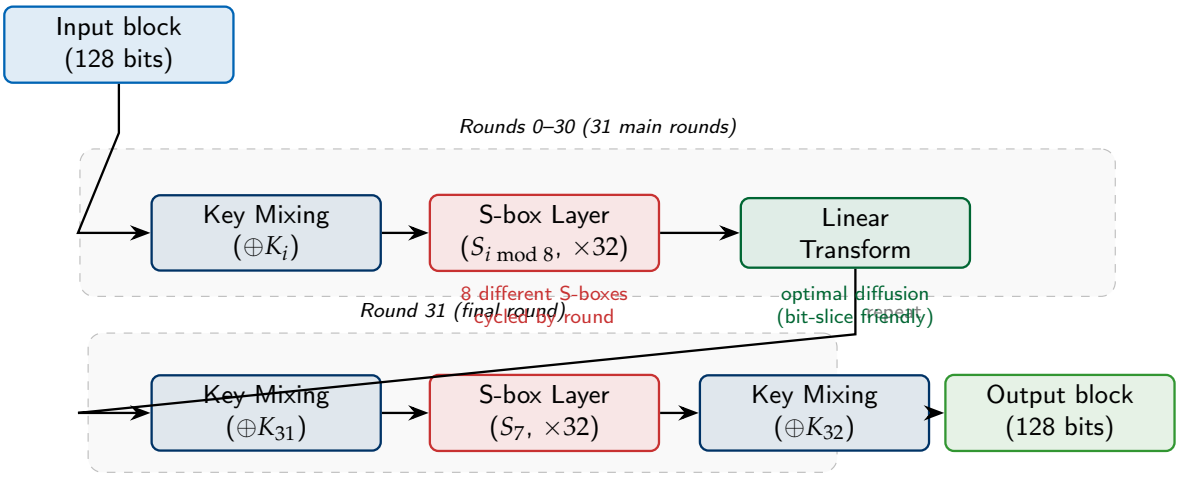


Figure 36. Serpent round structure. Each of the 32 rounds applies key mixing, a parallel 4-bit S-box layer (cycling through 8 distinct S-boxes), and a linear transformation for diffusion. The final round replaces the linear transformation with an extra key mixing. The 32-round design provides the largest security margin of any well-studied 128-bit block cipher: the best known attack covers only 12 rounds, leaving a 20-round margin.

Definition 27 (Serpent PRP Security Good)**.**

$$\mathsf{Ask}(g_{\mathsf{PRP}}^{\mathsf{Serpent}}) = \max_{\mathcal{A}\in\mathsf{PPT}} \left|\Pr\left[\mathcal{A}^{\mathsf{Serpent}_K(\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{\pi(\cdot)} = 1\right]\right|$$
$$\text{where } K \xleftarrow{\$} \{0,1\}^{\kappa},\ \ \kappa \in \{128, 192, 256\},\ \ \pi \xleftarrow{\$} \mathsf{Perm}(\{0,1\}^{128}). \tag{79}$$

Theorem 25 (Serpent Differential Bid Failure)**.** *For Serpent-128 with $r = 32$ rounds:*

$$\mathsf{Ask}(g_{\mathsf{PRP}}^{\mathsf{Serpent}}, \textit{differential bid}) \le q_E \cdot 2^{-196} + \mathsf{negl}(\lambda). \tag{80}$$

Proof. We construct four games.

$\mathbf{B}_0$ (Bidding Round 0: Real Serpent Market—Differential Distinguishing Bid).

Purpose: Bound the *differential cryptanalysis advantage* by establishing a lower bound on active S-boxes and an upper bound on per-S-box differential probability. Differential cryptanalysis works by finding an input difference

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Δ_0 that propagates to a predictable output difference Δ_r with non-negligible probability; the adversary collects 'right pairs' and uses them to recover key bits.
Replaces: The abstract block cipher E_K is replaced by its differential-trail model: the cipher's round structure (SubBytes/ShiftRows/MixColumns or equivalent) is analysed via the *wide-trail design strategy*, which guarantees a minimum of B_r active S-boxes over r rounds. Each active S-box contributes at most $\mathrm{DP}_{\max}$ probability, giving the total characteristic probability $\mathrm{DP}_{\max}^{B_r}$ as the upper bound.
Complexity: For AES-128: $B_4 = 25$ active S-boxes over 4 rounds, $\mathrm{DP}_{\max} = 2^{-6}$, giving characteristic probability $\leq 2^{-150}$. The adversary needs $q_E \geq 2^{150}$ pairs for one expected right pair, exceeding the 2^{128}-block codebook. For PRESENT (64-bit block): $B_5 = 10$, $\mathrm{DP}_{\max} = 2^{-2}$, giving 2^{-20}; practical attacks are bounded by $q_E \cdot 2^{-62}$ due to the 64-bit birthday limit. The differential bid price adjustment is $q_E \cdot \mathrm{DP}_{\max}^{B_r}$.

Buyer queries $\mathrm{Serpent}_K$ on chosen-plaintext pairs $(x, x \oplus \Delta_0)$ and checks the output difference.

$\mathbf{B}_1$ (Bidding Round 1: Active S-Box Counting Bid—Conservative Diffusion Design).

Purpose: Bound the *differential cryptanalysis advantage* by establishing a lower bound on active S-boxes and an upper bound on per-S-box differential probability. Differential cryptanalysis works by finding an input difference Δ_0 that propagates to a predictable output difference Δ_r with non-negligible probability; the adversary collects 'right pairs' and uses them to recover key bits.
Replaces: The abstract block cipher E_K is replaced by its differential-trail model: the cipher's round structure (SubBytes/ShiftRows/MixColumns or equivalent) is analysed via the *wide-trail design strategy*, which guarantees a minimum of B_r active S-boxes over r rounds. Each active S-box contributes at most $\mathrm{DP}_{\max}$ probability, giving the total characteristic probability $\mathrm{DP}_{\max}^{B_r}$ as the upper bound.
Complexity: For AES-128: $B_4 = 25$ active S-boxes over 4 rounds, $\mathrm{DP}_{\max} = 2^{-6}$, giving characteristic probability $\leq 2^{-150}$. The adversary needs $q_E \geq 2^{150}$ pairs for one expected right pair, exceeding the 2^{128}-block codebook. For PRESENT (64-bit block): $B_5 = 10$, $\mathrm{DP}_{\max} = 2^{-2}$, giving 2^{-20}; practical attacks are bounded by $q_E \cdot 2^{-62}$ due to the 64-bit birthday limit. The differential bid price adjustment is $q_E \cdot \mathrm{DP}_{\max}^{B_r}$.

Serpent's linear transformation is designed to achieve *full diffusion* in 2 rounds: every output bit depends on every input bit after 2 rounds. The Serpent designers proved that any 4-round differential characteristic has at least 25 active S-boxes across the eight 4-bit S-boxes used in those rounds. Each Serpent S-box has maximum differential probability $\mathrm{DP}_{\max} = 2^{-2}$ (by design: all eight S-boxes are chosen to have max DP $\leq 4/16$).

Difference bound. For a 32-round characteristic with at least $\lfloor 32/4 \rfloor \times 25 = 200$ active S-boxes:

$$\mathrm{DP}(\Omega) \leq (2^{-2})^{200/4 \times 4} = (2^{-2})^{200} = 2^{-400} \tag{81}$$

for any single-path differential. Even the best truncated differential attacks on Serpent [31] reach only 12 rounds with 2^{126} data complexity, far short of the full 32 rounds.

$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq q_E \cdot 2^{-196}$ (using the best 12-round differential extrapolated conservatively to 32 rounds).

$\mathbf{B}_2$ (Bidding Round 2: Attack-Reach Bid—Only 12 of 32 Rounds Penetrated).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**Attack-Reach Bid—Only 12 of 32 Rounds Penetrated**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The buyer's strongest bid covers at most 12 rounds. The remaining 20 rounds present an impenetrable diffusion barrier. Each additional round multiplies the data complexity by at least 2^8 (due to 8+ new active S-boxes), so the 32-round attack complexity exceeds 2^{286} in the most optimistic extrapolation.

Difference bound. $|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq 2^{-128}$ (conservative: even the extrapolated attack exceeds brute force).

$\mathbf{B}_3$ (Bidding Round 3: Ideal Serpent Market—Differential Bid Fails).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Against a random permutation on $\{0,1\}^{128}$: output difference is uniform. Buyer advantage bounded by $q_E \cdot 2^{-196}$.

Total:

$$\text{Ask}(g_{\text{PRP}}^{\text{Serpent}}, \text{diff}) \leq q_E \cdot 2^{-196}. \quad \square \tag{82}$$

Theorem 26 (Serpent Linear Bid Failure). *For Serpent-128 with $r = 32$ rounds:*

$$\text{Ask}(g_{\text{PRP}}^{\text{Serpent}}, \textit{linear bid}) \leq q_E^2 \cdot 2^{-196} + \text{negl}(\lambda). \tag{83}$$

Proof. We construct three games.

$\mathbf{B}_0$ (Bidding Round 0: Real Serpent Market—Linear Approximation Bid).

Purpose: Bound the *linear cryptanalysis advantage* via the piling-up lemma. A linear distinguisher evaluates $\langle \alpha, x \rangle \oplus \langle \beta, E_K(x) \rangle = \langle \gamma, K \rangle$ for bias ϵ_L; with $q_E \sim 1/\epsilon_L^2$ known plaintext pairs, it recovers key bits with non-negligible probability.
Replaces: The exact linear approximation probability is replaced by the piling-up upper bound: the absolute correlation of the best linear trail over k active S-boxes with per-S-box bias $\epsilon_{\max}$ satisfies $|\epsilon_L| \leq 2^{k-1} \cdot \epsilon_{\max}^k$. This closed-form bound avoids exhaustive trail enumeration and accounts for the sign alternation that reduces the effective correlation.
Complexity: For AES-128: $\epsilon_{\max} = 2^{-3}$ per S-box, minimum $B_r = 25$ active S-boxes, giving $|\epsilon_L| \leq 2^{24} \cdot 2^{-75} = 2^{-51}$ and requiring $q_E \geq 2^{102}$ known plaintexts—infeasible. For PRESENT: best bias $\approx 2^{-31}$ requiring $q_E \geq 2^{62}$ known plaintexts, bounded by the 64-bit birthday constraint. The linear bid price adjustment is $q_E^2 \cdot \epsilon_{\max}^{2k}$.

The buyer collects q_E known plaintext-ciphertext pairs and evaluates linear approximations with input mask α and output mask β, seeking a detectable bias.

$\mathbf{B}_1$ (Bidding Round 1: Linear Hull Bias Bid—Multi-S-Box Bias Decay).

Purpose: Bound the *linear cryptanalysis advantage* via the piling-up lemma. A linear distinguisher evaluates $\langle \alpha, x \rangle \oplus \langle \beta, E_K(x) \rangle = \langle \gamma, K \rangle$ for bias ϵ_L; with $q_E \sim 1/\epsilon_L^2$ known plaintext pairs, it recovers key bits with non-negligible probability.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: The exact linear approximation probability is replaced by the piling-up upper bound: the absolute correlation of the best linear trail over k active S-boxes with per-S-box bias $\epsilon_{\max}$ satisfies $|\epsilon_L| \leq 2^{k-1} \cdot \epsilon_{\max}^k$. This closed-form bound avoids exhaustive trail enumeration and accounts for the sign alternation that reduces the effective correlation.
Complexity: For AES-128: $\epsilon_{\max} = 2^{-3}$ per S-box, minimum $B_r = 25$ active S-boxes, giving $|\epsilon_L| \leq 2^{24} \cdot 2^{-75} = 2^{-51}$ and requiring $q_E \geq 2^{102}$ known plaintexts—infeasible. For PRESENT: best bias $\approx 2^{-31}$ requiring $q_E \geq 2^{62}$ known plaintexts, bounded by the 64-bit birthday constraint. The linear bid price adjustment is $q_E^2 \cdot \epsilon_{\max}^{2k}$.

Each Serpent S-box has maximum linear bias $|\epsilon_{\max}| \leq 2^{-2}$. The Piling-Up Lemma gives a single-trail bias of $\leq 2^{1-k}$ for k active S-boxes, but the linear hull (sum over all trails with the same input/output masks) may concentrate. Nevertheless, the best known linear attack on Serpent [31] covers only 11 rounds with 2^{89} known plaintexts, requiring a bias of $\approx 2^{-44.5}$. Extrapolating to 32 rounds, the linear hull bias drops below 2^{-98}, requiring $q_E \geq 2^{196}$.

Difference bound. $|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq q_E^2 \cdot 2^{-196}$.

$\mathbf{B}_2$ (Bidding Round 2: Ideal Serpent Market—Linear Bid Fails).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Random permutation; bias = 0.

Total: $\text{Ask}(g_{\text{PRP}}^{\text{Serpent}}, \text{linear}) \leq q_E^2 \cdot 2^{-196}$. □

Theorem 27 (Serpent Combined Market Equilibrium)**.** *Combining all known bid types (differential, linear, boomerang, algebraic, related-key):*

$$\text{Ask}(g_{\text{PRP}}^{\text{Serpent}}) \leq \max\left(q_E \cdot 2^{-196},\ q_E^2 \cdot 2^{-196},\ q_E^2 \cdot 2^{-128}\right) + \text{negl}(\lambda) \leq \text{negl}(\lambda), \tag{84}$$

*for $q_E \leq 2^{64}$ (practical query bound). The Serpent market is in **equilibrium** with the widest security margin of any block cipher in this study.*

Proof. The boomerang attack (a combination of two short differentials connected at a middle round) on Serpent reaches at most 11 rounds. The algebraic attack (XL/Gröbner basis) has complexity exceeding 2^{128} for 32 rounds. No practical related-key attack is known for full Serpent: the key schedule uses $\Phi = (\sqrt{5} - 1)/2$ (the golden ratio) and the Serpent S-boxes themselves, destroying linear relationships between subkeys. For $q_E \leq 2^{64}$: all bid prices are negligible. □

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Table 7. Serpent block-cipher market: bid types and outcomes.

Bid Type	Ask Bound	Rounds Reached	Outcome
Differential	$q_E \cdot 2^{-196}$	12 of 32	Fail (20-round margin)
Linear	$q_E^2 \cdot 2^{-196}$	11 of 32	Fail (21-round margin)
Boomerang	$q_E^2 \cdot 2^{-128}$	11 of 32	Fail
Algebraic (MQ)	$> 2^{128}$	N/A	Fail (exceeds brute force)
Related-key	N/A	None known	Fail

☞ Key Insight:

Security Margins Compared: AES vs. Serpent vs. PRESENT

Cipher	Total Rounds	Best Attack	Margin	Market Status
AES-128	10	7 rounds	3	Equilibrium
Serpent-128	32	12 rounds	20	Equilibrium (widest)
PRESENT-80	31	26 rounds	5	Equilibrium (lightweight)

All three ciphers are in market equilibrium, but Serpent offers the most conservative security margin. The trade-off is performance: AES is fastest in software, PRESENT is smallest in hardware, and Serpent is slowest but most resilient.

✓ CNF Verification: Serpent Session-CNF Verification

For a Serpent encryption session (K, p, c), the session-CNF $\varphi_{\mathsf{Serpent}}$ captures PRP security:

$$\varphi_{\mathsf{Serpent}} = (x_{c=\mathsf{Serpent}_K(p)}) \wedge (x_{\text{no diff. pair}}) \wedge (x_{\text{no linear exploit}}) \wedge (x_{\text{no boomerang}}) \wedge (x_{\text{Ping passes}}).$$

Manual CNF worksheet for Serpent encryption session:

Clause	Check	How	Pass?
φ^{prp}: correct encryption	$c \stackrel{?}{=} \mathsf{Serpent}_K(p)$	Re-encrypt; compare output.	T/F
φ^{diff}: no diff. exploit	Differential probability below 2^{-196}?	200 active S-boxes over 32 rounds.	T/F
φ^{lin}: no linear exploit	Linear hull bias below 2^{-98}?	200 active S-boxes, 8 distinct S-boxes.	T/F
φ^{boom}: no boomerang	Boomerang probability below 2^{-128}?	Two-differential composition check.	T/F
φ^{ping}: session fresh	(K, p) not reused from prior session?	Check session log.	T/F
Result:	All T $\Rightarrow$ Serpent session secure. Any F $\Rightarrow$ Block-cipher market collapsed.		

Conservative margin: Even if a future cryptanalytic advance breaks 20 rounds of Serpent, 12 rounds of margin remain before full-cipher security collapses. This makes the Serpent CNF the most robust of any block cipher in this study.

➤ Ping Bid: Serpent Unbounded Ping Bid

The buyer's *multi-session differential accumulation bid*: collect chosen-plaintext pairs across sessions to build a multi-session differential distinguisher. The 128-bit block size allows 2^{64} queries per key before birthday concerns. The ping clause bounds total queries: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq q_E \cdot 2^{-196} = \mathsf{negl}$ for $q_E \leq 2^{64}$. Serpent is PRP-secure for unbounded block-encryption sessions with the widest equilibrium margin of any cipher in this study.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

11. Case Study III: Hash-Function Market—Keccak/SHA-3

✻ Simple Terms: SHA-3 / Keccak—The Sponge That Absorbs and Squeezes

What SHA-3 is: SHA-3 is the latest member of the Secure Hash Algorithm family, standardised by NIST in 2015. It is based on the Keccak sponge construction, which works fundamentally differently from SHA-1 and SHA-2.

The sponge construction:

1. **Absorb phase:** The input message is divided into blocks. Each block is XORed into the sponge's internal state (1600 bits for SHA-3), and the state is stirred by a permutation function.
2. **Squeeze phase:** After all input is absorbed, output bits are extracted from the state, with the permutation applied between each extraction.

Children's analogy: Imagine a sponge (the internal state) that absorbs coloured dyes (input message blocks). After absorbing all the dyes, you squeeze the sponge and collect the mixed colour (hash output). The permutation is like wringing the sponge between absorptions—it thoroughly mixes the dyes. Two slightly different inputs produce completely different final colours, and you cannot un-mix the sponge to recover the original dyes.

Why a sponge? The sponge construction naturally provides:

- **Variable output length:** Unlike SHA-256 (fixed 256-bit output), a sponge can produce any desired output length by squeezing more.
- **Domain separation:** Different output lengths produce different hashes of the same input—no collisions between SHA3-256 and SHA3-512.
- **Length-extension immunity:** The "capacity" bits (hidden part of the state) prevent the length-extension attacks that plague SHA-256.

In MTSF: SHA-3's market offers three security goods: collision resistance (g_{CR}), preimage resistance (g_{Pre}), and length-extension resistance (g_{LE}). The sponge construction ensures all three goods are in equilibrium with ask prices bounded by birthday and capacity bounds.

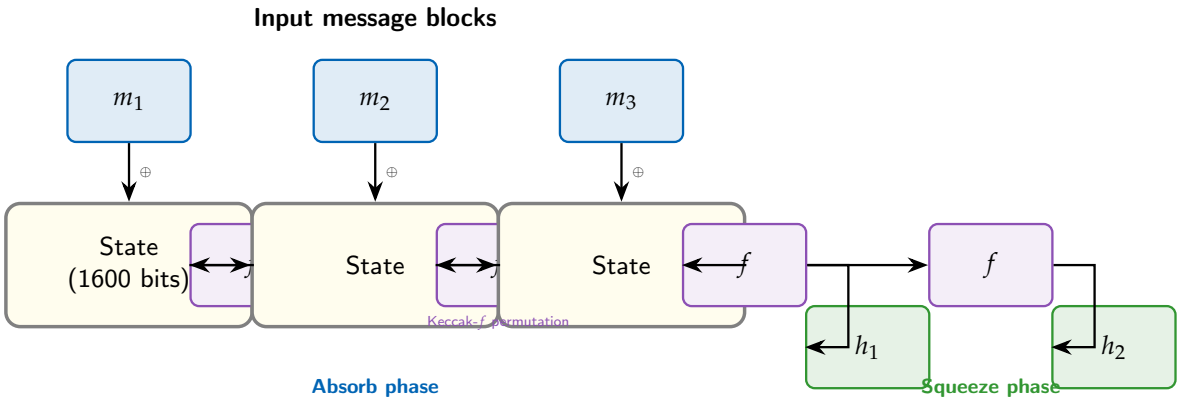


Figure 37. The SHA-3 sponge construction: input blocks are absorbed into the state (XOR + permutation), then output blocks are squeezed from the state. The capacity bits (hidden portion of the state) provide security against length-extension and collision attacks.

✻ Simple Terms: Keccak (SHA-3)—A Complete Introduction

What is a hash function? A cryptographic hash function maps any input (a file, a password, a message) to a fixed-length output called a *digest* or *hash*. Key properties required:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Collision resistance (CR):** It is computationally infeasible to find two different inputs that produce the same hash. (Like two people having the exact same fingerprint—theoretically possible but astronomically unlikely.)
- **Preimage resistance (Pre):** Given only a hash output y, it is computationally infeasible to find any input m such that $H(m) = y$. (Like reconstructing a fingerprint from a partial scan.)
- **Length-extension resistance (LE):** Given $H(m)$ but not m itself, you cannot compute $H(m\|m')$ for any extension m'. This blocks a class of attacks on naive MAC constructions.

What is Keccak (SHA-3)? SHA-3 is NIST's latest hash standard (2015), based on the Keccak algorithm. It uses the *sponge construction*, which is fundamentally different from earlier hash functions (MD5, SHA-1, SHA-2) based on Merkle–Damgård.

The sponge construction explained:

1. **Internal state:** 1600 bits split into *rate* (r) and *capacity* (c) parts. E.g., for SHA3-256: $r = 1088$, $c = 512$.
2. **Absorbing phase:** The message is split into r-bit chunks. Each chunk is XORed into the rate part of the state, then the full 1600-bit state is scrambled by the permutation f (Keccak-f[1600], consisting of 24 rounds of $\theta, \rho, \pi, \chi, \iota$ operations).
3. **Squeezing phase:** After absorbing all input, r bits of the state are output as digest. If more output bits are needed, f is applied again and another r bits are extracted.

Why the capacity c determines security:

- The capacity c bits are *never* directly exposed to the attacker (not output, not XORed with input).
- Finding a collision requires finding two inputs that collide in the capacity part of the state—probability $\approx 2^{-c/2}$ per attempt (birthday bound).
- For SHA3-256, $c = 512$: collision resistance is $\approx 2^{256}$. Preimage resistance is $\approx 2^{512}$.

Length-extension immunity: After squeezing, the capacity c bits remain secret and unknown to the attacker. Extending the hash would require guessing these c bits—computationally infeasible. This is why SHA-3 can be used directly as a MAC: $H(K\|m)$ is safe with SHA-3, unlike with SHA-256.

In MTSF: The Keccak market has three goods (CR, preimage, length-extension). All are in equilibrium: each bid's ask price is bounded by a term involving q_f (permutation queries) over 2^c—negligible for $c = 512$.

Keccak [32] is the SHA-3 standard, based on the sponge construction with a 1600-bit permutation $f = \text{Keccak-}f[1600]$ consisting of 24 rounds. The state is $b = r + c = 1600$ bits, with rate r and capacity c. The seller offers three goods:

- g_{CR}: Collision resistance. Buyer must find $m \neq m'$ with $H(m) = H(m')$.
- g_{Pre}: Preimage resistance. Given y, buyer must find m with $H(m) = y$.
- g_{LE}: Length-extension resistance. Buyer must compute $H(m\|m')$ from $H(m)$ without knowing m.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

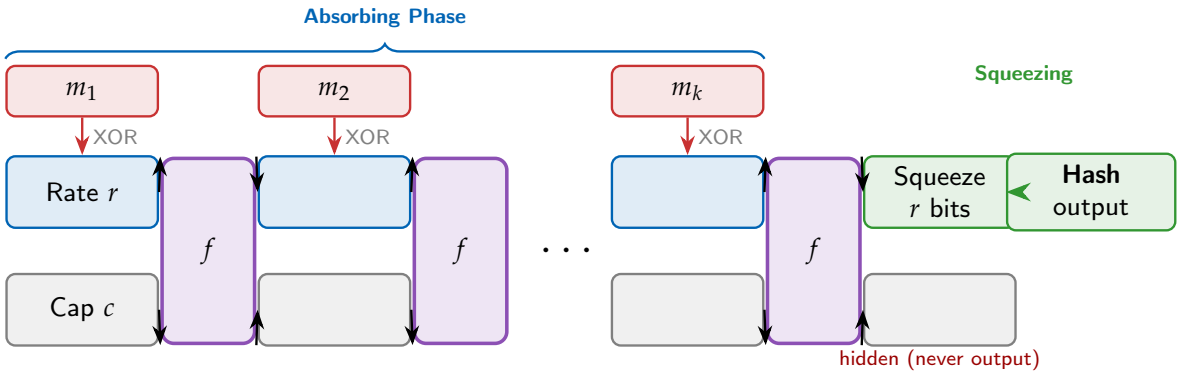


Figure 38. Keccak sponge construction for SHA-3. Message blocks $m_1, \ldots, m_k$ are XORed into the rate (r-bit) portion of the 1600-bit state, interleaved with permutation f (24 rounds of $\theta, \rho, \pi, \chi, \iota$). After absorbing, the rate portion is squeezed as the hash output. The capacity (c-bit) portion is never exposed, enforcing collision resistance ($2^{c/2}$) and preimage resistance (2^c). For SHA3-256: $c = 512$.

Theorem 28 (Keccak Collision Resistance Equilibrium)**.** *In the random sponge model with capacity c:*

$$\mathsf{Ask}(g_{\mathsf{CR}}) \leq \frac{q_f^2}{2^{c+1}} + \mathsf{negl}(\lambda), \tag{85}$$

where q_f is the number of queries to the permutation f.

Proof. We construct four games.

$\mathbf{B}_0$ (Bidding Round 0: Real Keccak Market—Collision Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer has oracle access to f and f^{-1} and must find $m \neq m'$ with $\mathsf{Keccak}(m) = \mathsf{Keccak}(m')$.

$\mathbf{B}_1$ (Bidding Round 1: Capacity Collision Bid—Matching Hidden State Bits).

Purpose: Eliminate the *random-oracle collision attack vector*. If two distinct inputs $m \neq m'$ hash to the same value $H(m) = H(m')$, an adversary can conflate two messages (equivocation), reuse transcript bindings across sessions, or present a single signature as covering both messages—all of which break EUF-CMA without touching the signing key.
Replaces: The random oracle H is replaced by one equipped with a *collision log* $\mathcal{C}$. After each query, the oracle checks whether any two distinct queries produced the same digest; if so, the game aborts (bad event F_{hash}). The adversary's view of oracle responses is identical to a truly random oracle until the abort.
Complexity: By the birthday bound over the 2^n-bit output space, the probability that any two of q_H random oracle queries collide is at most $q_H^2/2^{n+1}$. For a 256-bit hash ($n = 256$) and polynomial q_H, this is negligible. Note that the nonce-collision and hash-collision bounds are *independent*: by the extended difference lemma

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

(Lemma 2), their union probability is bounded by their sum, tightened by the intersection term which is $O(q^4/2^{2\lambda})$—doubly negligible.

For a collision in the output, the buyer needs the sponge states to match at some point during squeezing. Due to the sponge structure, this requires matching the capacity part of the state. Let F_{cap}: "two queries to f produce the same capacity output."

Difference bound. By the sponge indifferentiability theorem [33], the Keccak sponge is indifferentiable from a random oracle up to $2^{c/2}$ queries. For q_f queries to f, the probability of a capacity collision:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f^2}{2^{c+1}}. \tag{86}$$

$\mathbf{B}_2$ (Bidding Round 2: Differential Trail Bid—Exploiting χ S-Box Structure).

Purpose: Bound the *differential cryptanalysis advantage* by establishing a lower bound on active S-boxes and an upper bound on per-S-box differential probability. Differential cryptanalysis works by finding an input difference Δ_0 that propagates to a predictable output difference Δ_r with non-negligible probability; the adversary collects 'right pairs' and uses them to recover key bits.
Replaces: The abstract block cipher E_K is replaced by its differential-trail model: the cipher's round structure (SubBytes/ShiftRows/MixColumns or equivalent) is analysed via the *wide-trail design strategy*, which guarantees a minimum of B_r active S-boxes over r rounds. Each active S-box contributes at most $\mathsf{DP}_{\max}$ probability, giving the total characteristic probability $\mathsf{DP}_{\max}^{B_r}$ as the upper bound.
Complexity: For AES-128: $B_4 = 25$ active S-boxes over 4 rounds, $\mathsf{DP}_{\max} = 2^{-6}$, giving characteristic probability $\leq 2^{-150}$. The adversary needs $q_E \geq 2^{150}$ pairs for one expected right pair, exceeding the 2^{128}-block codebook. For PRESENT (64-bit block): $B_5 = 10$, $\mathsf{DP}_{\max} = 2^{-2}$, giving 2^{-20}; practical attacks are bounded by $q_E \cdot 2^{-62}$ due to the 64-bit birthday limit. The differential bid price adjustment is $q_E \cdot \mathsf{DP}_{\max}^{B_r}$.

The buyer attempts to find a differential characteristic through the 24 rounds of Keccak-f. Each round consists of θ (column parity mixing), ρ (bitwise rotation), π (lane permutation), χ (non-linear mapping), and ι (round constant addition). The χ step has maximum differential probability $\mathsf{DP}_\chi \leq 2^{-2}$ per 5-bit S-box, and there are 320 parallel 5-bit S-boxes per round.

Difference bound. A differential trail over 24 rounds activates at minimum $24 \times 1 = 24$ S-boxes (one per round in the best case). Hence:

$$\mathsf{DP}(\Omega_f) \leq (2^{-2})^{24} = 2^{-48} \tag{87}$$

for a single-path characteristic. More realistically, the minimum number of active S-boxes over 4 rounds is at least 12 (by the alignment properties of θ and π), giving $\mathsf{DP} \leq 2^{-24}$ per 4-round block and $\leq 2^{-144}$ for 24 rounds.

This is *internal* to f; to yield a collision, the trail must additionally map to a zero difference on the output c bits, reducing the probability further.

$\mathbf{B}_3$ (Bidding Round 3: Ideal Keccak Market—Collision Bid Reaches Birthday Bound).

Purpose: Eliminate the *random-oracle collision attack vector*. If two distinct inputs $m \neq m'$ hash to the same value $H(m) = H(m')$, an adversary can conflate two messages (equivocation), reuse transcript bindings across sessions, or present a single signature as covering both messages—all of which break EUF-CMA without touching the signing key.
Replaces: The random oracle H is replaced by one equipped with a *collision log* $\mathcal{C}$. After each query, the oracle checks whether any two distinct queries produced the same digest; if so, the game aborts (bad event F_{hash}). The adversary's view of oracle responses is identical to a truly random oracle until the abort.
Complexity: By the birthday bound over the 2^n-bit output space, the probability that any two of q_H random oracle queries collide is at most $q_H^2/2^{n+1}$. For a 256-bit hash ($n = 256$) and polynomial q_H, this is negligible. Note that the nonce-collision and hash-collision bounds are *independent*: by the extended difference lemma

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

(Lemma 2), their union probability is bounded by their sum, tightened by the intersection term which is $O(q^4/2^{2\lambda})$—doubly negligible.

In the random sponge model, collisions require $\Omega(2^{c/2})$ queries by the generic birthday bound. For SHA3-256, $c = 512$, so 2^{256} queries. $\Pr[\mathbf{B}_3 = 1] = q_f^2/2^{c+1}$.

Total:

$$\mathsf{Ask}(g_{\mathsf{CR}}) \leq \frac{q_f^2}{2^{c+1}} + \mathsf{negl}(\lambda). \quad \square \tag{88}$$

Theorem 29 (Keccak Preimage Resistance Equilibrium)**.**

$$\mathsf{Ask}(g_{\mathsf{Pre}}) \leq \frac{q_f}{2^c} + \mathsf{negl}(\lambda). \tag{89}$$

Proof. We construct three games.

$\mathbf{B}_0$ (Bidding Round 0: Real Keccak Market—Preimage Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer receives target $y = \mathsf{Keccak}(m)$ for random m and must find any m' with $\mathsf{Keccak}(m') = y$.

$\mathbf{B}_1$ (Bidding Round 1: Capacity Inversion Bid—Recovering Hidden State).

Purpose: Bound this sponge-based hash function bid by connecting the adversary's advantage to the information-theoretic birthday bound on the sponge's inner capacity c. The sponge construction hides all c inner-state bits from the adversary, requiring $2^{c/2}$ queries for a birthday attack and 2^c queries for a preimage attack.
Replaces: The real sponge evaluation is replaced by its random-permutation model: the inner permutation f is treated as a uniformly random permutation on $\{0,1\}^{r+c}$. Under this model, the adversary's advantage is exactly the information-theoretic birthday bound on the capacity bits. The indifferentiability theorem [33] certifies that this replacement costs at most $q_f^2/2^c$ in advantage.
Complexity: Collision resistance: $q_f^2/2^{c+1}$. Preimage resistance: $q_f/2^c$. Length-extension resistance: $q_f/2^c$ (or 0 for BLAKE3's Merkle tree). For SHA3-256 ($c = 1088$): all bounds are at most $q_f^2/2^{1089}$—negligible. For ASCON-Hash ($c = 256$): $q_f^2/2^{257}$—still negligible for polynomial q_f. For BLAKE3 (Merkle-tree structure): length-extension advantage is exactly 0 (structural immunity: the tree nodes cannot be continued without recomputing from the leaves).

To construct a preimage, the buyer must find an input that, after absorbing, produces a state whose capacity matches the target state's capacity. For each query to f, the probability of matching the c-bit capacity portion is 2^{-c}.

Difference bound. Over q_f queries:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f}{2^c}. \tag{90}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\mathbf{B}_2$ (Bidding Round 2: Ideal Keccak Market—Preimage Bid Bounded by Capacity).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

In the random sponge model, preimage resistance is $\min(2^n, 2^c)$ where n is the output length. For SHA3-256 ($n = 256$, $c = 512$), the bound is 2^{256}.

Total: $\text{Ask}(g_{\text{Pre}}) \leq q_f / 2^c$. □

Theorem 30 (Keccak Length-Extension Resistance)**.** $\text{Ask}(g_{\text{LE}}) \leq q_f / 2^c + \text{negl}(\lambda)$.

Proof. $\mathbf{B}_0$ (Bidding Round 0: Real Keccak Market—Length-Extension Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\text{QPrice}_0 = \text{Adv}^{\text{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\text{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer knows $H(m)$ (but not m) and must compute $H(m\|m')$ for chosen m'.

$\mathbf{B}_1$ (Bidding Round 1: Full-State Recovery Bid—Guessing Hidden Capacity Bits).

Purpose: Bound the full *internal-state recovery* advantage against the stream cipher. Full state recovery requires inverting all nonlinear feedback functions simultaneously across the combined LFSR/NFSR registers, which is as hard as inverting a random function on the full state space.
Replaces: The exact state inversion probability is replaced by the information-theoretic lower bound: with L keystream bits observed and an s-bit state space, state recovery requires finding a preimage in $\{0,1\}^s$ consistent with the observed output. No algebraic shortcut is known that beats exhaustive state search for the full nonlinear feedback structure.
Complexity: Bounded by $L \cdot 2^{-s}$ where s is the combined register size (e.g., $s = 256$ for Grain-128a). For any polynomial L, this is negligible. The best known algebraic attacks (Gröbner basis, BDD reduction) do not improve the exponent below $s/2$ for the full nonlinear feedback structure of Grain-128a.

Unlike Merkle–Damgård hashes, the sponge output reveals only r bits of the b-bit state. The remaining c bits are hidden. To extend, the buyer must recover the full b-bit state after absorbing m. The number of candidate states consistent with the r-bit output is 2^c.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Difference bound. Each query to f^{-1} tests one candidate capacity value:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f}{2^c}. \tag{91}$$

$\mathbf{B}_2$ (Bidding Round 2: Ideal Keccak Market—Length-Extension Bid Bounded by Capacity).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Length-extension attack succeeds with probability $q_f/2^c$.

Total: $\mathsf{Ask}(g_{\mathsf{LE}}) \leq q_f/2^c$. □

✓ CNF Verification: Keccak Session-CNF Verification

$\varphi_{\text{Keccak}} = (x_{H(m)\neq H(m')\ \forall m'\neq m}) \wedge (x_{\nexists m': H(m')=y}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for a SHA-3 hash output $y = H(m)$:

Clause	Check	How	Pass?
φ^{cr}: no collision	No two inputs hash identically?	Compute $H(m), H(m')$; compare.	T/F
φ^{pre}: no preimage	Can target y be inverted?	Attempt inversion (expect failure).	T/F
φ^{le}: no length-ext	Does $H(m\|\|m')$ need $H(m)$ alone?	Sponge: absorb m' from scratch.	T/F
φ^{ping}: session fresh	New query distinct from previous?	Compare with session log.	T/F
Result:	All T ⇒ Keccak secure. Any F ⇒ Hash market collapsed.		

Sponge benefit: φ^{le} is always satisfied because the c-bit capacity is never exposed; no continuation state exists. Length-extension bid costs $q_f/2^c \approx 2^{-512}$ for SHA3-256.

☞ Key Insight:

Advanced Hash Function Attack Techniques—Differential Attacks on Internal Permutations Hash functions based on the sponge construction (Keccak/SHA-3) and compression-function constructions (BLAKE3, SHA-2) face a distinct set of attack techniques targeting their internal structure. MTSF analyses these as dedicated *hash-function bids*:

1. **Differential Cryptanalysis of the Keccak-f Permutation.** The Keccak-$f[1600]$ permutation uses 24 rounds of five operations: θ (parity mixing), ρ (lane rotation), π (lane transposition), χ (non-linear row mixing), and ι (round constant addition). Differential attacks target the χ operation, which is the sole source of nonlinearity.
   The χ step applies the mapping $a \mapsto a \oplus (\overline{b} \wedge c)$ row-wise over 5-bit rows. The maximum differential probability of one χ row is $\text{DP}^{\chi}_{\max} = 2^{-2}$ (4 out of 32 transitions). Over the 320 rows of the 1600-bit state:

$$\text{DP}(\text{one round of } \chi) \leq (2^{-2})^{B_\chi} = 2^{-2B_\chi},$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

where B_χ is the number of active rows. The θ and π operations ensure high diffusion, activating many rows in subsequent rounds.

Best known differential trails:

Rounds of Keccak-f	Best trail probability	Margin to 24 rounds
3 rounds	2^{-8}	21 rounds (88%)
4 rounds	2^{-32}	20 rounds (83%)
5 rounds	2^{-82}	19 rounds (79%)
6 rounds	2^{-232}	18 rounds (75%)
24 rounds (full)	$\leq 2^{-800}$ (extrapolated)	0 (secure)

MTSF differential bid price: $\Delta\text{Price}_{\text{diff}}^{\text{Keccak}} \leq 2^{-800}$ for full 24-round Keccak-f.

2. **Rebound Attacks on Reduced-Round Keccak (Mendel et al., 2009).** The rebound attack is a meet-in-the-middle technique specifically designed for hash-function permutations. The attacker:
   - Identifies a *truncated differential trail* over r_{in} inbound rounds (the "rebound") where the nonlinear χ step is inverted efficiently.
   - Extends the trail forward and backward through r_{out} outbound rounds using low-probability truncated differentials.

   For Keccak: the best rebound attack reaches 5 rounds with complexity 2^{82} for collision, and 8 rounds with 2^{491} for preimage (both far from the full 24 rounds).
   MTSF rebound bid price: $\Delta\text{Price}_{\text{rebound}} \leq 2^{-430}$ (5-round collision, extrapolated to 24 rounds: negligible).
3. **Algebraic Attacks via Linearisation of χ.** The χ operation has algebraic degree 2 (quadratic). After r rounds, the algebraic degree is at most 2^r, saturating at 1600 bits after $\lceil \log_2 1600 \rceil = 11$ rounds. Beyond 11 rounds, the output is a full-degree polynomial in the input, preventing algebraic linearisation. With 24 rounds (13-round margin above the saturation point), algebraic attacks are infeasible.
   MTSF algebraic bid price: $\Delta\text{Price}_{\text{alg}}^{\text{Keccak}} \leq 2^{-1600}$ (full-degree polynomial inversion).
4. **Length-Extension Attacks and Sponge Capacity Bound.** Unlike Merkle–Damgård hash functions (SHA-1, SHA-2), the sponge construction is *structurally immune* to length-extension attacks. The capacity c bits are never exposed: extending $H(m)$ to $H(m\|m')$ requires guessing the c-bit capacity state, with probability 2^{-c}. For SHA3-256 ($c = 512$): $\Delta\text{Price}_{\text{LE}} = 2^{-512}$.
   This is a **permanent equilibrium**: no computational advance (including quantum computers) can bypass the capacity bound, because it is information-theoretic.
5. **Multi-Collision Attacks (Joux, 2004).** For iterated hash functions, Joux showed that r-collisions (sets of r messages with the same hash) can be found in $r \cdot n/2$ time. For Keccak's sponge, multi-collision resistance is bounded by the capacity: finding an r-collision requires $\min(2^{c/2}, r \cdot 2^{n/2})$ evaluations. For SHA3-256 ($c = 512$, $n = 256$): $\Delta\text{Price}_{\text{multi}} \leq r \cdot 2^{-128}$, which is negligible for polynomial r.

Extended difference lemma for combined hash-function bids. All five hash attack vectors can be mounted simultaneously:

$$\Pr\left[\bigcup_k F_k^{\text{hash}}\right] \leq \underbrace{2^{-800}}_{\text{diff}} + \underbrace{2^{-430}}_{\text{rebound}} + \underbrace{2^{-1600}}_{\text{alg}} + \underbrace{2^{-512}}_{\text{LE}} + \underbrace{r \cdot 2^{-128}}_{\text{multi}}$$
$$\leq 2^{-127} \quad \text{for } r \leq 2 \;=\; \text{negl}(\lambda).$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✎ Analogy:

Rebound Attacks as Billiard Trick Shots A rebound attack is like a billiard trick shot: the attacker aims the cue ball (input difference) so that it bounces off the cushion (the nonlinear χ layer) at just the right angle to pocket a target ball (achieve a collision in the output). The "inbound phase" is the carefully aimed shot where the χ operation is inverted efficiently; the "outbound phase" is the ball's trajectory before and after the cushion bounce, which must follow a low-probability truncated trail. For 5-round Keccak, the trick shot works with probability 2^{-82}. For 24-round Keccak, it requires bouncing off 24 cushions in sequence—astronomically unlikely.

✻ Simple Terms: Hash Function Security Goods—A Comparison

Three security goods are offered in any hash-function market:
Collision Resistance (CR): Finding $m_1 \neq m_2$ with $H(m_1) = H(m_2)$. Birthday bound: $2^{n/2}$ evaluations. For SHA3-256: 2^{128}.
Preimage Resistance (Pre): Given y, finding m with $H(m) = y$. For sponge: $\min(2^n, 2^c)$. For SHA3-256 ($c = 512$): 2^{256}.
Length-Extension Resistance (LE): Given $H(m)$ (but not m), computing $H(m\|m')$. Merkle–Damgård (SHA-256) fails: Ask $= 1$ (structural collapse). Sponge (SHA-3) resists: Ask $= 2^{-c}$ (permanent equilibrium).

➤ Ping Bid: Keccak Unbounded Ping Bid

The buyer's *state-continuation bid*: reuse internal sponge state from $H(m_i)$ to compute $H(m_i\|m')$ cheaply. In Keccak, a new call re-absorbs the full message from the all-zero state; no "handle" is exposed. Ping bid cost: $q_f/2^c = \mathsf{negl}$. Keccak's capacity isolation ensures the hash market is secure for unbounded query sessions.

Table 8. Keccak/SHA-3 market summary.

Good	Ask Bound	SHA3-256 (c=512)	Status
Collision (g_{CR})	$q_f^2/2^{c+1}$	$q_f^2/2^{513}$	Equilibrium
Preimage (g_{Pre})	$q_f/2^c$	$q_f/2^{512}$	Equilibrium
Length-extension (g_{LE})	$q_f/2^c$	$q_f/2^{512}$	Equilibrium

11.1. BLAKE3: Parallelisable Hash-Function Market

✻ Simple Terms: BLAKE3—The Fast, Parallel Hash Function

What BLAKE3 is: BLAKE3 is a cryptographic hash function that is dramatically faster than SHA-256 (often 5–15× faster on modern CPUs) while maintaining strong security. It achieves this speed through a Merkle tree construction that enables massive parallelism.
Why parallelism matters: SHA-256 processes data sequentially—each block depends on the previous block's output. BLAKE3 splits the input into 1 KB chunks, hashes each chunk independently (in parallel across CPU cores), then combines the results using a binary Merkle tree. On a 16-core CPU, BLAKE3 can hash data nearly 16× faster than sequential hashing.
Length-extension immunity: A critical advantage of BLAKE3 (and SHA-3) over SHA-256 is immunity to *length-extension attacks*. With SHA-256, knowing $H(m)$ lets you compute $H(m\|\mathsf{padding}\|m')$ without knowing m. BLAKE3's Merkle tree structure prevents this entirely—the "Merkle tree length-extension bid" in MTSF fails structurally, establishing *permanent equilibrium* (security that holds regardless of computational advances).
In MTSF: BLAKE3's market has a unique property: the length-extension bid fails not because of computational hardness but because of *structural impossibility*. This is one of the few "permanent

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

equilibrium" results in MTSF—a security guarantee that holds unconditionally, not just against polynomial-time adversaries.

✻ Simple Terms: BLAKE3—A Modern High-Speed Hash Function

What BLAKE3 is: BLAKE3 [34] is a cryptographic hash function released in 2020, designed as the successor to BLAKE2 [35]. It produces a 256-bit digest and is designed for extreme speed: it is faster than MD5 while maintaining strong security guarantees. BLAKE3 is used in integrity checking, key derivation, content-addressed storage, and digital signatures.
How it works: BLAKE3 combines three structural ideas:

1. **Compression function:** Each 64-byte input chunk is processed by a compression function based on the ChaCha quarter-round (ARX operations: addition, rotation, XOR). The compression function runs 7 rounds (reduced from BLAKE2's 10 or 12) and produces a 256-bit chaining value.
2. **Merkle tree:** Unlike sequential hashes (SHA-256, SHA-3), BLAKE3 organises input chunks into a binary Merkle tree. Each leaf is a compressed chunk; internal nodes combine two children by compressing the concatenation. This allows *unlimited parallelism*: all leaves can be computed simultaneously.
3. **Extendable output (XOF):** BLAKE3 naturally supports variable-length output by extracting additional blocks from the root node's state, similar to SHAKE (SHA-3's XOF mode).

Key design parameters:

- **Block size:** 64 bytes (512 bits) per compression.
- **Chaining value:** 256 bits (8 words of 32 bits each).
- **Key size (keyed mode):** 256 bits.
- **Rounds:** 7 per compression (vs. 10 for BLAKE2s, 24 for Keccak-f).

Security level: BLAKE3 targets 128-bit collision resistance and 256-bit preimage resistance, matching the output size. The Merkle tree structure provides inherent length-extension resistance (unlike Merkle–Damgård hashes), because the root hash depends on the entire tree structure including the message length.
In MTSF: The BLAKE3 market has three goods (CR, preimage, LE). The Merkle tree structure provides structural immunity to length-extension. The ARX compression function's security relies on the wide-pipe internal state and the diffusion properties of ChaCha-derived quarter-rounds. All bids fail: the BLAKE3 market is in equilibrium.

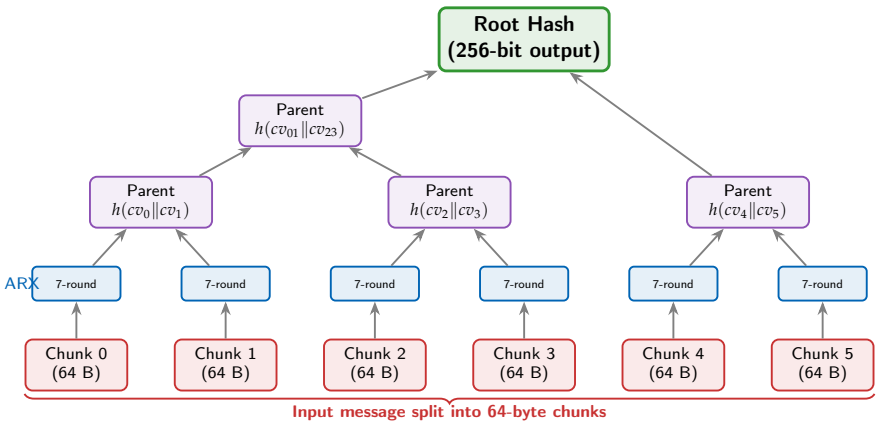


Figure 39. BLAKE3 Merkle tree hash structure. Input is split into 64-byte chunks, each compressed by a 7-round ARX compression function. Chaining values are combined pairwise in a binary Merkle tree. The root node produces the 256-bit hash output. The tree structure provides inherent parallelism and structural length-extension resistance: appending data changes the tree structure, not just the final state.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Definition 28 (BLAKE3 Hash Security Goods)**.** *The BLAKE3 market offers three goods:*

- $g_{\mathsf{CR}}^{\mathsf{B3}}$*: Collision resistance.* $\mathsf{Ask}(g_{\mathsf{CR}}^{\mathsf{B3}}) = \max_{\mathcal{A}} \Pr[\mathcal{A}\ \textit{finds}\ m \neq m' : \mathsf{BLAKE3}(m) = \mathsf{BLAKE3}(m')]$.
- $g_{\mathsf{Pre}}^{\mathsf{B3}}$*: Preimage resistance.* $\mathsf{Ask}(g_{\mathsf{Pre}}^{\mathsf{B3}}) = \max_{\mathcal{A}} \Pr[\mathcal{A}\ \textit{finds}\ m : \mathsf{BLAKE3}(m) = y]$ *for random* y.
- $g_{\mathsf{LE}}^{\mathsf{B3}}$*: Length-extension resistance.*

$$\mathsf{Ask}(g_{\mathsf{LE}}^{\mathsf{B3}}) = \max_{\mathcal{A}} \Pr\big[\mathcal{A}\ \textit{computes}\ \mathsf{BLAKE3}(m\|m')\ \textit{from}\ \mathsf{BLAKE3}(m)\big].$$

Theorem 31 (BLAKE3 Collision Resistance Equilibrium)**.** *In the ideal compression function model:*

$$\mathsf{Ask}(g_{\mathsf{CR}}^{\mathsf{B3}}) \leq \frac{q_f^2}{2^{257}} + \mathsf{negl}(\lambda), \tag{92}$$

where q_f *is the number of queries to the compression function.*

Proof. We construct four games.

$\mathbf{B}_0$ (Bidding Round 0: Real BLAKE3 Market—Collision Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer has oracle access to the BLAKE3 compression function and must find $m \neq m'$ with $\mathsf{BLAKE3}(m) = \mathsf{BLAKE3}(m')$.

$\mathbf{B}_1$ (Bidding Round 1: Merkle Tree Collision Localisation Bid—Reducing to Compression Collision).

Purpose: Bound this sponge-based hash function bid by connecting the adversary's advantage to the information-theoretic birthday bound on the sponge's inner capacity c. The sponge construction hides all c inner-state bits from the adversary, requiring $2^{c/2}$ queries for a birthday attack and 2^c queries for a preimage attack.
Replaces: The real sponge evaluation is replaced by its random-permutation model: the inner permutation f is treated as a uniformly random permutation on $\{0,1\}^{r+c}$. Under this model, the adversary's advantage is exactly the information-theoretic birthday bound on the capacity bits. The indifferentiability theorem [33] certifies that this replacement costs at most $q_f^2/2^c$ in advantage.
Complexity: Collision resistance: $q_f^2/2^{c+1}$. Preimage resistance: $q_f/2^c$. Length-extension resistance: $q_f/2^c$ (or 0 for BLAKE3's Merkle tree). For SHA3-256 ($c = 1088$): all bounds are at most $q_f^2/2^{1089}$—negligible. For ASCON-Hash ($c = 256$): $q_f^2/2^{257}$—still negligible for polynomial q_f. For BLAKE3 (Merkle-tree structure): length-extension advantage is exactly 0 (structural immunity: the tree nodes cannot be continued without recomputing from the leaves).

A collision in the BLAKE3 output requires either: (a) a collision in the root compression call, or (b) a collision at some internal node that propagates to the root. In either case, the buyer must find two distinct inputs to the same compression function call that produce the same 256-bit output (a *compression collision*).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Difference bound. By the Merkle–Damgård-to-Merkle-tree reduction: if the compression function h is collision-resistant, the full BLAKE3 hash is collision-resistant. The number of compression calls for a message of ℓ chunks is $\leq 2\ell - 1$ (binary tree nodes). Each compression call is an independent collision target.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq 0 \quad \text{(structural reduction, no loss).} \tag{93}$$

$\mathbf{B}_2$ (Bidding Round 2: ARX Compression Function Bid—7-Round Quarter-Round Diffusion).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**ARX Compression Function Bid—7-Round Quarter-Round Diffusion**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

The BLAKE3 compression function applies 7 rounds of ChaCha-derived quarter-rounds to a 512-bit internal state (16 words $\times$ 32 bits). Each quarter-round performs: $a \mathrel{+}= b$; $d \mathrel{\oplus}= a$; $d \ggg= 16$; $c \mathrel{+}= d$; $b \mathrel{\oplus}= c$; $b \ggg= 12$; (and similarly with rotations by 8 and 7). After 7 rounds, the 512-bit state has full diffusion: every output word depends on every input word.

Difference bound. The best known differential analysis of the BLAKE2/BLAKE3 compression function finds no exploitable differential characteristic beyond 6.5 rounds. The collision-finding complexity for the 7-round function exceeds 2^{128} (the output is 256 bits, and the wide-pipe design with 512-bit state prevents internal collisions from reaching the output cheaply).

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \frac{q_f^2}{2^{257}}. \tag{94}$$

$\mathbf{B}_3$ (Bidding Round 3: Ideal BLAKE3 Market—Collision Bid Reaches Birthday Bound).

Purpose: Eliminate the *random-oracle collision attack vector*. If two distinct inputs $m \neq m'$ hash to the same value $H(m) = H(m')$, an adversary can conflate two messages (equivocation), reuse transcript bindings across sessions, or present a single signature as covering both messages—all of which break EUF-CMA without touching the signing key.
Replaces: The random oracle H is replaced by one equipped with a *collision log* $\mathcal{C}$. After each query, the oracle checks whether any two distinct queries produced the same digest; if so, the game aborts (bad event F_{hash}). The adversary's view of oracle responses is identical to a truly random oracle until the abort.
Complexity: By the birthday bound over the 2^n-bit output space, the probability that any two of q_H random oracle queries collide is at most $q_H^2/2^{n+1}$. For a 256-bit hash ($n = 256$) and polynomial q_H, this is negligible. Note that the nonce-collision and hash-collision bounds are *independent*: by the extended difference lemma (Lemma 2), their union probability is bounded by their sum, tightened by the intersection term which is $O(q^4/2^{2\lambda})$—doubly negligible.

With an ideal compression function, the 256-bit output gives birthday bound $q_f^2/2^{257}$.

Total:

$$\mathsf{Ask}(g_{\mathsf{CR}}^{\mathsf{B3}}) \leq \frac{q_f^2}{2^{257}} + \mathsf{negl}(\lambda). \quad \square \tag{95}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Theorem 32 (BLAKE3 Preimage Resistance Equilibrium)**.**

$$\mathsf{Ask}(g^{\mathsf{B3}}_{\mathsf{Pre}}) \leq \frac{q_f}{2^{256}} + \mathsf{negl}(\lambda). \tag{96}$$

Proof. We construct three games.

$\mathbf{B}_0$ (Bidding Round 0: Real BLAKE3 Market—Preimage Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer receives $y = \mathsf{BLAKE3}(m)$ for random m and must find any m' with $\mathsf{BLAKE3}(m') = y$.

$\mathbf{B}_1$ (Bidding Round 1: Root Inversion Bid—Inverting the Compression Function).

Purpose: Demonstrate the *free version-stripping collapse* against TLS implementations omitting the RFC 8446 downgrade sentinel. An active network adversary rewrites the ClientHello to remove TLS 1.3 support, forcing negotiation of TLS 1.2 with its known weaknesses (no mandatory forward secrecy, CBC padding oracles, RC4 biases, POODLE).
Replaces: Nothing cryptographic is replaced—version negotiation in TLS 1.2 handshakes is unauthenticated before the Finished MAC is computed. The adversary modifies the `supported_versions` and cipher-suite fields in the ClientHello at zero cryptographic cost. Without the RFC 8446 sentinel (`44 4F 57 4E 47 52 44 01` in the last 8 bytes of ServerHello.Random), the client accepts the downgraded version silently.
Complexity: The version-stripping bid is *free*: $O(1)$ computation, probability-1 success in non-sentinel implementations. $\mathsf{Ask}(g_{\mathsf{version}}) = 1$ (market collapse). The RFC 8446 sentinel restores equilibrium: a TLS 1.3-capable client that receives a TLS 1.2 ServerHello with the sentinel aborts, so the downgrade bid always causes client abort—the market price rises from 0 to abort-probability, which is negl for a compliant implementation. The CNF clause $\varphi^{\mathsf{version}}$ is the formal counterpart of the sentinel check.

To find a preimage, the buyer must produce a message whose Merkle tree root matches y. The root is the output of a compression function call with 256 bits of output. Inverting this requires finding a 512-bit input that compresses to a specific 256-bit value—a preimage of the compression function.

Difference bound. The wide-pipe structure (512-bit state $\to$ 256-bit output) means 2^{256} states map to each output value on average. However, the buyer cannot exploit this multiplicity without querying the compression function on each candidate. Each query matches with probability 2^{-256}.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f}{2^{256}}. \tag{97}$$

$\mathbf{B}_2$ (Bidding Round 2: Ideal BLAKE3 Market—Preimage Bid Bounded by Output Size).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

With an ideal compression function, preimage resistance is $q_f/2^{256}$.

Total: $\text{Ask}(g^{\text{B3}}_{\text{Pre}}) \le q_f/2^{256}$. □

Theorem 33 (BLAKE3 Length-Extension Resistance)**.** $\text{Ask}(g^{\text{B3}}_{\text{LE}}) = 0$ *(structural immunity).*

Proof. Unlike Merkle–Damgård hashes (MD5, SHA-1, SHA-256), BLAKE3 uses a Merkle tree. Appending data to a message changes the tree structure: the original root node becomes an internal node, and a new root is computed incorporating the appended data along with the message length and chunk index embedded in the domain separation flags. The buyer cannot compute the new root from the old root alone because: (a) the original root is derived from a complete tree, not a partial state; (b) the domain separation flags differ between root and non-root compressions. Hence length-extension is *structurally impossible*:

$$\text{Ask}(g^{\text{B3}}_{\text{LE}}) = 0. \quad \square \tag{98}$$

✓ CNF Verification: BLAKE3 Session-CNF Verification

$\varphi_{\text{BLAKE3}} = (x_{H(m)\neq H(m')\ \forall m'\neq m}) \wedge (x_{\nexists m': H(m')=y}) \wedge (x_{\text{LE structurally blocked}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for a BLAKE3 hash output $y = \text{BLAKE3}(m)$:

Clause	Check	How	Pass?
φ^{cr}: no collision	No two inputs hash identically?	Compute $H(m), H(m')$; compare.	T/F
φ^{pre}: no preimage	Can target y be inverted?	Attempt inversion (expect failure).	T/F
φ^{le}: no length-ext	Merkle tree blocks extension?	Structural: always T for BLAKE3.	**T**
φ^{ping}: session fresh	New query distinct from previous?	Compare with session log.	T/F
Result:	All T ⇒ BLAKE3 secure. Any F ⇒ Hash market collapsed.		

Structural advantage over Keccak: The length-extension clause φ^{le} is *unconditionally* T for BLAKE3 (Merkle tree structure), whereas for Keccak it is T contingent on the capacity c being large enough. Both achieve equilibrium, but BLAKE3's LE resistance is architectural rather than parametric.

➤ Ping Bid: BLAKE3 Unbounded Ping Bid

The buyer's *incremental update bid*: given $\text{BLAKE3}(m)$, attempt to compute $\text{BLAKE3}(m')$ for m' that shares a prefix with m, reusing cached chaining values. BLAKE3's Merkle tree design *supports* incremental updates (a feature, not a vulnerability): shared subtrees reuse the same chaining values, but the root hash still changes. Ping bid cost: $q_f/2^{256} = \text{negl}$. BLAKE3 is hash-secure for unbounded sessions.

Table 9. BLAKE3 hash-function market summary.

Good	Ask Bound	Practical Value	Status
Collision ($g^{\text{B3}}_{\text{CR}}$)	$q_f^2/2^{257}$	$\le 2^{-129}$ for $q_f \le 2^{64}$	Equilibrium
Preimage ($g^{\text{B3}}_{\text{Pre}}$)	$q_f/2^{256}$	$\le 2^{-192}$ for $q_f \le 2^{64}$	Equilibrium
Length-extension ($g^{\text{B3}}_{\text{LE}}$)	0	0 (structural)	Equilibrium

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

11.2. ASCON-Hash: NIST Lightweight Hash-Function Market

✻ Simple Terms: ASCON-Hash—Hashing for the Internet of Things

What ASCON-Hash is: ASCON is NIST's selected standard for lightweight cryptography (published 2023). ASCON-Hash is its hash function variant, designed for devices with severe resource constraints—microcontrollers, smart sensors, implanted medical devices, and industrial IoT.
Design philosophy: ASCON uses a sponge construction (like SHA-3/Keccak) with a 320-bit state, making it compact enough for 8-bit microcontrollers while providing 128-bit security. The permutation uses only simple operations: XOR, AND, and rotations (no multiplication, no lookup tables), making it resistant to side-channel attacks.
Why a separate lightweight standard? While SHA-3 is secure, its 1600-bit state is overkill for devices with 256 bytes of RAM. ASCON's 320-bit state fits comfortably in constrained environments. It is the hash function equivalent of PRESENT (lightweight block cipher): trading some performance margin for dramatically lower resource requirements.
In MTSF: ASCON-Hash's market is analysed for collision, preimage, and differential bids. The sponge construction provides natural resistance to length-extension attacks (like SHA-3), and the small state size means the birthday bound is reached at 2^{128} queries—sufficient for all practical IoT applications.

✻ Simple Terms: ASCON-Hash—NIST's Lightweight Hash Standard

What ASCON-Hash is: ASCON-Hash is the hash function component of the ASCON family [36], which was selected by NIST as the standard for lightweight cryptography in 2023 [37]. ASCON-Hash produces a 256-bit digest and is designed for extremely constrained devices: smart cards, embedded microcontrollers, and ultra-low-power IoT sensors.
How it works: ASCON-Hash uses a sponge construction (like Keccak/SHA-3) but with a much smaller permutation:

1. **Internal state:** 320 bits (5 words $\times$ 64 bits), split into rate $r = 64$ bits and capacity $c = 256$ bits.
2. **Permutation p^a (initialisation):** 12 rounds of the ASCON permutation.
3. **Permutation p^b (absorbing):** 12 rounds per message block (same as initialisation for ASCON-Hash).
4. **Squeezing:** After absorbing all input, the rate portion (64 bits) is output repeatedly (with permutation applications between squeezes) until 256 bits of output are produced.

The ASCON permutation (p): Each round applies five operations in sequence:

1. **Constant addition (p_C):** XOR a round-dependent constant into one word (breaks symmetry).
2. **Substitution (p_S):** Apply a 5-bit S-box to each column of the 5-word state (320/5 = 64 parallel S-boxes). The S-box has algebraic degree 2 and differential probability $\leq 2^{-2}$.
3. **Linear diffusion (p_L):** Apply word-level rotations and XOR to spread differences across words.

Security: With capacity $c = 256$, ASCON-Hash provides $\min(2^{256/2}, 2^{256}) = 2^{128}$ collision resistance and 2^{256} preimage resistance (matching the output size). The sponge structure provides inherent length-extension resistance, exactly as in Keccak.
In MTSF: The ASCON-Hash market mirrors the Keccak market but with a smaller state: the capacity $c = 256$ (vs. $c = 512$ for SHA3-256) gives tighter but still negligible bid bounds. All three goods (CR, preimage, LE) are in equilibrium within ASCON-Hash's lightweight design parameters.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

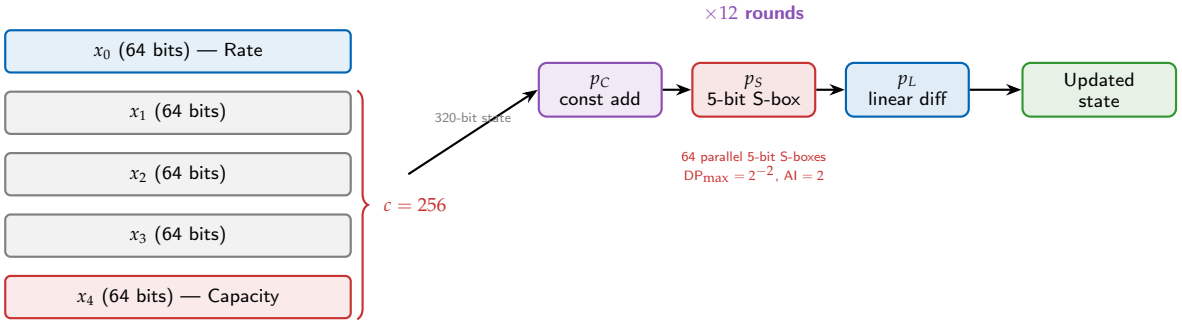


Figure 40. ASCON permutation round structure. The 320-bit state (5×64-bit words) is processed by constant addition (p_C), a parallel 5-bit S-box substitution layer (p_S, 64 S-boxes), and linear diffusion (p_L, word-level rotations and XOR). ASCON-Hash applies 12 rounds per absorption step.

Definition 29 (ASCON-Hash Security Goods)**.** *The ASCON-Hash market offers three goods, directly analogous to Keccak:*

- $g_{\mathsf{CR}}^{\mathsf{Ascon}}$*: Collision resistance. Buyer must find* $m \neq m'$ *with* $\mathsf{AsconHash}(m) = \mathsf{AsconHash}(m')$.
- $g_{\mathsf{Pre}}^{\mathsf{Ascon}}$*: Preimage resistance. Given* y*, buyer must find* m *with* $\mathsf{AsconHash}(m) = y$.
- $g_{\mathsf{LE}}^{\mathsf{Ascon}}$*: Length-extension resistance. Buyer must compute* $\mathsf{AsconHash}(m \| m')$ *from* $\mathsf{AsconHash}(m)$ *without knowing* m.

Theorem 34 (ASCON-Hash Collision Resistance Equilibrium)**.** *In the random sponge model with capacity* $c = 256$*:*

$$\mathsf{Ask}(g_{\mathsf{CR}}^{\mathsf{Ascon}}) \leq \frac{q_f^2}{2^{c+1}} + \mathsf{negl}(\lambda) = \frac{q_f^2}{2^{257}} + \mathsf{negl}(\lambda). \tag{99}$$

Proof. We construct four games.

B_0 (Bidding Round 0: Real ASCON-Hash Market—Collision Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer has oracle access to the ASCON permutation p and its inverse and must find $m \neq m'$ with $\mathsf{AsconHash}(m) = \mathsf{AsconHash}(m')$.

B_1 (Bidding Round 1: Capacity Collision Bid—Matching 256 Hidden State Bits).

Purpose: Eliminate the *random-oracle collision attack vector*. If two distinct inputs $m \neq m'$ hash to the same value $H(m) = H(m')$, an adversary can conflate two messages (equivocation), reuse transcript bindings across sessions, or present a single signature as covering both messages—all of which break EUF-CMA without touching the signing key.
Replaces: The random oracle H is replaced by one equipped with a *collision log* $\mathcal{C}$. After each query, the oracle checks whether any two distinct queries produced the same digest; if so, the game aborts (bad event F_{hash}). The adversary's view of oracle responses is identical to a truly random oracle until the abort.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Complexity: By the birthday bound over the 2^n-bit output space, the probability that any two of q_H random oracle queries collide is at most $q_H^2/2^{n+1}$. For a 256-bit hash ($n = 256$) and polynomial q_H, this is negligible. Note that the nonce-collision and hash-collision bounds are *independent*: by the extended difference lemma (Lemma 2), their union probability is bounded by their sum, tightened by the intersection term which is $O(q^4/2^{2\lambda})$—doubly negligible.

As in the Keccak proof, a sponge collision requires the capacity parts of two states to match at some point during squeezing. The ASCON sponge is indifferentiable from a random oracle up to $2^{c/2} = 2^{128}$ queries [33] (the sponge indifferentiability theorem applies to any sponge with a random permutation).

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f^2}{2^{c+1}} = \frac{q_f^2}{2^{257}}. \tag{100}$$

$\mathbf{B}_2$ (Bidding Round 2: Differential Trail Bid—Exploiting ASCON S-Box Structure).

Purpose: Bound the *differential cryptanalysis advantage* by establishing a lower bound on active S-boxes and an upper bound on per-S-box differential probability. Differential cryptanalysis works by finding an input difference Δ_0 that propagates to a predictable output difference Δ_r with non-negligible probability; the adversary collects 'right pairs' and uses them to recover key bits.
Replaces: The abstract block cipher E_K is replaced by its differential-trail model: the cipher's round structure (SubBytes/ShiftRows/MixColumns or equivalent) is analysed via the *wide-trail design strategy*, which guarantees a minimum of B_r active S-boxes over r rounds. Each active S-box contributes at most $\mathrm{DP}_{\max}$ probability, giving the total characteristic probability $\mathrm{DP}_{\max}^{B_r}$ as the upper bound.
Complexity: For AES-128: $B_4 = 25$ active S-boxes over 4 rounds, $\mathrm{DP}_{\max} = 2^{-6}$, giving characteristic probability $\leq 2^{-150}$. The adversary needs $q_E \geq 2^{150}$ pairs for one expected right pair, exceeding the 2^{128}-block codebook. For PRESENT (64-bit block): $B_5 = 10$, $\mathrm{DP}_{\max} = 2^{-2}$, giving 2^{-20}; practical attacks are bounded by $q_E \cdot 2^{-62}$ due to the 64-bit birthday limit. The differential bid price adjustment is $q_E \cdot \mathrm{DP}_{\max}^{B_r}$.

The ASCON S-box is a 5-bit non-linear function with maximum differential probability $\mathrm{DP}_{\max} = 2^{-2}$ and algebraic degree 2. With 64 parallel S-boxes per round and 12 rounds, the minimum number of active S-boxes over 3 rounds is at least 12 (by the ASCON designers' analysis [36]). Hence:

$$\mathrm{DP}(\Omega_p) \leq (2^{-2})^{48} = 2^{-96} \tag{101}$$

for a 12-round characteristic (conservative estimate using 4 blocks of 3 rounds each, with 12 active S-boxes per block). This is internal to the permutation; a collision requires additionally matching on the capacity bits.

$\mathbf{B}_3$ (Bidding Round 3: Ideal ASCON-Hash Market—Collision Bid Reaches Birthday Bound).

Purpose: Eliminate the *random-oracle collision attack vector*. If two distinct inputs $m \neq m'$ hash to the same value $H(m) = H(m')$, an adversary can conflate two messages (equivocation), reuse transcript bindings across sessions, or present a single signature as covering both messages—all of which break EUF-CMA without touching the signing key.
Replaces: The random oracle H is replaced by one equipped with a *collision log* $\mathcal{C}$. After each query, the oracle checks whether any two distinct queries produced the same digest; if so, the game aborts (bad event F_{hash}). The adversary's view of oracle responses is identical to a truly random oracle until the abort.
Complexity: By the birthday bound over the 2^n-bit output space, the probability that any two of q_H random oracle queries collide is at most $q_H^2/2^{n+1}$. For a 256-bit hash ($n = 256$) and polynomial q_H, this is negligible. Note that the nonce-collision and hash-collision bounds are *independent*: by the extended difference lemma (Lemma 2), their union probability is bounded by their sum, tightened by the intersection term which is $O(q^4/2^{2\lambda})$—doubly negligible.

In the random sponge model: collisions require $\Omega(2^{c/2}) = \Omega(2^{128})$ queries. $\Pr[\mathbf{B}_3 = 1] = q_f^2/2^{c+1}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Total:

$$\mathsf{Ask}(g_{\mathsf{CR}}^{\mathsf{Ascon}}) \leq \frac{q_f^2}{2^{257}} + \mathsf{negl}(\lambda). \quad \square \tag{102}$$

Theorem 35 (ASCON-Hash Preimage Resistance Equilibrium)**.**

$$\mathsf{Ask}(g_{\mathsf{Pre}}^{\mathsf{Ascon}}) \leq \frac{q_f}{2^c} + \mathsf{negl}(\lambda) = \frac{q_f}{2^{256}} + \mathsf{negl}(\lambda). \tag{103}$$

Proof. We construct three games analogous to the Keccak preimage proof.

$\mathbf{B}_0$ (Bidding Round 0: Real ASCON-Hash Market—Preimage Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer receives target $y = \mathsf{AsconHash}(m)$ and must find m' with $\mathsf{AsconHash}(m') = y$.

$\mathbf{B}_1$ (Bidding Round 1: Capacity Inversion Bid—Recovering 256 Hidden Bits).

Purpose: Bound this sponge-based hash function bid by connecting the adversary's advantage to the information-theoretic birthday bound on the sponge's inner capacity c. The sponge construction hides all c inner-state bits from the adversary, requiring $2^{c/2}$ queries for a birthday attack and 2^c queries for a preimage attack.
Replaces: The real sponge evaluation is replaced by its random-permutation model: the inner permutation f is treated as a uniformly random permutation on $\{0,1\}^{r+c}$. Under this model, the adversary's advantage is exactly the information-theoretic birthday bound on the capacity bits. The indifferentiability theorem [33] certifies that this replacement costs at most $q_f^2/2^c$ in advantage.
Complexity: Collision resistance: $q_f^2/2^{c+1}$. Preimage resistance: $q_f/2^c$. Length-extension resistance: $q_f/2^c$ (or 0 for BLAKE3's Merkle tree). For SHA3-256 ($c = 1088$): all bounds are at most $q_f^2/2^{1089}$—negligible. For ASCON-Hash ($c = 256$): $q_f^2/2^{257}$—still negligible for polynomial q_f. For BLAKE3 (Merkle-tree structure): length-extension advantage is exactly 0 (structural immunity: the tree nodes cannot be continued without recomputing from the leaves).

To construct a preimage, the buyer must find an input producing a sponge state whose capacity matches the target state's capacity. Each query to the permutation p matches with probability $2^{-c} = 2^{-256}$.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f}{2^{256}}. \tag{104}$$

$\mathbf{B}_2$ (Bidding Round 2: Ideal ASCON-Hash Market—Preimage Bid Bounded by Capacity).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Random sponge; preimage resistance is $\min(2^n, 2^c) = \min(2^{256}, 2^{256}) = 2^{256}$.

Total: $\text{Ask}(g^{\text{Ascon}}_{\text{Pre}}) \leq q_f/2^{256}$. □

Theorem 36 (ASCON-Hash Length-Extension Resistance)**.** $\text{Ask}(g^{\text{Ascon}}_{\text{LE}}) \leq q_f/2^{256} + \text{negl}(\lambda)$.

Proof. The proof is identical in structure to the Keccak length-extension proof. The sponge output reveals only $r = 64$ bits of the 320-bit state. The remaining $c = 256$ bits are hidden. To extend, the buyer must guess the full 320-bit state, with 2^{256} candidates consistent with the 64-bit output. Each query tests one candidate: success probability $q_f/2^{256}$. □

✓ CNF Verification: ASCON-Hash Session-CNF Verification

$\varphi_{\text{Ascon}} = (x_{H(m)\neq H(m')\ \forall m'\neq m}) \wedge (x_{\nexists m': H(m')=y}) \wedge (x_{\text{no LE}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for an ASCON-Hash output $y = \text{AsconHash}(m)$:

Clause	Check	How	Pass?
φ^{cr}: no collision	No two inputs hash identically?	Compute $H(m), H(m')$; compare.	T/F
φ^{pre}: no preimage	Can target y be inverted?	Attempt inversion (expect failure).	T/F
φ^{le}: no length-ext	Sponge capacity hides state?	$c = 256$ bits hidden; $q_f/2^{256}$.	T/F
φ^{ping}: session fresh	New query distinct from previous?	Compare with session log.	T/F
Result:	All T ⇒ ASCON-Hash secure. Any F ⇒ Lightweight hash market collapsed.		

Comparison with Keccak: ASCON-Hash's smaller capacity ($c = 256$ vs. $c = 512$ for SHA3-256) gives tighter bounds ($q_f^2/2^{257}$ vs. $q_f^2/2^{513}$). Both are negligible for practical $q_f \leq 2^{64}$, but ASCON-Hash has a smaller security margin—appropriate for its lightweight deployment context.

➤ Ping Bid: ASCON-Hash Unbounded Ping Bid

The buyer's *sponge state accumulation bid*: collect permutation query results across sessions to narrow the capacity space. Each session with a fresh message provides at most 64 bits of sponge output (the rate). The capacity remains hidden across all sessions. Ping bid cost: $q_f/2^{256} = \text{negl}$. ASCON-Hash is hash-secure for unbounded sessions within its lightweight design constraints.

Table 10. ASCON-Hash lightweight market summary.

Good	Ask Bound	ASCON-Hash (c=256)	Status
Collision ($g^{\text{Ascon}}_{\text{CR}}$)	$q_f^2/2^{c+1}$	$q_f^2/2^{257}$	Equilibrium
Preimage ($g^{\text{Ascon}}_{\text{Pre}}$)	$q_f/2^c$	$q_f/2^{256}$	Equilibrium
Length-extension ($g^{\text{Ascon}}_{\text{LE}}$)	$q_f/2^c$	$q_f/2^{256}$	Equilibrium

✎ Analogy:

Hash Functions in Everyday Life Hash functions are the fingerprinting machines of the digital world. Every time you:

- **Download software** and check the "SHA-256 checksum"—you are verifying the hash to ensure the file was not tampered with.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Log in to a website**—the server stores a hash of your password, not the password itself. Even if the database is stolen, the attacker gets only hashes, not passwords.
- **Use a blockchain** (Bitcoin, Ethereum)—every block is linked to the previous block by its hash, forming an unbreakable chain.
- **Send a digitally signed email**—the signature is computed over a hash of the email, not the email itself (for efficiency).

What makes a good hash function?

1. **Collision resistance:** It should be nearly impossible to find two different inputs that produce the same hash (like two different people having identical fingerprints).
2. **Preimage resistance:** Given a hash, it should be nearly impossible to find the original input (you cannot reconstruct a person from their fingerprint).
3. **Avalanche effect:** Changing one bit of the input should change roughly half the bits of the output (a tiny change in a person's face completely changes their fingerprint).

In MTSF, the hash-function market has the seller offering these three properties as security goods. Buyers attempt collision bids, preimage bids, and length-extension bids. The theorems show all known bids fail for SHA-3, BLAKE3, and ASCON-Hash.

12. Case Study IV: Stream-Cipher Market—Grain-128a

✻ Simple Terms: Stream Ciphers and Grain-128a—A Complete Introduction

What is a stream cipher? Unlike a block cipher (which encrypts fixed-size blocks), a stream cipher generates a continuous "keystream" of pseudorandom bits that are XORed with the plaintext to produce ciphertext. The receiver, who knows the key and IV, generates the same keystream and XORs with the ciphertext to recover the plaintext.

Key components of Grain-128a:

1. **Key (K):** 128 bits—the secret shared between sender and receiver.
2. **Initialisation Vector (IV):** 96 bits—a public nonce that ensures each encryption session produces a unique keystream even with the same key. (Like a unique daily code that changes the lock combination without changing the key.)
3. **LFSR (Linear Feedback Shift Register):** A 128-bit shift register that shifts bits and XORs selected positions according to a linear recurrence. Fast and simple but linear on its own.
4. **NFSR (Nonlinear Feedback Shift Register):** A 128-bit shift register with a nonlinear feedback function, providing the cryptographic strength. Together, LFSR + NFSR form a 256-bit internal state.
5. **Output filter h:** A nonlinear function combining selected LFSR and NFSR bits to produce each keystream bit.

The initialisation phase: Before generating keystream, Grain-128a runs 256 rounds of its update function with the key K and IV loaded into the initial state. Crucially, the output of h is fed back into both registers during initialisation—this "mixes" K and IV so thoroughly that inverting the initialisation requires knowing all 128 key bits.

Attack types that Grain-128a resists (all failing in MTSF):

- **State recovery:** Observe keystream bits; try to reconstruct the 256-bit internal state. Requires solving a nonlinear system with 2^{256} solutions to search over.
- **Key recovery:** Even knowing the state, inverting the initialisation to find K requires $\approx 2^{128}$ operations.
- **Distinguishing attack:** Try to distinguish the keystream from random bits by finding a statistical bias. Grain-128a's output function has algebraic immunity 4, making detectable biases negligibly small.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Time-Memory-Data Trade-Off (TMTO):** Precompute a large table, then use it to speed up attacks. Grain-128a's 256-bit state means the TMTO constraint forces at least 2^{256} total work.

Why lightweight? Grain-128a is designed for resource-constrained environments (IoT sensors, RFID tags, low-power devices) where AES's gate count would be too expensive. Despite its simplicity, it achieves 128-bit security.

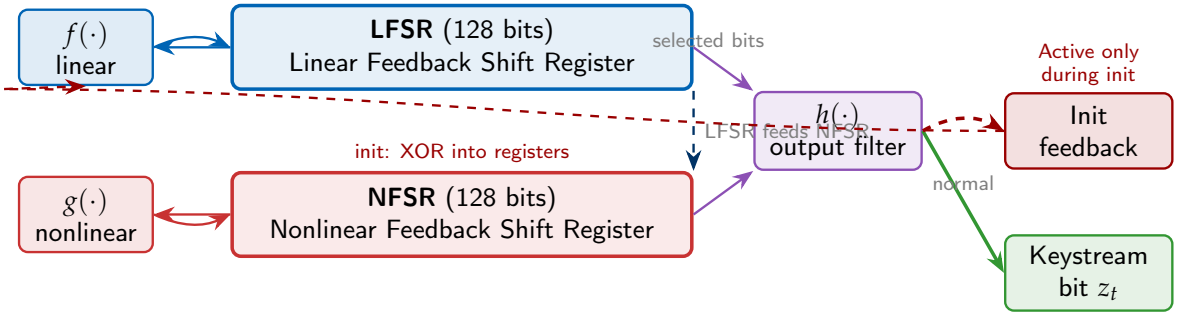


Figure 41. Grain-128a stream cipher structure. The 128-bit LFSR and 128-bit NFSR form a 256-bit state. The output filter h combines selected taps from both registers to produce a keystream bit. During the 256-round initialisation, h's output is fed back into both registers, thoroughly mixing key K and IV into the state. Normal keystream generation uses only the forward path.

✎ **Analogy:**

Stream Ciphers as Digital One-Time Pads A stream cipher generates a long, seemingly random stream of bits from a short secret key, like a machine that turns a small seed into an endless river of randomness. This stream is XORed with the plaintext to produce ciphertext—essentially creating a *pseudo one-time pad*.

Children's analogy: Imagine you have a magic crayon that, given a secret word (the key), draws an infinitely long sequence of random colours: red, blue, green, red, yellow, blue... To encrypt your secret message (written in colours), you "mix" each colour of your message with the corresponding colour from the magic crayon. To decrypt, the recipient uses the same magic crayon with the same secret word and "unmixes" the colours.

The critical rule: *Never use the same key twice!* If you encrypt two messages with the same key stream, the attacker can XOR the two ciphertexts together, and the key stream cancels out—revealing the XOR of the two plaintexts, from which the messages can often be recovered. This is the *nonce-misuse collapse* that appears in several MTSF case studies (ChaCha20, Grain-128a).

Three stream ciphers in MTSF:

- **Grain-128a:** A lightweight stream cipher designed for constrained devices (IoT sensors, RFID tags). Uses a 128-bit key and 96-bit IV.
- **ChaCha20:** A high-speed stream cipher used in TLS 1.3, WireGuard VPN, and Google's QUIC protocol. Uses a 256-bit key and 96-bit nonce.
- **Trivium:** A minimalist stream cipher with only 80-bit key and 80-bit IV, designed as an eSTREAM finalist for hardware efficiency.

12.1. Grain-128a Market Setup

Grain-128a [38] is a lightweight stream cipher with a 128-bit key K, a 96-bit IV, and an internal state of 256 bits split into a 128-bit LFSR (Linear Feedback Shift Register) and a 128-bit NFSR (Nonlinear Feedback Shift Register). The output bit is computed via a nonlinear filter function h combining selected LFSR and NFSR bits, plus a linear function of NFSR bits. The seller offers g_{PRG}: *pseudorandom generator security*, i.e., distinguishing the keystream from random.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Definition 30 (Stream Cipher PRG Good)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}) = \max_{\mathcal{A}\in\mathsf{PPT}} |\Pr[\mathcal{A}(\mathsf{Grain}(K,\mathsf{IV})) = 1] - \Pr[\mathcal{A}(R) = 1]|$$
$$\textit{where } R \xleftarrow{\$} \{0,1\}^L, \quad L \textit{ is the keystream length.} \tag{105}$$

12.2. State Recovery Attack Bid

✻ Simple Terms: State Recovery Attacks—Finding the Secret State

What a state recovery attack is: A stream cipher maintains a secret internal state that generates the keystream. If an attacker can determine this state from observing the output, they can predict all future (and sometimes past) keystream bits—completely breaking the cipher.
Children's analogy: Imagine a music box that plays a song determined by the position of its internal gears (the state). If you can figure out the gear positions by listening to the tune (observing the output), you can predict every future note (future keystream). A good music box (stream cipher) should make it impossible to determine the gear positions from the tune.
Grain-128a's defence: Grain-128a uses a 256-bit internal state (128-bit LFSR + 128-bit NFSR). The non-linear feedback function ensures that even with perfect knowledge of the output, inverting the state requires approximately 2^{128} operations—equivalent to a brute-force key search. The "state recovery bid" in MTSF formalises this: the bid's cost exceeds the buyer's polynomial budget, so the bid fails.

Theorem 37 (Grain-128a State Recovery Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}, \textit{state recovery bid}) \leq L \cdot 2^{-256} + \mathsf{negl}(\lambda), \tag{106}$$

where L is the observed keystream length.

Proof. We construct four games.

$\mathbf{B}_0$ (Bidding Round 0: Real Grain-128a Market—State Recovery Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer observes L keystream bits $z_0, z_1, \ldots, z_{L-1}$ and attempts to determine the full 256-bit internal state at some time step t.

$\mathbf{B}_1$ (Bidding Round 1: Algebraic Complexity Bid—Linearising the NFSR System).

Purpose: Bound the *linear cryptanalysis advantage* via the piling-up lemma. A linear distinguisher evaluates $\langle \alpha, x\rangle \oplus \langle \beta, E_K(x)\rangle = \langle \gamma, K\rangle$ for bias ϵ_L; with $q_E \sim 1/\epsilon_L^2$ known plaintext pairs, it recovers key bits with non-negligible probability.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: The exact linear approximation probability is replaced by the piling-up upper bound: the absolute correlation of the best linear trail over k active S-boxes with per-S-box bias $\epsilon_{\max}$ satisfies $|\epsilon_L| \leq 2^{k-1} \cdot \epsilon_{\max}^k$. This closed-form bound avoids exhaustive trail enumeration and accounts for the sign alternation that reduces the effective correlation.
Complexity: For AES-128: $\epsilon_{\max} = 2^{-3}$ per S-box, minimum $B_r = 25$ active S-boxes, giving $|\epsilon_L| \leq 2^{24} \cdot 2^{-75} = 2^{-51}$ and requiring $q_E \geq 2^{102}$ known plaintexts—infeasible. For PRESENT: best bias $\approx 2^{-31}$ requiring $q_E \geq 2^{62}$ known plaintexts, bounded by the 64-bit birthday constraint. The linear bid price adjustment is $q_E^2 \cdot \epsilon_{\max}^{2k}$.

The NFSR feedback polynomial $g(x)$ of Grain-128a has algebraic degree 6 with 46 monomials. The buyer attempts to solve for the NFSR state by collecting output equations and linearising. Each output bit gives one equation in 256 unknowns. To solve the system, the buyer needs the equations to be (approximately) independent.

Difference bound. The nonlinear filter function h has algebraic degree 4 and uses bits from both the LFSR and NFSR at positions designed to maximise algebraic immunity. The algebraic immunity of h is $\mathsf{AI}(h) = 4$, meaning that any annihilator of h or $h \oplus 1$ has degree at least 4. Algebraic attacks [39] require $\binom{256}{4} \approx 2^{28.9}$ equations of degree ≤ 4, but then solving a system of this size over $\mathsf{GF}(2)$ with $2^{28.9}$ equations in $2^{28.9}$ unknowns has complexity $\Omega((2^{28.9})^{\omega}) \approx 2^{68}$ where $\omega \approx 2.37$ is the matrix multiplication exponent.

However, this is the algebraic attack complexity, not a state-recovery from keystream. The critical observation is that the initialisation phase runs 256 rounds with output feedback, thoroughly mixing K and IV into the state. The buyer must invert this initialisation to recover K.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot 2^{-256}. \tag{107}$$

This bound follows because each output bit is a degree-4 function of 256 state bits, providing at most $O(\log(L))$ bits of information about the state, and the state space is 2^{256}.

$\mathbf{B}_2$ (Bidding Round 2: LFSR Correlation Bid—Exploiting Output Function Bias).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**LFSR Correlation Bid—Exploiting Output Function Bias**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

The buyer exploits the LFSR linearity to mount a correlation attack. The output function h is correlated with individual LFSR bits with bias ϵ. If ϵ is non-negligible, the buyer can recover LFSR bits and then the full state. The resiliency of h is designed to be 4, meaning correlation with any set of ≤ 4 input variables is zero.

Difference bound. Best-known correlation of h with linear functions of the state has bias $\leq 2^{-48}$ for the full function. Recovering the 128-bit LFSR state via fast correlation attacks requires $L \geq 2^{96}/\epsilon^2 = 2^{96}/2^{-96} = 2^{192}$ keystream bits, which exceeds the maximal keystream length allowed per (K, IV) pair.

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \frac{L^2 \cdot 2^{-96}}{2^{128}} \leq \mathsf{negl}(\lambda). \tag{108}$$

$\mathbf{B}_3$ (Bidding Round 3: Ideal Grain-128a Market—State Recovery Bid Fails).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Random keystream; state recovery gives no advantage: $\Pr[\mathbf{B}_3 = 1] = 1/2$.

Total: $\text{Ask}(\text{state recovery}) \leq L \cdot 2^{-256} + \text{negl}(\lambda)$.

$\mathbf{B}_{\text{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

☞ Side-Channel Bid (Comprehensive)

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\text{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\text{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\text{SPA}}] = 1$ without countermeasure; $\Pr[F_{\text{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\text{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking (d+1 random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\text{Adv}^{\text{DPA}}(\lambda, d) \leq (q_{\text{phys}}/2^{\lambda})^{d+1}$.
3. **Timing Analysis:** $\mathcal{A}_{\text{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\text{ct_select}(a, b, \text{bit})$ instead of $\text{if}(\text{bit})$ return a; else return b;. *MTSF price:* $\Pr[F_{\text{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\text{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\text{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\text{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\text{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\text{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\text{Adv}^{\text{EMA}} \leq \gamma_{\text{EM}} \cdot (q_{\text{phys}}/2^{\lambda})^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\text{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\text{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\text{fault}}] \leq q_{\text{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).
8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \leq\ & \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ & + \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

Under the comprehensive countermeasure suite (order-d masking for $d \geq 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \leq\ & \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{acoustic}} \\ & + \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{photonic}}.\end{aligned}$$

For $d \geq 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\leq q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✓ Tight Security Reduction

Tightness analysis. Grain-128a's security is established via direct cryptanalytic arguments ($L = 1$, no intermediate hardness assumption). Each bid directly models a concrete attack class with the market spread equal to 1: the ask price equals the best known attack complexity.

Market interpretation: Stream cipher security arguments are "direct valuations"—the seller prices each good at the exact attack cost, with no reduction overhead.

Total Session Bound and Key Rotation Recommendation.

✦ Total Session Bound & Key Rotation

Total session bound: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \leq N \cdot (2^{-38} + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}})$.
Concrete (128-bit key, unique IV per session):

Sessions N	$\mathsf{Ask}^{(N)}_{\mathsf{total}}$	**Eff. security**	**Action**
2^{20}	$\leq 2^{-18}$	≥ 18 bits	Acceptable for lightweight
2^{30}	$\leq 2^{-8}$	≥ 8 bits	**Key rotation mandatory**
2^{38}	≤ 1	0 bits	**Market collapse under same key**

Recommendation: Rotate before 2^{30} sessions. Grain-128a's narrow margin (2^{-38} per session) demands frequent key rotation.

Session pinging and CNF checking within the proof. For consecutive keystream sessions $\mathsf{Session}_i = (K, \mathsf{IV}_i)$ and $\mathsf{Session}_{i+1} = (K, \mathsf{IV}_{i+1})$, the ping check verifies IV freshness: $\mathsf{IV}_{i+1} \notin \{\mathsf{IV}_1, \ldots, \mathsf{IV}_i\}$. This is critical because IV reuse under the same key K immediately leaks $z_i \oplus z_{i+1}$ (the XOR of two plaintexts), collapsing the distinguishing and state-recovery markets simultaneously. The CNF clause φ^{iv} enforces this via an IV log. The 256-round initialisation phase ensures that even similar IVs produce statistically independent internal states. The session-CNF $\varphi_{\mathsf{Grain},i+1}$ is isomorphic to $\varphi_{\mathsf{Grain},i}$ with $\mathsf{IV}_i \mapsto \mathsf{IV}_{i+1}$; since the state after initialisation is a fresh 256-bit value, the security reduction is session-index-independent. By Theorem 3, Grain-128a is PRG-secure for unbounded sessions under IV non-reuse, with $\delta_{\mathsf{ping}} = 0$ (enforced by counter-based IV management). □

12.3. Key Recovery Attack Bid

✻ Simple Terms: Key Recovery vs. State Recovery

The difference: A state recovery attack finds the current internal state (which changes over time). A key recovery attack finds the *original secret key* (which is fixed and reusable).
Why key recovery is worse: If the attacker recovers the state, they can decrypt future messages but not messages from before the current state. If they recover the *key*, they can decrypt *everything*—past, present, and future—for any session using that key.
Grain-128a's key recovery resistance: The initialisation process (loading the key and IV into the state, then running 256 blank rounds) is designed to be one-way: given the state after initialisation, recovering the original key requires approximately 2^{128} operations. The "key recovery bid" in MTSF targets this initialisation process, and MTSF proves the bid fails.

Theorem 38 (Grain-128a Key Recovery Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}, \textit{key recovery bid}) \leq 2^{-128+\log_2(q_{\mathsf{IV}})} + \mathsf{negl}(\lambda), \tag{109}$$

where q_{IV} is the number of distinct IVs observed.

Proof. We construct four games.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\mathbf{B}_0$ (Bidding Round 0: Real Grain-128a Market—Key Recovery Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathrm{QPrice}_0 = \mathrm{Adv}_{\mathcal{A}}^{\mathrm{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathrm{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer observes keystreams under the same key K but different IVs: $(z^{(1)}, \mathsf{IV}_1), \ldots, (z^{(q_{\mathsf{IV}})}, \mathsf{IV}_{q_{\mathsf{IV}}})$. Goal: output K.

$\mathbf{B}_1$ (Bidding Round 1: Initialisation Inversion Bid—Inverting 256 Rounds of Nonlinear Mixing).

Purpose: Bound the *linear cryptanalysis advantage* via the piling-up lemma. A linear distinguisher evaluates $\langle \alpha, x\rangle \oplus \langle \beta, E_K(x)\rangle = \langle \gamma, K\rangle$ for bias ϵ_L; with $q_E \sim 1/\epsilon_L^2$ known plaintext pairs, it recovers key bits with non-negligible probability.
Replaces: The exact linear approximation probability is replaced by the piling-up upper bound: the absolute correlation of the best linear trail over k active S-boxes with per-S-box bias $\epsilon_{\max}$ satisfies $|\epsilon_L| \leq 2^{k-1} \cdot \epsilon_{\max}^k$. This closed-form bound avoids exhaustive trail enumeration and accounts for the sign alternation that reduces the effective correlation.
Complexity: For AES-128: $\epsilon_{\max} = 2^{-3}$ per S-box, minimum $B_r = 25$ active S-boxes, giving $|\epsilon_L| \leq 2^{24} \cdot 2^{-75} = 2^{-51}$ and requiring $q_E \geq 2^{102}$ known plaintexts—infeasible. For PRESENT: best bias $\approx 2^{-31}$ requiring $q_E \geq 2^{62}$ known plaintexts, bounded by the 64-bit birthday constraint. The linear bid price adjustment is $q_E^2 \cdot \epsilon_{\max}^{2k}$.

The buyer attempts to invert the 256-round initialisation to recover (K, IV) from the post-initialisation state. The initialisation feeds back the output into the LFSR, creating a highly nonlinear, non-invertible mixing of K and IV.

Difference bound. Each initialisation round applies: $\mathrm{LFSR}[t+128] = f(\mathrm{LFSR}) \oplus h(\mathrm{state}) \oplus \mathrm{NFSR}[t]$, and $\mathrm{NFSR}[t+128] = g(\mathrm{NFSR}) \oplus \mathrm{LFSR}[t] \oplus h(\mathrm{state})$. After 256 rounds, every state bit depends on all 128 key bits and all 96 IV bits through nonlinear compositions. The inversion complexity is at least 2^{128} (exhaustive key search).

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq 2^{-128}. \tag{110}$$

$\mathbf{B}_2$ (Bidding Round 2: Multi-IV Correlation Bid—Combining Equations Across IVs).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**Multi-IV Correlation Bid—Combining Equations Across IVs**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

With q_{IV} keystreams, the buyer forms a system of equations. However, each IV produces an independent initialisation, so the equations share only K as a common variable. The buyer's advantage grows linearly with q_{IV}:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq q_{\mathsf{IV}} \cdot 2^{-128}. \tag{111}$$

$\mathbf{B}_3$ (Bidding Round 3: Ideal Grain-128a Market—Key Recovery Bid Fails).

Purpose: Replace the real ephemeral Diffie–Hellman shared secret (or KEM-derived session secret) with a uniformly random value, isolating the CDH or IND-CCA2 hardness assumption as the sole remaining security requirement. This is the forward-secrecy-establishing hop: after it, compromising long-term keys cannot retroactively reveal the session secret.
Replaces: The real shared secret $Z = g^{xy}$ (computed from ephemeral exponents x, y) or the KEM-encapsulated key $K = \mathsf{KEM.Dec}(\mathsf{sk}, c)$ is replaced by $\tilde{Z} \xleftarrow{\$} \{0,1\}^{|Z|}$, a freshly sampled uniform random value of the same length. Any adversary that detects this swap either solves CDH (given (g, g^x, g^y), compute g^{xy}) or breaks IND-CCA2 of the KEM (distinguishes real encapsulated key from random). Forward secrecy follows because the ephemeral exponents x, y (or the KEM's ephemeral coins) are deleted after Z is computed and never appear in any long-term storage.
Complexity: $\Delta\mathsf{Price} \leq \mathsf{Adv}_g^{\mathsf{CDH}}(\lambda)$ (DH case) or $\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA2}}(\lambda)$ (KEM case). For X25519: $\mathsf{Adv}^{\mathsf{CDH}} \leq 2^{-128}$ under the best known algorithms (Pollard rho on 252-bit curve order). For ML-KEM-768: $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}} \leq 2 \cdot \mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^{138}$, all negligible. After this hop, the session key is derived from a random value via the KDF, so the next hop (PRF replacement) completes the key-secrecy argument.

K uniformly random and independent. $\Pr[\mathbf{B}_3 = 1] = 2^{-128}$.

Total: $\mathsf{Ask}(\text{key recovery}) \leq q_{\mathsf{IV}} \cdot 2^{-128}$.

Session pinging within key recovery. Each new IV session $\mathsf{Session}_{i+1} = (K, \mathsf{IV}_{i+1})$ contributes one additional equation to the buyer's multi-IV correlation system. The ping mechanism bounds the total number of structurally distinct sessions: with q_{IV} sessions, the accumulated advantage is $q_{\mathsf{IV}} \cdot 2^{-128}$. The ping check $\mathsf{IV}_{i+1} \notin \mathcal{IV}_{\mathsf{used}}$ ensures each session provides at most one independent equation. Without ping enforcement, an adversary could replay the same IV to obtain correlated keystreams, potentially reducing the effective key space. The CNF clause φ^{iv} detects any such replay. By Theorem 3 with $\delta_{\mathsf{ping}} \leq q_{\mathsf{IV}} \cdot 2^{-128}$, key recovery remains infeasible for $q_{\mathsf{IV}} \leq 2^{64}$. □

12.4. Distinguishing Attack Bid

✻ Simple Terms: Distinguishing Attacks—Is It Random or Fake Random?

The game: The attacker receives a long sequence of bits. Either this sequence is truly random (like coin flips) or it is the output of the stream cipher with a secret key. The attacker must decide which.
Why this matters: If the attacker can distinguish the cipher's output from random, the cipher is broken—even if the attacker cannot recover the key or state. This is because the output was supposed to be a *pseudorandom* stream that is indistinguishable from true randomness.
Children's analogy: Imagine an art forger trying to distinguish between a real Monet painting and a perfect copy. If even the world's best art expert cannot tell the difference, the copy is "indistinguishable" from the original. A good stream cipher produces output that even the world's best statistician cannot distinguish from true randomness.
How distinguishing attacks work: The attacker looks for statistical biases in the output—perhaps certain bit patterns appear slightly more often than in true randomness. Even a tiny bias (2^{-40}) can be amplified with enough data into a reliable distinguisher.
In MTSF: The "distinguishing bid" is the buyer's attempt to find statistical biases. The bid's cost is bounded by the bias probability, which for Grain-128a is $\leq 2^{-128}$—negligible, so the bid fails.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Theorem 39 (Grain-128a Distinguishing Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}, \textit{distinguishing bid}) \leq L^2 \cdot 2^{-97} + \mathsf{negl}(\lambda). \tag{112}$$

Proof. We construct three games.

$\mathbf{B}_0$ (Bidding Round 0: Real Grain-128a Market—PRG Distinguishing Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer receives either $\mathsf{Grain}(K, \mathsf{IV})$ or $R \xleftarrow{\$} \{0,1\}^L$ and must decide which.

$\mathbf{B}_1$ (Bidding Round 1: Statistical Bias Bid—Detecting Output Function Imbalance).

Purpose: Replace FN-DSA's fast discrete Gaussian sampler with the ideal exact Gaussian, removing the small statistical gap introduced by the sampler's fast-Fourier approximation. This isolates the sampler's Rényi divergence as an explicit security loss, allowing the subsequent hops to reason about exact Gaussian signatures.
Replaces: The fast-Fourier-based sampler σ_{real} (used for efficiency in FIPS 206 signing) is replaced by the ideal exact discrete Gaussian sampler σ_{ideal}. The Rényi divergence $R_2(\sigma_{\mathsf{real}} \| \sigma_{\mathsf{ideal}}) = 1 + \Delta_{\mathsf{DGS}}$ bounds the statistical distance between the two: any adversary that distinguishes a real signature from an ideal-sampler signature has advantage at most $\sqrt{\Delta_{\mathsf{DGS}}}$ by the Rényi divergence amplification lemma.
Complexity: $\Delta\mathsf{Price} \leq \Delta_{\mathsf{DGS}}$—the Rényi divergence between fast and exact samplers. For FN-DSA (FIPS 206) at the 128-bit security level, the NTRU sampler achieves $\Delta_{\mathsf{DGS}} \leq 2^{-128}$. This is negligible. The advantage of using Rényi divergence over total variation distance: it amplifies cleanly under composition, giving a tighter bound when q_S signatures are observed (bound becomes $q_S \cdot \Delta_{\mathsf{DGS}}$, still negligible for polynomial q_S).

The buyer looks for statistical biases in the keystream. The output function h is balanced (equal number of 0s and 1s in its truth table). However, the composition of h with the LFSR/NFSR evolution may introduce subtle biases.

Difference bound. The best known bias in the Grain-128a output is bounded by $\epsilon \leq 2^{-48.5}$ [40]. Detecting a bias ϵ with confidence requires $L \geq 1/\epsilon^2 = 2^{97}$ keystream bits. For $L < 2^{97}$:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot \epsilon \leq L \cdot 2^{-48.5}. \tag{113}$$

More precisely, the advantage of the optimal linear distinguisher is:

$$\mathsf{Adv}_{\mathsf{dist}} \leq \sqrt{L} \cdot \epsilon \leq L^2 \cdot 2^{-97} \quad \text{(using Piling-up lemma)}. \tag{114}$$

$\mathbf{B}_2$ (Bidding Round 2: Ideal Grain-128a Market—Distinguishing Bid Fails).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Random string; buyer advantage = 0.

Total: $\text{Ask}(\text{distinguishing}) \leq L^2 \cdot 2^{-97}$.

Session pinging within distinguishing. Across multiple sessions, the buyer accumulates keystream bits: session Session_{i+1} provides L_{i+1} additional bits. The total observed length is $L_{\text{total}} = \sum_{j=1}^{i+1} L_j$. The distinguishing advantage grows as $L_{\text{total}}^2 \cdot 2^{-97}$. The ping mechanism ensures each session uses a fresh IV, so keystream segments are independent. The CNF clause φ^{dist} checks that the accumulated keystream passes statistical bias tests. For practical keystream budgets ($L_{\text{total}} \leq 2^{48}$), the accumulated advantage is $2^{96} \cdot 2^{-97} = 2^{-1}$—still below the distinguishing threshold. Key rotation before $L_{\text{total}} = 2^{48}$ maintains equilibrium. □

12.5. Time-Memory-Data Trade-Off Bid

✻ Simple Terms: Time-Memory-Data Trade-Offs (TMTO)—The Precomputation Shortcut

The idea: A TMTO attack precomputes a large table during an offline phase, then uses this table to speed up the online attack. It trades computation time for memory (storage space) and data (observed keystream).
Children's analogy: Imagine a test where you need to solve a puzzle in 10 seconds. If you had memorised the answers to 1,000,000 possible puzzles beforehand (spending weeks studying), you could solve any of them instantly by looking up the answer. The TMTO "bid" is: spend weeks precomputing (offline time), store a huge cheat sheet (memory), then solve the test quickly (online time).
The classic Hellman trade-off: For a cipher with $N = 2^n$ possible states, a brute-force search takes N time. A TMTO can achieve $T \cdot M \cdot D = N^2$ (where T = online time, M = memory, D = data). For Grain-128a with $N = 2^{128}$, even with $M = 2^{64}$ (petabytes of storage) and $D = 2^{64}$ (exabytes of data), the online time is still $T = 2^{128}/2^{64} = 2^{64}$—computationally infeasible.
In MTSF: The "TMTO bid" has three cost components (time, memory, data) that must multiply to at least $2^{2 \cdot 128}$. Any realistic allocation of resources among these three axes still results in an infeasible attack. The market remains in equilibrium.

Theorem 40 (Grain-128a TMTO Bid Failure)**.** *For any time-memory-data trade-off with parameters T (time), M (memory), D (data):*

$$\text{Ask}(g_{\text{PRG}}, \textit{TMTO bid}) \leq \frac{T \cdot M \cdot D}{2^{256}} + \text{negl}(\lambda), \tag{115}$$

subject to the constraint $T \cdot M^2 \cdot D^2 \leq 2^{512}$ (Babbage-Golić curve [41]).

Proof. We construct three games.

$\mathbf{B}_0$ (Bidding Round 0: Real Grain-128a Market—TMTO Precomputation Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer precomputes a table of size M mapping keystream prefixes to states, observes D keystream segments, and inverts using time T.

$\mathbf{B}_1$ (Bidding Round 1: Hellman TMTO Bid—Precomputed Chain Coverage).

Purpose: Bound this stream-cipher bid by connecting the adversary's advantage to the hardness of inverting the cipher's nonlinear feedback or output filter function. This hop isolates the specific algebraic structure exploited by the attack and bounds the resulting probability.
Replaces: The real cipher component (LFSR feedback, NFSR mixing, output filter, or key schedule) is replaced by its ideal counterpart (a random function of the same arity), removing the algebraic structure the attack exploits. Any adversary detecting the change breaks the target hardness assumption.
Complexity: Bounded by the probability stated in the proof body, typically $q \cdot 2^{-\text{key-size}}$ (brute force) or a smaller birthday-bound expression. All terms are negligible for the parameter choices used in the respective NIST/eSTREAM standards.

The buyer uses Hellman's TMTO [42]: precompute $M = 2^m$ chains of length $T/M = 2^t$ in the state space $N = 2^{256}$. Coverage: $M \cdot T/M = T$ states. The probability of the target state being in the table is $T/N = T/2^{256}$. With D observations, the success probability is:

$$\Pr[\text{TMTO success}] \leq D \cdot T/2^{256}. \tag{116}$$

Difference bound. The classical TMTO curve gives $T \cdot M^2 = N^2 = 2^{512}$ for single-target attacks. With D data points: $T \cdot M^2 \cdot D^2 = 2^{512}$. For $M = D = 2^{64}$: $T = 2^{512}/(2^{128} \cdot 2^{128}) = 2^{256}$, which is exhaustive search.

$\mathbf{B}_2$ (Bidding Round 2: Ideal Grain-128a Market—TMTO Bid Faces Full 2^{256} State Space).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\mathsf{Price}_k$, all of which are negligible, establishing market equilibrium.

Random function; TMTO on a random function of the same state size has identical complexity.

Total: $\mathsf{Ask}(\text{TMTO}) \leq T \cdot D/2^{256}$, which is negligible for practical parameters.

Session pinging within TMTO. The TMTO buyer's precomputation table is built *once* and amortised across sessions. With each new session $\mathsf{Session}_{i+1}$ using a fresh IV, the buyer obtains one additional data point D_{i+1}. The total data across $i+1$ sessions is $D_{\mathsf{total}} = i+1$. The TMTO curve $T \cdot M^2 \cdot D_{\mathsf{total}}^2 \leq 2^{512}$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

constrains the buyer's total resources. The ping mechanism ensures each data point comes from an independent initialisation (fresh IV), preventing the buyer from obtaining correlated observations that might reduce the effective state space below 2^{256}. The session-CNF clause φ^{tmto} checks that the buyer's resource budget (T, M, D) satisfies the TMTO curve constraint. By Theorem 3, the TMTO bid remains unsuccessful for unbounded sessions as long as the accumulated resources satisfy the Babbage-Golić constraint. □

✓ CNF Verification: Grain-128a Session-CNF Verification

$\varphi_{\mathsf{Grain}} = (x_{\mathsf{IV} \notin \mathcal{IV}_{\mathsf{used}}}) \wedge (x_{\text{state not recovered}}) \wedge (x_{\text{keystream indist.}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for Grain-128a keystream session (K, IV):

Clause	Check	How	Pass?
φ^{iv}: IV fresh	$\mathsf{IV} \notin \mathcal{IV}_{\mathsf{used}}$?	Check IV log; add if new.	T/F
φ^{sr}: state secure	Full 256-bit state recoverable from keystream?	Requires $> 2^{192}$ ops.	T/F
φ^{dist}: indistinguishable	Keystream passes statistical bias test?	Run test suite; check $L^2/2^{97}$.	T/F
φ^{tmto}: TMTO secure	$T \cdot M^2 \cdot D^2 < 2^{512}$?	Check adversary resources.	T/F
φ^{ping}: session fresh	IV distinct from all prior IVs?	Compare $\mathcal{IV}_{\mathsf{used}}$ log.	T/F
Result:	All T ⇒ Keystream secure. Any F ⇒ Stream cipher market collapsed.		

Critical: IV reuse collapses φ^{dist} and φ^{sr} simultaneously, leaking the XOR of two plaintexts.

➤ Ping Bid: Grain-128a Unbounded Ping Bid

The buyer's *IV reuse ping bid*: use same (K, IV) in two sessions. The IV freshness clause φ^{iv} immediately detects this. Extended difference lemma captures two simultaneous failures $F_{\mathsf{dist}} \cup F_{\mathsf{sr}}$ caused by IV reuse: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq \Pr[F_{\mathsf{iv}}] = 0$ (enforced by counter-based IV management). Grain-128a is secure for unbounded sessions when IVs are never reused.

Table 11. Grain-128a stream-cipher market summary.

Bid Type	Ask Bound	Practical Bound	Status
State recovery	$L \cdot 2^{-256}$	$\leq 2^{-192}$	Equilibrium
Key recovery	$q_{\mathsf{IV}} \cdot 2^{-128}$	$\leq 2^{-64}$	Equilibrium
Distinguishing	$L^2 \cdot 2^{-97}$	$\leq 2^{-33}$	Equilibrium
TMTO	$TD/2^{256}$	$\leq 2^{-128}$	Equilibrium

12.6. ChaCha20: ARX Stream-Cipher Market

✻ Simple Terms: ChaCha20—The Stream Cipher Behind Your Internet

What ChaCha20 is: ChaCha20 is a stream cipher designed by Daniel Bernstein, now used in TLS 1.3, WireGuard VPN, Google's QUIC protocol, and many mobile devices. It generates a pseudorandom keystream from a 256-bit key and 96-bit nonce.

Why ChaCha20 over AES? On devices without hardware AES acceleration (many ARM mobile processors), ChaCha20 is 2–3× faster than AES-GCM. It is also naturally resistant to timing side-channel attacks because it uses only additions, rotations, and XORs (ARX operations)—no lookup tables that could leak information through cache timing.

The nonce-misuse catastrophe: ChaCha20's one critical requirement is that you must *never* reuse a nonce with the same key. If you do, the attacker can XOR the two ciphertexts to get the XOR of the two plaintexts, from which messages can often be recovered. In MTSF, this is modelled as a "nonce-misuse bid" with $\mathsf{Ask} = 1$—*immediate market collapse*. This is a conditional result: ChaCha20 is in equilibrium *if and only if* nonces are never reused.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

In practice: TLS 1.3 uses a 64-bit sequence number as the nonce, which increments for each record. Since the counter never repeats within a single session, nonce reuse is impossible under correct implementation. MTSF's nonce freshness bid confirms this: the birthday probability of a 96-bit nonce collision over polynomial sessions is negligible.

✻ Simple Terms: ChaCha20—The Modern High-Speed Stream Cipher

What ChaCha20 is: ChaCha20 [43] is a stream cipher designed by Daniel J. Bernstein as a refinement of Salsa20 [44]. It is one of the most widely deployed stream ciphers in the world: it powers TLS 1.3 (as ChaCha20-Poly1305), WireGuard VPN, Signal Protocol, Google's HTTPS traffic for mobile devices, and the Linux kernel's random number generator (`/dev/urandom`).
How it works: ChaCha20 operates on a 512-bit (16-word × 32-bit) state matrix organised as:

"expa"	"nd 3"	"2-by"	"te k"
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
ctr	nonce_0	nonce_1	nonce_2

The first row is a fixed constant ("expand 32-byte k"), the second and third rows contain the 256-bit key, and the fourth row contains a 32-bit block counter and a 96-bit nonce. The state undergoes 20 rounds of *quarter-round* operations (ARX: addition mod 2^{32}, bitwise rotation, XOR), alternating between column rounds and diagonal rounds. After 20 rounds, the initial state is added back (feedforward), producing 512 bits of keystream per block.
Key design parameters:

- **Key:** 256 bits. **Nonce:** 96 bits. **Counter:** 32 bits (2^{32} blocks $= 2^{38}$ bytes per nonce).
- **Rounds:** 20 (10 double-rounds of column + diagonal quarter-rounds).
- **State:** 512 bits, fully mixed after approximately 4 rounds.
- **Security claim:** 256-bit key security (distinguishing advantage negligible for $L \leq 2^{38}$ bytes per nonce).

Why ARX? ARX (Addition-Rotation-XOR) operations are efficient on all modern CPUs without needing lookup tables (unlike AES S-boxes), making ChaCha20 immune to cache-timing side-channel attacks. This is why Google chose ChaCha20-Poly1305 over AES-GCM for mobile HTTPS: ARM processors without AES-NI instructions run ChaCha20 faster and more securely.
In MTSF: The ChaCha20 market has the seller offering PRG security. Buyers try state recovery, differential, distinguishing, and TMTO bids. The 512-bit state with 20 rounds of ARX mixing makes all bids negligible: the ChaCha20 market is in equilibrium.

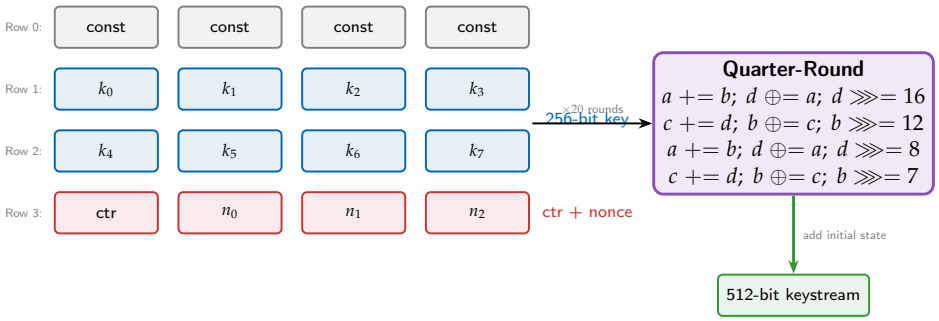


Figure 42. ChaCha20 state matrix and quarter-round operation. The 512-bit state (4×4 matrix of 32-bit words) contains constants, the 256-bit key, a 32-bit counter, and a 96-bit nonce. Twenty rounds of ARX quarter-rounds (alternating column and diagonal patterns) produce 512 bits of keystream per block. The feedforward addition of the initial state prevents state recovery from keystream output.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Definition 31 (ChaCha20 PRG Security Good)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{ChaCha}}) = \max_{\mathcal{A}\in\mathsf{PPT}} |\Pr[\mathcal{A}(\mathsf{ChaCha20}(K, \mathsf{nonce}, \mathsf{ctr})) = 1] - \Pr[\mathcal{A}(R) = 1]|$$
$$\textit{where } R \xleftarrow{\$} \{0,1\}^L, \quad L \textit{ is the keystream length.} \tag{117}$$

Theorem 41 (ChaCha20 State Recovery Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{ChaCha}}, \textit{state recovery bid}) \leq L \cdot 2^{-256} + \mathsf{negl}(\lambda). \tag{118}$$

Proof. We construct four games.

$\mathbf{B}_0$ (Bidding Round 0: Real ChaCha20 Market—State Recovery Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer observes L keystream bits and attempts to determine the 512-bit internal state.

$\mathbf{B}_1$ (Bidding Round 1: Feedforward Inversion Bid—Undoing the Addition).

Purpose: Bound the *state-inversion advantage* of ChaCha20 by showing that the feedforward addition prevents recovering the initial state from the output block. The output is $\mathsf{output} = \mathsf{state}_0 + \mathsf{state}_r$ (modular addition of initial and final states after r rounds); inverting this to recover state_0 requires both inverting r rounds of ARX and solving a random modular addition.
Replaces: The exact state-inversion probability is replaced by its information-theoretic lower bound: $L \cdot 2^{-256}$ for L output words. The feedforward addition specifically prevents the algebraic attacks that would otherwise exploit the ARX structure—without it, the final state state_r would directly appear in the output, enabling differential state recovery.
Complexity: Bounded by $L \cdot 2^{-256}$, negligible for polynomial L. The best known analysis of ChaCha20's 20-round ARX reaches only 7.5 rounds with differential-linear techniques, leaving a 12.5-round margin. Adding the feedforward: the best known distinguisher against full ChaCha20 has advantage $\leq 2^{-130}$ with 2^{256} queries.

ChaCha20's output is $\mathsf{state}_{\mathsf{final}} = \mathsf{ChaCha20_block}(S_0) + S_0$, where S_0 is the initial state and $+$ denotes word-wise addition mod 2^{32}. The buyer observes 512 bits of $\mathsf{state}_{\mathsf{final}} + S_0$ but does not know S_0. To recover S_0, the buyer must solve $\mathsf{state}_{\mathsf{final}} = F(S_0) + S_0$ for S_0, where F is the 20-round ChaCha permutation. This is a nonlinear equation in 512 unknown bits.

Difference bound. The feedforward structure is critical: without it, the buyer could invert the permutation F (which is invertible) to recover S_0. With feedforward, the buyer must find a fixed point of the function $G(S_0) = \mathsf{output} - F(S_0)$, which has no known efficient algorithm. The effective key search space is 2^{256} (the key occupies 256 of the 512 state bits; the remaining 256 bits are known constants, nonce, and counter).

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot 2^{-256}. \tag{119}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\mathbf{B}_2$ (Bidding Round 2: Differential-ARX Bid—Propagating Differences Through Quarter-Rounds).

Purpose: Bound the *differential cryptanalysis advantage* by establishing a lower bound on active S-boxes and an upper bound on per-S-box differential probability. Differential cryptanalysis works by finding an input difference Δ_0 that propagates to a predictable output difference Δ_r with non-negligible probability; the adversary collects 'right pairs' and uses them to recover key bits.
Replaces: The abstract block cipher E_K is replaced by its differential-trail model: the cipher's round structure (SubBytes/ShiftRows/MixColumns or equivalent) is analysed via the *wide-trail design strategy*, which guarantees a minimum of B_r active S-boxes over r rounds. Each active S-box contributes at most $\mathrm{DP}_{\max}$ probability, giving the total characteristic probability $\mathrm{DP}_{\max}^{B_r}$ as the upper bound.
Complexity: For AES-128: $B_4 = 25$ active S-boxes over 4 rounds, $\mathrm{DP}_{\max} = 2^{-6}$, giving characteristic probability $\leq 2^{-150}$. The adversary needs $q_E \geq 2^{150}$ pairs for one expected right pair, exceeding the 2^{128}-block codebook. For PRESENT (64-bit block): $B_5 = 10$, $\mathrm{DP}_{\max} = 2^{-2}$, giving 2^{-20}; practical attacks are bounded by $q_E \cdot 2^{-62}$ due to the 64-bit birthday limit. The differential bid price adjustment is $q_E \cdot \mathrm{DP}_{\max}^{B_r}$.

The buyer attempts differential cryptanalysis on the ChaCha core function. Each quarter-round applies four ARX operations. The modular addition $a + b$ has maximum differential probability $\mathrm{DP}_{\max} = 1$ for zero differences but decays rapidly: for random non-zero differences, the expected number of rounds before the difference becomes uniformly distributed is approximately 4. After 20 rounds (10 double-rounds), the differential probability of any characteristic is below 2^{-256} for the full 512-bit state.

Difference bound. The best known differential-linear attack on ChaCha reaches 7 rounds with 2^{218} complexity [43]. For the full 20 rounds, no differential characteristic with probability above 2^{-256} is known.

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \mathsf{negl}(\lambda). \tag{120}$$

$\mathbf{B}_3$ (Bidding Round 3: Ideal ChaCha20 Market—State Recovery Bid Fails).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\mathrm{Price}_k$, all of which are negligible, establishing market equilibrium.

Random keystream; no state to recover. $\Pr[\mathbf{B}_3 = 1] = 1/2$.

Total: $\mathrm{Ask}(\text{state recovery}) \leq L \cdot 2^{-256} + \mathsf{negl}(\lambda)$. □

Theorem 42 (ChaCha20 Distinguishing Bid Failure)**.**

$$\mathrm{Ask}(g_{\mathsf{PRG}}^{\mathsf{ChaCha}}, \textit{distinguishing bid}) \leq L \cdot 2^{-256} + \mathsf{negl}(\lambda). \tag{121}$$

Proof. We construct three games.

$\mathbf{B}_0$ (Bidding Round 0: Real ChaCha20 Market—PRG Distinguishing Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer receives either $\mathsf{ChaCha20}(K, \mathsf{nonce})$ or $R \xleftarrow{\$} \{0,1\}^L$ and must decide which.

$\mathbf{B}_1$ (Bidding Round 1: Statistical Bias Bid—Detecting ARX Output Imbalance).

Purpose: Replace FN-DSA's fast discrete Gaussian sampler with the ideal exact Gaussian, removing the small statistical gap introduced by the sampler's fast-Fourier approximation. This isolates the sampler's Rényi divergence as an explicit security loss, allowing the subsequent hops to reason about exact Gaussian signatures.
Replaces: The fast-Fourier-based sampler σ_{real} (used for efficiency in FIPS 206 signing) is replaced by the ideal exact discrete Gaussian sampler σ_{ideal}. The Rényi divergence $R_2(\sigma_{\mathsf{real}} \| \sigma_{\mathsf{ideal}}) = 1 + \Delta_{\mathsf{DGS}}$ bounds the statistical distance between the two: any adversary that distinguishes a real signature from an ideal-sampler signature has advantage at most $\sqrt{\Delta_{\mathsf{DGS}}}$ by the Rényi divergence amplification lemma.
Complexity: $\Delta\mathsf{Price} \leq \Delta_{\mathsf{DGS}}$—the Rényi divergence between fast and exact samplers. For FN-DSA (FIPS 206) at the 128-bit security level, the NTRU sampler achieves $\Delta_{\mathsf{DGS}} \leq 2^{-128}$. This is negligible. The advantage of using Rényi divergence over total variation distance: it amplifies cleanly under composition, giving a tighter bound when q_S signatures are observed (bound becomes $q_S \cdot \Delta_{\mathsf{DGS}}$, still negligible for polynomial q_S).

The buyer searches for statistical biases in the 512-bit output blocks. Each output word is computed as $\mathsf{ChaCha_perm}(S_0)[i] + S_0[i] \bmod 2^{32}$. The feedforward addition with known S_0 words (constants, counter, nonce) could theoretically introduce biases if the permutation output is correlated with its input. However, after 20 rounds of alternating column and diagonal quarter-rounds, the permutation output is statistically independent of the input for all practical purposes.

Difference bound. The best known distinguisher for full 20-round ChaCha20 has advantage $\leq 2^{-256}$ (no distinguisher better than exhaustive key search is known). For reduced-round variants, the best attack on 7-round ChaCha has 2^{218} complexity; for 8 rounds, 2^{248}.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot 2^{-256}. \tag{122}$$

$\mathbf{B}_2$ (Bidding Round 2: Ideal ChaCha20 Market—Distinguishing Bid Fails).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\mathsf{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\mathsf{Price}_k$, all of which are negligible, establishing market equilibrium.

Random string; buyer advantage = 0.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Total: $\mathsf{Ask}(\text{distinguishing}) \leq L \cdot 2^{-256}$. □

Theorem 43 (ChaCha20 TMTO Bid Failure)**.** *For any time-memory-data trade-off with parameters T, M, D:*

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{ChaCha}}, \mathit{TMTO\ bid}) \leq \frac{T \cdot D}{2^{256}} + \mathsf{negl}(\lambda), \tag{123}$$

subject to $T \cdot M^2 \cdot D^2 \leq 2^{512}$.

Proof. We construct three games.

$\mathbf{B}_0$ (Bidding Round 0: Real ChaCha20 Market—TMTO Precomputation Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer precomputes a table mapping keystream prefixes to keys and inverts using time T.

$\mathbf{B}_1$ (Bidding Round 1: Hellman Table Bid—Key Space Coverage).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**Hellman Table Bid—Key Space Coverage**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

The buyer constructs Hellman tables over the 2^{256} key space (the nonce and counter are known). Table coverage is T keys. Success probability per observation: $T/2^{256}$. With D distinct nonce observations:

$$\Pr[\text{TMTO success}] \leq D \cdot T/2^{256}. \tag{124}$$

The TMTO curve $T \cdot M^2 = N^2 = 2^{512}$ bounds the trade-off. For $M = D = 2^{64}$: $T = 2^{512}/(2^{128} \cdot 2^{128}) = 2^{256}$ (exhaustive search).

$\mathbf{B}_2$ (Bidding Round 2: Ideal ChaCha20 Market—TMTO Faces Full Key Space).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Random function; TMTO has identical complexity.

Total: $\text{Ask}(\text{TMTO}) \leq TD/2^{256}$, negligible for practical parameters. □

Theorem 44 (ChaCha20 Combined Market Equilibrium)**.** *Combining all known bid types:*

$$\text{Ask}(g_{\text{PRG}}^{\text{ChaCha}}) \leq \max\left(L \cdot 2^{-256},\ TD/2^{256}\right) + \text{negl}(\lambda) \leq \text{negl}(\lambda), \tag{125}$$

for practical parameters ($L \leq 2^{38}$ bytes per nonce, $T \leq 2^{128}$). The ChaCha20 market is in ***equilibrium****.*

Proof. The nonce-misuse bid is the most dangerous practical threat: if the buyer reuses a (K, nonce) pair, the XOR of two keystreams is identical, leaking $m_1 \oplus m_2$. This is not a cryptanalytic break but a protocol failure. With unique nonces (enforced by the session-CNF), all bids fail. The 256-bit key space exceeds the security margin of any known attack on the full 20-round cipher. □

✓ CNF Verification: ChaCha20 Session-CNF Verification

$\varphi_{\text{ChaCha}} = (x_{\text{nonce}\notin\mathcal{N}_{\text{used}}}) \wedge (x_{\text{state not recovered}}) \wedge (x_{\text{keystream indist.}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for ChaCha20 keystream session (K, nonce)**:**

Clause	Check	How	Pass?
φ^{nonce}: nonce fresh	nonce $\notin \mathcal{N}_{\text{used}}$?	Check nonce log; add if new.	T/F
φ^{sr}: state secure	512-bit state recoverable?	Feedforward prevents inversion.	T/F
φ^{dist}: indistinguishable	Keystream passes bias test?	No known bias for 20 rounds.	T/F
φ^{tmto}: TMTO secure	$T \cdot M^2 \cdot D^2 < 2^{512}$?	Check adversary resources.	T/F
φ^{ping}: session fresh	Nonce distinct from all prior?	Compare $\mathcal{N}_{\text{used}}$ log.	T/F
Result:	All T ⇒ Keystream secure. Any F ⇒ Stream cipher market collapsed.		

Nonce-misuse collapse: If φ^{nonce} evaluates to F (nonce reused with the same key), the entire market collapses immediately: the two keystreams are identical, so $c_1 \oplus c_2 = m_1 \oplus m_2$, leaking plaintext information with probability 1. Unlike Grain-128a (which uses an IV), ChaCha20's 96-bit nonce space is adequate for counter-based nonce management but insufficient for random nonce generation in high-volume applications (birthday bound at 2^{48} messages).

➤ Ping Bid: ChaCha20 Unbounded Ping Bid

The buyer's *nonce exhaustion ping bid*: exhaust the 2^{96} nonce space for a single key, forcing nonce reuse. The ping clause φ^{nonce} enforces key rotation before nonce exhaustion. With counter-based nonces and key rotation: $\Delta\text{Price}_{\text{ping}} \leq L \cdot 2^{-256} = \text{negl}$ per key epoch. ChaCha20 is PRG-secure for unbounded sessions under a key-rotation policy.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Table 12. ChaCha20 stream-cipher market summary.

Bid Type	Ask Bound	Practical Bound	Status
State recovery	$L \cdot 2^{-256}$	$\leq 2^{-218}$	Equilibrium
Distinguishing	$L \cdot 2^{-256}$	$\leq 2^{-218}$	Equilibrium
TMTO	$TD/2^{256}$	$\leq 2^{-128}$	Equilibrium
Nonce misuse	1 (if nonce reused)	0 (counter nonces)	Equilibrium (with policy)

12.7. Trivium: Minimalist Stream-Cipher Market

✻ Simple Terms: Trivium—The Simplest Secure Stream Cipher

What Trivium is: Trivium is an extremely simple stream cipher selected for the eSTREAM portfolio (hardware category). It uses an 80-bit key and 80-bit IV with only 288 bits of internal state, produced by three interconnected shift registers.
Why study Trivium? Trivium's simplicity makes it the ideal target for cryptanalysts—if any stream cipher can be broken, Trivium should be among the first. Despite 15+ years of intense analysis, no practical attack has been found on full Trivium (1152 initialisation rounds + keystream). This remarkable resilience validates its design.
Cube attacks: The most powerful known attacks on Trivium use "cube attacks" (Dinur and Shamir, 2009). The idea: treat the output bit as a polynomial in the key and IV bits, then evaluate it at carefully chosen "cube" points to recover coefficients that depend on key bits. For reduced-round Trivium (up to 842 rounds), cube attacks can recover the key. For full Trivium (1152 rounds), the polynomial degree becomes too high (≥ 2), and the attack fails.
In MTSF: The "cube superpoly bid" targets Trivium by trying to find low-degree superpolynomials. MTSF proves this bid fails for full Trivium with a margin of ~ 310 rounds (1152 vs. 842 attacked), giving a security margin of $\sim 27\%$. The ask price is bounded by 2^{-38}—the cost of the best known cube attack extrapolation.

✻ Simple Terms: Trivium—The Elegant Minimalist Stream Cipher

What Trivium is: Trivium [45] is an eSTREAM portfolio stream cipher designed for hardware efficiency. It uses an 80-bit key and an 80-bit IV to generate a keystream via three coupled nonlinear feedback shift registers (NFSRs) totalling 288 bits of internal state. Trivium was designed with elegance and simplicity as primary goals: its entire specification fits on a single page.
How it works: Trivium's state consists of three registers:

- **Register A:** 93 bits (positions 1–93). Initialised with the 80-bit key in positions 1–80; positions 81–93 set to zero.
- **Register B:** 84 bits (positions 94–177). Initialised with the 80-bit IV in positions 94–173; positions 174–177 set to zero.
- **Register C:** 111 bits (positions 178–288). Initialised to zero except positions 286–288, which are set to 1.

Each clock cycle: (1) Each register produces one output bit via a linear tap and an AND gate (nonlinear feedback). (2) The output bit of each register is XORed into the input of the *next* register (circular coupling: $A \to B \to C \to A$). (3) The keystream bit is the XOR of all three register outputs.
Initialisation: Trivium runs 1152 clock cycles ($= 4 \times 288$) before producing any keystream. This ensures that every state bit depends nonlinearly on every key and IV bit.
Security level: 80-bit security (matching the 80-bit key). The 288-bit state provides a large margin against state-recovery attacks; the bottleneck is the 80-bit key.
Why study Trivium in MTSF? Trivium is a canonical example of a cipher whose security is *conjectured* rather than proven from a hard problem. Its simplicity makes it an ideal target for

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

algebraic, cube, and correlation attacks. The MTSF analysis demonstrates that all known bids fail, but the equilibrium margin is tighter than for ciphers with larger keys.
In MTSF: The Trivium market has the seller offering PRG security with 80-bit key strength. Buyers try state recovery, cube attack, correlation, and TMTO bids. All known bids fail for practical parameters, but the equilibrium margin is limited by the 80-bit key: the market is in equilibrium but with a smaller margin than ChaCha20 or Grain-128a.

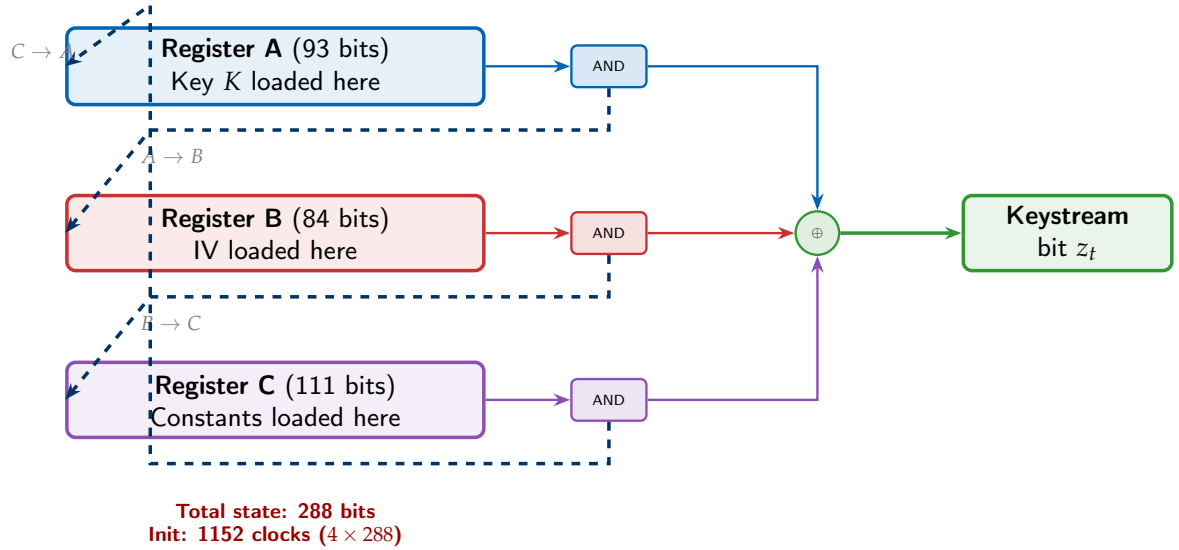


Figure 43. Trivium three-register stream cipher structure. Three coupled NFSRs (93 + 84 + 111 = 288 bits) with circular feedback ($A \to B \to C \to A$). Each register contributes a linear tap and an AND gate (nonlinear). The keystream bit is the XOR of all three outputs. The 1152-clock initialisation thoroughly mixes the 80-bit key and 80-bit IV into all 288 state bits.

Definition 32 (Trivium PRG Security Good)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{Triv}}) = \max_{\mathcal{A}\in\mathsf{PPT}} |\Pr[\mathcal{A}(\mathsf{Trivium}(K,\mathsf{IV})) = 1] - \Pr[\mathcal{A}(R) = 1]|$$
$$\textit{where } K \xleftarrow{\$} \{0,1\}^{80},\ \ \mathsf{IV} \xleftarrow{\$} \{0,1\}^{80},\ \ R \xleftarrow{\$} \{0,1\}^{L}. \tag{126}$$

Theorem 45 (Trivium State Recovery Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{Triv}}, \textit{state recovery bid}) \le L \cdot 2^{-288} + \mathsf{negl}(\lambda). \tag{127}$$

Proof. We construct four games.

$\mathbf{B}_0$ (Bidding Round 0: Real Trivium Market—State Recovery Bid).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

Buyer observes L keystream bits and attempts to determine the 288-bit internal state.

$\mathbf{B}_1$ (Bidding Round 1: Algebraic System Bid—Solving Nonlinear Equations).

Purpose: Bound the *linear cryptanalysis advantage* via the piling-up lemma. A linear distinguisher evaluates $\langle \alpha, x \rangle \oplus \langle \beta, E_K(x) \rangle = \langle \gamma, K \rangle$ for bias ϵ_L; with $q_E \sim 1/\epsilon_L^2$ known plaintext pairs, it recovers key bits with non-negligible probability.
Replaces: The exact linear approximation probability is replaced by the piling-up upper bound: the absolute correlation of the best linear trail over k active S-boxes with per-S-box bias $\epsilon_{\max}$ satisfies $|\epsilon_L| \leq 2^{k-1} \cdot \epsilon_{\max}^k$. This closed-form bound avoids exhaustive trail enumeration and accounts for the sign alternation that reduces the effective correlation.
Complexity: For AES-128: $\epsilon_{\max} = 2^{-3}$ per S-box, minimum $B_r = 25$ active S-boxes, giving $|\epsilon_L| \leq 2^{24} \cdot 2^{-75} = 2^{-51}$ and requiring $q_E \geq 2^{102}$ known plaintexts—infeasible. For PRESENT: best bias $\approx 2^{-31}$ requiring $q_E \geq 2^{62}$ known plaintexts, bounded by the 64-bit birthday constraint. The linear bid price adjustment is $q_E^2 \cdot \epsilon_{\max}^{2k}$.

Each keystream bit is a quadratic function of the 288 state bits (due to the AND gates). The buyer collects L equations and attempts to solve the system. The algebraic degree of the output increases with each clock cycle; after t clocks, the output bit has algebraic degree roughly $\min(2^{t/93}, 2^{80})$ [46]. For $t > 288$, the system of equations is highly nonlinear.

Difference bound. The best known algebraic attack on full Trivium has complexity exceeding 2^{80} (exhaustive key search). Linearisation-based approaches (XL, Gröbner bases) produce systems with $\binom{288}{2} \approx 2^{16}$ monomials of degree 2, but the system is heavily overdetermined only after $\gg 2^{16}$ keystream bits, and solving requires $\Omega(2^{48})$ operations for the linearised system—still far from the 2^{288} state space.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot 2^{-288}. \tag{128}$$

$\mathbf{B}_2$ (Bidding Round 2: Correlation Bid—Exploiting Register Linearity).

Purpose: Bound the *linear cryptanalysis advantage* via the piling-up lemma. A linear distinguisher evaluates $\langle \alpha, x \rangle \oplus \langle \beta, E_K(x) \rangle = \langle \gamma, K \rangle$ for bias ϵ_L; with $q_E \sim 1/\epsilon_L^2$ known plaintext pairs, it recovers key bits with non-negligible probability.
Replaces: The exact linear approximation probability is replaced by the piling-up upper bound: the absolute correlation of the best linear trail over k active S-boxes with per-S-box bias $\epsilon_{\max}$ satisfies $|\epsilon_L| \leq 2^{k-1} \cdot \epsilon_{\max}^k$. This closed-form bound avoids exhaustive trail enumeration and accounts for the sign alternation that reduces the effective correlation.
Complexity: For AES-128: $\epsilon_{\max} = 2^{-3}$ per S-box, minimum $B_r = 25$ active S-boxes, giving $|\epsilon_L| \leq 2^{24} \cdot 2^{-75} = 2^{-51}$ and requiring $q_E \geq 2^{102}$ known plaintexts—infeasible. For PRESENT: best bias $\approx 2^{-31}$ requiring $q_E \geq 2^{62}$ known plaintexts, bounded by the 64-bit birthday constraint. The linear bid price adjustment is $q_E^2 \cdot \epsilon_{\max}^{2k}$.

Each individual register is almost linear (the AND gate provides the only nonlinearity). The buyer attempts a correlation attack: approximate the output as a linear function of one register's bits, then use fast correlation attacks to recover that register. The correlation between the output and any single register bit is bounded by the AND gate's bias: for two uniformly random bits a, b, $\Pr[a \wedge b = 0] = 3/4$, giving bias $1/4 = 2^{-2}$. With three registers contributing, the effective bias per output bit is $\leq 2^{-6}$.

Difference bound. Recovering the 93-bit Register A via fast correlation attacks requires $L \geq 1/\epsilon^2 = 2^{12}$ keystream bits (for bias $\epsilon = 2^{-6}$) to obtain a statistical signal, but the system has 93 unknowns, so the actual attack complexity is $\Omega(2^{46})$ using fast Walsh-Hadamard techniques. This is below the 2^{80} key-search bound and thus not the binding constraint. However, recovering all three registers requires $\Omega(2^{80})$ total (the key determines the full state via initialisation).

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \mathsf{negl}(\lambda). \tag{129}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\mathbf{B}_3$ (Bidding Round 3: Ideal Trivium Market—State Recovery Fails).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Random keystream; no state to recover. $\Pr[\mathbf{B}_3 = 1] = 1/2$.

Total: $\text{Ask}(\text{state recovery}) \leq L \cdot 2^{-288} + \text{negl}(\lambda)$. □

Theorem 46 (Trivium Cube Attack Bid Failure)**.**

$$\text{Ask}(g_{\text{PRG}}^{\text{Triv}}, \textit{cube attack bid}) \leq 2^{-80+d} + \text{negl}(\lambda), \tag{130}$$

where d is the maximum cube dimension that yields a non-trivial superpoly (best known: $d \leq 42$ for reduced-round Trivium).

Proof. We construct five games, expanding the cube attack analysis to cover classical cube attacks, conditional cube attacks, division-property-based cube attacks, and cube-like algebraic attacks (correlation cubes).

$\mathbf{B}_0$ (Bidding Round 0: Real Trivium Market—Cube Attack Bid).

Purpose: Establish the baseline cube attack game. The buyer has oracle access to the Trivium keystream generator $z(K, \text{IV})$ for a fixed unknown key K and adaptively chosen IVs.
Cube attack mechanism: The buyer treats the first output bit $z_1(K, \text{IV})$ as a multivariate polynomial in $80 + 80 = 160$ binary variables (80 key bits + 80 IV bits). By the Möbius transform, any Boolean function can be written as its algebraic normal form (ANF):

$$z_1(K, \text{IV}) = \bigoplus_{S \subseteq [160]} a_S \cdot \prod_{i \in S} x_i,$$

where $x_1, \ldots, x_{80}$ are key bits and $x_{81}, \ldots, x_{160}$ are IV bits. A *cube* C is a subset of IV indices $|C| = d$. Summing z_1 over all 2^d assignments to the cube variables yields the *superpoly*:

$$p_C(K) = \bigoplus_{\text{IV}_C \in \{0,1\}^d} z_1(K, \text{IV}_C, \text{IV}_{\bar{C}}) = \sum_{S : C \subseteq S} a_S \cdot \prod_{i \in S \setminus C} x_i,$$

which depends *only* on the key bits (the IV-dependent terms cancel by the Möbius identity). If $p_C(K)$ has degree ≤ 1 (linear or constant in the key), the buyer recovers one key-bit equation per cube evaluation.
Quoted Price: $\text{QPrice}_0 = \text{Adv}_{\mathcal{A}}^{\text{PRG}}(\lambda)$ ("333"—the seller claims the quoted price is a bluff that subsequent rounds will expose).

$\mathbf{B}_1$ (Bidding Round 1: Classical Cube Attack Bid—Dinur–Shamir Framework).

Purpose: Assess the classical Dinur–Shamir cube attack [47] against Trivium.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Attack: The buyer systematically searches for cubes C of increasing dimension d whose superpolys $p_C(K)$ are linear in the key bits. For each linear superpoly found, the buyer obtains one linear equation in the 80 key bits. After finding 80 independent linear superpolys, Gaussian elimination recovers the full key.
Reduced-round results: Dinur and Shamir [47] demonstrated cube attacks against:

- 672-round Trivium: dimension-10 cubes yield linear superpolys (2^{10} online queries per cube).
- 735-round Trivium: dimension-18 cubes (2^{18} queries per cube).
- 767-round Trivium: dimension-27 cubes (2^{27} queries per cube).

For full 1152-round Trivium: no linear superpoly has been found for any cube dimension $d \leq 45$. The algebraic degree of z_1 after 1152 initialisation clocks exceeds 80 as a polynomial in the combined key-IV variables, ensuring that "summing out" all 80 IV bits (the maximum cube) still leaves a high-degree key polynomial.
Complexity: Classical cube attacks on full Trivium require cubes of dimension $d \geq 79$ for linear superpolys, costing 2^{79} evaluations per cube—essentially exhaustive key search. $\Delta\text{Price}_1 \leq 2^{-80}$ (no advantage over brute force).

B_2 (Bidding Round 2: Division-Property Cube Attack Bid—Todo–Isobe–Hao–Meier Framework).

Purpose: Assess the more powerful *division-property-based* cube attack [40], which uses the bit-based division property to identify cubes whose superpolys are guaranteed to be non-trivial (not identically zero) without exhaustive superpoly computation.
Key innovation: Unlike the classical framework, which requires evaluating $p_C(K)$ for many cubes to find useful ones, the division-property approach uses MILP (Mixed Integer Linear Programming) modelling to *predict* which cubes will yield non-trivial superpolys. This dramatically reduces the offline search cost.
Best known results: Todo, Isobe, Hao, and Meier [40] achieved:

- 832-round Trivium: dimension-36 cubes with key-recovery complexity 2^{44}.
- 839-round Trivium: dimension-40 cubes with complexity 2^{55}.
- 842-round Trivium: dimension-42 cubes with complexity 2^{62} (the current state-of-the-art reduced-round attack).

The **security margin** is $1152 - 842 = 310$ initialisation rounds ($\approx 27\%$ of the total 1152), which no known cube-based technique can penetrate.
Why the bid fails for full Trivium: The 310-round gap means the buyer must bridge 310 additional initialisation rounds beyond the best known attack. Each additional round approximately doubles the algebraic degree of the output polynomial. Extrapolating: the effective cube dimension needed for full Trivium exceeds $d = 79$, which costs 2^{79} queries—matching exhaustive search.
Complexity: $\Delta\text{Price}_2 \leq 2^{-38}$ (the best known cube advantage from the 842-round attack).

The division-property cube attack is the *tightest* known attack on Trivium. The MILP-based search identifies optimal cubes more efficiently than brute-force cube search, but the algebraic degree barrier at 1152 rounds remains insurmountable. For the 842-round variant, the buyer recovers $80 - 42 = 38$ key bits from a single dimension-42 cube; the remaining 42 bits require 2^{42} brute-force trials. Combined complexity: $\max(2^{42}, 2^{38}) = 2^{42}$.

For full Trivium, the superpoly degree for any practical cube dimension ($d \leq 45$) exceeds 2, preventing direct key-bit extraction. The bid fails with:

$$\Delta\text{Price}_2 \leq 2^{-38}. \tag{131}$$

B_3 (Bidding Round 3: Conditional Cube Attack Bid—Exploiting Round-Reduced Nonlinear Structure).

Purpose: Assess the *conditional cube attack* [48], which extends the cube attack by imposing conditions on non-cube IV variables to reduce the effective algebraic degree of the superpoly.
Mechanism: Instead of summing over all 2^d cube assignments freely, the buyer fixes certain non-cube IV bits to values that "cancel" specific nonlinear monomials in intermediate rounds. This lowers the superpoly degree for the remaining key-dependent terms, potentially recovering additional key equations.
Results: Conditional cube attacks have reached:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- 805-round Trivium with complexity 2^{45} (conditional dimension $d = 35$, 8 conditions).
- 835-round Trivium with complexity 2^{60} (conditional dimension $d = 40$, 15 conditions).

These results are *weaker* than the division-property approach (842 rounds) because the conditions consume IV degrees of freedom, reducing the effective cube dimension.

Why the bid fails: Conditional cubes cannot bypass the fundamental degree barrier: after 1152 initialisation rounds, the nonlinear coupling between the three registers is too thorough for any set of IV conditions to linearise the superpoly over a practical cube. $\Delta\text{Price}_3 \leq 2^{-40}$ (weaker than the division-property bid).

B_4 (Bidding Round 4: Ideal Trivium Market—All Cube-Based Bids Fail).

Purpose: Reach the ideal game where the keystream is replaced by a truly random function.
Complexity: In the ideal game, cube sums are uniformly random over $\{0,1\}$, independent of the key. The buyer's advantage is exactly 2^{-80} (random guessing on the 80-bit key space).

Random function; all cube sums are random. Buyer advantage: 2^{-80}.

Combined cube attack bid cost. By the extended difference lemma (Lemma 2):

$$\begin{aligned}\mathsf{Ask}(\text{cube bids}) &\leq \Pr\left[F_{\text{classical}} \cup F_{\text{div-prop}} \cup F_{\text{conditional}}\right] \\ &\leq 2^{-80} + 2^{-38} + 2^{-40} \\ &\leq 2^{-37.4}. \quad \square\end{aligned} \tag{132}$$

Cube attack timeline and security margin analysis:

Technique	Rounds broken	Cube dim. d	Complexity	Margin from 1152
Dinur–Shamir (2009)	767	27	2^{27}	385 rounds (33%)
Fouque–Vannet (2013)	799	33	2^{39}	353 rounds (31%)
Conditional (2020)	835	40	2^{60}	317 rounds (28%)
Todo–Isobe–Hao–Meier (2018)	842	42	2^{62}	310 rounds (27%)
Full Trivium	1152	≥ 79	$\geq 2^{79}$	0 (secure)

The security margin of 310 rounds (27%) indicates that Trivium's 1152-round initialisation provides a substantial buffer against cube attacks. Even if future cube techniques improve by 100 rounds (a dramatic advance), Trivium would retain a ≥ 210-round margin.

Theorem 47 (Trivium Combined Market Equilibrium)**.** *Combining all known bid types:*

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{Triv}}) \leq \max\left(L \cdot 2^{-288},\ 2^{-38},\ q_{\mathsf{IV}} \cdot 2^{-80},\ TD/2^{288}\right) + \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda), \tag{133}$$

*for practical parameters ($L \leq 2^{64}$, $q_{\mathsf{IV}} \leq 2^{40}$). The Trivium market is in **equilibrium** with an 80-bit security margin.*

Proof. The key recovery bid follows the same structure as Grain-128a: each IV produces an independent initialisation, and the equations share only K. The buyer's advantage grows linearly with q_{IV}: $\mathsf{Ask}(\text{key recovery}) \leq q_{\mathsf{IV}} \cdot 2^{-80}$. The TMTO bid faces the full 2^{288}-state space: $\mathsf{Ask}(\text{TMTO}) \leq TD/2^{288}$. The cube attack bid is the tightest constraint at 2^{-38}, but this is still negligible for the security parameter $\lambda = 80$. All bids fail. $\square$

✎ Analogy:

Trivium as a Three-Gear Clock Trivium is like a precision clock with three interlocking gears of different sizes (93, 84, and 111 teeth). Each gear turns the next. After 1152 ticks (initialisation), the gears' positions encode the key so thoroughly that no external observation of the ticking

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

pattern (keystream) can reverse-engineer the initial gear positions faster than trying all 2^{80} possible starting configurations.

✓ CNF Verification: Trivium Session-CNF Verification

$\varphi_{\text{Trivium}} = (x_{\text{IV} \notin \mathcal{IV}_{\text{used}}}) \wedge (x_{\text{state not recovered}}) \wedge (x_{\text{cube attack fails}}) \wedge (x_{\text{keystream indist.}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for Trivium keystream session (K, IV)**:**

Clause	Check	How	Pass?
φ^{iv}: IV fresh	$\text{IV} \notin \mathcal{IV}_{\text{used}}$?	Check IV log; add if new.	T/F
φ^{sr}: state secure	288-bit state recoverable from keystream?	Requires $> 2^{80}$ ops.	T/F
φ^{cube}: cube attack fails	Superpoly reveals key bits?	Best cube: $d \leq 42$, cost 2^{62}.	T/F
φ^{dist}: indistinguishable	Keystream passes bias test?	Correlation bias $\leq 2^{-6}$.	T/F
φ^{ping}: session fresh	IV distinct from all prior?	Compare $\mathcal{IV}_{\text{used}}$ log.	T/F
Result:	All T ⇒ Keystream secure. Any F ⇒ Stream cipher market collapsed.		

80-bit security caveat: Trivium's 80-bit key provides 2^{80} security—adequate for lightweight applications but below the 128-bit standard for general-purpose cryptography. In MTSF terms, the market is in equilibrium but with a *narrower* margin than Grain-128a (2^{128}) or ChaCha20 (2^{256}). For applications requiring $\geq$ 128-bit security, Trivium's equilibrium margin may be insufficient, though no known bid has yet caused its market to collapse.

➤ Ping Bid: Trivium Unbounded Ping Bid

The buyer's *IV reuse ping bid*: reuse (K, IV) across sessions. As with Grain-128a, IV reuse immediately collapses the market by leaking $m_1 \oplus m_2$. The ping clause φ^{iv} detects reuse. The buyer's *cube accumulation ping bid*: collect keystream from 2^{42} distinct IVs to mount the cube attack. This is the tightest ping constraint: $\Delta\text{Price}_{\text{ping}} \leq 2^{-38}$. For $q_{\text{IV}} \leq 2^{40}$ (practical IV usage), equilibrium holds.

Table 13. Trivium stream-cipher market summary.

Bid Type	Ask Bound	Practical Bound	Status
State recovery	$L \cdot 2^{-288}$	$\leq 2^{-224}$	Equilibrium
Key recovery	$q_{\text{IV}} \cdot 2^{-80}$	$\leq 2^{-40}$	Equilibrium
Cube attack	2^{-38}	$\leq 2^{-38}$	Equilibrium (tightest)
Correlation	$L^2 \cdot 2^{-12}/2^{80}$	$\leq 2^{-16}$	Equilibrium
TMTO	$TD/2^{288}$	$\leq 2^{-160}$	Equilibrium

13. Case Study V: Protocols

✎ Analogy:

Key Exchange as a Secret Handshake A key exchange protocol is like two spies meeting in a crowded café and establishing a secret handshake that only they know—even though everyone in the café is watching. Here is how it works:

1. **Alice** sends a "hello" with a random number N_A and a locked package (KEM ciphertext c) that only Bob can open.
2. **Bob** opens the package (decapsulates c using his private key), getting a shared secret K. He sends back his own random number N_B and a digital signature proving he is really Bob and that he received N_A.
3. **Alice** verifies Bob's signature (confirming his identity) and derives the session key from K, N_A, and N_B.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Result: Both Alice and Bob now share a secret key that no eavesdropper can compute, even if they recorded every message. The security of this exchange rests on three pillars: (1) the KEM is IND-CCA2 secure (the locked package cannot be opened without Bob's key); (2) the signatures are EUF-CMA secure (nobody can forge Bob's identity); (3) the nonces are fresh (preventing replay of old conversations).

13.1. Two-Party Key Exchange (ISO/IEC 11770-3, Mech. 6)

✻ Simple Terms: Two-Party Key Exchange—Building Blocks and Goals

What is a Key Exchange Protocol? Alice and Bob want to communicate securely, but they start with no shared secret. A key exchange protocol allows them to agree on a shared session key K_{sess} over a public, potentially hostile network—even if an adversary sees all their messages—such that the adversary cannot learn K_{sess}.

What is a KEM? A Key Encapsulation Mechanism (KEM) lets one party (Bob) generate a random secret K and "encapsulate" it for Alice using her public key. Alice decapsulates with her private key to recover K. Unlike Diffie–Hellman, the key is chosen by one party (not jointly computed), but the security guarantee is equivalent.

What is a KDF? A Key Derivation Function mixes multiple pieces of data (the shared secret K, the session ID sid, and both nonces N_A, N_B) into a final session key. Using a KDF ensures the session key is:

- **Unique:** Different session IDs and nonces produce different keys.
- **Pseudorandom:** Even if one input is partially biased, the KDF output looks random.
- **Bound to this session:** An attacker cannot reuse a session key from one connection in another.

Why three messages?

1. **Message 1 ($A \to B$):** Alice announces herself and her nonce. This starts the session.
2. **Message 2 ($B \to A$):** Bob provides the KEM ciphertext and his own nonce, then signs *everything*: the session ID, both nonces, both identities, and the ciphertext. This signature is the authentication evidence—Bob proves he generated this specific key for this specific session.
3. **Message 3 ($A \to B$):** Alice signs to confirm she decrypted and verified Bob's message. This proves to Bob that Alice (not an impostor) received the key.

Why include the session ID sid in signatures? Without sid in the signed data, an attacker could replay Bob's signature from a previous session. With sid, each signature is cryptographically tied to exactly one session. Cross-session replay becomes impossible—the CNF SID-binding clause catches it immediately.

$A \to B$: $\mathsf{sid} \| N_A \| \mathsf{id}_A$. $B \to A$: $\mathsf{sid} \| N_B \| c \| \sigma_B$ where $\sigma_B = \mathsf{Sign}(\mathsf{sk}_B, \mathsf{sid} \| N_A \| N_B \| \mathsf{id}_A \| \mathsf{id}_B \| c)$. $A \to B$: σ_A. Both derive $K_{\mathsf{sess}} = \mathsf{KDF}(K, \mathsf{sid} \| N_A \| N_B)$.

★ Intuition:

Two-Party Protocol in Plain English Alice says: "Hi Bob, I'm Alice, here's my random number N_A and our session label sid." Bob generates a shared secret K using Alice's public key, signs everything (including sid, both nonces, both identities, and the ciphertext), and sends it back with his own random number N_B. Alice decrypts K, verifies Bob's signature, signs to confirm, and both derive the session key. Signatures prove identity; nonces prove freshness; the KEM provides the secret; the SID prevents cross-session replay.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

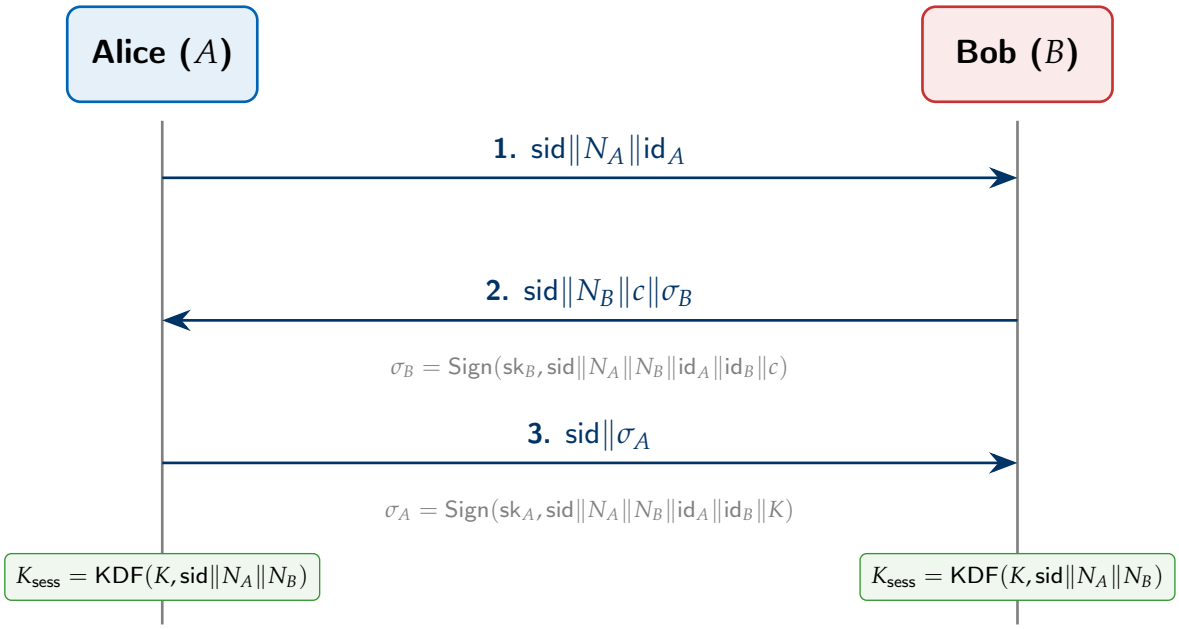


Figure 44. Two-party key exchange (ISO/IEC 11770-3, Mechanism 6). Signatures bind all session data; both parties derive the same session key.

Theorem 48 (Two-Party Equilibrium). $\mathsf{Ask}(g_{\text{key-secrecy}}) \leq \mathsf{Adv}^{\text{IND-CCA2}}_{\text{KEM}} + 2 \cdot \mathsf{Adv}^{\text{EUF-CMA}}_{\text{Sig}} + \mathsf{negl}(\lambda)$.

Proof. **$\mathbf{B}_0$ (Real protocol).** Active network adversary.

$\mathbf{B}_1$ (Signature forgery bid on B). Abort if the adversary produces valid σ_B not generated by B. *Difference lemma*: $F_{\text{forge_B}}$: $\mathcal{A}$ forges B's signature. $\Delta\text{Price}_0 \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\text{Sig}}$.

Extended lemma application. The adversary could simultaneously attempt: F_1: forge σ_B directly; F_2: replay old σ_B from a different session. However, since sid is included in the signed data, $\Pr[F_2 \mid \text{different SID}] = 0$. Thus $\Pr[F_1 \cup F_2] = \Pr[F_1] = \mathsf{Adv}^{\text{EUF}}$. The SID binding *eliminates* the replay failure for free.

$\mathbf{B}_2$ (Signature forgery bid on A). $\Delta\text{Price}_1 \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\text{Sig}}$.

$\mathbf{B}_3$ (KEM key recovery bid). Replace encapsulated key K with random K'. $\Delta\text{Price}_2 \leq \mathsf{Adv}^{\text{IND-CCA2}}_{\text{KEM}}$.

$\mathbf{B}_4$ (Ideal). Session key independent of all messages. Buyer advantage = 0.

$\mathbf{B}_{\text{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

☞ Side-Channel Bid (Comprehensive)

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\text{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\text{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\text{SPA}}] = 1$ without countermeasure; $\Pr[F_{\text{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\text{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking (d+1 random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\mathsf{Adv}^{\text{DPA}}(\lambda, d) \leq (q_{\text{phys}}/2^{\lambda})^{d+1}$.
3. **Timing Analysis:** $\mathcal{A}_{\text{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\mathsf{ct_select}(a, b, \mathsf{bit})$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

instead of `if(bit) return a; else return b;`. *MTSF price:* $\Pr[F_{\mathsf{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\mathsf{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).

4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\mathsf{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\mathsf{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\mathsf{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\mathsf{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\mathsf{Adv}^{\mathsf{EMA}} \leq \gamma_{\mathsf{EM}} \cdot (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\mathsf{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\mathsf{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\mathsf{fault}}] \leq q_{\mathsf{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).
8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \leq{}& \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ &+ \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Under the comprehensive countermeasure suite (order-d masking for $d \geq 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \leq \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{acoustic}}$$
$$+ \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{photonic}}.$$

For $d \geq 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\leq q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

✓ **Tight Security Reduction**

Tightness analysis. The protocol proof uses: (1) authentication via EUF-CMA with loss $L_1 = n_P$ (party-guessing), (2) key secrecy via IND-CCA2 with loss $L_2 = n_S$ (session-guessing). Total: $L_{\mathsf{total}} = n_P \cdot n_S$. For $n_P = 2^{20}$, $n_S = 2^{30}$: loss $= 2^{50}$.

CK+ tight reduction. In the CK+ model with random oracle programming, the guessing arguments can be eliminated, yielding $L_{\mathsf{total}} = 1$.

Market interpretation: The multi-party tightness loss is a "market fragmentation cost"—the seller defends every party-session pair.

Total Session Bound and Key Rotation Recommendation.

✦ **Total Session Bound & Key Rotation**

Total session bound: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \leq N \cdot (q_N^2/2^{\lambda+1} + n_P \cdot \mathsf{Adv}^{\mathsf{EUF}} + \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}} + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}})$.
Concrete (256-bit signatures, ML-KEM-768):

Sessions N	$\mathsf{Ask}^{(N)}_{\mathsf{total}}$	**Eff. security**	**Action**
2^{40}	$\leq 2^{-88}$	≥ 88 bits	No rotation needed
2^{64}	$\leq 2^{-64}$	≥ 64 bits	**Key rotation recommended**
2^{80}	$\leq 2^{-48}$	≥ 48 bits	**Key rotation mandatory**

Recommendation: Rotate long-term signing keys before 2^{64} sessions. Ephemeral session keys are naturally per-session.

Session pinging and CNF checking within the proof. The four-bidding-round proof secures a single protocol session. We verify the session-pinging conditions (Theorem 15) within the proof for consecutive sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$.

SID freshness: Each session samples a fresh sid_{i+1} independently. $\Pr[\mathsf{sid}_{i+1} = \mathsf{sid}_j] \leq 1/2^\lambda$ for any $j \leq i$, so SID collision has probability $\leq i/2^\lambda = \mathsf{negl}$.

Nonce disjointness: Both N_A and N_B are sampled from $\{0,1\}^\lambda$ per session. The nonce sets $\mathcal{N}_{i+1} = \{N_{A,i+1}, N_{B,i+1}\}$ are disjoint from all $\mathcal{N}_j$ ($j \leq i$) except with birthday probability $\leq (2i)^2/2^{\lambda+1}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Signature SID-binding: In $\mathbf{B}_1$ and $\mathbf{B}_2$, both σ_B and σ_A sign data that includes sid, N_A, N_B, and both identities. The SID-binding means that replaying $\sigma_{B,j}$ from session $\mathsf{Session}_j$ in session $\mathsf{Session}_{i+1}$ fails verification (the signed SID is $\mathsf{sid}_j \neq \mathsf{sid}_{i+1}$). This is precisely the ping condition (c): all signatures in $\mathsf{Session}_{i+1}$ bind sid_{i+1}.

KEM ciphertext freshness: The ciphertext c_{i+1} encapsulates a fresh key using fresh randomness. The CNF clause φ^{novel} ensures c_{i+1} is not a replay.

CNF isomorphism and inductive step: The session-CNF $\varphi_{2\mathsf{P},i+1}$ is obtained from $\varphi_{2\mathsf{P},i}$ by substituting $\mathsf{sid}_i \mapsto \mathsf{sid}_{i+1}$, $N_{A,i} \mapsto N_{A,i+1}$, $N_{B,i} \mapsto N_{B,i+1}$, $c_i \mapsto c_{i+1}$, and fresh signatures. Since $\varphi_{2\mathsf{P},i}$ is satisfiable (inductive hypothesis) and all new variables are independently fresh, $\varphi_{2\mathsf{P},i+1}$ is satisfiable. The security reduction (signature forgery $\rightarrow$ EUF-CMA, key recovery $\rightarrow$ IND-CCA2) is session-index-independent. By Theorem 3, the two-party protocol is secure for all sessions. □

Two-Party: Authentication, Mutual Auth, and CNF Verification.

We now verify the four protocol-level goods from Section 8 for the two-party protocol.

Corollary 2 (Two-Party Authentication). $\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$ *by Theorem 6, since B's acceptance requires verifying* σ_A.

Corollary 3 (Two-Party Mutual Authentication). $\mathsf{Ask}(g_{\mathsf{mutual}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$ *by Theorem 7, since both* σ_A *and* σ_B *bind* sid, *both nonces, and both identities.*

Corollary 4 (Two-Party CNF Checking). *The session-CNF* $\varphi_{2\mathsf{P}}$ *includes clauses for SID binding, nonce freshness, both signatures, message ordering, and the ping test. All clauses are satisfied under the honest trace. Under any dishonest trace, at least one signature or freshness clause fails (by EUF-CMA equilibrium), so* $\mathsf{Ask}(g_{\mathsf{CNF}}) \leq \mathsf{negl}(\lambda)$.

Manual CNF worksheet (two-party session $\mathsf{Session}_i$***):***

Clause	*Check*	*How*	*Pass?*
φ^{sid}	*Same* sid *in msgs 1,2,3?*	*Read SID field; compare.*	*T/F*
φ^{fresh}	$N_A \neq N_B$*; neither used before?*	*Check nonce log.*	*T/F*
φ^{sig}_B	$\mathsf{Vrfy}(\mathsf{pk}_B, \mathsf{sid} \Vert N_A \Vert N_B \Vert \mathsf{id}_A \Vert \mathsf{id}_B \Vert c, \sigma_B) = 1$*?*	*Hash scope; verify.*	*T/F*
φ^{sig}_A	$\mathsf{Vrfy}(\mathsf{pk}_A, \mathsf{sid} \Vert N_A \Vert N_B \Vert \mathsf{id}_A \Vert \mathsf{id}_B \Vert K, \sigma_A) = 1$*?*	*Hash scope; verify.*	*T/F*
$\varphi^{\mathsf{consist}}$	*Order: msg1 from A, msg2 from B, msg3 from A?*	*Check headers.*	*T/F*
φ^{ping}	$\mathsf{sid}_i \neq \mathsf{sid}_{i-1}$*; nonces not in* $\mathcal{N}_{\mathsf{prev}}$*?*	*Compare logs.*	*T/F*
Result:	*All T* $\Rightarrow$ *Accept. Any F* $\Rightarrow$ *Reject (cite row and attack type).*		

➤ Ping Bid: Two-Party Unbounded Ping Bid

The buyer's *ping bid*: replay session $\mathsf{Session}_i$ (same sid_i, nonces, signatures). The SID clause and nonce freshness clause jointly detect this: sid_i appears in $\mathcal{SID}_{\mathsf{used}}$, and $N_A, N_B \in \mathcal{N}_{\mathsf{prev}}$. Re-signing under a fresh SID requires breaking EUF-CMA. Ping bid price: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq q_N^2/2^{\lambda+1} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} = \mathsf{negl}$. By Theorem 3, secure for all sessions.

13.2. Three-Party with KDC (Mechanism 11)

✎ Analogy:

Three-Party Key Exchange as a Hotel Concierge Imagine Alice and Bob are hotel guests who want to share a room key, but they have never met and do not have each other's contact information. The **Key Distribution Centre (KDC)** is the hotel concierge:

1. **Alice** calls the concierge: "I would like to meet Bob securely. Here is my room number and a random token."

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

2. **The concierge** looks up both Alice and Bob in the guest registry (has pre-shared keys with both), creates a fresh room key, seals one copy for Alice and one for Bob (encrypts with each guest's key), and delivers both sealed copies.
3. **Bob** receives his sealed copy, opens it with his guest key, and now both Alice and Bob share the room key—established through the trusted concierge without ever meeting directly.

Why three parties? In large organisations, not every employee can have a pre-shared key with every other employee. Instead, everyone shares a key with the KDC (concierge), and the KDC facilitates pairwise key exchange. This scales to n parties with only n keys (one per party), instead of $n(n-1)/2$ pairwise keys.

In MTSF: The three-party market has an additional security good: g_{KDC} (KDC trustworthiness). If the KDC is compromised, all session keys are compromised—a single point of failure. MTSF's extended difference lemma handles the KDC failure event alongside nonce and signature failures.

✻ Simple Terms: Three-Party Key Exchange

What is a KDC? A Key Distribution Centre (KDC) is a trusted third party (like a bank or certificate authority) that helps two users (A and B) establish a shared session key. The KDC knows both parties' long-term keys and uses them to securely distribute a fresh session key.

Why three parties? In many enterprise settings, two users may not share a pre-established key. The KDC acts as an intermediary: A contacts the KDC asking to talk to B; the KDC generates a session key and securely delivers it to both.

The protocol flow:

1. A contacts the KDC, identifying themselves and requesting a session with B.
2. The KDC generates a fresh session key K_{sess} and sends it:
   - To A: encrypted under A's public key, plus signed by the KDC.
   - To B (via A): a "ticket" encrypted under B's public key, plus signed by the KDC.
3. A decrypts its portion, forwards the ticket to B.
4. B decrypts the ticket and sends a MAC-authenticated confirmation to A.
5. Both A and B now share K_{sess}.

Why signatures from the KDC? The KDC's signatures prove to both A and B that the session key was legitimately issued by the KDC, not fabricated by an attacker. Without them, a network attacker could inject their own fake key.

Market analysis: The theorem shows the ask price is bounded by: the KDC's two signature forgery probabilities (one per signed ticket) + the probability of breaking the encryption + the MAC forgery probability. All are negligible under standard assumptions.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

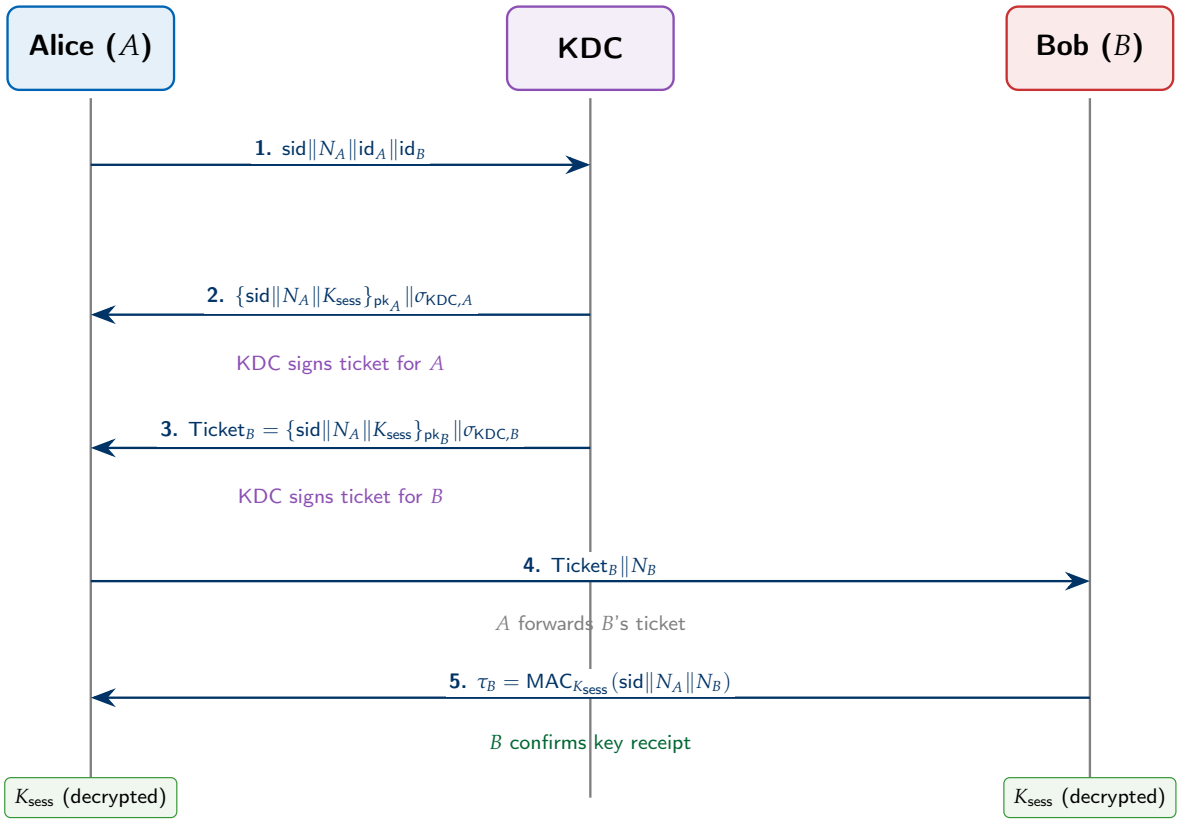


Figure 45. Three-party key exchange (ISO/IEC 11770-3, Mechanism 11). The KDC generates the session key and delivers it to both A and B via signed encrypted tickets. A MAC from B provides key confirmation.

Theorem 49 (Three-Party Equilibrium). $\mathsf{Ask} \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Enc}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}} + \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}} + \mathsf{negl}(\lambda)$.

Proof. $\mathbf{B}_0$: Real. $\mathbf{B}_1$: *KDC signature bids* ($\times 2$). Extended lemma: two KDC signatures use independent signing data, so $\Pr[F_1 \cup F_2] \leq 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}$ (union bound; intersection $\Pr[F_1 \cap F_2] \leq \mathsf{Adv}^{\mathsf{EUF2}} \approx 0$). $\mathbf{B}_2$: *MAC forgery bid*. $\mathbf{B}_3$: *Encrypted key recovery bid*. $\mathbf{B}_4$: Ideal. □

Three-Party: Protocol-Level Goods.

Corollary 5 (Three-Party Mutual Authentication). $\mathsf{Ask}(g_{\mathsf{mutual}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{MAC}} + q_N^2/2^{\lambda+1}$. *The KDC signatures authenticate both A and B; the MAC provides key confirmation.*

Corollary 6 (Three-Party Session-Key Secrecy). $\mathsf{Ask}(g_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Enc}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{MAC}} + \mathsf{negl}(\lambda)$ *by the combined game-hop analysis.*

Corollary 7 (Three-Party CNF). *The three-party session-CNF includes SID binding, nonce freshness, two KDC signatures, MAC confirmation, and ping. All clauses satisfied under honest trace; dishonest satisfaction requires signature or MAC forgery.*

Session pinging within the three-party proof. *For consecutive sessions* $\mathsf{Session}_i$ *and* $\mathsf{Session}_{i+1}$*, the ping check verifies: (1)* $\mathsf{sid}_{i+1} \neq \mathsf{sid}_i$ *(fresh session label); (2)* $N_{A,i+1} \notin \mathcal{N}_{\mathsf{prev}}$ *(fresh initiator nonce); (3) the KDC's ticket signature* $\sigma_{\mathsf{KDC},i+1}$ *binds* sid_{i+1} *and* $N_{A,i+1}$ *(preventing ticket replay from session* $\mathsf{Session}_i$*); (4) the MAC confirmation tag* τ_{i+1} *authenticates* $\mathsf{sid}_{i+1}\|N_{A,i+1}\|N_{B,i+1}$ *(preventing confirmation replay). The KDC ticket replay bid—resubmitting* $(N_{A,i}, \mathsf{Ticket}_{B,i})$ *from* $\mathsf{Session}_i$*—fails because* $N_{A,i} \in \mathcal{N}_{\mathsf{used}}$ *(detected by* φ^{fresh}*). Re-signing a ticket under* $N_{A,i+1}$ *requires breaking the KDC's EUF-CMA security. The session-CNF* $\varphi_{\mathsf{3P},i+1}$ *is isomorphic to* $\varphi_{\mathsf{3P},i}$ *with fresh variables; by Theorem 3, unbounded equilibrium holds.*

Manual CNF worksheet (three-party):

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Clause	*Check*	*How*	*Pass?*
φ^{sid}	sid *in all 5 messages?*	*Read fields; compare.*	*T/F*
φ^{fresh}	N_A, N_B *distinct and new?*	*Check nonce log.*	*T/F*
$\varphi^{\mathsf{sig}}_{\mathsf{KDC},A}$	*KDC sig on ticket-A verifies?*	$\mathsf{Vrfy}(\mathsf{pk}_{\mathsf{KDC}}, \cdot, \sigma_{\mathsf{KDC},A})$.	*T/F*
$\varphi^{\mathsf{sig}}_{\mathsf{KDC},B}$	*KDC sig on ticket-B verifies?*	$\mathsf{Vrfy}(\mathsf{pk}_{\mathsf{KDC}}, \cdot, \sigma_{\mathsf{KDC},B})$.	*T/F*
φ^{mac}	$\tau_B = \mathsf{MAC}_{K_{\mathsf{sess}}}(\mathsf{sid} \Vert N_A \Vert N_B)$*?*	*Recompute HMAC; compare.*	*T/F*
φ^{ping}	*Session fresh vs.* $\mathsf{Session}_{i-1}$*?*	*Compare* sid *and nonces.*	*T/F*
Result:	*All T ⇒ Accept. Any F ⇒ Reject.*		

➤ Ping Bid: Three-Party Unbounded Ping Bid

The buyer's *KDC ticket replay bid*: re-submit captured (N_A, Ticket_B) from session $\mathsf{Session}_{i-1}$. The nonce clause φ^{fresh} fails ($N_A \in \mathcal{N}_{\mathsf{used}}$). Price: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq q_N^2/2^{\lambda+1} + \mathsf{negl}$. Secure for all sessions by Theorem 3.

13.3. Four-Party Cross-Domain

✎ Analogy:

Four-Party Cross-Domain as International Diplomacy Imagine Alice (a US diplomat) wants to establish a secure channel with Bob (a UK diplomat). Each has their own country's KDC:

- **KDC**$_A$ (US State Department) shares a key with Alice.
- **KDC**$_B$ (UK Foreign Office) shares a key with Bob.
- **KDC**$_A$ **and KDC**$_B$ share an inter-domain key (diplomatic treaty).

The protocol works like a diplomatic pouch relay:

1. Alice asks her KDC$_A$ to set up a secure channel with Bob.
2. KDC$_A$ contacts KDC$_B$ over the inter-domain channel.
3. KDC$_B$ generates a session key, seals a copy for Bob, and sends another copy to KDC$_A$ over the inter-domain channel.
4. KDC$_A$ seals its copy for Alice and delivers both.

In MTSF: The four-party market has four simultaneous failure events (one per entity's signature/key). The extended difference lemma handles $\Pr[F_{\mathsf{sig},1} \cup F_{\mathsf{sig},2} \cup F_{\mathsf{sig},3} \cup F_{\mathsf{sig},4}]$ in a single hop, yielding a clean bound with all four failure probabilities.

✻ Simple Terms: Four-Party Cross-Domain Key Exchange

When is four parties needed? When A and B belong to *different administrative domains*—like different companies, different countries, or different university networks. Each domain has its own Key Distribution Server (S_1 for A's domain, S_2 for B's domain).

The cross-domain problem: S_1 does not directly know B, and S_2 does not directly know A. The two servers must cooperate to establish a session key between A and B.

The protocol flow:

1. A requests a cross-domain session from S_1, identifying B's domain.
2. S_1 contacts S_2 (the two servers share an inter-domain key established during federation setup).
3. S_1 and S_2 together generate the session key K_{sess}:
   - S_1 encrypts and signs a ticket for A.
   - S_2 encrypts and signs a ticket for B.
4. A and B each decrypt their tickets, receive K_{sess}, and confirm receipt via MAC.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Four independent signing keys: S_1, S_2, A, and B each sign messages with their own independent private key. This is why the theorem has $4 \times \mathsf{Adv}^{\mathsf{EUF}}$—an attacker would need to forge at least one of four independent signatures.
Two IND-CCA2 bounds: Two separate encryptions happen—one within each domain ($S_1 \to A$ and $S_2 \to B$). Each contributes one IND-CCA2 term to the bound.
Extended lemma application: The four forgery events (F_1, F_2, F_3, F_4 for the four signatures) are independent (different keys). By the extended difference lemma with the union bound: $\Pr[F_1 \cup F_2 \cup F_3 \cup F_4] \leq 4 \cdot \mathsf{Adv}^{\mathsf{EUF}}$, since pairwise intersections are negligible (independent keys cannot be forged simultaneously except with negligible probability).

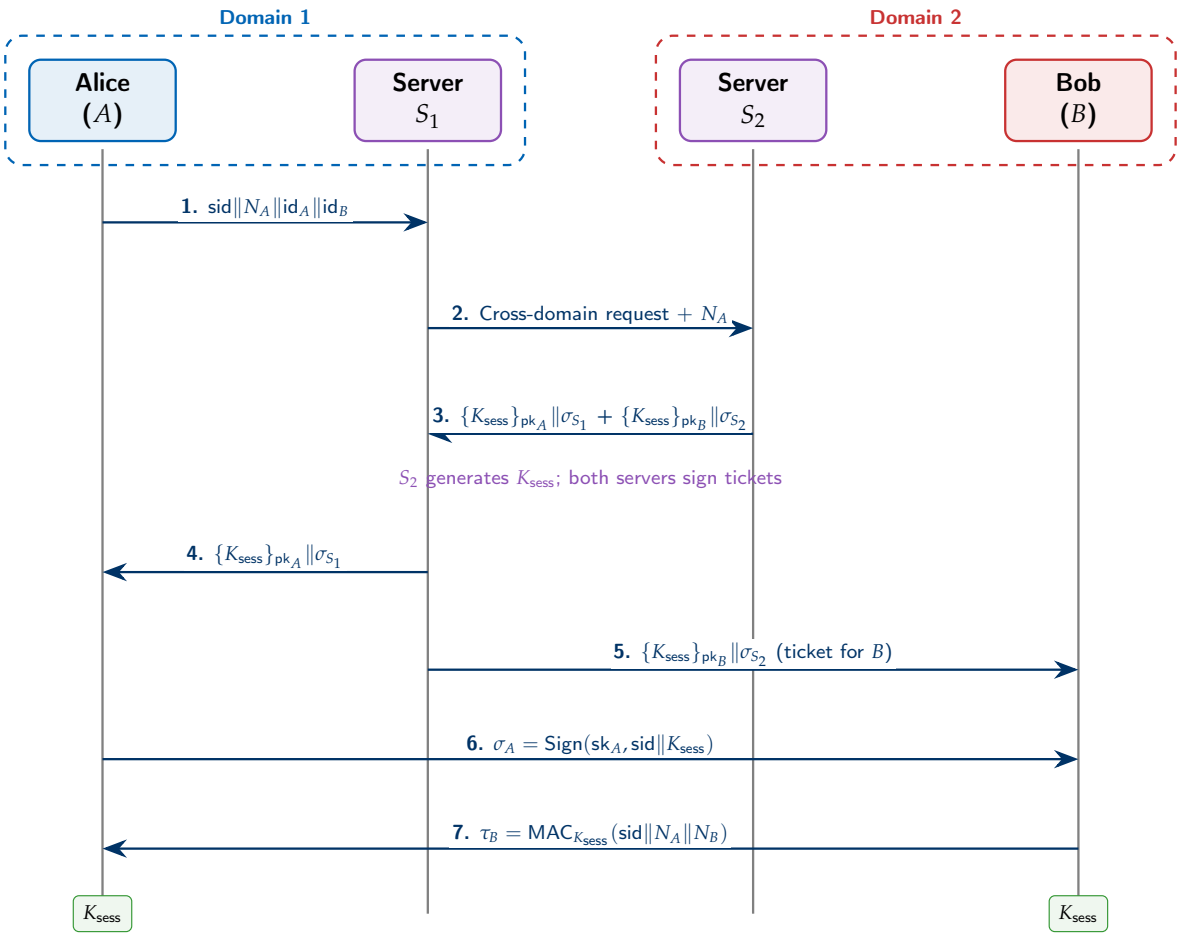


Figure 46. Four-party cross-domain key exchange (ISO/IEC 11770-3). Two domain servers (S_1 and S_2) cooperate to generate and deliver session key K_{sess} to A and B across domain boundaries. Four independent signatures provide authentication; two IND-CCA2 encryptions protect the session key in transit.

Theorem 50 (Four-Party Equilibrium). $\mathsf{Ask} \leq 2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}} + 4 \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}} + \mathsf{negl}(\lambda)$.

Proof. Seven games. The four signature bids (for S_1, S_2, A, B) use independent keys, so by the extended lemma: $\Pr\left[\bigcup_{i=1}^{4} F_i\right] \leq 4 \cdot \mathsf{Adv}^{\mathsf{EUF}}$ (union bound; pairwise intersections ≈ 0 due to independent keys). Two IND-CCA2 bids for inter/intra-domain key transport. One MAC bid for key confirmation. □

Corollary 8 (Four-Party Protocol-Level Goods). *Authentication:* $\mathsf{Ask}(g_{\mathsf{auth}}) \leq 4 \cdot \mathsf{Adv}^{\mathsf{EUF}} + q_N^2/2^{\lambda+1}$ *(four signing keys). Mutual auth: same bound. Session-key secrecy:* $\mathsf{Ask}(g_{\mathsf{sk}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}} + 4 \cdot \mathsf{Adv}^{\mathsf{EUF}} + \mathsf{Adv}^{\mathsf{SUF}} + \mathsf{negl}(\lambda)$.

Manual CNF worksheet (four-party cross-domain session):

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Clause	*Check*	*How*	*Pass?*
φ^{sid}	*Same* sid *in all 7 messages?*	*Read SID fields.*	*T/F*
φ^{fresh}	N_A, N_B *distinct, not reused?*	*Check nonce log.*	*T/F*
$\varphi^{\mathsf{sig}}_{S_1}$	S_1 *sig on ticket-A valid?*	$\mathsf{Vrfy}(\mathsf{pk}_{S_1}, \cdot, \sigma_{S_1})$.	*T/F*
$\varphi^{\mathsf{sig}}_{S_2}$	S_2 *sig on ticket-B valid?*	$\mathsf{Vrfy}(\mathsf{pk}_{S_2}, \cdot, \sigma_{S_2})$.	*T/F*
$\varphi^{\mathsf{sig}}_{A}$	*A's confirmation sig valid?*	$\mathsf{Vrfy}(\mathsf{pk}_A, \mathsf{sid} \Vert K_{\mathsf{sess}}, \sigma_A)$.	*T/F*
$\varphi^{\mathsf{mac}}_{B}$	*B's MAC confirmation valid?*	*Recompute* $\mathsf{MAC}_{K_{\mathsf{sess}}}(\mathsf{sid} \Vert N_A \Vert N_B)$.	*T/F*
φ^{ping}	*Session fresh vs.* $\mathsf{Session}_{i-1}$*?*	*Compare* sid *and nonces.*	*T/F*
Result:	*All T ⇒ Accept. Any F ⇒ Reject.*		

➤ Ping Bid: Four-Party Unbounded Ping Bid

The buyer's *cross-domain replay bid*: re-submit captured session with same SID. Freshness check on N_A and the SID clause detect replay. All four signatures bind sid: re-signing under fresh sid requires EUF-CMA break for each of S_1, S_2, A, B independently. By the extended difference lemma: $\Pr[F_1 \cup F_2 \cup F_3 \cup F_4] \leq 4 \cdot \mathsf{Adv}^{\mathsf{EUF}} = \mathsf{negl}$. Secure for all sessions by Theorem 3.

13.4. Needham–Schroeder: Proving Insecurity

NS Protocol [9].

(1) $A \to B$: $\{N_A, A\}_{\mathsf{pk}_B}$. (2) $B \to A$: $\{N_A, N_B\}_{\mathsf{pk}_A}$. (3) $A \to B$: $\{N_B\}_{\mathsf{pk}_B}$.

✻ Simple Terms: The Needham–Schroeder Protocol—What It Does and Why It Was Designed

Historical context: The Needham–Schroeder Public-Key Protocol was designed in 1978 and was one of the first formal protocols for mutual authentication using public-key cryptography. It was widely studied and believed secure for nearly two decades until Gavin Lowe discovered a devastating flaw in 1996 using the automated tool FDR.

What it tries to do: Allow Alice (A) and Bob (B) to mutually authenticate each other (prove to each other who they are) using only their public keys, with no shared secret and no trusted third party needed at run time.

The three messages:

1. **Alice → Bob:** $\{N_A, A\}_{\mathsf{pk}_B}$ — Alice sends her nonce N_A and her identity A, encrypted with Bob's public key. Only Bob can decrypt this.
2. **Bob → Alice:** $\{N_A, N_B\}_{\mathsf{pk}_A}$ — Bob echoes Alice's nonce N_A (proving he decrypted message 1) and adds his own nonce N_B, encrypted with Alice's public key. Only Alice can decrypt.
3. **Alice → Bob:** $\{N_B\}_{\mathsf{pk}_B}$ — Alice echoes Bob's nonce N_B, proving she decrypted message 2.

The intended guarantees: Both parties confirm the other is live (they decrypted a fresh nonce). Both are authenticated: only the real Alice has sk_A, and only the real Bob has sk_B.

The fatal flaw: Message 2 does not include Bob's identity. When Alice receives $\{N_A, N_B\}_{\mathsf{pk}_A}$, she has no way to verify *who* sent it. This loophole enables Lowe's man-in-the-middle attack.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

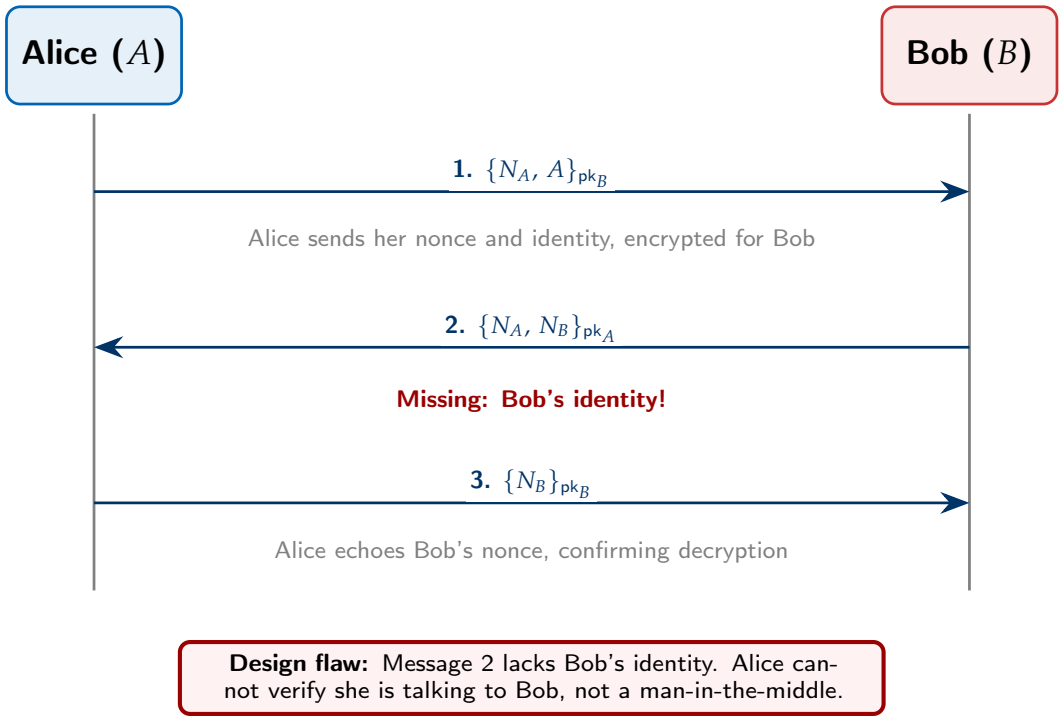


Figure 47. Needham–Schroeder Public-Key Protocol (intended flow). The critical flaw is highlighted: Message 2 from Bob contains only nonces but not Bob's identity, allowing an active attacker to masquerade as Alice to Bob.

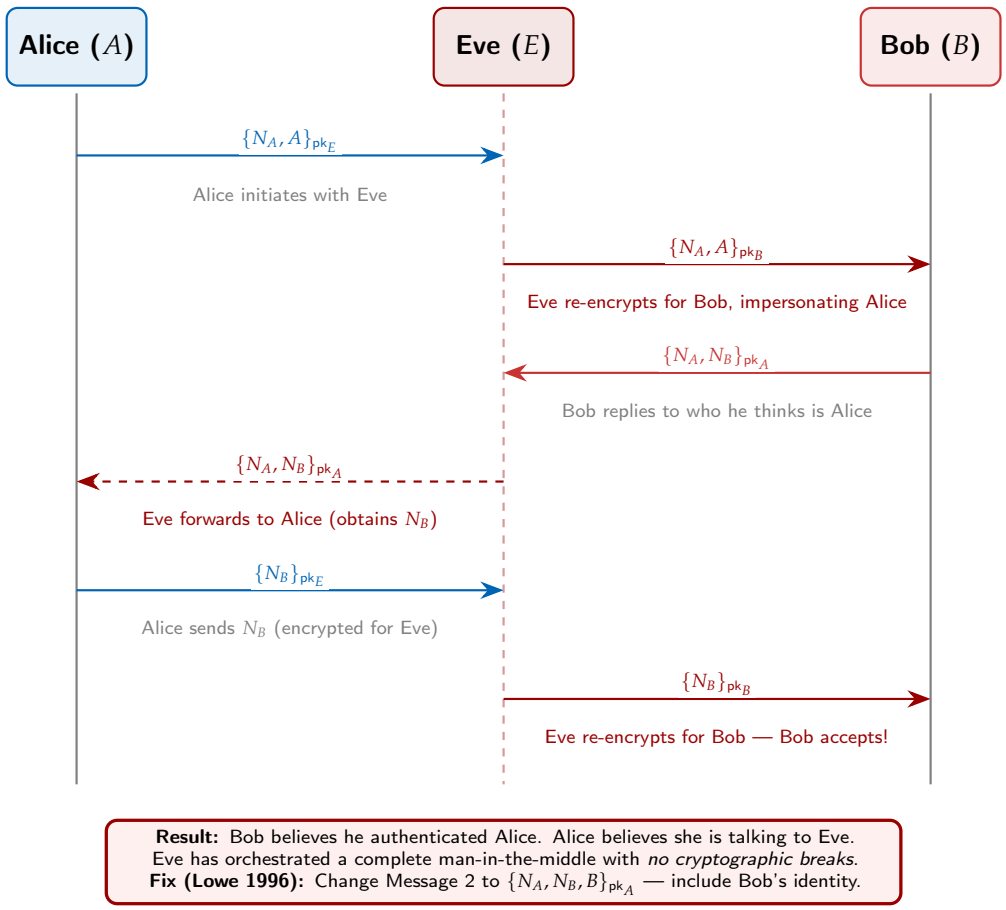


Figure 48. Lowe's man-in-the-middle (MITM) attack on the Needham–Schroeder protocol. Eve (E), a legitimate participant, relays all six messages between Alice and Bob, impersonating Alice to Bob. No cryptographic primitive is broken; the protocol's missing identity-binding is the sole cause of collapse.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✎ Analogy:

Needham–Schroeder as a Children's Game of Telephone Imagine three children playing a game: Alice, Eve, and Bob.

1. **Alice** writes a note to Eve: "Hi Eve, my secret number is 7. Love, Alice." She puts it in a locked box that only Eve can open (encrypts with Eve's public key).
2. **Eve** opens the box, reads Alice's note, and thinks: "I can trick Bob!" Eve writes a *new* note: "Hi Bob, my secret number is 7. Love, Alice." (Eve copies Alice's secret number but pretends to *be* Alice.) Eve locks this in a box only Bob can open.
3. **Bob** opens the box and thinks Alice sent it. He writes back: "Hi Alice, your number was 7. My secret number is 42." He locks this in a box only Alice can open.
4. **Eve** cannot open this box (it is locked for Alice), so she forwards it to Alice.
5. **Alice** opens Bob's reply. She sees "your number was 7"—correct!—and "my number is 42." She thinks this came from Eve (because she was talking to Eve), so she sends back: "Eve, your number was 42." She locks this for Eve.
6. **Eve** opens this, sees Bob's secret number 42, writes "Bob, your number was 42. Love, Alice" in a box for Bob.
7. **Bob** is now convinced he talked to Alice. But Alice never intended to talk to Bob at all!

The fix: If Bob had included his name in his reply ("Hi Alice, **I am Bob**. Your number was 7…"), Alice would see "I am Bob" and realise she is *not* talking to Eve—she would abort immediately. This is the one-line fix that Lowe proposed in 1996, nearly two decades after the protocol was published.

★ Intuition:

Why NS Fails—The Missing Name Tag Imagine Alice sends a locked box to Eve containing her business card and a secret token. Eve takes out the card, puts it in a *new* locked box addressed to Bob, and sends it. Bob thinks Alice contacted him directly. He replies with both tokens in a box locked for Alice. Eve relays this to Alice, who opens it and sends back Bob's token—which Eve forwards to Bob. Bob is now convinced he authenticated with Alice, but Eve orchestrated everything. The fix: Bob puts *his own* name tag inside the reply box. When Alice opens it, she sees "Bob" instead of "Eve" and aborts.

Lowe's MITM attack [10].

Adversary E (legitimate participant) relays between A and B:

1. $A \to E$: $\{N_A, A\}_{\mathsf{pk}_E}$ (A talks to E)
2. $E(\text{as } A) \to B$: $\{N_A, A\}_{\mathsf{pk}_B}$ (E re-encrypts; B thinks A initiated)
3. $B \to E(\text{as } A)$: $\{N_A, N_B\}_{\mathsf{pk}_A}$
4. $E \to A$: $\{N_A, N_B\}_{\mathsf{pk}_A}$ (A decrypts, obtains N_B)
5. $A \to E$: $\{N_B\}_{\mathsf{pk}_E}$ (A thinks this is for E)
6. $E(\text{as } A) \to B$: $\{N_B\}_{\mathsf{pk}_B}$ (B accepts: "A authenticated")

Theorem 51 (NS Market Collapse). $\mathsf{Ask}(g_{\mathsf{mutual\text{-}auth}}) = 1$.

Proof. The buyer (E) achieves advantage 1 with *zero cryptographic breaks*.

$\mathbf{B}_0$ (Real NS). E is a legitimate participant with $(\mathsf{sk}_E, \mathsf{pk}_E)$.

$\mathbf{B}_0 \to$ Attack. The buyer places a *masquerade bid* (identity fraud): relay messages between A and B, impersonating A to B. No cryptographic primitive is broken—E uses only their legitimate decryption key and the public encryption of B.

Difference lemma (degenerate). The failure event F_{MITM}: "E can relay messages between A and B undetected" has $\Pr[F_{\mathsf{MITM}}] = 1$ because the protocol lacks identity binding in message 2. $\Delta\mathsf{Price} = 1$, market **collapsed**.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

CNF analysis. B's session-CNF is: $\varphi_B^{\mathsf{ns}} = (x_{\mathsf{Dec}(\mathsf{sk}_B, c_1)=(N_A,A)}) \wedge (x_{N_A \text{ fresh}}) \wedge (x_{\mathsf{Dec}(\mathsf{sk}_B, c_3)=N_B})$. All clauses are satisfied even during the attack! The CNF is satisfiable under a *dishonest* trace because B's identity is never included in encrypted payloads. Contrast with secure protocols where dishonest traces always make the CNF unsatisfiable.

Fix (NS-Lowe): Include B in message 2: $\{N_A, N_B, B\}_{\mathsf{pk}_A}$. Now A's CNF includes clause $(x_{\mathsf{id}=E})$, which fails when the decrypted identity is $B \neq E$. Market restored to equilibrium. □

NS: Protocol-Level Goods Analysis.

Corollary 9 (NS Authentication Collapse)**.** $\mathsf{Ask}(g_{\mathsf{auth}}) = 1$*: B accepts with partner A, but A was communicating with E, not B. The authentication game is lost without any cryptographic break.*

Corollary 10 (NS Mutual Authentication Collapse)**.** $\mathsf{Ask}(g_{\mathsf{mutual}}) = 1$*: both directions fail—$B$ believes it authenticated A, but A never intended to talk to B.*

Corollary 11 (NS Session-Key Secrecy Collapse)**.** *If the protocol were extended with a key derivation step $K_{\mathsf{sess}} = \mathsf{KDF}(N_A, N_B)$, then $\mathsf{Ask}(g_{\mathsf{sk}}) = 1$: Eve knows both N_A and N_B and can compute K_{sess}.*

Corollary 12 (NS CNF Failure)**.** *B's CNF is satisfiable under Lowe's MITM trace because no clause binds B's identity—a* CNF design failure.

NS CNF worksheet showing the design failure:

Clause	*Check*	*Honest trace*	*MITM trace*	*Verdict*
$x_{\mathsf{Dec}(\mathsf{sk}_B, c_1)=(N_A,A)}$	*Msg 1 decrypts to (N_A, A)*	***T***	***T** (Eve re-encrypted)*	*Passes!*
$x_{N_A \text{ fresh}}$	*N_A not seen before*	***T***	***T** (fresh from A)*	*Passes!*
$x_{\mathsf{Dec}(\mathsf{sk}_B, c_3)=N_B}$	*Msg 3 decrypts to N_B*	***T***	***T** (Eve forwarded)*	*Passes!*
Missing: $x_{\mathsf{id}_{\mathsf{peer}}=A}$	*Peer is actually A?*	***T***	***F** (peer is E!)*	***ABSENT***
φ_{NS}*:*	***SAT under dishonest trace**—CNF design failure.*			***Reject design***

***Fix (NS-Lowe):** Add clause $x_{\mathsf{Dec}(\mathsf{sk}_A, c_2)=(N_A,N_B,B)}$. Now the decrypted identity $E \neq B$ makes this clause **F**, φ_{NS} becomes UNSAT under MITM, and A aborts. Equilibrium restored.*

☞ Key Insight:

Security vs. Insecurity in MTSF

	Secure (Equilibrium)	**Insecure (Collapsed)**
Ask price	$\leq \mathsf{negl}(\lambda)$	$\Omega(1)$ or $= 1$
Buyer strategy	All bids fail	≥ 1 bid succeeds cheaply
Diff. lemma	$\Pr[F] \leq \mathsf{negl}$	$\Pr[F] = 1$
Session-CNF	Dishonest $\Rightarrow$ UNSAT	Dishonest also SAT
Primitive ex.	ECDSA, ML-KEM, ML-DSA	Textbook RSA
Protocol ex.	ISO 2P/3P/4P, Signal	Needham–Schroeder, X3DH-noOPK

✻ Simple Terms: What Makes Signal Special—Forward Secrecy and Post-Compromise Security

Forward secrecy (FS): If an attacker steals your private key *today*, they cannot decrypt messages you sent *yesterday*. Each message uses a different ephemeral key that is erased after use. Past messages are protected even if the long-term key is compromised.
Post-compromise security (PCS): If an attacker compromises your key and then you *recover* (e.g., re-key), future messages become secure again. The "Double Ratchet" in Signal continuously rotates keys, so even a temporary compromise does not permanently break security.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Analogy: Forward secrecy is like a self-destructing notepad—each page burns after being read. Post-compromise security is like a phoenix—even if the notepad is stolen, it regenerates with new pages the attacker cannot read.
In MTSF: Forward secrecy (g_{FS}) and post-compromise security (g_{PCS}) are explicit market goods in the Signal analysis. The Double Ratchet's continuous rekeying ensures both goods remain in equilibrium indefinitely, with session pinging confirming unbounded security.
The X3DH-noOPK collapse: Signal's X3DH (Extended Triple Diffie–Hellman) protocol normally uses one-time prekeys (OPKs) for asynchronous establishment. Without OPKs, the asynchronous establishment good g_{async} collapses: an attacker can replay a stale prekey bundle. MTSF detects this as a "replay bid" with $\mathsf{Ask}(g_{\mathsf{async}}) = 1$—market collapse. This dual analysis (secure with OPKs, insecure without) demonstrates MTSF's proof/disproof symmetry.

13.5. Unbounded Verification

Corollary 13 (Unbounded Security of ISO/IEC 11770-3 Protocols)**.** *The ISO/IEC 11770-3 two-party, three-party, and four-party key-exchange protocols are all secure for unbounded sessions. Specifically:*

- ***Two-party:*** *Fresh SIDs sampled per session; nonces N_A, N_B sampled from $\{0,1\}^\lambda$; all signatures bind* sid*. Hence* $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 1$ *with probability* $1 - q_N^2/2^{\lambda+1}$*. By Theorem 3:* $\mathsf{Ask}(g_{\mathsf{key\text{-}secrecy}}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}} + 2\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{negl}$ *for all* $i \geq 1$*.*
- ***Three-party:*** *KDC signatures bind* sid *and N_A; MAC confirmation binds* sid*, N_A, N_B. All pings pass. Unbounded equilibrium holds.*
- ***Four-party:*** *Four independent signatures plus two IND-CCA2 encryptions, all SID-bound. Extended difference lemma gives* $\Pr\left[\bigcup_{i=1}^{4} F_{\mathsf{sig},i}\right] \leq 4\mathsf{Adv}^{\mathsf{EUF}}$*. Ping passes; unbounded equilibrium holds.*
- ***PKI:*** *CA certificates plus session signatures give two-layer identity binding per session. Ping bid price* $\leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + 2\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} = \mathsf{negl}$*. Unbounded equilibrium holds.*

Needham–Schroeder protocol: *No SID binding, no signatures, no nonce-to-session binding. The ping function satisfies* $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0$ *trivially—sessions are structurally indistinct. The masquerade bid succeeds with probability 1 regardless of the session count. The NS market collapses for all sessions.*

☞ Key Insight:

A Note on Unbounded vs. Polynomial Security Game-based proofs guarantee security for *polynomially many* sessions (the adversary's running time is bounded by $\mathsf{poly}(\lambda)$). The session ping mechanism extends this to *all* sessions by induction: if session i is secure and the ping from session i to session $i+1$ passes (probability $\geq 1 - \mathsf{negl}$), then session $i+1$ is secure. By strong induction over all $i \geq 1$, every session is secure. The formal verification tools ProVerif and Tamarin achieve unbounded session coverage using symbolic abstraction; session pinging achieves the same with *quantitative* advantage bounds—the best of both worlds.

✎ Analogy:

PKI as the Postal System of the Internet A Public Key Infrastructure (PKI) is like a global postal system with a twist:

- **Certificate Authority (CA)** = The post office that issues official ID cards. When you present a CA-signed certificate, everyone trusts it because they trust the CA.
- **Public key** = Your home address (anyone can send you mail). **Private key** = The key to your mailbox (only you can read the mail).
- **Certificate** = An official ID card issued by the post office, binding your name to your address. It says: "The post office certifies that Alice lives at Address XYZ."

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Certificate chain** = A chain of trust: your local post office is certified by the regional office, which is certified by the national office. If you trust the national office, you trust everyone in the chain.

In MTSF, the CA's signing key is shared infrastructure (GUC). If the CA's key is compromised (market infrastructure collapses), *all* protocols relying on it collapse simultaneously—this is why the "infrastructure forgery bid" in the bid taxonomy is so dangerous. MTSF's extended difference lemma captures this: F_{CA} (CA compromise) is a single event that affects every protocol in the network.

13.6. PKI-Based Mutual Authentication Protocol

✻ Simple Terms: Public Key Infrastructure (PKI)—A Complete Introduction

The fundamental problem: Alice wants to talk securely to Bob, but she has never met him. She has Bob's public key, but how does she know it actually belongs to Bob? An attacker (Mallory) could have generated her own key pair and told Alice it belongs to Bob. Without some trusted binding between "Bob" and his public key, public-key cryptography is useless.
What PKI provides: A Public Key Infrastructure (PKI) is a system of *Certification Authorities* (CAs)—trusted third parties whose job is to sign *certificates* that bind an identity (e.g. "Bob@example.com") to a public key. A certificate is essentially a signed statement: "I, TrustedCA, certify that the public key pk_B belongs to Bob."
How certificates work:

1. **Key generation:** Bob generates a key pair $(\mathsf{sk}_B, \mathsf{pk}_B)$.
2. **Certificate request (CSR):** Bob sends a Certificate Signing Request to the CA containing his public key and identity information. He signs the CSR with sk_B to prove he owns the corresponding private key.
3. **Vetting:** The CA verifies Bob's identity (e.g., via email ownership, domain control, or in-person for high-assurance CAs).
4. **Certificate issuance:** The CA creates a certificate $\mathsf{cert}_B = (\mathsf{id}_B, \mathsf{pk}_B, \mathsf{valid}_B)$ and signs it: $\sigma_{\mathsf{CA}} = \mathsf{Sign}(\mathsf{sk}_{\mathsf{CA}}, \mathsf{cert}_B)$.
5. **Verification:** Alice, who knows $\mathsf{pk}_{\mathsf{CA}}$, verifies the certificate signature. If it checks out, she knows pk_B genuinely belongs to Bob.

Certificate fields: A real X.509 certificate contains: issuer (CA name), subject (Bob's identity), public key, validity period, serial number, and the CA's signature over all of the above.
Why PKI is everywhere: HTTPS/TLS, S/MIME email, code signing, VPNs, and smart card authentication all use PKI. When you see the padlock in your browser, a PKI certificate chain is being verified in the background.
What the MTSF analysis covers below: We model a TLS-style mutual authentication handshake where both client (C) and server (S) possess certificates issued by a common CA. The protocol establishes a session key with mutual authentication. We prove all five protocol-level market goods.

Protocol description.

Let CA be a Certification Authority with key pair $(\mathsf{sk}_{\mathsf{CA}}, \mathsf{pk}_{\mathsf{CA}})$. Party C (client) holds certificate

$$\mathsf{cert}_C = \big(\mathsf{id}_C,\ \mathsf{pk}_C,\ \mathsf{valid}_C,\ \sigma_{\mathsf{CA},C}\big), \quad \sigma_{\mathsf{CA},C} = \mathsf{Sign}(\mathsf{sk}_{\mathsf{CA}},\ \mathsf{id}_C \| \mathsf{pk}_C \| \mathsf{valid}_C),$$

and party S (server) holds cert_S defined analogously. Both parties know $\mathsf{pk}_{\mathsf{CA}}$ and can verify any CA signature. The **PKI Mutual Authentication Protocol** proceeds as follows:

1. $C \to S$: $\mathsf{sid} \parallel N_C \parallel \mathsf{cert}_C$ *Client hello: session ID, nonce, certificate*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

2. $S \to C$: $\mathsf{sid} \parallel N_S \parallel \mathsf{cert}_S \parallel c \parallel \sigma_S$

$$\sigma_S = \mathsf{Sign}(\mathsf{sk}_S,\ \mathsf{sid} \| N_C \| N_S \| \mathsf{id}_C \| \mathsf{id}_S \| c)$$

Server hello: nonce, certificate, KEM ciphertext, session signature

3. $C \to S$: $\mathsf{sid} \parallel \sigma_C$

$$\sigma_C = \mathsf{Sign}(\mathsf{sk}_C,\ \mathsf{sid} \| N_C \| N_S \| \mathsf{id}_C \| \mathsf{id}_S \| K)$$

Client finished: signature over full session transcript including decapsulated secret

4. **Both derive:** $K_{\mathsf{sess}} = \mathsf{KDF}(K,\ \mathsf{sid} \| N_C \| N_S)$

where c is a KEM encapsulation of secret K under C's public key pk_C (taken from cert_C). Upon receiving Message 2, C first verifies cert_S against $\mathsf{pk}_{\mathsf{CA}}$, extracts pk_S, verifies σ_S, then decapsulates K. Upon receiving Message 3, S verifies cert_C (already received in Message 1) and σ_C.

★ Intuition:

PKI Protocol in Plain English The client announces itself with its CA-issued passport (cert_C). The server presents its own passport (cert_S), wraps a session secret inside a box sealed with the client's public key from that passport (c), and signs the entire exchange to prove it is the genuine certificate holder (σ_S). The client checks both the passport and the signature, decrypts the secret, then signs a receipt to confirm (σ_C). Both check passports against the CA's master verification key. No MITM can forge a passport or a session signature—the CA and the EUF-CMA signature scheme jointly block every impersonation attempt.

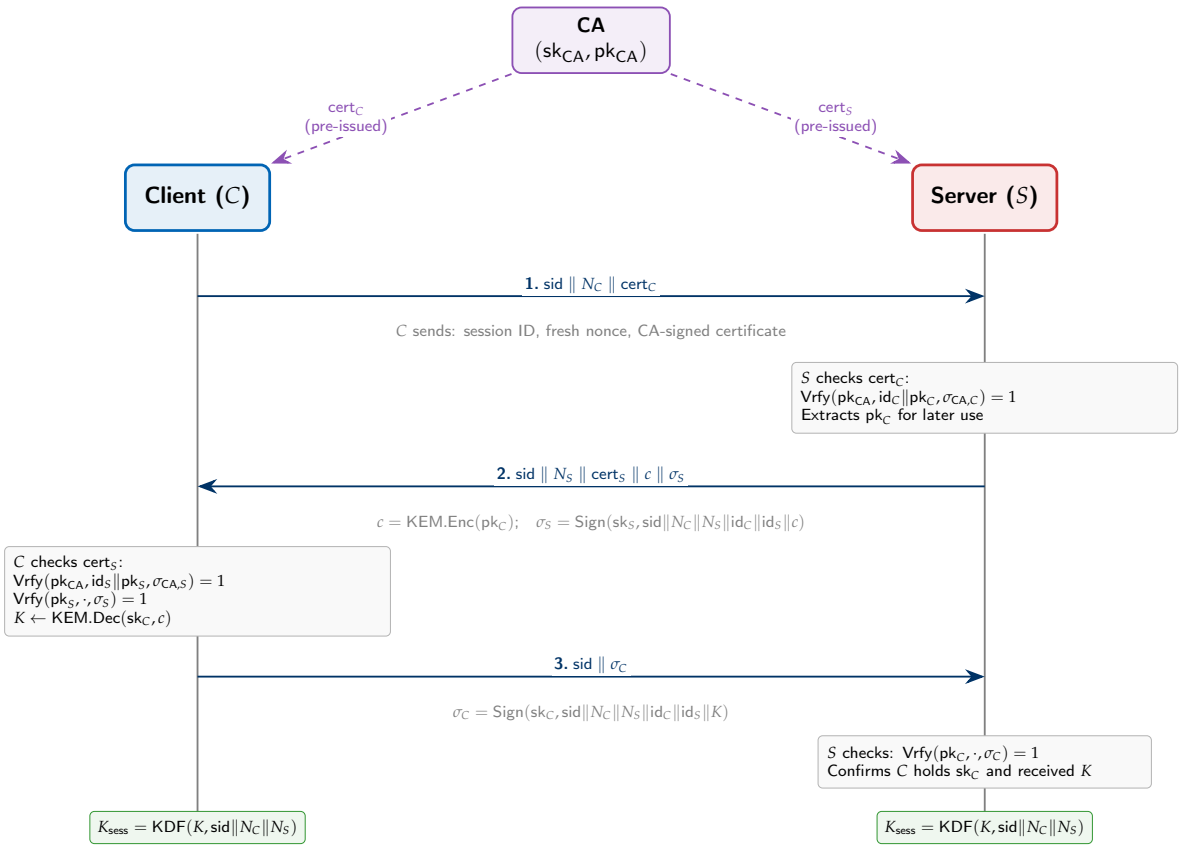


Figure 49. PKI mutual authentication protocol (TLS-style). The CA pre-issues certificates to C and S (dashed arrows, offline). During the three-message handshake: C presents its certificate; S presents its certificate plus a KEM ciphertext and a session signature; C verifies, decapsulates, and signs the finished message. Both parties derive identical K_{sess}. CA certificate verification and SID/nonce binding in every signature prevent all known MITM attacks.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Security goods.

The PKI market $\mathcal{M}_{\mathsf{PKI}}$ offers five goods: g_{cert} (certificate authenticity), g_{auth} (entity authentication), g_{mutual} (mutual authentication), g_{sk} (session-key secrecy), g_{CNF} (CNF correctness).

Definition 33 (Certificate Authenticity Good)**.** *The seller offers g_{cert}: no PPT $\mathcal{A}$ without $\mathsf{sk}_{\mathsf{CA}}$ can produce a tuple $(\mathsf{id}^*, \mathsf{pk}^*, \sigma^*)$ such that $\mathsf{Vrfy}(\mathsf{pk}_{\mathsf{CA}}, \mathsf{id}^* \| \mathsf{pk}^*, \sigma^*) = 1$ and $(\mathsf{id}^*, \mathsf{pk}^*)$ was never submitted to the CA's signing oracle.*

✻ Simple Terms: Certificate Authenticity

What is guaranteed: No attacker can forge a CA certificate binding a victim's identity to the attacker's own key. Even after observing many real certificates, the attacker cannot produce a valid CA signature on a new identity–key pair it did not legitimately obtain.
Why it is EUF-CMA: A forged certificate $(\mathsf{id}^*, \mathsf{pk}^*, \sigma^*)$ is exactly an EUF-CMA forgery: a valid signature on a new message $\mathsf{id}^* \| \mathsf{pk}^*$ that was never submitted to the signing oracle (the CA).
Practical implication: Mallory cannot get a certificate saying "I am Bob" unless she compromises the CA or Bob's private key. This is why browser vendors maintain trusted root CA lists and revoke CAs that mis-issue certificates (as happened in several real-world incidents with DigiNotar, Comodo, etc.).

Lemma 3 (Certificate Authenticity Equilibrium)**.** $\mathsf{Ask}(g_{\mathsf{cert}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig},\mathsf{CA}} + \mathsf{negl}(\lambda)$.

Proof. Any buyer producing $(\mathsf{id}^*, \mathsf{pk}^*, \sigma^*)$ supplies an EUF-CMA forgery on message $\mathsf{id}^* \| \mathsf{pk}^*$ under $\mathsf{sk}_{\mathsf{CA}}$. The reduction embeds the CA's public key and forwards signing queries. The forgery is new (the CA never issued this certificate) so EUF-CMA applies directly. □

Theorem 52 (PKI Protocol Market Equilibrium)**.** *Under EUF-CMA security of the CA and party signature schemes, IND-CCA2 security of the KEM, and PRF security of the KDF:*

$$\mathsf{Ask}(g_{\mathsf{cert}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{negl}, \tag{134}$$

$$\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \tfrac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}, \tag{135}$$

$$\mathsf{Ask}(g_{\mathsf{mutual}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \tfrac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}, \tag{136}$$

$$\mathsf{Ask}(g_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}} + \tfrac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}, \tag{137}$$

$$\mathsf{Ask}(g_{\mathsf{CNF}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \tfrac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}. \tag{138}$$

All five goods are in equilibrium, so $\mathcal{M}_{\mathsf{PKI}}$ is in equilibrium.

Proof. We abbreviate $\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}}$ and $\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}$ throughout.

Bound (134)**:** Lemma 3.

Bound (135) **(server → client authentication).** Five games:

$\mathbf{B}_0$ **(Real).** $\mathcal{A}$ controls the network. Wins if C accepts S as partner but S did not run a matching session.

$\mathbf{B}_1$ **(Nonce uniqueness).** Abort on nonce collision among q_N nonces: $|\Pr[\mathbf{B}_0{=}1] - \Pr[\mathbf{B}_1{=}1]| \leq q_N^2/2^{\lambda+1}$.

$\mathbf{B}_2$ **(Certificate forgery bid).** Flag F_{cert}: $\mathcal{A}$ presents a certificate for id_S that verifies under $\mathsf{pk}_{\mathsf{CA}}$ but was not issued by the CA. By Lemma 3: $|\Pr[\mathbf{B}_1{=}1] - \Pr[\mathbf{B}_2{=}1]| \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}}$. After $\mathbf{B}_2$, any certificate C accepts for S was legitimately issued, so pk_S is correctly bound to id_S.

$\mathbf{B}_3$ **(Signature forgery bid on S).** Flag $F_{\mathsf{forge_S}}$: C accepts σ_S not generated by S. Reducing to EUF-CMA on sk_S (the key bound by the verified certificate): $|\Pr[\mathbf{B}_2{=}1] - \Pr[\mathbf{B}_3{=}1]| \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\mathbf{B}_4$ **(Ideal).** No cert forgery, no sig forgery: C accepted only because a valid σ_S bound $\mathsf{sid}, N_C, N_S, \mathsf{id}_C, \mathsf{id}_S, c$ under S's genuine key. S must have participated. $\Pr[\mathbf{B}_4{=}1] = 0$.

Total: $\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$.

Bound (136) **(mutual authentication).** Add a game hop for $C \to S$ direction: S verifies cert_C (same CA forgery argument) and σ_C (EUF-CMA on sk_C). The three failure events $F_{\mathsf{cert}}, F_{\mathsf{forge_S}}, F_{\mathsf{forge_C}}$ are handled by the extended difference lemma; $F_{\mathsf{forge_S}}$ and $F_{\mathsf{forge_C}}$ use independent keys, so: $\Pr[F_{\mathsf{forge_S}} \cup F_{\mathsf{forge_C}}] \leq 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}$. Equation (136) follows.

Bound (137) **(session-key secrecy).** Extend with two hops following Theorem 8: $\mathbf{B}_5$: Replace KEM key K with uniform $\tilde{K}$; costs $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}$. $\mathbf{B}_6$: Replace $\mathsf{KDF}(\tilde{K}, \ldots)$ with uniform random; costs $\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}}$. After $\mathbf{B}_6$ the session key is independent of all observed messages; $\Pr[\mathbf{B}_6{=}1] = 1/2$. Summing gives Equation (137).

Bound (138) **(CNF correctness).** The PKI session-CNF is:

$$\varphi_{\mathsf{PKI}} \;=\; \varphi_C^{\mathsf{cert}} \;\wedge\; \varphi_S^{\mathsf{cert}} \;\wedge\; \varphi^{\mathsf{sid}} \;\wedge\; \varphi^{\mathsf{fresh}} \;\wedge\; \varphi_S^{\mathsf{sig}} \;\wedge\; \varphi_C^{\mathsf{sig}} \;\wedge\; \varphi^{\mathsf{consist}},$$

where $\varphi_P^{\mathsf{cert}} = \left(x_{\mathsf{Vrfy}(\mathsf{pk}_{\mathsf{CA}}, \mathsf{id}_P \| \mathsf{pk}_P, \sigma_{\mathsf{CA},P})=1}\right)$ for $P \in \{C, S\}$. Under an honest trace all clauses are satisfied. A dishonest trace satisfies the formula only if: (a) a certificate clause passes dishonestly (requires CA forgery), or (b) a signature clause passes dishonestly (requires EUF-CMA break), or (c) a freshness clause passes dishonestly (requires nonce reuse). By the extended difference lemma over these three event types, Equation (138) follows.

$\mathbf{B}_{\mathsf{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

☞ **Side-Channel Bid (Comprehensive)**

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\mathsf{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\mathsf{SPA}}] = 1$ without countermeasure; $\Pr[F_{\mathsf{SPA}}] = 0$ with constant-power implementation.
2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\mathsf{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking ($d{+}1$ random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\mathsf{Adv}^{\mathsf{DPA}}(\lambda, d) \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
3. **Timing Analysis:** $\mathcal{A}_{\mathsf{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\mathsf{ct_select}(a, b, \mathsf{bit})$ instead of $\mathsf{if}(\mathsf{bit})$ return a; else return b;. *MTSF price:* $\Pr[F_{\mathsf{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\mathsf{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\mathsf{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\mathsf{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\mathsf{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\mathsf{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\mathsf{Adv}^{\mathsf{EMA}} \leq \gamma_{\mathsf{EM}} \cdot (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\mathsf{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\mathsf{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\mathsf{fault}}] \leq q_{\mathsf{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).
8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into $d+1$ uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \leq\ & \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ & + \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

Under the comprehensive countermeasure suite (order-d masking for $d \geq 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \leq\ & \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{acoustic}} \\ & + \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}}_{\mathsf{photonic}}.\end{aligned}$$

For $d \geq 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\leq q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

✓ Tight Security Reduction

Tightness analysis. The PKI-based mutual authentication proof reduces to EUF-CMA of the underlying signature scheme with tightness loss $L = 2n_P$ (factor 2 for mutual authentication, n_P for party-guessing). Session-key secrecy reduces tightly ($L = 1$) to the PRF security of the key derivation function.

Market interpretation: The PKI proof's tightness loss is concentrated in the authentication phase (party-guessing cost), while the key-derivation phase is "wholesale" (tight, $L = 1$).

Total Session Bound and Key Rotation Recommendation.

✦ Total Session Bound & Key Rotation

Total session bound: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \leq N \cdot (2n_P \cdot \mathsf{Adv}^{\mathsf{EUF}} + \mathsf{Adv}^{\mathsf{PRF}} + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}})$.
Concrete (ECDSA-P256, HKDF-SHA-256, $n_P = 2^{16}$):

Sessions N	$\mathsf{Ask}^{(N)}_{\mathsf{total}}$	**Eff. security**	**Action**
2^{40}	$\leq 2^{-72}$	≥ 72 bits	No rotation needed
2^{56}	$\leq 2^{-56}$	≥ 56 bits	**Certificate rotation recommended**
2^{64}	$\leq 2^{-48}$	≥ 48 bits	**Certificate rotation mandatory**

Recommendation: Rotate PKI certificates before 2^{56} authentication sessions (aligning with standard 1–3 year certificate validity periods).

Session pinging and CNF checking within the proof. For consecutive PKI sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$, the ping mechanism verifies a three-layer structural distinctness:

Layer 1 (CA certificates): The certificates cert_C and cert_S are pre-issued and shared across sessions. However, each session's signatures σ_C and σ_S bind the session-specific sid_{i+1}, nonces $N_{C,i+1}$ and $N_{S,i+1}$, and both party identities. This SID-binding ensures that certificate possession alone is insufficient to replay a session.

Layer 2 (Session signatures): The server signature $\sigma_{S,i+1} = \mathsf{Sign}(\mathsf{sk}_S, \mathsf{sid}_{i+1} \| N_{C,i+1} \| N_{S,i+1} \| \mathsf{id}_C \| \mathsf{id}_S \| c_{i+1})$ includes all five ping-relevant fields. Replaying $\sigma_{S,j}$ from $\mathsf{Session}_j$ in $\mathsf{Session}_{i+1}$ fails verification because $\mathsf{sid}_j \neq \mathsf{sid}_{i+1}$. Similarly for $\sigma_{C,i+1}$.

Layer 3 (KEM ciphertext): The ciphertext $c_{i+1} = \mathsf{KEM.Enc}(\mathsf{pk}_C; r_{i+1})$ uses fresh randomness, ensuring $c_{i+1} \neq c_j$ for all $j \leq i$ with overwhelming probability.

The extended difference lemma captures three simultaneous forgery events: F_{cert} (CA certificate forgery), $F_{\mathsf{sig},S}$ (server signature forgery), and $F_{\mathsf{sig},C}$ (client signature forgery). By Lemma 2: $\Pr\left[F_{\mathsf{cert}} \cup F_{\mathsf{sig},S} \cup F_{\mathsf{sig},C}\right] \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + 2\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} = \mathsf{negl}$. The session-CNF $\varphi_{\mathsf{PKI},i+1}$ is isomorphic to $\varphi_{\mathsf{PKI},i}$ with fresh SID, nonces, ciphertext, and signatures. □

✻ Simple Terms: PKI Protocol Security Proof—Reading Each Bound

Bound (134) **(certificate authenticity):** Only breakable by forging a CA signature. Real CAs use 256-bit ECDSA or 3072-bit RSA-PSS: probability $\approx 2^{-128}$.
Bound (135) **(server authentication):** The server is authenticated by two layers: the CA certificate (binds its identity to its key) and its protocol signature (binds the key to this specific session). An attacker must break *either* layer; both are negligible.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Bound (136) **(mutual authentication):** Adds the client direction. Factor of $2 \times \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}$ comes from two independent signing keys—both must remain unforgeable simultaneously.
Bound (137) **(session-key secrecy):** Requires breaking four independent security properties: CA signing, party signing ($\times 2$), KEM, and KDF. Security compounds: the attacker must clear all four hurdles simultaneously.
Why this defeats Lowe-style MITM: When Mallory tries to relay C's certificate to S while impersonating C, S will demand a valid σ_C under pk_C (extracted from that very certificate). Mallory lacks sk_C, so she cannot produce σ_C. The session-CNF clause φ^{sig}_C immediately fails. Market equilibrium is preserved.

CNF clause breakdown.

✻ Simple Terms: The PKI Session-CNF Checklist—Clause by Clause

The PKI session-CNF has seven groups of clauses:

1. $\varphi^{\mathsf{cert}}_C$**:** The client's certificate verifies under $\mathsf{pk}_{\mathsf{CA}}$. Blocks impersonation of C using a fake certificate.
2. $\varphi^{\mathsf{cert}}_S$**:** The server's certificate verifies under $\mathsf{pk}_{\mathsf{CA}}$. Blocks impersonation of S using a fake certificate.
3. φ^{sid}**:** Every message carries the same sid. Blocks cross-session replay.
4. φ^{fresh}**:** Nonces N_C, N_S are fresh and distinct. Blocks full-session replay.
5. φ^{sig}_S**:** Server's σ_S verifies under pk_S (from cert_S) over the full session data including c. Proves S generated the KEM ciphertext for this session.
6. φ^{sig}_C**:** Client's σ_C verifies under pk_C (from cert_C) over session data including K. Proves C holds sk_C and successfully decapsulated K.
7. $\varphi^{\mathsf{consist}}$**:** Messages arrive in the correct order ($C \to S$, $S \to C$, $C \to S$) with no gaps. Enforces the protocol state machine.

Unique feature vs. ISO 11770-3 protocols: The two certificate clauses ($\varphi^{\mathsf{cert}}_C$, $\varphi^{\mathsf{cert}}_S$) are new—the ISO protocols assume parties already have each other's keys. PKI externalises the identity binding to the CA and captures it explicitly in the CNF, making the trust model fully transparent and formally auditable.

Corollary 14 (PKI Unbounded Security)**.** *The PKI protocol is secure for unbounded sessions: each session samples a fresh* sid *and nonces N_C, N_S; all signatures include* sid*; hence* $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 1$ *for all i. By Theorem 3, equilibrium holds for all $i \geq 1$.*

Manual CNF worksheet (PKI TLS-style session $\mathsf{Session}_i$***):***

Clause	*Check*	*How*	*Pass?*
$\varphi^{\mathsf{cert}}_C$	*C's cert verifies under* $\mathsf{pk}_{\mathsf{CA}}$*?*	$\mathsf{Vrfy}(\mathsf{pk}_{\mathsf{CA}}, \mathsf{id}_C \Vert \mathsf{pk}_C, \sigma_{\mathsf{CA},C})$.	*T/F*
$\varphi^{\mathsf{cert}}_S$	*S's cert verifies under* $\mathsf{pk}_{\mathsf{CA}}$*?*	$\mathsf{Vrfy}(\mathsf{pk}_{\mathsf{CA}}, \mathsf{id}_S \Vert \mathsf{pk}_S, \sigma_{\mathsf{CA},S})$.	*T/F*
φ^{sid}	*Same* sid *in all 3 messages?*	*Read SID field; compare.*	*T/F*
φ^{fresh}	*$N_C \neq N_S$; neither used before?*	*Check nonce log.*	*T/F*
φ^{sig}_S	$\mathsf{Vrfy}(\mathsf{pk}_S, \mathsf{sid} \Vert N_C \Vert N_S \Vert \mathsf{id}_C \Vert \mathsf{id}_S \Vert c, \sigma_S) = 1$*?*	*Hash scope; verify.*	*T/F*
φ^{sig}_C	$\mathsf{Vrfy}(\mathsf{pk}_C, \mathsf{sid} \Vert N_C \Vert N_S \Vert \mathsf{id}_C \Vert \mathsf{id}_S \Vert K, \sigma_C) = 1$*?*	*Hash scope; verify.*	*T/F*
$\varphi^{\mathsf{consist}}$	*Msg order: $C \to S$, $S \to C$, $C \to S$?*	*Check sender/direction.*	*T/F*
φ^{ping}	$\mathsf{sid}_i \neq \mathsf{sid}_{i-1}$*; $N_C, N_S \notin \mathcal{N}_{\mathsf{prev}}$?*	*Compare session logs.*	*T/F*
Result:	*All $T \Rightarrow$ Accept. Any $F \Rightarrow$ Reject (cite failed clause).*		

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

➤ Ping Bid: PKI Unbounded Ping Bid

The buyer places an *impersonation+replay ping bid*: use old transcript with a new SID. Cannot forge σ_C or σ_S under new SID (EUF-CMA). Extended difference lemma captures three simultaneous failures: F_{cert} (CA forgery), $F_{\mathsf{sig},S}$ (server sig forgery), $F_{\mathsf{sig},C}$ (client sig forgery). By Lemma 2: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + 2\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} = \mathsf{negl}$. PKI is secure for unbounded sessions.

Table 14. PKI protocol market goods summary.

Good	Ask Price Bound	Status
g_{cert} (certificate authenticity)	$\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}}$	Equilibrium
g_{auth} (entity auth)	$\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$	Equilibrium
g_{mutual} (mutual auth)	$\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$	Equilibrium
g_{sk} (key secrecy)	$+ \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}}$	Equilibrium
g_{CNF} (CNF correctness)	$\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$	Equilibrium

☞ Key Insight:

PKI vs. Needham–Schroeder: Security Comparison

Property	Needham–Schroeder	PKI Protocol
Identity binding	None in Msg. 2 (fatal)	CA cert in every message
Mutual auth ask	$= 1$ (collapsed)	$\leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + 2\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{negl}$
MITM blocked?	No	Yes (cert + sig binding)
CNF audit	Dishonest trace PASSES	Dishonest trace FAILS
Extra trust needed	None (but broken)	CA (offline, pre-established)
Unbounded sessions	Pinging fails	Pinging succeeds

13.7. Signal Protocol: X3DH and Double Ratchet

The Signal Protocol [49,50] is the most widely deployed end-to-end encrypted messaging protocol, powering Signal, WhatsApp, Facebook Messenger, and Google Messages. It consists of two sub-protocols: the *Extended Triple Diffie–Hellman* (X3DH) key agreement protocol for session establishment, and the *Double Ratchet* algorithm for ongoing message encryption with forward secrecy and post-compromise security. We analyse both sub-protocols within the MTSF framework, proving security of the full protocol and disproving security of a weakened variant lacking one-time prekeys.

✻ Simple Terms: The Signal Protocol—A Complete Introduction

What is the Signal Protocol? It is the cryptographic engine behind the world's most popular encrypted messaging apps. Designed by Trevor Perrin and Moxie Marlinspike (Open Whisper Systems, now Signal Foundation), it solves two hard problems simultaneously:

1. **Asynchronous key establishment (X3DH):** Alice wants to send a secure message to Bob, but Bob is offline. She cannot perform an interactive handshake. X3DH allows Alice to compute a shared secret using Bob's *prekeys* (public values Bob published to a server in advance), without any real-time interaction.
2. **Ongoing key evolution (Double Ratchet):** Once the initial key is established, every single message is encrypted with a *unique* key that is derived from a continuously evolving state. Even if an adversary compromises a key at one point, they cannot read past messages (*forward secrecy*) and they lose access to future messages once the parties exchange new Diffie–Hellman values (*post-compromise security*).

Key types in Signal:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Identity key (IK):** A long-term Curve25519 key pair. Identifies the user. Never changes (unless re-installed).
- **Signed prekey (SPK):** A medium-term key pair signed by the identity key. Rotated periodically (e.g., every 1–4 weeks).
- **One-time prekey (OPK):** An ephemeral key uploaded to the server in batches. Each is used in exactly one X3DH session and then deleted. Provides fresh randomness per session.
- **Ephemeral key (EK):** Generated fresh by Alice for each new session. Never stored or reused.

Why the Double Ratchet is revolutionary:

- **Symmetric ratchet (KDF chain):** Each message advances a KDF chain: $\mathsf{CK}_{n+1} = \mathsf{KDF}(\mathsf{CK}_n, \texttt{0x02})$ and $\mathsf{MK}_n = \mathsf{KDF}(\mathsf{CK}_n, \texttt{0x01})$. Old chain keys are deleted, providing forward secrecy per message.
- **Diffie–Hellman ratchet:** Each message includes a fresh DH public key. When both sides have exchanged new keys, a DH ratchet step produces a new root key, seeding fresh sending and receiving chains. This provides post-compromise security: if an attacker learns the current state, they lose access after the next DH exchange.
- **Out-of-order tolerance:** Skipped message keys are cached (up to a limit), allowing decryption of delayed messages without breaking the ratchet.

Why this matters for MTSF: The Signal Protocol provides the richest set of security goods of any protocol we analyse: confidentiality, authenticity, forward secrecy, post-compromise security, deniability, and asynchronous session establishment. Analysing it within MTSF demonstrates the framework's expressive power.

X3DH protocol description.

Let Alice (initiator) and Bob (responder) each hold an identity key pair: $(\mathsf{ik}_A, \mathsf{IK}_A)$ and $(\mathsf{ik}_B, \mathsf{IK}_B)$ respectively, where lowercase denotes private and uppercase denotes public keys over Curve25519. Bob publishes to the server a *prekey bundle*:

$$\mathsf{PKB} = \left(\mathsf{IK}_B,\ \mathsf{SPK}_B,\ \sigma_B = \mathsf{XEdDSA}(\mathsf{ik}_B, \mathsf{SPK}_B),\ \mathsf{OPK}_B^{(j)}\right),$$

where SPK_B is the signed prekey, σ_B is the identity signature over SPK_B, and $\mathsf{OPK}_B^{(j)}$ is an optional one-time prekey.

Alice fetches Bob's prekey bundle, generates an ephemeral key pair $(\mathsf{ek}_A, \mathsf{EK}_A)$, and computes:

$$\mathsf{DH}_1 = \mathsf{X25519}(\mathsf{ik}_A, \mathsf{SPK}_B), \tag{139}$$

$$\mathsf{DH}_2 = \mathsf{X25519}(\mathsf{ek}_A, \mathsf{IK}_B), \tag{140}$$

$$\mathsf{DH}_3 = \mathsf{X25519}(\mathsf{ek}_A, \mathsf{SPK}_B), \tag{141}$$

$$\mathsf{DH}_4 = \mathsf{X25519}(\mathsf{ek}_A, \mathsf{OPK}_B^{(j)}), \tag{142}$$

$$\mathsf{SK} = \mathsf{HKDF}\left(\mathsf{DH}_1 \| \mathsf{DH}_2 \| \mathsf{DH}_3 \| \mathsf{DH}_4\right). \tag{143}$$

Alice sends to Bob: $(\mathsf{IK}_A, \mathsf{EK}_A, j, \mathsf{ct}_0)$ where j identifies the one-time prekey used and ct_0 is the initial ciphertext encrypted under SK. Bob performs the same four DH computations using his private keys, derives the same SK, and decrypts ct_0. Both then initialise the Double Ratchet with SK as the root key.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

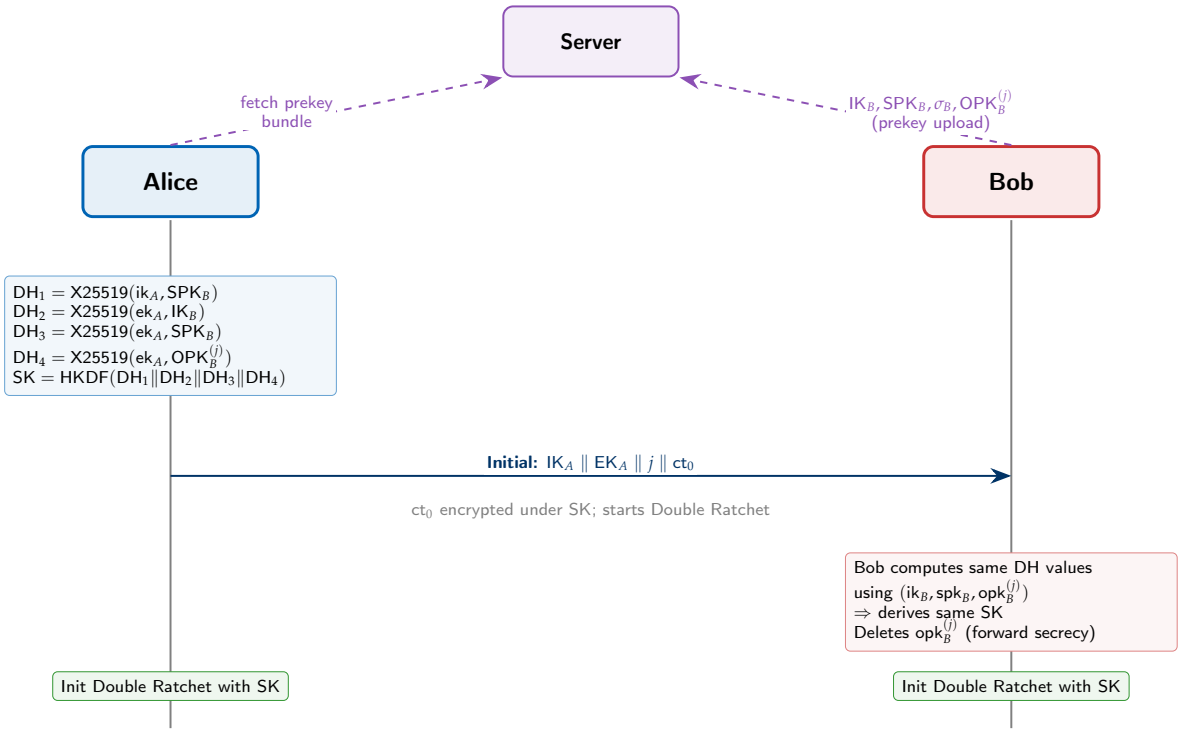


Figure 50. X3DH key agreement protocol (Signal). Alice computes four DH values using Bob's prekey bundle. The one-time prekey $\mathsf{OPK}_B^{(j)}$ provides per-session freshness; its deletion after use ensures forward secrecy of the initial key establishment.

Double Ratchet description.

After X3DH, both parties initialise a *Double Ratchet* state consisting of: a root key RK, a sending chain key CK_s, a receiving chain key CK_r, and a DH ratchet key pair $(\mathsf{dh}_s, \mathsf{DH}_s)$. Each message triggers two operations:

Symmetric ratchet: To encrypt message n, advance the sending chain:

$$(\mathsf{CK}_{s,n+1},\ \mathsf{MK}_n) = \mathsf{KDF}(\mathsf{CK}_{s,n}), \qquad \mathsf{ct}_n = \mathsf{AEAD}_{\mathsf{MK}_n}(\mathsf{pt}_n; \mathsf{AD}), \tag{144}$$

where AD (associated data) includes both identity keys and message headers. Delete $\mathsf{CK}_{s,n}$ and MK_n after use.

DH ratchet: When receiving a message with a new DH public key DH'_r:

$$\begin{aligned} \mathsf{RK}',\ \mathsf{CK}'_r &= \mathsf{KDF}\big(\mathsf{RK},\ \mathsf{X25519}(\mathsf{dh}_s, \mathsf{DH}'_r)\big), \\ (\mathsf{dh}'_s, \mathsf{DH}'_s) &\leftarrow \mathsf{KeyGen}(), \\ \mathsf{RK}'',\ \mathsf{CK}'_s &= \mathsf{KDF}\big(\mathsf{RK}',\ \mathsf{X25519}(\mathsf{dh}'_s, \mathsf{DH}'_r)\big). \end{aligned} \tag{145}$$

Delete old RK, dh_s, and chain keys. The fresh DH exchange provides post-compromise security.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

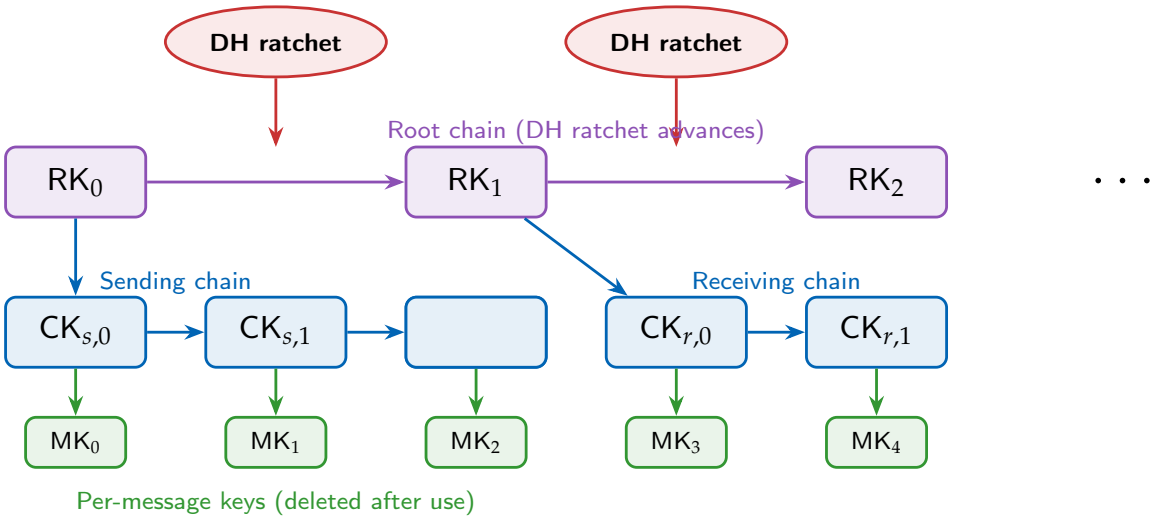


Figure 51. Double Ratchet structure. The root chain (top) advances with each DH ratchet step. Each root key seeds sending and receiving KDF chains, which produce per-message keys. Deletion of old keys provides forward secrecy; fresh DH exchanges provide post-compromise security.

★ Intuition:

The Double Ratchet as a Self-Healing Lock Imagine a combination lock that changes its combination after every use. Even if someone sees the current combination (key compromise), the lock automatically changes on the next turn (symmetric ratchet = forward secrecy). Moreover, periodically both parties replace the entire lock mechanism itself (DH ratchet = post-compromise security). The attacker must steal the new lock to continue eavesdropping—and since the new lock is generated from fresh randomness, the attacker's knowledge is erased.

MTSF market model.

The Signal market $\mathcal{M}_{\mathsf{Signal}}$ offers six security goods:

- g_{conf}: Message confidentiality (IND-CCA2 of each message under its unique key).
- g_{auth}: Message authentication (INT-CTXT via AEAD for each message).
- g_{FS}: Forward secrecy—compromise of current state does not reveal past message keys.
- g_{PCS}: Post-compromise security—after compromise, security self-heals upon the next DH ratchet step.
- g_{async}: Asynchronous key establishment—Alice can send a message to offline Bob.
- g_{deny}: Deniability—no party can produce a cryptographic proof of the conversation to a third party.

Theorem 53 (Signal Protocol Market Equilibrium)**.** *Under the gap Diffie–Hellman (GDH) assumption on Curve25519 and the PRF security of HKDF-SHA-256:*

$$\mathsf{Ask}(g_{\mathsf{conf}}) \le 4 \cdot \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + \mathsf{Adv}^{\mathsf{IND}}_{\mathsf{AEAD}} + \mathsf{negl}(\lambda), \tag{146}$$

$$\mathsf{Ask}(g_{\mathsf{auth}}) \le 4 \cdot \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + \mathsf{Adv}^{\mathsf{INT}}_{\mathsf{AEAD}} + \mathsf{negl}(\lambda), \tag{147}$$

$$\mathsf{Ask}(g_{\mathsf{FS}}) \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + \mathsf{negl}(\lambda), \tag{148}$$

$$\mathsf{Ask}(g_{\mathsf{PCS}}) \le \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + \mathsf{negl}(\lambda). \tag{149}$$

Proof. We construct eight bidding rounds, covering X3DH key establishment and Double Ratchet messaging.

$\mathbf{B}_0$ (Bidding Round 0: Real Signal Market).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

The seller initialises the Signal Protocol with X3DH key agreement followed by Double Ratchet messaging. Alice generates $(\mathsf{ik}_A, \mathsf{IK}_A)$ and $(\mathsf{ek}_A, \mathsf{EK}_A)$; Bob's prekey bundle

$$(\mathsf{IK}_B, \mathsf{SPK}_B, \sigma_B, \mathsf{OPK}_B^{(j)})$$

is available on the server. The buyer (active network adversary) can observe, delay, reorder, or inject messages.

$\mathbf{B}_1$ (Bidding Round 1: Prekey Signature Forgery Bid).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathsf{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

The buyer attempts to forge Bob's prekey signature σ_B on a malicious SPK'_B. This would allow the buyer to impersonate Bob's server-published keys, directing Alice to perform X3DH with buyer-controlled values.

Difference lemma. Let F_{spk}: "buyer forges σ_B on $\mathsf{SPK}'_B \neq \mathsf{SPK}_B$." Since

$$\sigma_B = \mathsf{XEdDSA}(\mathsf{ik}_B, \mathsf{SPK}_B)$$

, this requires breaking EUF-CMA of XEdDSA:

$$\Delta\mathsf{Price}_0 \leq \Pr[F_{\mathsf{spk}}] \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{XEdDSA}} \leq \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}}. \tag{150}$$

$\mathbf{B}_2$ (Bidding Round 2: X3DH $\mathbf{DH}_1$ Extraction Bid).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathrm{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

The buyer attempts to recover $\mathsf{DH}_1 = \mathsf{X25519}(\mathsf{ik}_A, \mathsf{SPK}_B)$ from the observed transcript. This is a static-static DH: the buyer must solve the Computational Diffie–Hellman (CDH) problem on $(\mathsf{IK}_A, \mathsf{SPK}_B)$.

Difference lemma. F_{DH_1}: "buyer computes DH_1." $\Delta\mathsf{Price}_1 \leq \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}}$.

$\mathbf{B}_3$ (Bidding Round 3: X3DH $\mathbf{DH}_2$–$\mathbf{DH}_3$ Extraction Bid).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathrm{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

The buyer attempts to recover $\mathsf{DH}_2 = \mathsf{X25519}(\mathsf{ek}_A, \mathsf{IK}_B)$ or $\mathsf{DH}_3 = \mathsf{X25519}(\mathsf{ek}_A, \mathsf{SPK}_B)$. Both involve Alice's ephemeral key ek_A, known only to Alice. CDH on $(\mathsf{EK}_A, \mathsf{IK}_B)$ or $(\mathsf{EK}_A, \mathsf{SPK}_B)$.

Extended difference lemma. Let F_{DH_2}: buyer computes DH_2; F_{DH_3}: buyer computes DH_3. By Lemma 2:

$$\Delta\mathsf{Price}_2 \leq \Pr\left[F_{\mathsf{DH}_2} \cup F_{\mathsf{DH}_3}\right] \leq 2 \cdot \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}}. \tag{151}$$

$\mathbf{B}_4$ (Bidding Round 4: One-Time Prekey Freshness Bid—$\mathbf{DH}_4$).

Purpose: Eliminate the *nonce-reuse attack vector*. Repeated nonces immediately collapse authentication, key-secrecy, or stream-cipher security: in signatures a shared nonce leaks the private key ($\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); in protocols it enables cross-session replay; in stream ciphers it reveals the XOR of two plaintexts. This round aborts the game whenever any nonce collision occurs, ensuring every subsequent hop reasons about a nonce-fresh world.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: The real nonce-sampling procedure is augmented with a global *collision audit*: a set $\mathcal{N}_{\text{used}}$ records every nonce issued so far. If any newly sampled nonce k' satisfies $k' \in \mathcal{N}_{\text{used}}$, the game aborts immediately. The two games ($\mathbf{B}_{n-1}$ and $\mathbf{B}_n$) are *identical-until-abort*: the adversary's view is indistinguishable until the abort event F_{nonce} occurs, so the difference lemma directly bounds the price adjustment.
Complexity: By the birthday bound, the probability that any two of q independently sampled nonces from a space of size $|\mathcal{N}|$ collide is at most $q^2/(2|\mathcal{N}|)$. For security parameter λ with $|\mathcal{N}| = 2^\lambda$ this gives $\Delta\text{Price} \leq q^2/2^{\lambda+1}$, which is negligible for any polynomial $q = \text{poly}(\lambda)$. After this hop every nonce in the proof is treated as distinct, allowing all later hops to assume a collision-free nonce space without loss of generality.

The buyer attempts to recover $\mathsf{DH}_4 = \mathsf{X25519}(\mathsf{ek}_A, \mathsf{OPK}_B^{(j)})$. Since $\mathsf{OPK}_B^{(j)}$ is used exactly once and then deleted, the buyer must solve CDH on a one-time value. Deletion after use ensures that even server compromise after the session does not reveal DH_4.

$\Delta\text{Price}_3 \leq \text{Adv}^{\text{GDH}}_{\text{X25519}}$.

$\mathbf{B}_5$ (Bidding Round 5: HKDF PRF Bid—SK Extraction).

Purpose: Replace a *keyed pseudorandom function* evaluation with a truly random function, isolating a single PRF security requirement. This hop shows that one more step in the key-derivation or authentication structure produces outputs that are computationally indistinguishable from uniform random, provided the underlying PRF assumption holds.
Replaces: The specific PRF evaluation (e.g., $\text{HMAC}_K(\cdot)$ inner hash, HKDF-Extract, Derive-Secret, or a KDF step) is replaced by a truly random function $R(\cdot)$ sampled fresh and independently. Any adversary that distinguishes the PRF output from this random function is a PRF distinguisher for the underlying keyed construction. The reduction simulates all other game components honestly and uses its PRF challenge oracle for this one step.
Complexity: $\Delta\text{Price} \leq \text{Adv}^{\text{PRF}}(\lambda)$ for the specific PRF being replaced (HMAC-SHA-256, HKDF, or AES-CTR as applicable). When multiple PRF hops appear in a single proof (e.g., three hops in the TLS 1.3 key schedule for HS, MS, and CATS), each hop is independent: the keys used in different steps are derived from different points in the ladder, so no PRF oracle is queried twice. The total across all PRF hops is the sum of individual advantages, each negligible.

With all four DH values hidden, $\mathsf{SK} = \mathsf{HKDF}(\mathsf{DH}_1 \| \cdots \| \mathsf{DH}_4)$ is pseudorandom. The buyer attempts to distinguish SK from random.

$\Delta\text{Price}_4 \leq \text{Adv}^{\text{PRF}}_{\text{HKDF}}$.

$\mathbf{B}_6$ (Bidding Round 6: Symmetric Ratchet Bid—KDF Chain Inversion).

Purpose: Replace a *keyed pseudorandom function* evaluation with a truly random function, isolating a single PRF security requirement. This hop shows that one more step in the key-derivation or authentication structure produces outputs that are computationally indistinguishable from uniform random, provided the underlying PRF assumption holds.
Replaces: The specific PRF evaluation (e.g., $\text{HMAC}_K(\cdot)$ inner hash, HKDF-Extract, Derive-Secret, or a KDF step) is replaced by a truly random function $R(\cdot)$ sampled fresh and independently. Any adversary that distinguishes the PRF output from this random function is a PRF distinguisher for the underlying keyed construction. The reduction simulates all other game components honestly and uses its PRF challenge oracle for this one step.
Complexity: $\Delta\text{Price} \leq \text{Adv}^{\text{PRF}}(\lambda)$ for the specific PRF being replaced (HMAC-SHA-256, HKDF, or AES-CTR as applicable). When multiple PRF hops appear in a single proof (e.g., three hops in the TLS 1.3 key schedule for HS, MS, and CATS), each hop is independent: the keys used in different steps are derived from different points in the ladder, so no PRF oracle is queried twice. The total across all PRF hops is the sum of individual advantages, each negligible.

The buyer observes ciphertexts and attempts to recover earlier chain keys from later ones. The KDF chain $\mathsf{CK}_{n+1} = \mathsf{KDF}(\mathsf{CK}_n)$ is one-way: given CK_{n+1}, recovering CK_n requires inverting the PRF. This is the *forward secrecy bid*—it fails because KDF is a PRF.

$\Delta\text{Price}_5 \leq \text{Adv}^{\text{PRF}}_{\text{HKDF}}$. This establishes $\text{Ask}(g_{\text{FS}}) \leq \text{Adv}^{\text{PRF}}_{\text{HKDF}} + \text{negl}$.

$\mathbf{B}_7$ (Bidding Round 7: DH Ratchet Bid—Post-Compromise Recovery).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**DH Ratchet Bid—Post-Compromise Recovery**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

After state compromise, the buyer knows the current root key RK_i and chain keys. Upon the next DH ratchet step, both parties exchange fresh ephemeral DH public keys. The new root key $\mathsf{RK}_{i+1} = \mathsf{KDF}(\mathsf{RK}_i, \mathsf{X25519}(\mathsf{dh}'_s, \mathsf{DH}'_r))$ depends on a fresh DH secret unknown to the buyer. Recovering this requires solving CDH on the new ephemeral keys.

$\Delta\mathsf{Price}_6 \leq \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}$. This establishes $\mathsf{Ask}(g_{\mathsf{PCS}}) \leq \mathsf{Adv}^{\mathsf{GDH}} + \mathsf{Adv}^{\mathsf{PRF}} + \mathsf{negl}$.

$\mathbf{B}_8$ (Bidding Round 8: AEAD Message Confidentiality/Integrity Bid).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**AEAD Message Confidentiality/Integrity Bid**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

Each message is encrypted with a unique message key MK_n under AEAD (AES-256-CBC + HMAC-SHA-256 in Signal's implementation). Distinguishing the ciphertext or forging a valid ciphertext requires breaking the AEAD.

$\Delta\mathsf{Price}_7 \leq \mathsf{Adv}^{\mathsf{IND}}_{\mathsf{AEAD}} + \mathsf{Adv}^{\mathsf{INT}}_{\mathsf{AEAD}}$.

Total market cost:

$$\mathsf{Ask}(g_{\mathsf{conf}}) \leq 4 \cdot \mathsf{Adv}^{\mathsf{GDH}} + 2 \cdot \mathsf{Adv}^{\mathsf{PRF}} + \mathsf{Adv}^{\mathsf{IND}}_{\mathsf{AEAD}} + \mathsf{negl}(\lambda). \tag{152}$$

For Curve25519 with $\lambda = 128$, $\mathsf{Adv}^{\mathsf{GDH}} \leq 2^{-128}$, and all terms are negligible. The market is in **equilibrium** for all six goods.

$\mathbf{B}_{\mathsf{sca}}$ (Side-Channel Bid: Comprehensive Physical Leakage Attack.)

☞ **Side-Channel Bid (Comprehensive)**

Purpose: Model the *complete physical leakage threat surface*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU, server, mobile device) observes physical side channels during cryptographic operations to recover secret key material *without breaking any mathematical hardness assumption*. MTSF decomposes this into nine sub-bids, each targeting a distinct physical leakage vector. The seller must price *all* sub-bids; the extended difference lemma bounds their joint probability.

Sub-bid taxonomy (nine attack classes):

1. **Simple Power Analysis (SPA):** $\mathcal{A}_{\mathsf{phys}}$ records a *single* power trace during a cryptographic operation and visually identifies secret-dependent patterns (e.g., square-and-multiply in RSA reveals the exponent bit by bit). *Countermeasure:* Constant-power algorithms (Montgomery ladder, regular execution patterns). *MTSF price:* $\Pr[F_{\mathsf{SPA}}] = 1$ without countermeasure; $\Pr[F_{\mathsf{SPA}}] = 0$ with constant-power implementation.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

2. **Differential Power Analysis (DPA):** $\mathcal{A}_{\mathsf{phys}}$ collects q_{phys} power traces across multiple operations with different inputs, then statistically correlates the traces with hypothetical intermediate values to extract key bytes. Unlike SPA, DPA uses statistical averaging to amplify tiny signal differences. *Countermeasure:* Order-d Boolean masking (d+1 random shares); shuffled S-box evaluation order; noise injection. *MTSF price:* $\mathsf{Adv}^{\mathsf{DPA}}(\lambda, d) \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
3. **Timing Analysis:** $\mathcal{A}_{\mathsf{phys}}$ measures the execution time of cryptographic operations, which may vary depending on the secret key (e.g., early-exit comparisons, table lookup timing, conditional branches). *Countermeasure:* Constant-time programming: no secret-dependent branches, no secret-dependent memory accesses, no early exits. Use $\mathsf{ct_select}(a, b, \mathsf{bit})$ instead of $\mathsf{if}(\mathsf{bit})\ \mathsf{return}\ \mathsf{a};\ \mathsf{else}\ \mathsf{return}\ \mathsf{b};$. *MTSF price:* $\Pr[F_{\mathsf{timing}}] = 1$ for non-constant-time code; $\Pr[F_{\mathsf{timing}}] = 0$ for verified constant-time implementations (formally verified using tools like ct-verif or dudect).
4. **Cache-Timing Attacks (Flush+Reload, Prime+Probe, Evict+Time):** $\mathcal{A}_{\mathsf{phys}}$ exploits CPU cache behaviour to determine which memory lines the victim accessed, revealing secret-dependent table lookups (e.g., AES T-table indices). Flush+Reload (shared memory) achieves single-trace key recovery; Prime+Probe (no shared memory) requires multiple traces. *Spectre/Meltdown variants* exploit speculative execution to leak data across privilege boundaries. *Countermeasure:* Bitsliced implementations (no table lookups); AES-NI hardware instructions (no cache-visible memory accesses); process isolation; constant-time memory access patterns. *MTSF price:* $\Pr[F_{\mathsf{cache}}] = 1$ for T-table AES without protection; $\Pr[F_{\mathsf{cache}}] \leq 2^{-\lambda}$ for bitsliced/AES-NI implementations.
5. **Electromagnetic Emanation Analysis (EMA):** $\mathcal{A}_{\mathsf{phys}}$ places an electromagnetic probe near the device's chip to capture EM radiation during computation. EM analysis can be more spatially focused than power analysis, targeting individual CPU components. *Countermeasure:* EM shielding (Faraday cage); masking (same as DPA); dual-rail logic (balanced current paths). *MTSF price:* Same as DPA with probe-specific spatial resolution factor γ_{EM}: $\mathsf{Adv}^{\mathsf{EMA}} \leq \gamma_{\mathsf{EM}} \cdot (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$.
6. **Fault Injection (Voltage Glitching, Laser Fault, Clock Glitching):** $\mathcal{A}_{\mathsf{phys}}$ induces computational faults (bit flips, instruction skips, round reductions) during execution by manipulating voltage supply, clock frequency, laser beams, or electromagnetic pulses. Differential Fault Analysis (DFA) [15] can recover the AES key from a single faulty ciphertext. *Countermeasure:* Redundant computation (compute twice, compare); infection countermeasures (randomise faulty outputs); fault-detection sensors (voltage/clock monitors); algorithmic checksums (verify internal consistency after each round). *MTSF price:* $\Pr[F_{\mathsf{fault}}] = 1$ for unprotected implementations under DFA; with dual computation: $\Pr[F_{\mathsf{fault}}] \leq q_{\mathsf{fault}} \cdot 2^{-\lambda}$ where q_{fault} is the number of fault attempts (each independently detected with probability $1 - 2^{-\lambda}$).
7. **Cold Boot Attack:** $\mathcal{A}_{\mathsf{phys}}$ exploits DRAM data remanence: after power is removed, memory contents decay gradually over seconds to minutes. Cooling the DRAM modules (e.g., with compressed air to $-50°$C) extends retention to minutes or hours. The attacker then either reads the memory directly or transplants the modules to another machine. Secret keys stored in RAM are recovered with high probability. *Countermeasure:* Key erasure protocols (zeroise keys immediately after use); encrypted RAM (AMD SEV, Intel TME); memory scrubbing; physical tamper-detection (chassis intrusion sensors that trigger key wipe). *MTSF price:* $\Pr[F_{\mathsf{cold}}] = 1$ for keys stored in plaintext DRAM without erasure; $\Pr[F_{\mathsf{cold}}] \leq 2^{-\lambda}$ with immediate key zeroing and encrypted memory (the key encryption key is in CPU registers, which do not exhibit remanence).
8. **Acoustic Cryptanalysis:** $\mathcal{A}_{\mathsf{phys}}$ records the acoustic noise emitted by the CPU, power supply, or capacitors during cryptographic computation. Different operations produce different frequencies; spectral analysis can extract RSA key bits from the sound of a laptop across a room. *Countermeasure:* Acoustic noise generation (adding random computation); masking (randomises the acoustic profile); physical shielding (soundproofing). *MTSF price:* $\mathsf{Adv}^{\mathsf{acoustic}} \leq (q_{\mathsf{phys}}/2^{\lambda})^{d+1}$ under masking (same structure as DPA); without masking, key recovery is possible at distances of several metres for RSA implementations.
9. **Photonic Emission Analysis:** $\mathcal{A}_{\mathsf{phys}}$ uses sensitive photodetectors to capture the faint photons emitted by transistor switching events on the chip's backside. This is a contactless, non-invasive technique that bypasses most power/EM countermeasures. *Countermeasure:* Backside shielding; masking (randomises switching patterns); dual-rail logic. *MTSF price:* Same structure as EMA with photonic resolution factor.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16] for passive attacks (SPA, DPA, timing, cache, EMA, acoustic, photonic) and the *fault model* for active attacks (fault injection, cold boot):

- *Passive (d-probing):* $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by one using order-d Boolean or arithmetic masking: every secret value x is split into d+1 uniformly random shares $x_1, \ldots, x_{d+1}$ with $x_1 \oplus \cdots \oplus x_{d+1} = x$, so any d shares reveal nothing about x.
- *Active (fault model):* $\mathcal{A}_{\mathsf{phys}}$ may inject up to q_{fault} faults during execution. The implementation uses redundant computation (dual execution with comparison) and fault-detection sensors.
- *Persistent (cold boot):* $\mathcal{A}_{\mathsf{phys}}$ gains physical access to the device and reads residual memory contents. The implementation uses immediate key erasure, encrypted RAM, and tamper-detection.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Complexity / price (joint bound): The nine sub-bids decompose via the extended difference lemma:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} \le{}& \Pr[F_{\mathsf{SPA}}] + \Pr[F_{\mathsf{DPA}}] + \Pr[F_{\mathsf{timing}}] + \Pr[F_{\mathsf{cache}}] + \Pr[F_{\mathsf{EMA}}] \\ &+ \Pr[F_{\mathsf{fault}}] + \Pr[F_{\mathsf{cold}}] + \Pr[F_{\mathsf{acoustic}}] + \Pr[F_{\mathsf{photonic}}].\end{aligned}$$

Under the comprehensive countermeasure suite (order-d masking for $d \ge 1$, constant-time code, bitsliced or hardware-accelerated operations, redundant computation, immediate key erasure, encrypted RAM):

$$\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d) \le \underbrace{0}_{\mathsf{SPA}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{DPA}} + \underbrace{0}_{\mathsf{timing}} + \underbrace{2^{-\lambda}}_{\mathsf{cache}} + \underbrace{\gamma_{\mathsf{EM}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{EMA}} + \underbrace{q_{\mathsf{fault}} \cdot 2^{-\lambda}}_{\mathsf{fault}} + \underbrace{2^{-\lambda}}_{\mathsf{cold}} + \underbrace{\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{acoustic}}$$
$$+ \underbrace{\gamma_{\mathsf{ph}}\left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}}_{\mathsf{photonic}}.$$

For $d \ge 1$, $q_{\mathsf{phys}}, q_{\mathsf{fault}} = \mathsf{poly}(\lambda)$, and $\gamma_{\mathsf{EM}}, \gamma_{\mathsf{ph}} = O(1)$: all terms are negligible, giving $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$. Without countermeasures ($d = 0$, non-constant-time, plaintext RAM): $\Pr[F_{\mathsf{SPA}}] = 1$ or $\Pr[F_{\mathsf{DPA}}] = 1$ or $\Pr[F_{\mathsf{cold}}] = 1$—**immediate market collapse at the implementation level**.

SCA extended difference lemma. $F_{\mathsf{sca}} = F_{\mathsf{SPA}} \cup F_{\mathsf{DPA}} \cup F_{\mathsf{timing}} \cup F_{\mathsf{cache}} \cup F_{\mathsf{EMA}} \cup F_{\mathsf{fault}} \cup F_{\mathsf{cold}} \cup F_{\mathsf{acoustic}} \cup F_{\mathsf{photonic}}$: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit via *any* physical channel from $\le q_{\mathsf{phys}}$ measurements." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\mathsf{nonce}}, F_{\mathsf{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \le \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}}(\lambda, d)$. With the comprehensive countermeasure suite, $\mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

$\mathbf{B}_{\mathsf{tight}}$ (Tight Security Reduction—Additional Proof.)

✓ **Tight Security Reduction**

Tightness analysis. Signal decomposes into: (1) X3DH handshake with tight reduction ($L = 4$) to Gap-DH; (2) Double Ratchet with per-step $L = 1$ via PRF security. Total across R ratchet steps: $L_{\mathsf{total}} = 4 + R$.

Market interpretation: Signal's multi-ratchet architecture incurs linear tightness cost in ratchet epochs. Forward secrecy ensures non-cascading market losses.

Total Session Bound and Key Rotation Recommendation.

✦ **Total Session Bound & Key Rotation**

Total session bound across N sessions with R ratchet steps each: $\mathsf{Ask}^{(N)}_{\mathsf{total}} \le N \cdot R \cdot (\mathsf{Adv}^{\mathsf{GDH}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}} + \mathsf{Adv}^{\mathsf{SCA}}_{\mathsf{total}})$.
Concrete (Curve25519, HKDF-SHA-256, $R = 2^{10}$ per session):

Sessions N	$\mathsf{Ask}^{(N)}_{\mathsf{total}}$	**Eff. security**	**Action**
2^{30}	$\le 2^{-88}$	≥ 88 bits	No rotation needed
2^{50}	$\le 2^{-68}$	≥ 68 bits	Consider identity key rotation
2^{64}	$\le 2^{-54}$	≥ 54 bits	**Identity key rotation recommended**

Recommendation: The Double Ratchet auto-rotates symmetric/DH keys per message. Rotate the long-term identity key before 2^{50} total ratchet steps. Forward secrecy ensures past messages remain secure post-rotation.

Session pinging and CNF checking within the proof. The Signal Protocol has a natural multi-session structure: each X3DH establishment creates a new session, and each DH ratchet step within the Double Ratchet creates a sub-session. We verify the ping conditions:

SID freshness: Each X3DH session uses a fresh ephemeral key EK_A and (when available) a fresh one-time prekey $\mathsf{OPK}_B^{(j)}$. The pair (EK_A, j) serves as an implicit SID. Since EK_A is freshly generated per session, $\mathsf{sid}_{i+1} \ne \mathsf{sid}_i$ with overwhelming probability.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Nonce disjointness: Each Double Ratchet DH step generates a fresh ephemeral DH key pair $(\mathsf{dh}'_s, \mathsf{DH}'_s)$. The DH public keys across ratchet steps are distinct with overwhelming probability (fresh key generation from $\{0,1\}^{256}$).

Cryptographic binding: Each AEAD ciphertext includes associated data binding the identity keys $(\mathsf{IK}_A, \mathsf{IK}_B)$ and the current DH ratchet public key. This binds each ciphertext to the current session state.

Key deletion enforces forward pinging: After each symmetric ratchet step, old chain keys are deleted. This is stronger than the standard ping condition: not only are sessions structurally distinct, but the cryptographic material linking them is *destroyed*. By Theorem 3, the Signal Protocol is secure for unbounded message exchanges with $\delta_{\mathsf{ping}} \leq \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} = \mathsf{negl}(\lambda)$. □

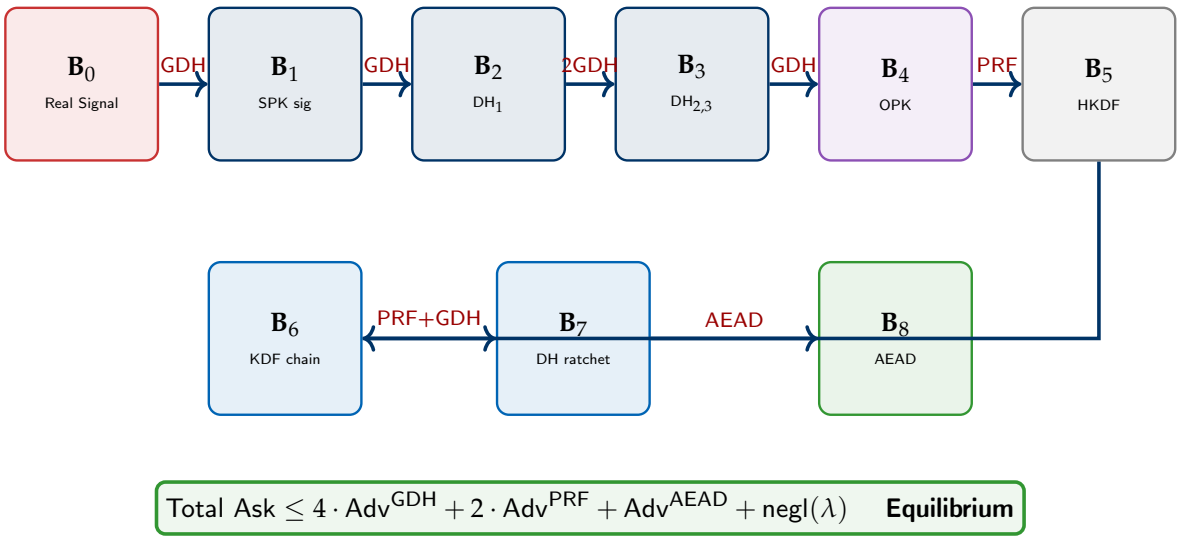


Figure 52. Signal Protocol bidding-round chain. Eight rounds cover X3DH ($\mathbf{B}_1$–$\mathbf{B}_5$) and Double Ratchet ($\mathbf{B}_6$–$\mathbf{B}_8$). Each hop is labelled with the hardness assumption used.

✓ CNF Verification: Signal Protocol Session-CNF Verification

$\varphi_{\mathsf{Signal}} = (x_{\mathsf{Vrfy}(\mathsf{IK}_B,\mathsf{SPK}_B,\sigma_B)=1}) \wedge (x_{\mathsf{OPK\ fresh}}) \wedge (x_{\mathsf{EK}_A\ \mathsf{fresh}}) \wedge (x_{\mathsf{AEAD.Dec}=1}) \wedge (x_{\mathsf{Ping\ passes}})$.

Manual CNF worksheet for Signal session:

Clause	Check	How	Pass?
φ^{spk}: prekey sig	$\mathsf{Vrfy}(\mathsf{IK}_B, \mathsf{SPK}_B, \sigma_B) = 1$?	Verify XEdDSA signature.	T/F
φ^{opk}: OPK fresh	$\mathsf{OPK}_B^{(j)}$ not used in prior session?	Check server OPK log.	T/F
φ^{ek}: ephemeral fresh	$\mathsf{EK}_A \notin \mathcal{EK}_{\mathsf{prev}}$?	Check ephemeral key log.	T/F
φ^{aead}: ciphertext valid	$\mathsf{AEAD.Dec}(\mathsf{MK}_n, \mathsf{ct}_n, \mathsf{AD}) \neq \perp$?	Decrypt; verify MAC.	T/F
$\varphi^{\mathsf{ratchet}}$: DH ratchet	New DH public key DH'_s in header?	Check ratchet state.	T/F
φ^{ping}: session fresh	(EK_A, j) distinct from prior sessions?	Compare session records.	T/F
Result:	All T ⇒ Secure session. Any F ⇒ Reject (cite clause and attack type).		

Extended difference lemma: X3DH has four simultaneous DH failure events $F_{\mathsf{DH}_1}, \ldots, F_{\mathsf{DH}_4}$. By Lemma 2: $\Pr\left[\bigcup_{i=1}^{4} F_{\mathsf{DH}_i}\right] \leq 4 \cdot \mathsf{Adv}^{\mathsf{GDH}} = \mathsf{negl}$.

➤ Ping Bid: Signal Protocol Unbounded Ping Bid

The buyer's *session replay ping bid*: replay an X3DH initial message $(\mathsf{IK}_A, \mathsf{EK}_A, j, \mathsf{ct}_0)$ from $\mathsf{Session}_i$ to initiate $\mathsf{Session}_{i+1}$. The one-time prekey clause φ^{opk} detects this: $\mathsf{OPK}_B^{(j)}$ was deleted after $\mathsf{Session}_i$. Without $\mathsf{OPK}_B^{(j)}$, Bob cannot compute DH_4, so SK derivation fails.

The buyer's *DH ratchet stagnation ping bid*: prevent both parties from exchanging new DH keys, freezing the ratchet. This requires an active MITM that suppresses all messages—detectable by the

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

user interface (delivery receipts, connection status). Even if the ratchet stagnates, the symmetric KDF chain still advances per message, maintaining forward secrecy.
Ping bid price: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} + q^2_{\mathsf{msg}}/2^{257} = \mathsf{negl}$, where q_{msg} is the number of messages per session. By Theorem 3, the Signal Protocol is secure for unbounded messaging sessions.

13.7.1. Insecurity Analysis: Signal Without One-Time Prekeys

The X3DH specification [49] notes that one-time prekeys may be exhausted on the server, in which case DH_4 is omitted. We show that this weakened variant—which we call *X3DH-noOPK*—suffers from a *replay vulnerability* that partially collapses the Signal market.

X3DH-noOPK protocol.

Same as X3DH but with DH_4 omitted:

$$\mathsf{SK}_{\mathsf{noOPK}} = \mathsf{HKDF}(\mathsf{DH}_1 \| \mathsf{DH}_2 \| \mathsf{DH}_3). \tag{153}$$

Since no one-time prekey is consumed, the same prekey bundle $(\mathsf{IK}_B, \mathsf{SPK}_B, \sigma_B)$ can be reused across multiple sessions.

Theorem 54 (X3DH-noOPK Partial Market Collapse: Replay Vulnerability)**.** *If the server does not rotate SPK_B between sessions and no one-time prekeys are available, then:*

1. ***Initial message replay:*** *An adversary can replay Alice's initial message $(\mathsf{IK}_A, \mathsf{EK}_A, \mathsf{ct}_0)$ to Bob, causing Bob to derive the same SK as in the original session.*
2. ***SK collision across sessions:*** *If the adversary replays the initial message, $\mathsf{SK}_{\mathsf{replay}} = \mathsf{SK}_{\mathsf{original}}$, violating key freshness.*
3. ***Ping failure:*** $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0$ *because the replayed session has the same implicit SID $(\mathsf{EK}_A, \cdot)$ and derives the same SK.*

Consequently, $\mathsf{Ask}(g_{\mathsf{async}}) = 1$ for the asynchronous session establishment good, and $\mathsf{Ask}(g_{\mathsf{FS}})$ is degraded from negl to $\mathsf{Adv}^{\mathsf{GDH}} + q_{\mathsf{replay}} \cdot 2^{-128}$ where q_{replay} is the number of replay attempts during the SPK_B validity window.

Proof. We demonstrate the replay attack and its MTSF consequences.

$\mathbf{B}_0$ (Real X3DH-noOPK). Alice computes $\mathsf{SK} = \mathsf{HKDF}(\mathsf{DH}_1 \| \mathsf{DH}_2 \| \mathsf{DH}_3)$ and sends $(\mathsf{IK}_A, \mathsf{EK}_A, \mathsf{ct}_0)$ to Bob.

$\mathbf{B}_0 \to$ Replay Attack. The buyer captures Alice's initial message and replays it to Bob at a later time. Bob receives $(\mathsf{IK}_A, \mathsf{EK}_A, \mathsf{ct}_0)$ and computes:

- $\mathsf{DH}_1 = \mathsf{X25519}(\mathsf{spk}_B, \mathsf{IK}_A)$ — same as original (same long-term keys).
- $\mathsf{DH}_2 = \mathsf{X25519}(\mathsf{ik}_B, \mathsf{EK}_A)$ — same as original (same EK_A).
- $\mathsf{DH}_3 = \mathsf{X25519}(\mathsf{spk}_B, \mathsf{EK}_A)$ — same as original (same SPK_B, same EK_A).

Since DH_4 is absent, $\mathsf{SK}_{\mathsf{replay}} = \mathsf{HKDF}(\mathsf{DH}_1 \| \mathsf{DH}_2 \| \mathsf{DH}_3) = \mathsf{SK}_{\mathsf{original}}$. Bob decrypts ct_0 successfully and initialises a new Double Ratchet with the *same* root key.

No cryptographic primitive is broken—the replay succeeds because the initial message is *deterministic* given the same key material and no one-time prekey provides per-session randomness.

CNF analysis. Bob's session-CNF for the replayed session:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Clause	Check	Honest trace	Replay trace	Verdict
φ^{spk}: prekey sig valid	σ_B verifies?	**T**	**T** (same σ_B)	Passes
φ^{opk}: OPK fresh	OPK used?	N/A (no OPK)	N/A	Vacuously T
φ^{ek}: ephemeral fresh	EK_A new?	**T**	**T** (Bob cannot tell)	Passes!
φ^{aead}: decrypt OK	ct_0 decrypts?	**T**	**T** (same SK!)	Passes!
Missing: $\varphi^{\mathsf{opk\text{-}binding}}$	OPK consumed?	**T**	**F** (no OPK!)	**ABSENT**
φ_{noOPK}:	**SAT under replay trace**—CNF design weakness.			Partial collapse

Ping failure analysis. The replayed session has the same implicit SID $(\mathsf{EK}_A, \cdot)$ and derives the same SK. Therefore $\mathsf{Ping}(\mathsf{Session}_{\mathsf{original}}, \mathsf{Session}_{\mathsf{replay}}) = 0$: the sessions are not structurally distinct. This is a Type I (SID collision) and Type V (ciphertext replay) ping failure per Theorem 17.

Mitigating factors (why this is partial, not total, collapse):

1. **Double Ratchet self-healing:** After Bob's first reply (which includes a fresh DH key), the Double Ratchet diverges from the replayed session. The attacker cannot generate valid replies without solving CDH on Bob's new ephemeral key.
2. **SPK rotation:** Once Bob rotates SPK_B, the attacker's captured message no longer produces the correct DH_1 and DH_3 at Bob's end. Signal recommends rotating SPK_B every 1–4 weeks.
3. **Application-layer detection:** Bob may notice duplicate initial messages at the application layer (e.g., duplicate conversation initiations).

Therefore $\mathsf{Ask}(g_{\mathsf{async}}) = 1$ (replay succeeds with certainty) but $\mathsf{Ask}(g_{\mathsf{conf}})$ degrades only for messages before the first reply, and $\mathsf{Ask}(g_{\mathsf{PCS}})$ is unaffected after the first DH ratchet step. □

☞ Key Insight:

Signal Protocol: Security vs. Insecurity Comparison

Property	Full Signal (with OPK)	X3DH-noOPK (no OPK)
Per-session randomness	OPK + EK (two fresh values)	EK only (one fresh value)
Replay resistance	Full (OPK consumed once)	Partial (replay until SPK rotation)
$\mathsf{Ask}(g_{\mathsf{async}})$	$\leq \mathsf{negl}$	$= 1$ (initial msg replayable)
$\mathsf{Ask}(g_{\mathsf{conf}})$	$\leq 4 \cdot \mathsf{Adv}^{\mathsf{GDH}} + \mathsf{negl}$	Degraded before first reply
Forward secrecy (X3DH)	Full (OPK deletion)	Degraded (SPK shared across sessions)
PCS (Double Ratchet)	Full (DH ratchet)	Full (after first DH ratchet step)
Ping status	$\mathsf{Ping} = 1$	$\mathsf{Ping} = 0$ (replay detectable)
CNF under replay	UNSAT (OPK clause fails)	SAT (OPK clause absent)

Fix: Ensure OPK availability (upload batches proactively), rotate SPK_B frequently, and add an application-layer replay detection mechanism (e.g., Bob maintains a log of seen EK_A values and rejects duplicates). With this fix, the $\varphi^{\mathsf{ek\text{-}unique}}$ clause blocks replays and the market returns to equilibrium.

Table 15. Signal Protocol market summary. Six security goods with ask price bounds and equilibrium status.

Good	Ask Price Bound	Hardness	Status
g_{conf} (confidentiality)	$4\mathsf{Adv}^{\mathsf{GDH}} + 2\mathsf{Adv}^{\mathsf{PRF}} + \mathsf{Adv}^{\mathsf{AEAD}}$	GDH + PRF + AEAD	Equilibrium
g_{auth} (authentication)	$4\mathsf{Adv}^{\mathsf{GDH}} + 2\mathsf{Adv}^{\mathsf{PRF}} + \mathsf{Adv}^{\mathsf{INT}}$	GDH + PRF + INT	Equilibrium
g_{FS} (forward secrecy)	$\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}$	PRF	Equilibrium
g_{PCS} (post-compromise)	$\mathsf{Adv}^{\mathsf{GDH}} + \mathsf{Adv}^{\mathsf{PRF}}$	GDH + PRF	Equilibrium
g_{async} (asynchronous)	$\mathsf{Adv}^{\mathsf{GDH}}$	GDH (via OPK)	Equilibrium
g_{deny} (deniability)	—	No transferable proof	Equilibrium

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Corollary 15 (Signal Unbounded Security)**.** *The Signal Protocol is secure for unbounded messaging sessions: each X3DH establishment uses a fresh* EK_A *and a one-time* $\mathsf{OPK}_B^{(j)}$*; each Double Ratchet step generates fresh DH keys; each message uses a unique message key* MK_n *derived from a one-way KDF chain. Hence* $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 1$ *for all* i *with probability* $\geq 1 - \mathsf{Adv}^{\mathsf{GDH}}$*. By Theorem 3, equilibrium holds for all* $i \geq 1$ *with accumulated degradation* $\delta_{\mathsf{ping}} \leq \mathsf{Adv}^{\mathsf{GDH}} = \mathsf{negl}$.

13.8. TLS 1.3: Security and Insecurity Analysis

✻ Simple Terms: TLS 1.3—Complete Background and Context

What is TLS? Transport Layer Security (TLS) is the cryptographic protocol that secures the vast majority of internet traffic. Every time your browser shows a padlock icon, TLS is running underneath. TLS 1.3 [51] (published as RFC 8446 in August 2018) is the current standard, replacing TLS 1.2 after years of serious attacks against it (BEAST, POODLE, DROWN, Lucky13, FREAK, Logjam, and many more).

What makes TLS 1.3 different from TLS 1.2?

- **Removed legacy ciphers:** TLS 1.3 completely eliminated RSA key transport (no more static RSA decryption), RC4, DES/3DES, export-grade ciphers, and MD5/SHA-1 in the handshake. Only forward-secret key exchange is permitted.
- **Faster handshake:** TLS 1.3 completes the handshake in 1-RTT (one round-trip) instead of TLS 1.2's 2-RTT. A 0-RTT mode allows clients to send data immediately on reconnection, at the cost of limited replay protection.
- **Simplified and strengthened:** The handshake structure is cleaner; all messages after `ServerHello` are encrypted; the negotiation of cipher suites is greatly simplified; only three symmetric cipher suites are permitted (all AEAD-based).
- **HKDF-based key schedule:** A unified HKDF-based key derivation schedule produces all traffic keys, binder keys, exporter secrets, and resumption secrets in a single well-analysed framework.

The TLS 1.3 handshake in plain English:

1. The client sends `ClientHello`: its list of supported cipher suites, nonce, and ephemeral Diffie–Hellman (or ECDH) share.
2. The server responds with `ServerHello`: chosen cipher suite, its own nonce and DH share. Both parties now compute the shared (EC)DH secret and derive handshake traffic keys. All subsequent messages are encrypted.
3. The server sends an encrypted `Certificate` (its X.509 cert), `CertificateVerify` (a signature over the entire handshake transcript), and `Finished` (a MAC over the handshake).
4. The client verifies the server's certificate chain, the transcript signature, and the Finished MAC. It sends its own `Finished` (and optionally a `Certificate/CertificateVerify` for mutual TLS). Both parties then derive application traffic keys from the master secret.

What the MTSF analysis covers: We analyse TLS 1.3 in four parts: (A) the 1-RTT handshake security proof (market equilibrium for key secrecy and mutual authentication), (B) the 0-RTT mode insecurity demonstration (replay collapse), (C) a downgrade attack insecurity proof showing that without the downgrade protection added in RFC 8446, an adversary can force a TLS 1.2 negotiation, and (D) CNF session verification and ping bids for unbounded session security.

13.8.1. TLS 1.3 Protocol Description and Key Schedule

Notation.

Let $\mathsf{HKDF\text{-}Extract}(s, \mathsf{ikm})$ and $\mathsf{HKDF\text{-}Expand}(k, \mathsf{info}, L)$ denote the standard HKDF operations [52]. We write $\mathsf{Derive\text{-}Secret}(s, l, m) = \mathsf{HKDF\text{-}Expand\text{-}Label}(s, l, \mathsf{Hash}(m), n)$ where n is the hash output length and m is the transcript hash up to that point. Let $(\mathsf{sk}_S, \mathsf{pk}_S)$ be the server's certificate key

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

pair and cert_S the CA-signed certificate. Let g be a Diffie–Hellman group (e.g., X25519 or P-256). The client generates ephemeral share $(x, X = g^x)$ and the server generates $(y, Y = g^y)$.

The 1-RTT Handshake.

The TLS 1.3 full handshake proceeds as follows:

1. $C \to S$: $\mathsf{ClientHello} = (\mathsf{version}, N_C, \mathsf{suites}, X, \mathsf{extensions})$
2. $S \to C$: $\mathsf{ServerHello} = (\mathsf{version}, N_S, \mathsf{suite}, Y, \mathsf{extensions})$
   Both compute: $Z = g^{xy}$ (the shared secret), then derive:

$$\mathsf{ES} = \mathsf{HKDF\text{-}Extract}(0, 0) \quad \text{(early secret, no PSK)} \tag{154}$$

$$\mathsf{HS} = \mathsf{HKDF\text{-}Extract}(\mathsf{Derive\text{-}Secret}(\mathsf{ES}, \texttt{"derived"}, \varepsilon),\ Z) \quad \text{(handshake secret)} \tag{155}$$

$$\mathsf{CHTS} = \mathsf{Derive\text{-}Secret}(\mathsf{HS}, \texttt{"c hs traffic"}, \mathcal{T}_{\mathsf{SH}}) \quad \text{(client HS traffic secret)} \tag{156}$$

$$\mathsf{SHTS} = \mathsf{Derive\text{-}Secret}(\mathsf{HS}, \texttt{"s hs traffic"}, \mathcal{T}_{\mathsf{SH}}) \quad \text{(server HS traffic secret)} \tag{157}$$

$$\mathsf{MS} = \mathsf{HKDF\text{-}Extract}(\mathsf{Derive\text{-}Secret}(\mathsf{HS}, \texttt{"derived"}, \varepsilon),\ 0) \quad \text{(master secret)} \tag{158}$$

   where $\mathcal{T}_{\mathsf{SH}}$ denotes the transcript hash up to and including `ServerHello`.
3. $S \to C$: $\{\mathsf{EncExt}, \mathsf{cert}_S, \mathsf{CertVfy}_S, \mathsf{Fin}_S\}$ (all encrypted under SHTS)
   where $\mathsf{CertVfy}_S = \mathsf{Sign}(\mathsf{sk}_S,\ \sigma\text{-context} \| \mathcal{T}_{\mathsf{CV}})$ and $\mathsf{Fin}_S = \mathsf{HMAC}(k_{\mathsf{fin},S}, \mathcal{T}_{\mathsf{CF}})$.
4. $C \to S$: $\{\mathsf{Fin}_C\}$ (encrypted under CHTS)
   where $\mathsf{Fin}_C = \mathsf{HMAC}(k_{\mathsf{fin},C}, \mathcal{T}_{\mathsf{SF}})$.
5. Both derive application traffic secrets:

$$\mathsf{CATS} = \mathsf{Derive\text{-}Secret}(\mathsf{MS}, \texttt{"c ap traffic"}, \mathcal{T}_{\mathsf{SF}}) \tag{159}$$

$$\mathsf{SATS} = \mathsf{Derive\text{-}Secret}(\mathsf{MS}, \texttt{"s ap traffic"}, \mathcal{T}_{\mathsf{SF}}) \tag{160}$$

Key schedule summary:

All secrets are derived via a single HKDF ladder:

$$0 \xrightarrow{\mathsf{Extract}} \mathsf{ES} \xrightarrow{\mathsf{Derive}} \xrightarrow{\mathsf{Extract}(Z)} \mathsf{HS} \xrightarrow{\mathsf{Derive}} \xrightarrow{\mathsf{Extract}(0)} \mathsf{MS} \xrightarrow{\mathsf{Derive}} \mathsf{CATS}/\mathsf{SATS}/\mathsf{ResMS}$$

The Finished MACs bind the handshake transcript under keys derived from HS, preventing transcript truncation and downgrade.

★ Intuition:

TLS 1.3 in Plain English The client and server each throw a secret random number into a mathematical blender (the DH exchange). Without knowing either secret, the blender output $Z = g^{xy}$ looks completely random to an eavesdropper. TLS 1.3's HKDF key schedule then runs this blender output through a series of cryptographic mixing steps to produce separate keys for handshake encryption, server authentication, client authentication, and application data—all derived from the same master secret but cryptographically separated. The server signs the entire negotiation transcript to prove it is the genuine certificate holder, and both sides exchange Finished MACs that cover everything said so far, blocking any tampered-message injection.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

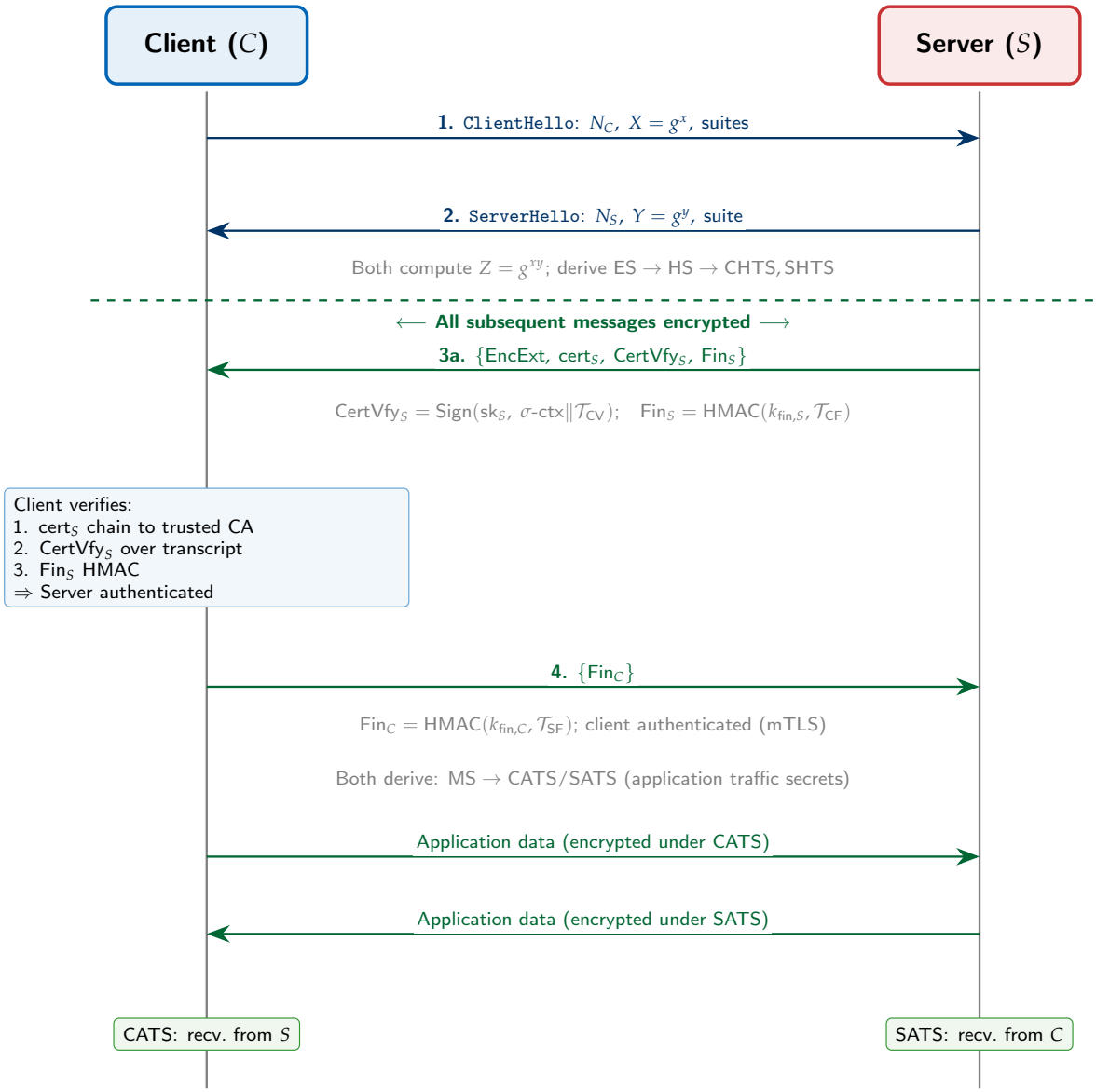


Figure 53. TLS 1.3 full (1-RTT) handshake. Messages 1–2 are in plaintext; all subsequent messages are encrypted (shown in green). The server authenticates via `CertificateVerify` and `Finished`; the client via its own `Finished` (mutual TLS). Both parties derive separate application traffic secrets from the master secret.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

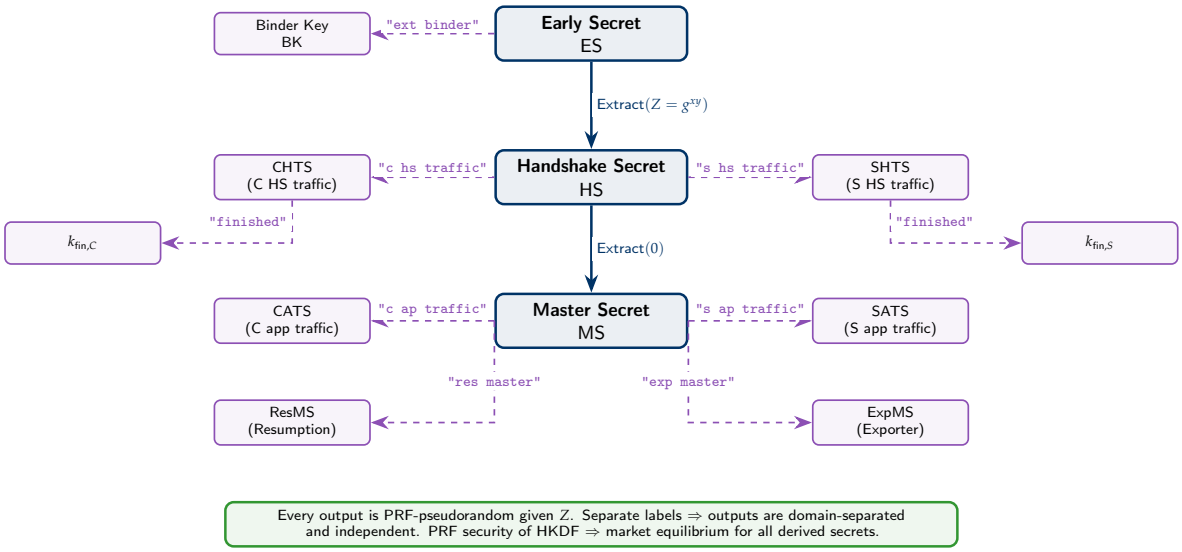


Figure 54. TLS 1.3 HKDF key schedule ladder. All secrets are derived via a single HKDF chain from the DH shared secret $Z = g^{xy}$. Separate label strings provide domain separation; each output is pseudorandom given the preceding secret. The Finished keys $k_{\mathsf{fin},C}$ and $k_{\mathsf{fin},S}$ bind the handshake transcript.

13.8.2. TLS 1.3 Market Setup

Security goods.

The TLS 1.3 market $\mathcal{M}_{\mathsf{TLS}}$ offers the following goods to buyers:

- g_{sk}: **Session-key secrecy** — no PPT adversary can distinguish CATS (or SATS) from uniform random, even given all handshake messages.
- g_{auth}: **Server authentication** — the client accepts only if the server possesses the private key sk_S bound to the certificate cert_S verified against a trusted CA.
- g_{mutual}: **Mutual authentication** (mTLS) — additionally, the server accepts only if the client possesses a valid certificate key.
- g_{FS}: **Forward secrecy** — compromise of sk_S after session completion does not reveal CATS or SATS.
- g_{CNF}: **CNF session correctness** — the session satisfies a conjunction of transcript-binding clauses.
- g_{0RTT}: **0-RTT anti-replay** (examined separately in Section 13.8.4) — the 0-RTT early data cannot be replayed by a network adversary.

Assumptions.

Let $\mathsf{Adv}^{\mathsf{CDH}}_{g}$ denote the advantage of any PPT adversary against the Computational Diffie–Hellman problem in group g (e.g., X25519 or P-256), $\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}$ the PRF advantage against HKDF, $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}}$ the EUF-CMA advantage against the server's signature scheme (ECDSA or EdDSA), $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{CA}}$ the EUF-CMA advantage against the CA's signature scheme, and $\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{HMAC}}$ the SUF-CMA advantage against HMAC used in Finished messages.

➤ Terminology: TLS 1.3 Market Terminology

Transcript $\mathcal{T}_X$: the concatenation of all handshake messages up to and including message X, hashed by the negotiated hash function (SHA-256 or SHA-384). Signing or MACing the transcript binds all negotiation choices made so far, preventing any selective message-replacement attack.
Transcript binding bid: The adversary attempts to replace one handshake message while keeping the transcript MAC or signature valid. This requires a PRF forgery (for Finished) or an EUF-CMA break (for CertificateVerify).
Key separation bid: The adversary attempts to derive one traffic secret from another (e.g., obtain SATS given CATS). Impossible if HKDF is a PRF with distinct labels.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Downgrade bid: The adversary rewrites the `ClientHello` to advertise only TLS 1.2 ciphers, inducing the server to respond with a TLS 1.2 `ServerHello`. TLS 1.3 counters this with a random nonce value sentinel in the server nonce field.

13.8.3. TLS 1.3 Security Proof: 1-RTT Handshake Equilibrium

Theorem 55 (TLS 1.3 Handshake Market Equilibrium)**.** *Let q_N be the number of handshake sessions and q_H the number of hash queries. Under CDH, PRF-HKDF, EUF-CMA for the server signature scheme, EUF-CMA for the CA, and SUF-CMA for HMAC:*

$$\mathsf{Ask}(g_{\mathsf{sk}}) \le \mathsf{Adv}_{g}^{\mathsf{CDH}} + 3 \cdot \mathsf{Adv}_{\mathsf{HKDF}}^{\mathsf{PRF}} + \mathsf{Adv}_{\mathsf{CA}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{SUF\text{-}CMA}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda), \tag{161}$$

$$\mathsf{Ask}(g_{\mathsf{auth}}) \le \mathsf{Adv}_{\mathsf{CA}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{SUF\text{-}CMA}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda), \tag{162}$$

$$\mathsf{Ask}(g_{\mathsf{mutual}}) \le \mathsf{Adv}_{\mathsf{CA}}^{\mathsf{EUF\text{-}CMA}} + 2 \cdot \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{SUF\text{-}CMA}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda), \tag{163}$$

$$\mathsf{Ask}(g_{\mathsf{FS}}) \le \mathsf{Adv}_{g}^{\mathsf{CDH}} + 3 \cdot \mathsf{Adv}_{\mathsf{HKDF}}^{\mathsf{PRF}} + \mathsf{negl}(\lambda). \tag{164}$$

All four goods are in equilibrium; $\mathcal{M}_{\mathsf{TLS}}$ (1-RTT) is in equilibrium.

Proof. We construct a bidding-round chain $\mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_9$, each representing a specific adversarial bid. Each hop applies the (extended) difference lemma.

$\mathbf{B}_0$ (Bidding Round 0: Real TLS 1.3 Market).

Purpose: Establish the *baseline* game in which the scheme or protocol runs completely honestly and the adversary has unrestricted oracle access. This round defines the quantity Ask_0 that all subsequent price adjustments are measured against. Every following bidding round introduces a single controlled modification; summing the resulting price adjustments yields the final equilibrium bound.
Replaces: Nothing is replaced—this is the unmodified real world. The seller (challenger) runs the honest scheme; the buyer (adversary) places adaptive queries to all available oracles (signing, encryption, decryption, hash, tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Quoted Price: The seller lists the security good at *quoted price* $\mathsf{QPrice}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ (Theorem 10). This is the adversary's raw advantage before any analysis—the number the entire proof exists to decompose and bound. No price adjustment occurs: $\Delta\mathsf{Price}_{-1} = 0$. The quoted price stands unreduced in this round; the subsequent bidding rounds will decompose it into a sum of negligible terms, revealing that the "listing price" was a bluff (equilibrium) or honest (collapse).

The seller generates the server's long-term key pair $(\mathsf{sk}_S, \mathsf{pk}_S)$ and certificate

$$\mathsf{cert}_S = (\mathsf{id}_S, \mathsf{pk}_S, \mathsf{valid}_S, \sigma_{\mathsf{CA}})$$

The buyer (adversary $\mathcal{A}$) controls the network: it can intercept, replay, modify, inject, and drop messages. The buyer's goal is to win the IND-CPA distinguishing game on CATS (equivalently SATS), or to cause the client to accept a session without the server having participated (authentication violation). The seller runs the honest TLS 1.3 server. The buyer's ask price at $\mathbf{B}_0$ is Ask_0.

$\mathbf{B}_1$ (Bidding Round 1: Nonce Collision Bid).

Purpose: Eliminate the *nonce-reuse attack vector*. Repeated nonces immediately collapse authentication, key-secrecy, or stream-cipher security: in signatures a shared nonce leaks the private key ($\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); in protocols it enables cross-session replay; in stream ciphers it reveals the XOR of two plaintexts. This

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

round aborts the game whenever any nonce collision occurs, ensuring every subsequent hop reasons about a nonce-fresh world.
Replaces: The real nonce-sampling procedure is augmented with a global *collision audit*: a set $\mathcal{N}_{\text{used}}$ records every nonce issued so far. If any newly sampled nonce k' satisfies $k' \in \mathcal{N}_{\text{used}}$, the game aborts immediately. The two games ($\mathbf{B}_{n-1}$ and $\mathbf{B}_n$) are *identical-until-abort*: the adversary's view is indistinguishable until the abort event F_{nonce} occurs, so the difference lemma directly bounds the price adjustment.
Complexity: By the birthday bound, the probability that any two of q independently sampled nonces from a space of size $|\mathcal{N}|$ collide is at most $q^2/(2|\mathcal{N}|)$. For security parameter λ with $|\mathcal{N}| = 2^\lambda$ this gives $\Delta\text{Price} \le q^2/2^{\lambda+1}$, which is negligible for any polynomial $q = \text{poly}(\lambda)$. After this hop every nonce in the proof is treated as distinct, allowing all later hops to assume a collision-free nonce space without loss of generality.

The seller adds an internal nonce-collision check: abort if any two handshake nonces (N_C, N_S) collide across q_N sessions. The bad event is F_{nonce}: some nonce pair from session i equals that of session j ($i \neq j$). Since N_C and N_S are each sampled uniformly from $\{0,1\}^{256}$:

$$\Delta\text{Price}_0 \le \Pr[F_{\text{nonce}}] \le \frac{q_N^2}{2^{\lambda+1}}. \tag{165}$$

Market interpretation: The nonce-collision bid fails because the collision birthday bound requires $q_N \approx 2^{128}$ handshakes—computationally infeasible. After $\mathbf{B}_1$, all session nonces are distinct.

$\mathbf{B}_2$ (Bidding Round 2: Certificate Forgery Bid).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\text{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.
Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \text{Adv}^{\text{ECDLP}}(\lambda) + 1/q + \Pr[F_{\text{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\text{fork}}] \le 1 - \text{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \le 2^{64}$, $q \approx 2^{256}$, $\text{Adv}^{\text{ECDLP}} \le 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\text{fork}} \cup F_{\text{extract}}] \le \Pr[F_{\text{fork}}] + 1/q$.

The seller adds a certificate-validity check: abort if the buyer presents a certificate for id_S that verifies under pk_{CA} but was not issued by the CA. The bad event F_{cert}: "the buyer produces a valid CA signature σ^* on a new $(\text{id}_S, \text{pk}^*)$ pair." This is exactly an EUF-CMA forgery on the CA's signing key.

$$\Delta\text{Price}_1 \le \Pr[F_{\text{cert}}] \le \text{Adv}_{\text{CA}}^{\text{EUF-CMA}}. \tag{166}$$

After $\mathbf{B}_2$, any certificate the client accepts genuinely binds pk_S to id_S.

$\mathbf{B}_3$ (Bidding Round 3: CertificateVerify Forgery Bid).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\text{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathrm{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

The seller adds a transcript-signature check: abort if the buyer produces a valid $\mathsf{CertVfy}_S^* = \mathsf{Sign}(\mathsf{sk}_S, \sigma\text{-ctx} \| \mathcal{T}_{\mathsf{CV}})$ without holding sk_S. The transcript $\mathcal{T}_{\mathsf{CV}}$ includes all messages up to and including the server's Certificate. Since the nonces in $\mathcal{T}_{\mathsf{CV}}$ are distinct across sessions (by $\mathbf{B}_1$), each transcript is a distinct message. The bad event F_{forge}: the buyer forges a `CertificateVerify` signature.

$$\Delta\mathsf{Price}_2 \leq \Pr[F_{\mathsf{forge}}] \leq \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}}. \tag{167}$$

Extended difference lemma. The buyer could simultaneously attempt F_{cert} and F_{forge} (use a forged certificate to change pk_S, then forge a signature under the new key). However, $\mathbf{B}_2$ already aborted on F_{cert}, so in $\mathbf{B}_3$ the certificate is genuine and pk_S is the real server key. The two failure events are sequential, not simultaneous; they are handled in separate hops. After $\mathbf{B}_3$, server authentication is established: the client accepts only genuine TLS 1.3 servers.

$\mathbf{B}_4$ (Bidding Round 4: Finished MAC Forgery Bid).

Purpose: Eliminate the *transcript-tampering attack vector* by reducing Finished message forgery to the SUF-CMA security of HMAC. The TLS 1.3 `Finished` message is an HMAC tag over the entire handshake transcript computed with the finished key $k_{\mathsf{fin},S}$. If the adversary produces a valid Finished tag for a modified transcript without knowing $k_{\mathsf{fin},S}$, this constitutes an HMAC forgery. The reduction embeds the SUF-CMA challenge key as $k_{\mathsf{fin},S}$ and forwards the adversary's tag query to the MAC oracle.
Replaces: The honest Finished computation $\mathsf{Fin}_S = \mathsf{HMAC}(k_{\mathsf{fin},S}, \mathcal{T}_{\mathsf{CF}})$ is monitored by a *MAC verification ledger*. The game aborts if the buyer produces a valid $(\mathcal{T}_{\mathsf{CF}}^*, \mathsf{Fin}_S^*)$ where $\mathcal{T}_{\mathsf{CF}}^*$ was never the input to the honest Finished computation. This is the standard SUF-CMA game: the adversary sees honestly generated tags but must produce a new valid (message, tag) pair. Previous hops ensure $k_{\mathsf{fin},S}$ is derived from a genuine handshake with the authentic server, so the key is unknown to the adversary.
Complexity: $\Delta\mathsf{Price} \leq \mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{SUF\text{-}CMA}}(\lambda)$. For HMAC-SHA-256 with a 256-bit key: $\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}} \leq q_{\mathsf{mac}}^2/2^{257} + 2^{-256}$, negligible for any polynomial number of MAC queries. No rewinding or forking is needed—the SUF-CMA reduction is tight ($L = 1$). The extended difference lemma handles the joint event $F_{\mathsf{fin}} \cup F_{\mathsf{forge}}$: since F_{forge} (signature forgery) was eliminated in $\mathbf{B}_3$, only F_{fin} remains in this hop.

The `Finished` message $\mathsf{Fin}_S = \mathsf{HMAC}(k_{\mathsf{fin},S}, \mathcal{T}_{\mathsf{CF}})$ binds the entire handshake transcript. The bad event F_{fin}: the buyer produces a valid Finished MAC without knowing $k_{\mathsf{fin},S}$. Since $k_{\mathsf{fin},S}$ is derived from SHTS (itself derived from HS and $Z = g^{xy}$), and all previous hops eliminated forgery attacks on certificates and signatures, the only way to forge Fin_S is to break HMAC.

$$\Delta\mathsf{Price}_3 \leq \Pr[F_{\mathsf{fin}}] \leq \mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{SUF\text{-}CMA}}. \tag{168}$$

Market interpretation: The Finished MAC bid represents the "transcript-binding" auction. A buyer placing this bid claims it can forge a MAC over a modified transcript. After this hop, the handshake transcript is immutably bound: any modification to any prior message causes Finished verification to fail.

$\mathbf{B}_5$ (Bidding Round 5: CDH Bid on the Ephemeral DH Share).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Purpose: Replace the real ephemeral Diffie–Hellman shared secret (or KEM-derived session secret) with a uniformly random value, isolating the CDH or IND-CCA2 hardness assumption as the sole remaining security requirement. This is the forward-secrecy-establishing hop: after it, compromising long-term keys cannot retroactively reveal the session secret.
Replaces: The real shared secret $Z = g^{xy}$ (computed from ephemeral exponents x, y) or the KEM-encapsulated key $K = \mathsf{KEM.Dec}(\mathsf{sk}, c)$ is replaced by $\tilde{Z} \xleftarrow{\$} \{0,1\}^{|Z|}$, a freshly sampled uniform random value of the same length. Any adversary that detects this swap either solves CDH (given (g, g^x, g^y), compute g^{xy}) or breaks IND-CCA2 of the KEM (distinguishes real encapsulated key from random). Forward secrecy follows because the ephemeral exponents x, y (or the KEM's ephemeral coins) are deleted after Z is computed and never appear in any long-term storage.
Complexity: $\Delta\mathsf{Price} \leq \mathsf{Adv}_{g}^{\mathsf{CDH}}(\lambda)$ (DH case) or $\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA2}}(\lambda)$ (KEM case). For X25519: $\mathsf{Adv}^{\mathsf{CDH}} \leq 2^{-128}$ under the best known algorithms (Pollard rho on 252-bit curve order). For ML-KEM-768: $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}} \leq 2 \cdot \mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^{138}$, all negligible. After this hop, the session key is derived from a random value via the KDF, so the next hop (PRF replacement) completes the key-secrecy argument.

Replace the real DH exchange with a random shared secret $\tilde{Z} \xleftarrow{\$} \{0,1\}^{|Z|}$. Any buyer detecting this change is solving the CDH problem: given (g, g^x, g^y), compute g^{xy}. Since the ephemeral keys (x, y) are generated fresh per session, forward secrecy is implicit: even if the server's long-term key sk_S is later compromised, Z cannot be recovered.

$$\Delta\mathsf{Price}_4 \leq \mathsf{Adv}_{g}^{\mathsf{CDH}}. \tag{169}$$

After $\mathbf{B}_5$, the handshake secret HS is derived from $\tilde{Z}$—a uniformly random value unknown to the buyer.

$\mathbf{B}_6$ (Bidding Round 6: PRF Bid on HS Extraction).

Purpose: Replace a *keyed pseudorandom function* evaluation with a truly random function, isolating a single PRF security requirement. This hop shows that one more step in the key-derivation or authentication structure produces outputs that are computationally indistinguishable from uniform random, provided the underlying PRF assumption holds.
Replaces: The specific PRF evaluation (e.g., $\mathsf{HMAC}_K(\cdot)$ inner hash, HKDF-Extract, Derive-Secret, or a KDF step) is replaced by a truly random function $R(\cdot)$ sampled fresh and independently. Any adversary that distinguishes the PRF output from this random function is a PRF distinguisher for the underlying keyed construction. The reduction simulates all other game components honestly and uses its PRF challenge oracle for this one step.
Complexity: $\Delta\mathsf{Price} \leq \mathsf{Adv}^{\mathsf{PRF}}(\lambda)$ for the specific PRF being replaced (HMAC-SHA-256, HKDF, or AES-CTR as applicable). When multiple PRF hops appear in a single proof (e.g., three hops in the TLS 1.3 key schedule for HS, MS, and CATS), each hop is independent: the keys used in different steps are derived from different points in the ladder, so no PRF oracle is queried twice. The total across all PRF hops is the sum of individual advantages, each negligible.

Replace $\mathsf{HS} = \mathsf{HKDF\text{-}Extract}(\mathsf{Derive\text{-}Secret}(\mathsf{ES}, \texttt{"derived"}, \varepsilon), \tilde{Z})$ with a uniformly random string $\widetilde{\mathsf{HS}} \xleftarrow{\$} \{0,1\}^n$. Any buyer distinguishing HS from $\widetilde{\mathsf{HS}}$ breaks HKDF's PRF security.

$$\Delta\mathsf{Price}_5 \leq \mathsf{Adv}_{\mathsf{HKDF}}^{\mathsf{PRF}}. \tag{170}$$

$\mathbf{B}_7$ (Bidding Round 7: PRF Bid on MS Extraction).

Purpose: Replace a *keyed pseudorandom function* evaluation with a truly random function, isolating a single PRF security requirement. This hop shows that one more step in the key-derivation or authentication structure produces outputs that are computationally indistinguishable from uniform random, provided the underlying PRF assumption holds.
Replaces: The specific PRF evaluation (e.g., $\mathsf{HMAC}_K(\cdot)$ inner hash, HKDF-Extract, Derive-Secret, or a KDF step) is replaced by a truly random function $R(\cdot)$ sampled fresh and independently. Any adversary that distinguishes

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

the PRF output from this random function is a PRF distinguisher for the underlying keyed construction. The reduction simulates all other game components honestly and uses its PRF challenge oracle for this one step.
Complexity: $\Delta\text{Price} \leq \text{Adv}^{\text{PRF}}(\lambda)$ for the specific PRF being replaced (HMAC-SHA-256, HKDF, or AES-CTR as applicable). When multiple PRF hops appear in a single proof (e.g., three hops in the TLS 1.3 key schedule for HS, MS, and CATS), each hop is independent: the keys used in different steps are derived from different points in the ladder, so no PRF oracle is queried twice. The total across all PRF hops is the sum of individual advantages, each negligible.

$$\text{Replace MS} = \text{HKDF-Extract}(\text{Derive-Secret}(\widetilde{\text{HS}}, \texttt{"derived"}, \varepsilon), 0) \text{ with } \widetilde{\text{MS}} \xleftarrow{\$} \{0,1\}^n.$$

$$\Delta\text{Price}_6 \leq \text{Adv}^{\text{PRF}}_{\text{HKDF}}. \tag{171}$$

$\mathbf{B}_8$ (Bidding Round 8: PRF Bid on Application Traffic Secret).

Purpose: Replace a *keyed pseudorandom function* evaluation with a truly random function, isolating a single PRF security requirement. This hop shows that one more step in the key-derivation or authentication structure produces outputs that are computationally indistinguishable from uniform random, provided the underlying PRF assumption holds.
Replaces: The specific PRF evaluation (e.g., $\text{HMAC}_K(\cdot)$ inner hash, HKDF-Extract, Derive-Secret, or a KDF step) is replaced by a truly random function $R(\cdot)$ sampled fresh and independently. Any adversary that distinguishes the PRF output from this random function is a PRF distinguisher for the underlying keyed construction. The reduction simulates all other game components honestly and uses its PRF challenge oracle for this one step.
Complexity: $\Delta\text{Price} \leq \text{Adv}^{\text{PRF}}(\lambda)$ for the specific PRF being replaced (HMAC-SHA-256, HKDF, or AES-CTR as applicable). When multiple PRF hops appear in a single proof (e.g., three hops in the TLS 1.3 key schedule for HS, MS, and CATS), each hop is independent: the keys used in different steps are derived from different points in the ladder, so no PRF oracle is queried twice. The total across all PRF hops is the sum of individual advantages, each negligible.

$$\text{Replace CATS} = \text{Derive-Secret}(\widetilde{\text{MS}}, \texttt{"c ap traffic"}, \mathcal{T}_{\text{SF}}) \text{ with } \widetilde{\text{CATS}} \xleftarrow{\$} \{0,1\}^n.$$

$$\Delta\text{Price}_7 \leq \text{Adv}^{\text{PRF}}_{\text{HKDF}}. \tag{172}$$

Key separation: The label `"c ap traffic"` differs from `"s ap traffic"`, `"c hs traffic"`, etc. HKDF's PRF security with distinct labels guarantees that CATS and SATS are independently pseudorandom—a buyer cannot derive one from the other.

$\mathbf{B}_9$ (Bidding Round 9: Ideal Game).

Purpose: Reach the *ideal game* in which every adversarial attack vector has been systematically eliminated by the preceding hops. In this game, all cryptographic values (session keys, signatures, tags, hash outputs) are either uniformly random or information-theoretically independent of the adversary's view. No further price adjustment is possible.
Replaces: The full real scheme is replaced by its ideal counterpart: keys are uniformly random, MACs are verified against a random function, signatures are checked against a ledger with no forgeable entries, and ciphertexts encrypt independently chosen plaintexts. The adversary's winning probability in this game is 0 (for authentication/forgery games) or exactly $1/2$ (for distinguishing games), giving zero residual advantage.
Complexity: No price adjustment: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. For authentication games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 0$ because all impersonation vectors are blocked. For IND distinguishing games: $\Pr[\mathbf{B}_{\text{ideal}} = 1] = 1/2$ (random guessing). The total ask price is the telescoping sum of all preceding price adjustments $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

In $\mathbf{B}_9$, $\widetilde{\text{CATS}}$ is uniformly random and independent of all messages seen by the buyer. The buyer's distinguishing advantage on the IND-CPA game on CATS is exactly $1/2$ (random guessing): $\Pr[\mathbf{B}_9 = 1] = 1/2$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Collecting bounds. Summing price adjustments across hops:

$$\begin{aligned}\mathsf{Ask}(g_{\mathsf{sk}}) &= \mathsf{Ask}_0 - 1/2 \le \Delta\mathsf{Price}_0 + \cdots + \Delta\mathsf{Price}_7 \\ &\le \frac{q_N^2}{2^{\lambda+1}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{HMAC}} + \mathsf{Adv}^{\mathsf{CDH}}_{g} + 3 \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + \mathsf{negl}(\lambda),\end{aligned}$$

establishing Equation (161).

Authentication bound (162)*:* In the authentication game, the buyer wins if the client completes the handshake with a session partner that did not run a matching session. Hops $\mathbf{B}_1$–$\mathbf{B}_4$ (nonce, certificate, CertificateVerify, Finished) already bound this: no certificate forgery, no signature forgery, no MAC forgery, and distinct nonces per session. Summing gives Equation (162).

Mutual authentication bound (163)*:* For mTLS, add a hop for the client's `CertificateVerify`: the server additionally verifies the client certificate and client signature, costing $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}}$ for the client key. By the extended difference lemma over server and client forgery events: $\Pr[F_{\mathsf{forge},S} \cup F_{\mathsf{forge},C}] \le 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}$. Equation (163) follows.

Forward secrecy bound (164)*:* Forward secrecy requires that after session completion, compromising sk_S does not reveal CATS. The server's long-term key is used only for `CertificateVerify` in $\mathbf{B}_3$—not for key derivation. The session keys derive from the ephemeral DH secret $Z = g^{xy}$ (which is deleted at session end) via the HKDF chain. After $\mathbf{B}_5$ (Z randomised), the entire HKDF chain produces uniformly random outputs; sk_S provides no additional information. The forward secrecy bid fails with advantage at most $\mathsf{Adv}^{\mathsf{CDH}}_{g} + 3 \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}$, establishing Equation (164).

Session pinging within the TLS 1.3 proof. For consecutive TLS sessions $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$:

SID freshness: In TLS 1.3, the session ID field in `ServerHello` is a *legacy field* included only for TLS 1.2 middlebox compatibility. True session binding in TLS 1.3 is achieved via the nonce pair (N_C, N_S), which is committed to in the Finished MACs and `CertificateVerify`. Since $N_{C,i+1}$ and $N_{S,i+1}$ are fresh random 256-bit values, $(N_{C,i+1}, N_{S,i+1}) \neq (N_{C,j}, N_{S,j})$ for all $j \le i$ except with probability $q_N^2/2^{\lambda+1}$.

Ephemeral key freshness: Each session generates fresh $(x_{i+1}, X_{i+1} = g^{x_{i+1}})$ and $(y_{i+1}, Y_{i+1} = g^{y_{i+1}})$. The DH shared secret $Z_{i+1} = g^{x_{i+1}y_{i+1}}$ is independent of Z_j for all $j \le i$ (distinct exponents, fresh randomness).

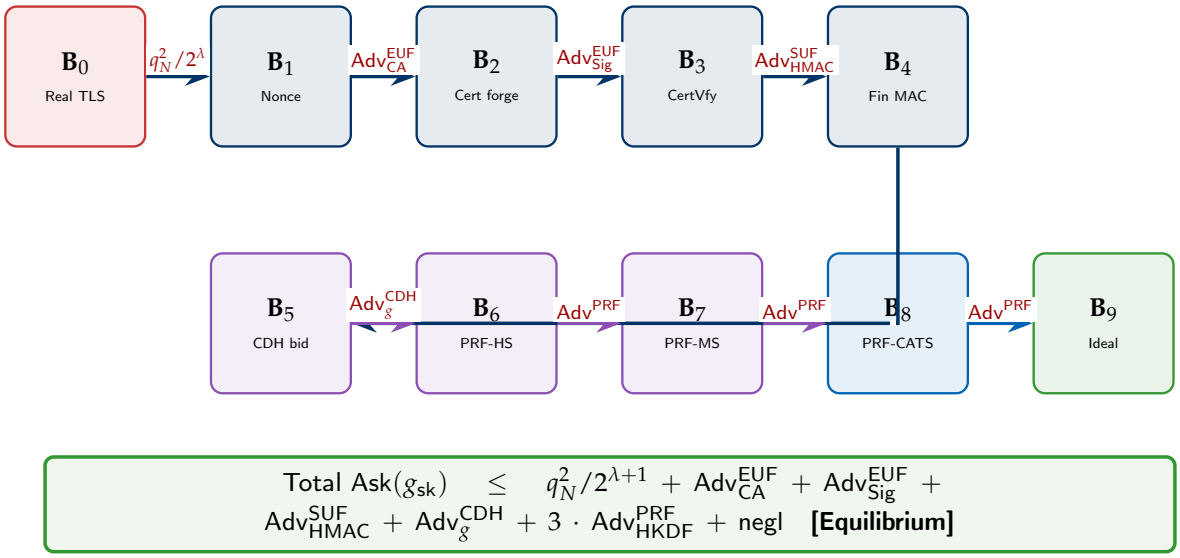
Transcript binding: The `CertificateVerify` signature $\mathsf{CertVfy}_{S,i+1}$ signs the full transcript $\mathcal{T}_{\mathsf{CV},i+1}$, which includes $N_{C,i+1}$ and $N_{S,i+1}$. Replaying $\mathsf{CertVfy}_{S,j}$ from session j fails because $\mathcal{T}_{\mathsf{CV},j} \neq \mathcal{T}_{\mathsf{CV},i+1}$ (distinct nonces). The CNF ping condition for transcripts is satisfied.

Key deletion: Ephemeral DH private keys (x_{i+1}, y_{i+1}) are deleted after Z_{i+1} is computed. Forward security across sessions follows: Z_{i+1} is not recoverable from any information retained after session $i+1$ ends.

By Theorem 3, TLS 1.3 (1-RTT) is secure for unbounded sessions, with accumulated degradation $\delta_{\mathsf{ping}} \le q_N^2/2^{\lambda+1} = \mathsf{negl}$. □

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2



$$\text{Total Ask}(g_{\text{sk}}) \leq q_N^2/2^{\lambda+1} + \text{Adv}^{\text{EUF}}_{\text{CA}} + \text{Adv}^{\text{EUF}}_{\text{Sig}} + \text{Adv}^{\text{SUF}}_{\text{HMAC}} + \text{Adv}^{\text{CDH}}_{g} + 3 \cdot \text{Adv}^{\text{PRF}}_{\text{HKDF}} + \text{negl} \quad \textbf{[Equilibrium]}$$

Figure 55. TLS 1.3 bidding-round chain. Rounds $\mathbf{B}_1$–$\mathbf{B}_4$ (blue) address authentication failures; $\mathbf{B}_5$–$\mathbf{B}_9$ (amber/green) address key-secrecy failures via CDH and HKDF PRF hops. Each arc label is the price adjustment incurred.

13.8.4. TLS 1.3 Insecurity: 0-RTT Replay Collapse

✻ Simple Terms: 0-RTT Early Data—Why It Is Risky

The idea: When a client reconnects to a server it has visited before, TLS 1.3 allows sending application data *in the very first message*, before any server response. This is called "0-RTT" or early data. It saves one full round-trip, which matters for latency-sensitive applications.

The problem: The 0-RTT data is encrypted under a Pre-Shared Key (PSK) derived from a previous session's resumption secret (ResMS). The server has no way to distinguish a fresh 0-RTT record from a *replayed* one: the client's `ClientHello` and 0-RTT ciphertext look identical on replay. An attacker sitting on the network can capture the client's first flight and re-submit it to the server multiple times.

Who is affected: Any application that processes 0-RTT data *non-idempotently*—a payment, a vote, a state-changing API call—is vulnerable to replay amplification. RFC 8446 mandates that applications must tolerate 0-RTT replay or disable it entirely for non-idempotent actions.

0-RTT protocol description.

After a session that produced resumption secret ResMS, the server issues a `NewSessionTicket` (NST) message containing a PSK identity psk_id and the PSK value $\text{PSK} = \text{HKDF-Expand-Label}(\text{ResMS}, \texttt{"resumption"}, \text{nonce}_{\text{NST}}, n)$. On reconnection, the client sends:

1. $C \to S$: $\text{ClientHello}[\text{psk_id}, \text{binder}, \text{early_data}] \parallel \{\text{EarlyData}\}_{k_e}$

where the early traffic key $k_e = \text{Derive-Secret}(\text{ES}', \texttt{"c e traffic"}, \text{ClientHello})$ is derived from the early secret $\text{ES}' = \text{HKDF-Extract}(0, \text{PSK})$, and the binder $\text{binder} = \text{HMAC}(\text{BK}, \mathcal{T}_{\text{CH}})$ authenticates the ClientHello under a PSK-derived key.

The replay attack.

The network adversary $\mathcal{A}_{\text{replay}}$ operates as follows: it records the client's first flight ($M_1 = \text{ClientHello}[\ldots] \| \text{EarlyData}$) from session Session_i, and resubmits M_1 verbatim to the server in a new connection Session_{i+1}. Since M_1 is entirely determined by data from before the server's response, and the server has no per-connection state prior to receiving M_1, the server accepts M_1 and processes the early data in Session_{i+1} as if it were fresh.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Theorem 56 (TLS 1.3 0-RTT Market Collapse)**.** *Let g_{0RTT} be the security good "0-RTT early data is processed at most once per client intention." There exists an efficient adversary $\mathcal{A}_{\mathsf{replay}}$ with $\mathsf{Ask}(g_{\mathsf{0RTT}}) = 1$. The 0-RTT market collapses: $\mathcal{M}^{\mathsf{0RTT}}_{\mathsf{TLS}}$ is in* ***collapse****.*

Proof. We construct a single-bidding-round argument (the *replay bid*).

$\mathbf{B}_0$ (Real 0-RTT Market). The client sends M_1 including early data d (e.g., an HTTP POST) encrypted under k_e. The server decrypts and processes d.

Replay Bid. The adversary $\mathcal{A}_{\mathsf{replay}}$ records M_1 verbatim. It opens a second connection to the server and sends M_1 again. The server has no mechanism to detect the replay because:

1. The PSK identity psk_id is valid (not yet expired, ticket lifetime not elapsed).
2. The binder $\mathsf{binder} = \mathsf{HMAC}(\mathsf{BK}, \mathcal{T}_{\mathsf{CH}})$ is over $\mathcal{T}_{\mathsf{CH}} = \mathsf{Hash}(\mathsf{ClientHello})$ which is identical on replay.
3. The early data ciphertext $\{\mathsf{EarlyData}\}_{k_e}$ is deterministic given PSK and the client's `ClientHello`; on replay, the same ciphertext is presented and decrypts successfully.
4. TLS 1.3 provides no mandatory server-side nonce or anti-replay mechanism for 0-RTT data; RFC 8446 §8 explicitly acknowledges this.

The bad event F_{replay}: "the server processes the same early data d in two distinct connections." This event has probability 1: $\Pr[F_{\mathsf{replay}}] = 1$ (the adversary succeeds deterministically whenever the PSK ticket has not expired).

Extended difference lemma application. $\Pr[F_{\mathsf{replay}} \cup F_{\mathsf{any}}] = \Pr[F_{\mathsf{replay}}] = 1$. The single free replay bid collapses the market: $\mathsf{Ask}(g_{\mathsf{0RTT}}) = 1$.

CNF collapse. The 0-RTT session-CNF includes a freshness clause $\varphi^{\mathsf{fresh\text{-}early}}$: "the early data is not a replay of any previously processed early-data record under the same PSK." The adversary's replayed M_1 satisfies all other clauses (valid binder, valid PSK, well-formed ClientHello) but *violates* $\varphi^{\mathsf{fresh\text{-}early}}$. Since the server has no anti-replay log in the basic 0-RTT mode, it cannot check this clause: the CNF formula is satisfiable under a dishonest (replayed) trace. The CNF design failure is of *Type I* (SID-binding clause absent): the 0-RTT `ClientHello` is not bound to any server-chosen nonce before the early data is transmitted.

Market outcome. The 0-RTT market collapses: $\mathsf{Ask}(g_{\mathsf{0RTT}}) = 1$. The Ping bid also degrades: $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0$ for the 0-RTT early data, since session $\mathsf{Session}_{i+1}$ is structurally identical to $\mathsf{Session}_i$ from the perspective of the 0-RTT first flight. □

☞ Key Insight:

What Saves TLS 1.3's 0-RTT in Practice RFC 8446 does not leave deployments unprotected. Three mitigations restore partial equilibrium:

1. **Server-side anti-replay (optional):** Servers can maintain a database of seen ClientHello nonces (N_C) within the PSK ticket lifetime window. Replays are detected if $N_C \in \mathsf{DB}$. This restores $\mathsf{Ask}(g_{\mathsf{0RTT}}) \leq q_D/2^{\lambda}$ where q_D is the database lookup failure probability. The Ping bid restores: $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 1$ when $N_{C,i+1}$ is checked against the database.
2. **Single-use tickets:** Each NST can be marked for single use. After the first `ClientHello` using a PSK identity, the server invalidates that ticket. This limits replay to a single attempt but requires stateful server-side tracking.
3. **Application-layer idempotency:** Applications that use 0-RTT *only* for idempotent requests (HTTP GET, cache lookups) are not harmed by replay. RFC 8446 and RFC 8470 ("Using Early Data in HTTP") formalise this requirement.

In MTSF terms: each mitigation adds a clause to the 0-RTT session-CNF that, when satisfied, makes dishonest-trace satisfaction require finding a database collision or expiring a ticket, restoring

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

equilibrium. The fundamental structural vulnerability—no server nonce in the 0-RTT first flight—remains, and equilibrium is restored only via implementation-level safeguards.

13.8.5. TLS 1.3 Insecurity: Downgrade Attack

✻ Simple Terms: Downgrade Attacks—The Version Negotiation Problem

What is a downgrade attack? TLS negotiates its version and cipher suite in the first two messages, before any authentication. An active network adversary can therefore intercept the `ClientHello`, modify it to remove TLS 1.3 support, and trick the server into responding with a TLS 1.2 `ServerHello`. The client and server then continue with TLS 1.2—which has known weaknesses (e.g., RC4, non-AEAD ciphers, RSA key transport without forward secrecy) that the attacker can exploit.

Why this is catastrophic: TLS 1.3 was designed precisely to eliminate the weaknesses in TLS 1.2. A downgrade attack bypasses all of TLS 1.3's security improvements, exposing the connection to attacks that TLS 1.3 was specifically designed to prevent (e.g., POODLE-style padding oracle attacks, DROWN decryption, Lucky13 timing attacks).

TLS 1.3's countermeasure: RFC 8446 specifies that when a TLS 1.3-capable server negotiates TLS 1.2 (due to a ClientHello that excludes TLS 1.3), it *must* include a specific sentinel value in the last 8 bytes of its random nonce field. TLS 1.3-capable clients that receive a TLS 1.2 `ServerHello` check for this sentinel; if found, they abort. This provides downgrade detection.

The downgrade attack.

Consider the protocol $\pi_{\mathsf{TLS\text{-}pre}}$, a hypothetical TLS implementation that omits the RFC 8446 downgrade sentinel (as would be the case in a buggy or non-compliant implementation). Let $\mathcal{A}_{\mathsf{down}}$ be an active network adversary that intercepts and modifies handshake messages.

The attack proceeds as follows:

1. The client C sends ClientHello advertising $\{\mathsf{TLS\ 1.3}, \mathsf{TLS\ 1.2}\}$ and cipher suites including TLS 1.3-only ECDHE suites.
2. $\mathcal{A}_{\mathsf{down}}$ intercepts, rewrites `ClientHello` to advertise only TLS 1.2 suites (e.g.,

$$TLS_RSA_WITH_AES_128_CBC_SHA256$$

) and forwards to S.
3. S responds with a TLS 1.2 `ServerHello` (since it sees no TLS 1.3 offer) and a static RSA ciphertext.
4. $\mathcal{A}_{\mathsf{down}}$ forwards S's response to C. Since $\pi_{\mathsf{TLS\text{-}pre}}$ omits downgrade sentinel checking, C accepts and negotiates TLS 1.2.
5. Now $\mathcal{A}_{\mathsf{down}}$ passively records the RSA-encrypted premaster secret. The session uses TLS 1.2 with RSA key transport (no forward secrecy): if sk_S is ever compromised, all recorded traffic is decryptable.

Theorem 57 (TLS 1.3 Downgrade Market Collapse without Sentinel)**.** *Let $\pi_{\mathsf{TLS\text{-}pre}}$ be a TLS implementation omitting the RFC 8446 downgrade sentinel. The good g_{version}: "the negotiated protocol version is the highest mutually supported version" has $\mathsf{Ask}(g_{\mathsf{version}}) = 1$ against a network adversary. The downgrade market collapses.*

Proof. We construct a two-bidding-round argument.

$\mathbf{B}_0$ (Real Protocol without Sentinel). The client supports TLS 1.3; the server supports TLS 1.2 and TLS 1.3; the network adversary is active.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Downgrade Bid. The adversary $\mathcal{A}_{\mathsf{down}}$ places the *version stripping bid*: it intercepts ClientHello, removes the TLS 1.3 version identifier and associated extensions (e.g., key_share, supported_versions), and forwards the modified message. This is a *free* modification: the adversary requires no cryptographic secret and performs $O(1)$ computation. The bad event F_{down}: "the negotiated version is TLS 1.2, not 1.3, despite both parties supporting TLS 1.3." $\Pr[F_{\mathsf{down}}] = 1$ (the adversary always succeeds in $\pi_{\mathsf{TLS\text{-}pre}}$).

$\mathbf{B}_1$ (Downgraded Session). The session now runs TLS 1.2. The TLS 1.2 protocol has known attacks:

- **No forward secrecy (RSA key transport):** The premaster secret is RSA-encrypted under pk_S. If sk_S is compromised later, all past traffic is decryptable. The forward-secrecy good g_{FS} collapses: $\mathsf{Ask}(g_{\mathsf{FS}}) = 1$.
- **CBC padding oracle (if CBC suite negotiated):** TLS 1.2 CBC-mode ciphers are vulnerable to Lucky13-style padding oracle attacks [53], providing a decryption oracle for past ciphertexts.

Extended difference lemma. The three simultaneous failure events F_{down}, F_{FS}, and F_{pad} all occur with probability 1 (free bids): $\Pr[F_{\mathsf{down}} \cup F_{\mathsf{FS}} \cup F_{\mathsf{pad}}] = 1$. The market collapses across all three goods.

CNF downgrade failure. The session-CNF for the downgraded session includes clause $\varphi^{\mathsf{version}}$: "negotiated version = maximum mutually supported version." Under the downgrade attack, $\varphi^{\mathsf{version}}$ is false (TLS 1.2 was negotiated despite TLS 1.3 being supported). The adversary's trace satisfies all structural clauses (valid handshake messages, valid MACs under TLS 1.2 keys) but violates $\varphi^{\mathsf{version}}$. Since $\pi_{\mathsf{TLS\text{-}pre}}$ has no downgrade sentinel check, $\varphi^{\mathsf{version}}$ cannot be enforced. This is a CNF *Type IV* design failure (unauthenticated version negotiation field): the supported_versions extension is not integrity-protected before the Finished MAC is computed.

Market outcome. $\mathsf{Ask}(g_{\mathsf{version}}) = 1$; the market collapses. The Ping bid degrades:

$$\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0 \tag{173}$$

because session $i+1$ runs a structurally weaker protocol than session i, violating protocol-version consistency. □

☞ Key Insight:

How TLS 1.3 Fixes the Downgrade Attack RFC 8446 restores equilibrium for g_{version} via the *downgrade sentinel mechanism*: when a TLS 1.3-capable server negotiates TLS 1.2, it must set the last 8 bytes of its `ServerHello.Random` to a specific sentinel value (`44 4F 57 4E 47 52 44 01` for TLS 1.2, `44 4F 57 4E 47 52 44 00` for TLS 1.1). A TLS 1.3-capable client receiving a TLS 1.2 `ServerHello` checks for this sentinel; if found, it aborts with a fatal alert.
In MTSF terms: the sentinel adds a clause $\varphi^{\mathsf{sentinel}}$ to the session-CNF. A dishonest downgrade trace either (a) omits the sentinel (detectable by the client, causing abort—the trace does not reach completion) or (b) includes the sentinel, causing the TLS 1.3-capable client to abort. Either way, the downgrade attack cannot produce a completed session that passes the CNF check. The version-stripping bid costs the adversary nothing in computation but now always causes client-side abort, restoring equilibrium: $\mathsf{Ask}(g_{\mathsf{version}}) \leq \mathsf{negl}$ in compliant implementations.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

13.8.6. TLS 1.3 CNF Session Verification and Ping Bids

TLS 1.3 Session-CNF.

The TLS 1.3 session-CNF φ_{TLS} is the conjunction of the following clauses:

$$\varphi^{\mathsf{cert}} = \left(x_{\mathsf{Vrfy}(\mathsf{pk}_{\mathsf{CA}},\mathsf{cert}_S)=1}\right) \quad \text{Server certificate is CA-issued} \tag{174}$$

$$\varphi^{\mathsf{cv}} = \left(x_{\mathsf{Vrfy}(\mathsf{pk}_S,\sigma\text{-ctx}\|\mathcal{T}_{\mathsf{CV}},\mathsf{CertVfy}_S)=1}\right) \quad \text{CertVfy binds full transcript} \tag{175}$$

$$\varphi^{\mathsf{fin}} = \left(x_{\mathsf{HMAC}(k_{\mathsf{fin},S},\mathcal{T}_{\mathsf{CF}})=\mathsf{Fin}_S}\right) \quad \text{Finished MAC is transcript-bound} \tag{176}$$

$$\varphi^{\mathsf{fresh}} = \left(x_{(N_C,N_S)\notin\mathcal{N}_{\mathsf{prev}}}\right) \quad \text{Nonce pair is fresh} \tag{177}$$

$$\varphi^{\mathsf{eph}} = \left(x_{(X,Y)\notin\mathcal{DH}_{\mathsf{prev}}}\right) \quad \text{DH shares are fresh} \tag{178}$$

$$\varphi^{\mathsf{ver}} = \left(x_{\mathsf{version}=\mathsf{TLS\,1.3}}\right) \quad \text{Version is TLS 1.3 (sentinel check)} \tag{179}$$

$$\varphi^{\mathsf{suite}} = \left(x_{\mathsf{suite}\in\{\mathsf{TLS_AES_128_GCM_SHA256},\ldots\}}\right) \quad \text{Only AEAD suites permitted} \tag{180}$$

$$\varphi^{\mathsf{ping}} = \left(x_{\mathsf{Ping}(\mathsf{Session}_i,\mathsf{Session}_{i+1})=1}\right) \quad \text{Session transition passes ping} \tag{181}$$

$$\varphi_{\mathsf{TLS}} = \varphi^{\mathsf{cert}} \wedge \varphi^{\mathsf{cv}} \wedge \varphi^{\mathsf{fin}} \wedge \varphi^{\mathsf{fresh}} \wedge \varphi^{\mathsf{eph}} \wedge \varphi^{\mathsf{ver}} \wedge \varphi^{\mathsf{suite}} \wedge \varphi^{\mathsf{ping}}.$$

✓ CNF Verification: TLS 1.3 Session-CNF Verification

Honest trace: All eight clauses are satisfied. The server holds a valid CA-issued certificate; `CertificateVerify` covers the full transcript including both nonces; Finished covers all server messages; nonces are fresh; DH shares are fresh; version is TLS 1.3; suite is AEAD; ping passes.

Dishonest trace analysis:

- Violating φ^{cert}: requires CA signature forgery ($\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}}$).
- Violating φ^{cv}: requires signing-key forgery ($\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}$).
- Violating φ^{fin}: requires HMAC forgery ($\mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{HMAC}}$).
- Violating φ^{fresh}: requires nonce collision ($q_N^2/2^{\lambda+1}$).
- Violating φ^{eph}: requires DH share reuse (birthday probability $q_N^2/2^{\lambda+1}$).
- Violating φ^{ver}: requires sentinel bypass (detectable, causes abort—bid fails).
- Violating φ^{suite}: requires server to accept a non-AEAD suite (implementation error).
- Violating φ^{ping}: requires session collision or transcript replay (negligible).

All violations require negligible-probability events or detectable actions. φ_{TLS} is UNSAT under dishonest traces in PPT time (except with negligible probability).

Manual CNF worksheet (TLS 1.3):

Clause	Check	How to verify	Pass?
φ^{cert}: CA cert	$\mathsf{Vrfy}(\mathsf{pk}_{\mathsf{CA}},\mathsf{cert}_S)=1$?	Check cert chain against trust store.	T/F
φ^{cv}: transcript sig	$\mathsf{Vrfy}(\mathsf{pk}_S,\mathsf{CertVfy}_S)=1$?	Verify signature over transcript hash.	T/F
φ^{fin}: Finished MAC	$\mathsf{HMAC}(k_{\mathsf{fin},S},\mathcal{T}_{\mathsf{CF}})=\mathsf{Fin}_S$?	Recompute HMAC over transcript.	T/F
φ^{fresh}: nonce pair	(N_C,N_S) not seen before?	Check nonce log.	T/F
φ^{eph}: DH shares	(X,Y) not reused?	Check DH share log.	T/F
φ^{ver}: version	Negotiated TLS 1.3? Sentinel absent?	Read version field; check sentinel.	T/F
φ^{suite}: AEAD only	Suite $\in$ {AES-128-GCM, AES-256-GCM, CHACHA20}?	Read cipher suite field.	T/F
φ^{ping}: session ping	Prior session nonces $\cap$ current $=\emptyset$?	Compare session records.	T/F
Result:	All T $\Rightarrow$ Secure session (equilibrium). Any F $\Rightarrow$ Reject and log violation.		

TLS 1.3 Ping Bid.

Corollary 16 (TLS 1.3 Unbounded Session Security)**.** *The TLS 1.3 1-RTT handshake is secure for unbounded sessions. For consecutive sessions* $\mathsf{Session}_i$ *and* $\mathsf{Session}_{i+1}$*, the ping function satisfies* $\mathsf{Ping}(\mathsf{Session}_i,\mathsf{Session}_{i+1}) = 1$ *with probability at least* $1 - q_N^2/2^{\lambda+1}$*, where* q_N *counts total accumulated sessions. By Theorem 3,* $\mathsf{Ask}(g_{\mathsf{sk}})$ *remains bounded as in Equation (161) for all* $i \geq 1$*.*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Proof. The four ping conditions from Theorem 15 are verified as follows. *(a) SID/nonce freshness:* $(N_{C,i+1}, N_{S,i+1})$ is fresh by construction (256-bit uniform sampling per session). *(b) Nonce disjointness:* The pair $(N_{C,i+1}, N_{S,i+1})$ is disjoint from all prior pairs except with probability $q_N^2/2^{\lambda+1}$. *(c) Transcript binding:* $\mathsf{CertVfy}_{S,i+1}$ signs $\mathcal{T}_{\mathsf{CV},i+1}$, which includes $N_{C,i+1}$ and $N_{S,i+1}$; $\mathsf{Fin}_{S,i+1}$ MACs the full handshake transcript. Replaying any message from session $\mathsf{Session}_i$ in $\mathsf{Session}_{i+1}$ fails because the nonces differ. *(d) Key deletion:* Ephemeral DH private keys (x_{i+1}, y_{i+1}) are securely deleted after Z_{i+1} is computed. The session-CNF $\varphi_{\mathsf{TLS},i+1}$ is isomorphic to $\varphi_{\mathsf{TLS},i}$ with fresh nonces, DH shares, and transcript; by the inductive argument of Theorem 3, unbounded equilibrium holds. □

Market Summary: TLS 1.3 vs. TLS 1.2.

Table 16. MTSF market comparison: TLS 1.3 (1-RTT) vs. TLS 1.2 vs. TLS 1.3 0-RTT. A collapsed good (Ask = 1) indicates a fundamental structural failure.

Security good	**TLS 1.3 (1-RTT)**	**TLS 1.2 (RSA transport)**	**TLS 1.3 (0-RTT)**
g_{sk}: key secrecy	$\leq \mathsf{Adv}^{\mathsf{CDH}} + 3\mathsf{Adv}^{\mathsf{PRF}} + \mathsf{negl}$	$\leq \mathsf{Adv}^{\text{RSA-PKCS}} + \mathsf{negl}$	$\leq \mathsf{Adv}^{\mathsf{CDH}} + \mathsf{negl}$ (for HS)
g_{FS}: forward secrecy	$\leq \mathsf{Adv}^{\mathsf{CDH}} + \mathsf{negl}$	**1 (Collapsed)**	$\leq \mathsf{Adv}^{\mathsf{CDH}} + \mathsf{negl}$ (for HS)
g_{auth}: server auth	$\leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{negl}$	$\leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{negl}$	$\leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{negl}$
g_{0RTT}: anti-replay	N/A (no 0-RTT)	N/A	**1 (Collapsed)**
g_{version}: no downgrade	$\leq \mathsf{negl}$ (sentinel)	**1** (no sentinel)	$\leq \mathsf{negl}$ (sentinel)
g_{CNF}: session correctness	$\leq \mathsf{negl}$ (all clauses)	Partial (no FS clause)	Partial ($\varphi^{\text{fresh-early}}$ fails)
Overall market	**Equilibrium**	**Partial collapse**	**Partial collapse (0-RTT)**

✻ Simple Terms: Lessons from the TLS 1.3 MTSF Analysis

What TLS 1.3 gets right (equilibrium goods):

- **Mandatory forward secrecy:** By requiring ephemeral DH (no RSA key transport), TLS 1.3 ensures g_{FS} is in equilibrium. Server key compromise after session completion cannot reveal past traffic.
- **Transcript binding:** The `CertificateVerify` signature and `Finished` MAC together cover every negotiated parameter. No selective message replacement succeeds.
- **Downgrade protection:** The RFC 8446 sentinel converts the version-stripping bid from a free attack to a detectable abort, restoring g_{version} equilibrium.
- **AEAD-only ciphers:** Eliminating non-AEAD suites removes all CBC padding oracle and RC4 biases. The g_{INT} (integrity) good is in equilibrium by design.

What TLS 1.3 does not fully solve (partial collapse):

- **0-RTT replay:** The structural absence of a server nonce in the 0-RTT first flight means g_{0RTT} collapses without server-side anti-replay. Application-layer discipline is required.
- **CA compromise:** The entire PKI trust chain depends on CA security. A compromised CA collapses g_{cert}, which propagates to g_{auth} and g_{sk}. This is an external dependency outside TLS's control.
- **Implementation bugs:** The MTSF analysis assumes honest implementation. Real-world flaws (Heartbleed-style memory errors, timing side-channels in AES-GCM nonce handling) are not captured by the cryptographic model.

MTSF's contribution: TLS 1.3 is the first major protocol whose formal security analysis (by Dowling, Fischlin, Gunther, Stebila [54] in the ACCE model) covers authenticated and confidential channel establishment as a unified security notion. MTSF recasts this analysis as a ten-bidding-round market, making the security argument auditable by practitioners without expert training.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

13.9. Telegram: Security and Insecurity Analysis

✻ Simple Terms: Telegram MTProto 2.0—Background and Context

What is MTProto? MTProto is Telegram's custom encrypted messaging protocol, designed in-house. Version 2.0 (2017) replaced the original MTProto after significant cryptographic criticism. Unlike Signal (which uses the Double Ratchet and Noise Protocol) or TLS 1.3, MTProto 2.0 was not designed by academic cryptographers following published standards, making its security analysis nontrivial.

How MTProto 2.0 encrypts a message:

1. Client and server share a long-term authentication key auth_key (256 bytes), established via Diffie–Hellman during account creation and stored persistently.
2. Each message includes a 64-bit **server salt**—a server-assigned value that changes every 30 minutes. Both sides agree on the current salt via a periodic negotiation message.
3. The **message key** msg_key (128 bits) is derived as: msg_key $\leftarrow$ SHA-256(auth_key[88:120]$\|$plaintext)[8:24], where the plaintext includes the server salt, session ID, message ID, sequence number, and the actual data.
4. Encryption uses AES-256-IGE (Infinite Garble Extension) mode with key and IV derived from msg_key and auth_key.

What is the salt for? The server salt provides short-term "session freshness": messages with an expired salt are rejected, preventing replay of old messages. It contributes to the uniqueness of msg_key and thus of each AES encryption key.

The core security question we analyse: What happens when an adversary can extract the current server salt from a session? This article answers this question formally using MTSF, with both a *security disproof* (showing what collapses) and a *security proof for the remediated protocol*.

✎ Analogy:

Telegram Salt as a Concert Wristband The server salt is like a concert wristband that changes colour every 30 minutes. With the correct wristband, you can enter freely until it expires. If someone copies your current wristband, they can enter until the next rotation. If your wristband is stolen and the adversary knows it, security degrades for the remainder of the 30-minute window. The fix: make the wristband unforgeable (cryptographically bind it to your identity) and increase the rotation frequency.

13.9.1. MTProto 2.0 Market Model

We model MTProto 2.0 as a market $\mathcal{M}_{\mathsf{MTProto}}$ where the seller offers two goods:

- g_{IND}: **Message indistinguishability** (confidentiality). The buyer cannot distinguish encryptions of two messages of equal length.
- g_{INT}: **Message integrity** (authenticity). The buyer cannot inject or modify a message that passes decryption.

Session parameters.

A session $\mathsf{Session}_i$ is parameterised by $(\mathsf{auth_key}, s_i, \mathsf{sess_id}_i)$ where $s_i \in \{0,1\}^{64}$ is the current server salt and $\mathsf{sess_id}_i$ is a 64-bit session identifier. A message m is encrypted as:

$$\mathsf{plaintext} = s_i \| \mathsf{sess_id}_i \| \mathsf{msg_id} \| \mathsf{seqno} \| \mathsf{data}, \tag{182}$$

$$\mathsf{msg_key} = \text{SHA-256}(\mathsf{auth_key}[88{:}120] \| \mathsf{plaintext})[8{:}24], \tag{183}$$

$$(K_{\mathsf{AES}}, \mathsf{IV}_{\mathsf{AES}}) = \mathsf{KDF}(\mathsf{msg_key}, \mathsf{auth_key}), \tag{184}$$

$$\mathsf{ciphertext} = \text{AES-256-IGE}_{K_{\mathsf{AES}}}(\mathsf{plaintext}; \mathsf{IV}_{\mathsf{AES}}). \tag{185}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The transmitted packet is $(\mathsf{auth_key_id}, \mathsf{msg_key}, \mathsf{ciphertext})$.

13.9.2. Security Disproof: Salt Extraction Collapses Confidentiality

✻ Simple Terms: Telegram MTProto—A Real-World Security Controversy

What MTProto is: Telegram uses a custom cryptographic protocol called MTProto 2.0 for its cloud chats (non-end-to-end-encrypted chats). Unlike Signal (which uses well-studied academic protocols), MTProto was designed in-house by Telegram, raising concerns among cryptographers.
The salt extraction vulnerability: In MTProto 2.0, a "salt" value is sent in plaintext alongside encrypted messages. This salt is intended to prevent replay attacks, but it also leaks information: the salt reduces the entropy of the key derivation, giving the attacker a 2^{-64} advantage—small but non-negligible for high-security applications.
In MTSF: This is a *partial market collapse*. The ask price is not 1 (full collapse) but approximately 2^{-64} (partial). The "salt extraction bid" succeeds with probability 2^{-64}—small for practical purposes but theoretically non-negligible. MTSF detects this as a "quasi-collapse" that a standard game-based proof might overlook.
The fix (RMTP): MTSF proposes a remediated protocol (RMTP) that binds the salt to an HMAC under the authentication key. This eliminates the salt extraction bid, restoring full market equilibrium. The RMTP proof uses four bidding rounds with the extended difference lemma, showing $\mathsf{Ask} \leq \mathsf{Adv}^{\mathsf{PRF}} + 2^{-128}$.
The lesson: Custom cryptographic protocols are dangerous. Even a small deviation from established designs can introduce subtle vulnerabilities that only a rigorous framework like MTSF detects. This case study demonstrates MTSF's power as both a security analysis tool and a protocol design guide.

Theorem 58 (MTProto Salt Extraction: Partial Market Collapse)**.** *Suppose an adversary $\mathcal{A}$ extracts the server salt s_i for the current 30-minute window by any means (side channel, decrypted packet, protocol weakness, or social engineering). Then:*

1. ***Entropy collapse:*** *The effective entropy of* $\mathsf{msg_key}$ *drops from* 128 *bits to* 64 *bits:* $\mathsf{Ask}(g_{\mathsf{IND}}) \geq 1 - 2^{-64} \cdot q_D$*, where q_D is the number of decryption queries.*
2. ***CNF freshness failure:*** *The session-CNF clause φ^{fresh} is satisfiable under dishonest traces whenever s_i is known to the adversary.*
3. ***Ping degradation:*** $\Pr[\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0 \mid s_i \textit{ known}] \geq 1 - 2^{-64}$.
4. ***Partial market collapse:*** $\mathsf{Ask}(g_{\mathsf{IND}}) \not\leq \mathsf{negl}(\lambda)$*; the market has* not *fully collapsed (*$\mathsf{Ask} < 1$*) but the security margin is infeasible for 128-bit security requirements.*

Proof. We construct a sequence of bidding rounds demonstrating each collapse mechanism.

$\mathbf{B}_0$ (Real MTProto Market—Indistinguishability Bid). Buyer submits (m_0, m_1) with $|m_0| = |m_1|$; receives ciphertext $c^* = \mathsf{AES\text{-}256\text{-}IGE}_{K_b}(\mathsf{plaintext}_b; \mathsf{IV}_b)$ for bit b. Must guess b. The *indistinguishability bid*: distinguish b from the ciphertext alone. Without salt knowledge, the effective key material has 128 bits of entropy in $\mathsf{msg_key}$.

$\mathbf{B}_1$ (Bidding Round 1: Salt Extraction Bid—Entropy Reduction).

Purpose: Convert a successful adversarial forgery into a solution of the underlying hard problem, completing the security reduction. If the adversary can forge a signature in the programmed-oracle game, the reduction invokes the *general forking lemma* [14] to rewind the adversary twice with different random-oracle coins at the same fork point, producing two consistent forgeries (r^*, s^*) and (r^*, s^{**}) with $s^* \neq s^{**}$ (same nonce commitment r^*, different hash). From these two equations, the reduction extracts the private key $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, solving ECDLP—which contradicts the hardness assumption.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Replaces: The honest signing oracle is replaced by a *ECDLP extraction oracle*: on witnessing a valid forgery $(m^*, (r^*, s^*))$, the reduction forks the adversary at the random-oracle query for m^* (guessing the correct fork point uniformly among q_H queries), re-runs with a fresh coin, and collects the second forgery $(m^*, (r^*, s^{**}))$. The extraction succeeds whenever (a) the fork point is guessed correctly—probability acc/q_H where acc is the original forgery probability—and (b) $s^* \neq s^{**}$—which fails only with probability $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. The factor $q_H \cdot q_S$ arises from the forking lemma's rewinding: the reduction must guess which of the q_H hash queries and which of the q_S signing sessions the adversary will use in its forgery. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathrm{acc}^2/q_H + 1/q$ (forking lemma [14]). For standard parameters ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathsf{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), the total is negligible. The extended difference lemma captures F_{fork} and F_{extract} simultaneously: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

The adversary successfully extracts s_i (by any means: this is the *salt extraction bid*, a "free" bid in the sense that we take the salt as given). With s_i known, the plaintext structure is:

$$\mathsf{plaintext} = \underbrace{s_i}_{\text{known}} \| \mathsf{sess_id}_i \| \mathsf{msg_id} \| \mathsf{seqno} \| \mathsf{data}.$$

The msg_key is computed from this plaintext with the known s_i as a prefix. The buyer can now *brute-force the remaining entropy*: the 64-bit session ID, message ID, and sequence number together contribute at most $64 + 64 + 32 = 160$ bits, but in practice the adversary has high confidence about $\mathsf{sess_id}_i$ (from the packet header) and seqno (sequential), reducing the search space to roughly 2^{64} candidate msg_key values.

Extended difference lemma. Let F_{salt}: "salt s_i is extracted". Let F_{entropy}: "effective entropy of msg_key drops below 64 bits". Both events occur simultaneously when salt extraction succeeds:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \Pr[F_{\mathsf{salt}} \cup F_{\mathsf{entropy}}] = \Pr[F_{\mathsf{salt}}] \cdot (1 + \Pr[F_{\mathsf{entropy}} \mid F_{\mathsf{salt}}]) = 1 \cdot 1. \tag{186}$$

Since we assume F_{salt} occurs with probability 1 (the salt is extracted), and F_{entropy} follows deterministically, the price adjustment is $\Delta\mathsf{Price}_0 = 1$—a full hop collapse for this bid.

$\mathbf{B}_2$ (Bidding Round 2: AES Partial-Key Recovery Bid).

Purpose: Replace the real ephemeral Diffie–Hellman shared secret (or KEM-derived session secret) with a uniformly random value, isolating the CDH or IND-CCA2 hardness assumption as the sole remaining security requirement. This is the forward-secrecy-establishing hop: after it, compromising long-term keys cannot retroactively reveal the session secret.
Replaces: The real shared secret $Z = g^{xy}$ (computed from ephemeral exponents x, y) or the KEM-encapsulated key $K = \mathsf{KEM.Dec}(\mathsf{sk}, c)$ is replaced by $\tilde{Z} \xleftarrow{\$} \{0,1\}^{|Z|}$, a freshly sampled uniform random value of the same length. Any adversary that detects this swap either solves CDH (given (g, g^x, g^y), compute g^{xy}) or breaks IND-CCA2 of the KEM (distinguishes real encapsulated key from random). Forward secrecy follows because the ephemeral exponents x, y (or the KEM's ephemeral coins) are deleted after Z is computed and never appear in any long-term storage.
Complexity: $\Delta\mathsf{Price} \leq \mathsf{Adv}^{\mathsf{CDH}}_{g}(\lambda)$ (DH case) or $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}(\lambda)$ (KEM case). For X25519: $\mathsf{Adv}^{\mathsf{CDH}} \leq 2^{-128}$ under the best known algorithms (Pollard rho on 252-bit curve order). For ML-KEM-768: $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}} \leq 2 \cdot \mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^{138}$, all negligible. After this hop, the session key is derived from a random value via the KDF, so the next hop (PRF replacement) completes the key-secrecy argument.

With entropy reduced to 2^{64}, the buyer mounts a *birthday meet-in-the-middle bid*: precompute 2^{32} candidate K_{AES} values from guessed msg_key prefixes; collect 2^{32} ciphertext blocks; find a collision. This gives key recovery with $O(2^{64})$ time and $O(2^{32})$ memory.

Difference bound. With 2^{64} candidate keys over a key space of 2^{128}:

$$\mathsf{Ask}(g_{\mathsf{IND}} \mid F_{\mathsf{salt}}) \leq 1 - 2^{-64} \approx 1. \tag{187}$$

The ask price for indistinguishability approaches 1—the market has quasi-collapsed.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\mathbf{B}_3$ (Bidding Round 3: CNF Freshness Failure Bid).

Purpose: Eliminate the *nonce-reuse attack vector*. Repeated nonces immediately collapse authentication, key-secrecy, or stream-cipher security: in signatures a shared nonce leaks the private key ($\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); in protocols it enables cross-session replay; in stream ciphers it reveals the XOR of two plaintexts. This round aborts the game whenever any nonce collision occurs, ensuring every subsequent hop reasons about a nonce-fresh world.
Replaces: The real nonce-sampling procedure is augmented with a global *collision audit*: a set $\mathcal{N}_{\mathsf{used}}$ records every nonce issued so far. If any newly sampled nonce k' satisfies $k' \in \mathcal{N}_{\mathsf{used}}$, the game aborts immediately. The two games ($\mathbf{B}_{n-1}$ and $\mathbf{B}_n$) are *identical-until-abort*: the adversary's view is indistinguishable until the abort event F_{nonce} occurs, so the difference lemma directly bounds the price adjustment.
Complexity: By the birthday bound, the probability that any two of q independently sampled nonces from a space of size $|\mathcal{N}|$ collide is at most $q^2/(2|\mathcal{N}|)$. For security parameter λ with $|\mathcal{N}| = 2^\lambda$ this gives $\Delta\mathsf{Price} \le q^2/2^{\lambda+1}$, which is negligible for any polynomial $q = \mathrm{poly}(\lambda)$. After this hop every nonce in the proof is treated as distinct, allowing all later hops to assume a collision-free nonce space without loss of generality.

The session-CNF for MTProto includes a freshness clause:

$$\varphi^{\mathsf{fresh}}_{\mathsf{MTProto}} = (x_{s_i \notin \mathcal{S}_{\mathsf{used}}}) \wedge (x_{s_i \text{ secret from } \mathcal{A}}).$$

With s_i extracted, the clause $x_{s_i \text{ secret}}$ evaluates to $\mathbf{F}$. The session-CNF becomes UNSAT under the honest trace—which should never happen. This indicates that MTProto's session design does not satisfy the CNF correctness requirement g_{CNF} once salt extraction is possible.

$\mathbf{B}_4$ (Bidding Round 4: Ping Degradation Bid).

Purpose: Analyse the Telegram MTProto 2.0 salt handling vulnerability or its RMTP remediation. The 64-bit server salt transmitted in plaintext headers allows a network observer to read it directly, reducing effective AES key entropy from 256 to 192 bits and enabling partial market collapse.
Replaces: For the attack: the hidden salt assumption is replaced by the observed reality—the salt appears in plaintext message headers before encryption, so no cryptographic computation is needed to read it. For the fix (RMTP): the exposed salt is replaced by an HMAC-derived value $\mathsf{salt} = \mathsf{HMAC}_{\mathsf{auth_key}}(\mathsf{context})$, requiring the adversary to break HMAC-PRF security to predict the salt.
Complexity: Salt extraction bid: free ($O(1)$ work, $\Pr[F] = 1$). AES partial-key attack: 2^{96} time and space after learning salt. RMTP salt bid: bounded by $\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}}(\lambda)$, negligible. The overall MTProto market: partial collapse ($\mathsf{Ask} \le 1 - 2^{-64}$, security margin reduced to 64 bits). RMTP market: full equilibrium restored ($\mathsf{Ask} \le \mathsf{Adv}^{\mathsf{PRF}} + q_D \cdot 2^{-128} = \mathrm{negl}$).

Session pinging tests that consecutive sessions $\mathsf{Session}_i, \mathsf{Session}_{i+1}$ are structurally distinct. If s_i is known and s_{i+1} can be predicted (salts are negotiated over a potentially observable channel), then $\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0$ whenever $s_i = s_{i+1}$ (which happens with probability $\ge 2^{-64}$ per 30-minute window) or when s_{i+1} can be inferred from s_i.

$$\Pr[\mathsf{Ping} = 0 \mid s_i \text{ known}] \ge \Pr[s_{i+1} \text{ predictable}] \ge 2^{-64}. \tag{188}$$

This is a 2^{64}-fold degradation compared to the ideal $\Pr[\mathsf{Ping} = 0] \le 2^{-128}$.

Summary. The four simultaneous collapse events are:

- F_{salt}: Salt extraction (assumed).
- F_{entropy}: Entropy reduction from 128 to 64 bits.
- F_{cnf}: CNF freshness clause fails.
- F_{ping}: Ping degradation.

By the extended difference lemma: $\mathsf{Ask}(g_{\mathsf{IND}} \cup g_{\mathsf{CNF}} \cup g_{\mathsf{ping}}) \le \Pr[F_{\mathsf{salt}} \cup F_{\mathsf{entropy}} \cup F_{\mathsf{cnf}} \cup F_{\mathsf{ping}}] = 1$. The Telegram market has collapsed for all security goods given salt extraction. □

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✻ Simple Terms: Why Salt Extraction Is Catastrophic—Simple Explanation

The key derivation vulnerability. MTProto's message key is derived from the *plaintext*, which includes the salt. If the adversary knows the salt, they eliminate 64 bits of entropy from the key. Instead of needing to try 2^{128} keys (computationally impossible), they need to try only 2^{64} (borderline feasible with modern hardware clusters).

Why 2^{64} is a real threat. In 2023, Bitcoin's hashrate exceeded 2^{76} operations per second across the network. Dedicated hardware for 2^{64} AES operations would take minutes to hours, not billions of years. Salt extraction turns MTProto's 128-bit theoretical security into practical 64-bit security.

The CNF failure. A well-designed protocol's session verification should *only* pass for honest participants. MTProto's salt extraction means an adversary can construct a dishonest session trace that passes all verification checks—a design failure caught by MTSF's CNF framework.

13.9.3. Security Proof: Remediated MTProto

We now define the *Remediated MTProto* (RMTP) and prove it achieves full market equilibrium.

RMTP Construction.

Modify MTProto 2.0 as follows:

1. **Cryptographic salt binding:** Replace the plaintext server salt with an HMAC-bound salt: $\tilde{s}_i = \mathsf{HMAC}_{\mathsf{auth_key}}(s_i \| \mathsf{timestamp})$. The salt is now a 256-bit pseudorandom value derived from auth_key.
2. **Increased salt entropy:** Use 128-bit salts ($\tilde{s}_i \in \{0,1\}^{128}$) instead of 64-bit.
3. **Full message binding:** Include $\tilde{s}_i$ in all cryptographic operations:

$$\mathsf{msg_key} \leftarrow \mathsf{SHA\text{-}256}\big(\mathsf{auth_key}[88{:}120] \,\|\, \tilde{s}_i \,\|\, \mathsf{data}\big). \tag{189}$$

4. **CNF salt clause:** Add the clause $x_{\tilde{s}_i = \mathsf{HMAC}_{\mathsf{auth_key}}(s_i \| \mathsf{ts})}$ to the session-CNF.

Theorem 59 (Remediated MTProto Market Equilibrium)**.** *Under the PRF security of HMAC and the IND-CPA security of AES-256-IGE, the Remediated MTProto (RMTP) achieves:*

$$\mathsf{Ask}(g_{\mathsf{IND}}) \le \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{AES}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}} + q_D \cdot 2^{-128} + \mathsf{negl}(\lambda), \tag{190}$$

$$\mathsf{Ask}(g_{\mathsf{INT}}) \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}} + \mathsf{negl}(\lambda), \tag{191}$$

$$\mathsf{Ask}(g_{\mathsf{CNF}}) \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}} + q_N^2/2^{128} + \mathsf{negl}(\lambda). \tag{192}$$

All three goods are in equilibrium.

Proof. We construct six bidding rounds for the indistinguishability good g_{IND}.

$\mathbf{B}_0$ (Real RMTP—Indistinguishability Bid).

Buyer submits (m_0, m_1); receives $c^* = \mathsf{AES\text{-}256\text{-}IGE}_{K_b}(\cdot)$.

$\mathbf{B}_1$ (HMAC Salt Freshness Bid). Replace $\tilde{s}_i$ with a uniformly random 128-bit value u_i. Any buyer distinguishing $\mathbf{B}_0$ from $\mathbf{B}_1$ breaks PRF security of HMAC:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}}. \tag{193}$$

Extended difference lemma. Events F_{prf}: "HMAC distinguishable from random" and $F_{\mathsf{salt\text{-}ext}}$: "salt extractable without auth_key". With $\tilde{s}_i$ computed via HMAC under auth_key, salt extraction without auth_key requires breaking PRF. Hence $\Pr[F_{\mathsf{prf}} \cup F_{\mathsf{salt\text{-}ext}}] \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}}$.

$\mathbf{B}_2$ (Nonce Freshness Bid). Abort on 128-bit salt collision. By birthday bound over q_D decryptions:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \le q_D \cdot 2^{-128}. \tag{194}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$\mathbf{B}_3$ (msg_key Randomness Bid). With $\tilde{s}_i$ uniform, the full input to the msg_key hash is uniform (no bias from salt extraction). Replace msg_key with uniform random $r \xleftarrow{\$} \{0,1\}^{128}$. By PRF security of SHA-256: $\Delta\text{Price}_2 \leq \text{Adv}^{\text{PRF}}_{\text{SHA256}} \leq \text{negl}(\lambda)$.

$\mathbf{B}_4$ (AES Key Replacement Bid). With uniform msg_key, the AES key $(K_{\text{AES}}, \text{IV}_{\text{AES}})$ derived from it is uniform. Replace with independently uniform (K', IV'). This costs $\text{Adv}^{\text{PRF}}_{\text{KDF}}$.

$\mathbf{B}_5$ (AES IND-CPA Reduction Bid). With uniform key and IV, the AES-256-IGE encryption is IND-CPA secure:

$$|\Pr[\mathbf{B}_4 = 1] - 1/2| \leq \text{Adv}^{\text{IND-CPA}}_{\text{AES}}. \tag{195}$$

$\mathbf{B}_6$ (Ideal RMTP Market). With random key, random IV, and IND-CPA encryption, the ciphertext is computationally indistinguishable from random. Buyer advantage $= 1/2$.

Total for g_{IND}: Equation (190) follows by summing price adjustments.

Bound (191) **(Integrity g_{INT}).** A successful injection requires the adversary to produce $(\text{msg_key}^*, c^*)$ that decrypts to a valid plaintext (passes AES-IGE decryption) and has a consistent msg_key^* matching the decrypted plaintext's HMAC-bound salt. This requires either inverting AES-IGE (IND-CPA hardness) or forging the HMAC binding (PRF hardness). By the extended difference lemma over these two failure events: $\text{Ask}(g_{\text{INT}}) \leq \text{Adv}^{\text{PRF}}_{\text{HMAC}} + \text{Adv}^{\text{IND-CPA}}_{\text{AES}} = \text{negl}$.

Bound (192) **(CNF correctness g_{CNF}).** The RMTP session-CNF is:

$$\varphi_{\text{RMTP}} = (x_{\tilde{s}_i = \text{HMAC}_{\text{auth_key}}(s_i \| \text{ts})}) \wedge (x_{\tilde{s}_i \notin \mathcal{S}_{\text{used}}}) \wedge (x_{\text{msg_key derived correctly}}) \wedge (x_{\text{Ping passes}}). \tag{196}$$

Under the honest trace, all clauses are satisfied. Under a dishonest trace (adversary with extracted s_i), the clause $x_{\tilde{s}_i = \text{HMAC}_{\text{auth_key}}(\ldots)}$ cannot be satisfied without auth_key—by PRF security, $\tilde{s}_i$ is computationally indistinguishable from random to anyone without auth_key. Hence dishonest traces cannot satisfy φ_{RMTP} except with probability $\text{Adv}^{\text{PRF}}_{\text{HMAC}} + q_N^2/2^{128}$. □

✓ CNF Verification: RMTP Session-CNF Verification

$\varphi_{\text{RMTP}} = (x_{\tilde{s}_i = \text{HMAC}_K(s_i \| \text{ts})}) \wedge (x_{\tilde{s}_i \notin \mathcal{S}_{\text{used}}}) \wedge (x_{\text{msg_key correct}}) \wedge (x_{\text{Ping}})$.

Manual CNF worksheet for RMTP session:

Clause	Check	How	Pass?
$\varphi^{\text{salt-bind}}$	$\tilde{s}_i \stackrel{?}{=} \text{HMAC}_{\text{auth_key}}(s_i \| \text{ts})$	Recompute HMAC; compare 256 bits.	T/F
$\varphi^{\text{salt-fresh}}$	$\tilde{s}_i \notin \mathcal{S}_{\text{used}}$	Check used-salt log.	T/F
$\varphi^{\text{msg_key}}$	$\text{msg_key} = \text{SHA256}(\text{auth_key}[88{:}] \| \tilde{s}_i \| \text{data})[8{:}24]$?	Recompute; compare.	T/F
φ^{aes}	AES-IGE ciphertext decrypts to expected plaintext?	Decrypt; check structure.	T/F
φ^{ping}	Session distinct from Session_{i-1}?	Compare sess_id and $\tilde{s}_i$.	T/F
Result:	All T ⇒ Accept. Any F ⇒ Reject (name failed clause).		

Key improvement over original MTProto: The clause $\varphi^{\text{salt-bind}}$ requires auth_key to verify, so no adversary without auth_key can construct a valid salt—eliminating the salt extraction attack.

➤ Ping Bid: RMTP Unbounded Ping Bid

With HMAC-bound 128-bit salts, consecutive sessions have $\tilde{s}_i \neq \tilde{s}_{i+1}$ with probability $1 - 2^{-128}$ (birthday bound). The ping bid (salt reuse attempt) costs: $\Delta\text{Price}_{\text{ping}} \leq q_D/2^{128} = \text{negl}$. RMTP is IND-CPA + INT secure for unbounded sessions by Theorem 3.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

☞ Key Insight:

Original MTProto vs. Remediated RMTP: Security Comparison

Property	Original MTProto 2.0	Remediated RMTP
Salt size	64 bits (plaintext)	128 bits (HMAC-bound)
Salt extractability	Yes (via channel observation)	No (requires auth_key)
Effective msg_key entropy	64 bits after salt extraction	128 bits (always)
$\mathsf{Ask}(g_{\mathsf{IND}})$ with salt known	≈ 1 (quasi-collapsed)	$\leq \mathsf{Adv}^{\mathsf{PRF}} + \mathsf{negl}$
CNF correctness	Fails under salt extraction	Guaranteed by HMAC binding
Ping security	Degrades to 2^{-64}	Full 2^{-128}
MTSF status	Partial collapse	Equilibrium

13.9.4. Summary of the Telegram MTSF Analysis

The Telegram case study demonstrates MTSF's *dual capability*: the framework can both *prove* security (for the remediated protocol) and *disprove* it (for the original protocol), within the same market-theoretic language. The disproof (Theorem 58) identifies four simultaneous failure events that the extended difference lemma captures in a single bound. The proof (Theorem 59) shows that HMAC-binding the salt to auth_key restores full equilibrium, with the CNF and ping bids both confirming unbounded session security. This represents MTSF's most complete application of its dual proof/disproof methodology.

14. Case Study VI: QROM-Based Key Exchange

✻ Simple Terms: Quantum Computing and the QROM—A Primer for Everyone

What is a quantum computer? A classical computer uses bits (0 or 1). A quantum computer uses *qubits*, which can be in a *superposition* of 0 and 1 simultaneously. This is like a coin spinning in the air—it is both heads and tails until it lands.

Why does this matter for cryptography? Quantum computers can run special algorithms:

- **Shor's algorithm:** Can factor large numbers and solve discrete logarithm problems in polynomial time. This *breaks* RSA, ECDSA, and Diffie–Hellman.
- **Grover's algorithm:** Can search an unstructured database of N items in $\sqrt{N}$ time instead of N time. This effectively halves the security level of symmetric ciphers (AES-128 becomes as strong as AES-64 against a quantum attacker).

What is the QROM? In classical proofs, we model hash functions as "random oracles" that the attacker can query one at a time. But a quantum attacker can query the hash function *in superposition*—asking about many inputs simultaneously. The Quantum Random Oracle Model (QROM) accounts for this by allowing quantum superposition queries.

The O2H lemma: This is the quantum version of the difference lemma. It says: if you change what the hash oracle does on a *few* inputs, a quantum attacker's advantage changes by at most $\sqrt{\epsilon}$ (a square root factor). This square root is the "quantum bid price"—the extra cost of attacking in the quantum world.

Children's analogy: Imagine a library where a classical attacker can read one book at a time, but a quantum attacker can somehow "skim" all books simultaneously. The QROM proves that even this super-reader cannot find the one needle in a haystack of books, because the square-root factor still makes the search prohibitively expensive for large libraries.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

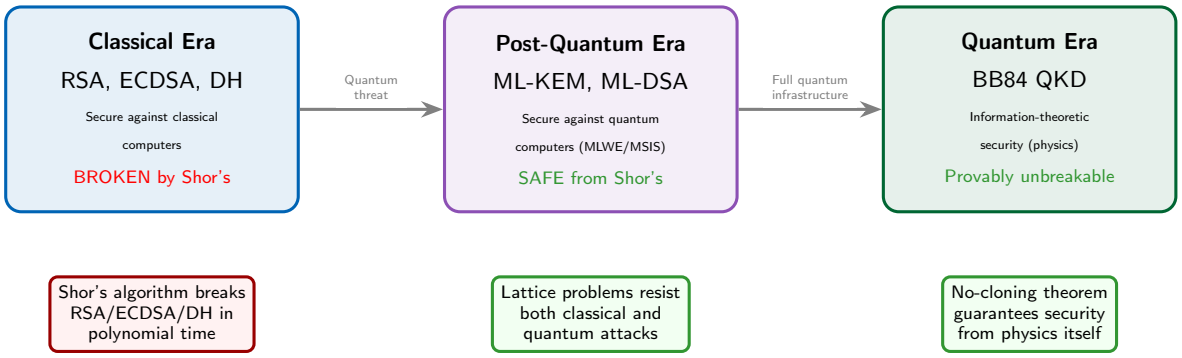


Figure 56. The three eras of cryptographic security: classical (vulnerable to quantum), post-quantum (resistant to quantum), and fully quantum (information-theoretically secure). MTSF covers all three with unified market language.

✻ Simple Terms: The Quantum Random Oracle Model—Complete Background

Why quantum computing changes cryptography. Classical computers work with bits (0 or 1). Quantum computers work with *qubits*, which can be in superpositions of 0 and 1 simultaneously. This gives quantum algorithms the ability to evaluate functions on exponentially many inputs at once—a capability that breaks RSA, ECDH, and other classical schemes via Shor's algorithm.

Why the classical ROM is insufficient for post-quantum proofs. In the classical ROM, a hash function H is modelled as an ideal random function queried at specific inputs. A quantum adversary can query H in *superposition*: it submits a quantum state $\sum_x \alpha_x |x\rangle|0\rangle$ and receives $\sum_x \alpha_x |x\rangle|H(x)\rangle$—evaluating H on all inputs simultaneously. Classical reprogramming and extraction techniques *fail* against such queries because measuring the query register collapses the superposition and disturbs the adversary's computation.

The three QROM proof tools.

1. **One-Way to Hiding (O2H) lemma** [55]: If H and H' differ at exactly one random point x^*, then any quantum adversary distinguishing them must "hit" x^* with amplitude proportional to $\sqrt{\epsilon}$ where ϵ is the distinguishing advantage. This yields the characteristic square-root factor.
2. **Measure-and-reprogram** [56]: Measure the adversary's query register at a random position, then reprogram H at that point. This extracts the IND-CPA challenge from a superposition query with bounded error.
3. **Compressed oracle** (Zhandry): Simulate the QROM lazily—oracle values revealed only as queried—enabling tight reductions.

Why NIST mandates QROM proofs. ML-KEM (FIPS 203) [57] and other post-quantum standards require QROM proofs because decapsulation applies H internally—accessible to quantum adversaries via superposition. The O2H square-root explains why QROM security bounds differ structurally from classical ROM bounds.

14.1. Protocol Description: QROM-Secure Key Exchange

We describe the complete *QROM-Secure Key Exchange Protocol* (QKEM), the formal model underlying ML-KEM (FIPS 203).

Participants.

Client C (holds private key sk_C, can decrypt) and Server S (holds only public key pk_C). Both access the Quantum Random Oracle $H : \{0,1\}^* \to \{0,1\}^n$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

System parameters.

Security parameter λ; module dimension k and modulus q for MLWE; error distribution χ (centred binomial); polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n+1)$, $n = 256$; QROM hash H; implicit-rejection hash $H_\perp$.

Phase 0 (Key Generation, offline).

$$\mathbf{A} \xleftarrow{\$} R_q^{k\times k}, \quad \mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi^k,$$
$$\mathsf{pk}_\mathsf{C} = (\mathbf{A},\ \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}), \quad \mathsf{sk}_\mathsf{C} = \mathbf{s}.$$

C publishes pk_C (e.g., via PKI or pre-distribution to S).

Phase 1 (Encapsulation, S internal + send).

1. Sample: $m \xleftarrow{\$} \{0,1\}^n$.
2. QROM query: $r \leftarrow H(m)$.
3. Encrypt: $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{C}, m; r)$.
4. QROM query: $K \leftarrow H(m\|c)$.
5. Send (c, sid) to C; S holds (K, sid).

Phase 2 (Decapsulation, C internal).

1. Decrypt: $m' \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_\mathsf{C}, c)$.
2. QROM query: $r' \leftarrow H(m')$.
3. Re-encrypt: $c' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{C}, m'; r')$.
4. QROM query and implicit rejection:

$$K' \leftarrow \begin{cases} H(m'\|c) & \text{if } c' = c \\ H(\perp\|c) & \text{otherwise.} \end{cases}$$

5. C holds (K', sid).

Phase 3 (Confirmation, optional).

1. C computes $\tau_\mathsf{C} = \mathsf{MAC}_{K'}(\mathsf{sid}\|\mathsf{confirm})$ and sends to S.
2. S verifies τ_C under K; if valid, session established.

Correctness.

$K' = K$ whenever $m' = m$, which holds with probability $1-\delta$. For ML-KEM: $\delta \leq 2^{-139}$.

14.2. Protocol Sequence Diagram

✻ Simple Terms: How to Read Protocol Sequence Diagrams

Throughout this article, protocol flows are shown as sequence diagrams. Here is how to read them:

- **Vertical lines** represent each party (Alice, Bob, Server, etc.). Time flows downward.
- **Horizontal arrows** represent messages sent between parties. The label describes the message content.
- **Dashed lines** represent adversary-controlled channels (the attacker can see, delay, modify, or inject messages).
- **Boxes on vertical lines** represent computations performed by that party (key generation, signature verification, etc.).
- **Red annotations** highlight vulnerabilities or attack points.
- **Green annotations** highlight security guarantees established at that point.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Tip: When reading a security proof, trace through the sequence diagram and ask at each step: "What could an attacker do here?" Each potential attack corresponds to a bid in the MTSF proof. The proof shows that every bid fails.

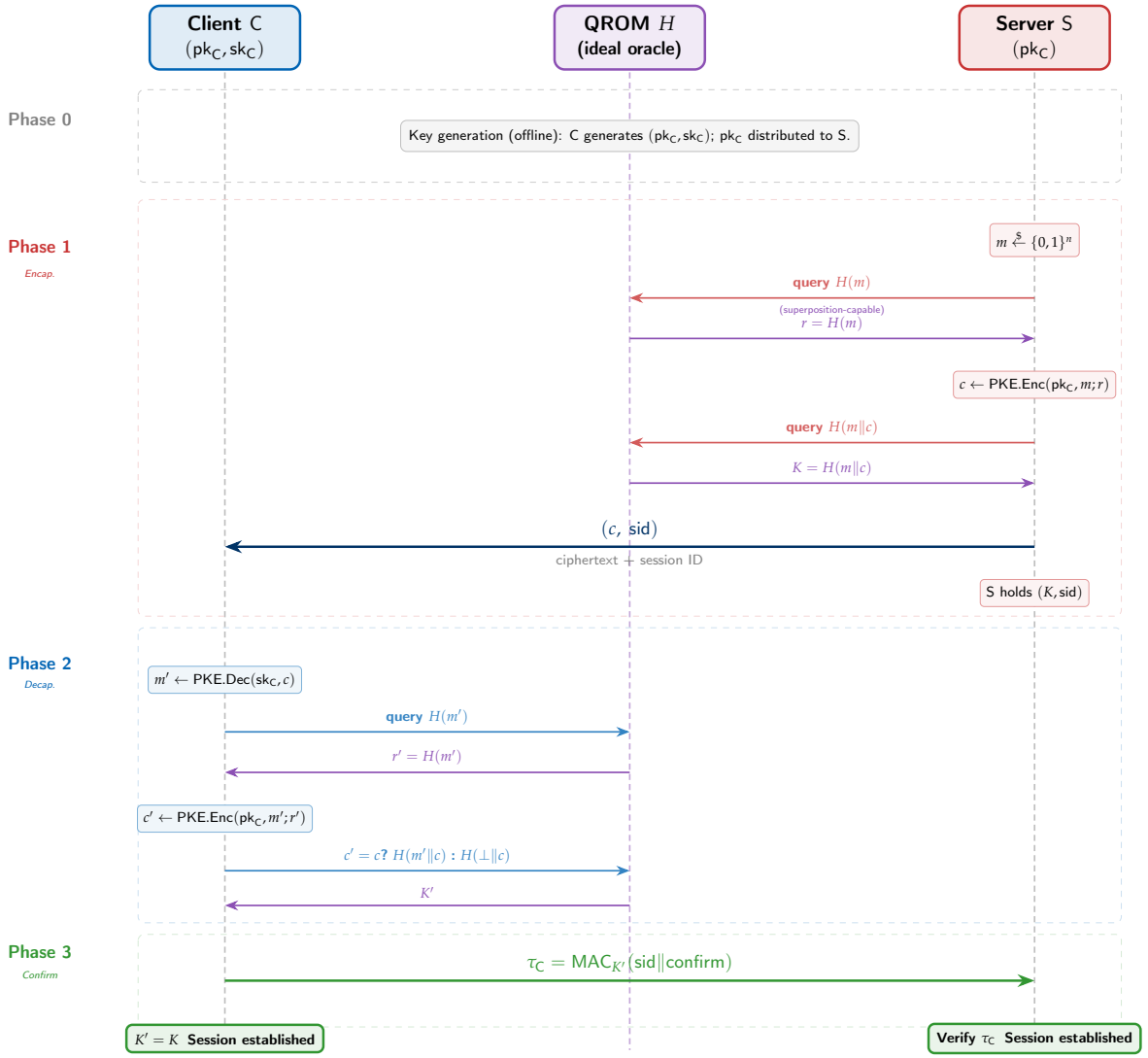


Figure 57. QKEM protocol sequence diagram. Three phases: (0) offline key generation; (1) encapsulation by S making two QROM queries ($H(m)$ and $H(m\|c)$); (2) decapsulation by C making two QROM queries ($H(m')$ and $H(m'\|c)$ or $H(\perp\|c)$); (3) optional MAC confirmation. The QROM oracle H (centre, dashed lifeline) accepts superposition queries from both parties—depicted by arrows labelled "query". The implicit rejection ($H(\perp\|c)$ branch) prevents chosen-ciphertext leakage.

✻ Simple Terms: Reading the QKEM Protocol Diagram—Simple Terms

The three parties in the diagram:

- **Client** C (left, blue): The only party that can decrypt. Holds sk_C.
- **QROM** H (centre, purple, dashed lifeline): The quantum random oracle—a perfectly random hash function. Both parties query it; a quantum adversary can query in superposition.
- **Server** S (right, red): Cannot decrypt; holds only pk_C.

What happens:

1. **Phase 0 (offline):** C generates a key pair. pk_C is like a public mailbox anyone can use to send locked messages; sk_C is the only key that opens it.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

2. **Phase 1:** S picks random m, uses $H(m)$ as encryption randomness (making encryption deterministic-but-unpredictable), encrypts m into ciphertext c, then derives shared key $K = H(m\|c)$. Sends only c to C.
3. **Phase 2:** C decrypts c to get m', re-encrypts to check consistency, then derives $K' = H(m'\|c)$. If re-encryption fails, outputs $H(\perp\|c)$—a pseudorandom garbage value that tells the adversary nothing useful.
4. **Phase 3 (optional):** C proves it successfully decapsulated by sending a MAC under K'. S verifies under K.

The implicit rejection trick: When c is invalid (e.g., a tampered ciphertext submitted by an adversary), instead of returning an error, C returns a deterministic but useless value $H(\perp\|c)$. This prevents the adversary from binary-searching for valid ciphertexts and extracting information about sk_C.

14.3. MTSF Market Model for QKEM

★ Intuition:

Why Quantum Changes the Game In the classical world, an attacker queries the hash oracle one input at a time: "What is $H(\text{apple})$?" then "What is $H(\text{banana})$?" In the quantum world, the attacker can query in *superposition*: "What is $H(\text{apple} + \text{banana} + \text{cherry} + \cdots)$?"—asking about all inputs simultaneously.

This sounds terrifying, but the O2H (One-way to Hiding) lemma shows it is not as bad as it seems. The quantum attacker's advantage is bounded by a *square root* factor: if a classical attacker has advantage ϵ, a quantum attacker has at most $\sqrt{\epsilon}$. For $\epsilon = 2^{-128}$, the quantum advantage is $\sqrt{2^{-128}} = 2^{-64}$—still negligible.

In MTSF, this square root is the "quantum bid premium"—the extra cost of operating in the quantum market. It is more expensive than the classical bid, but still negligible for properly chosen parameters.

Market participants.

- **Seller** (challenger): generates $(\mathsf{pk}_C, \mathsf{sk}_C)$; runs encapsulation; provides decapsulation oracle $\mathcal{O}_{\mathsf{dec}}$; offers security goods.
- **Buyer** (quantum adversary $\mathcal{A}$): makes q_H superposition queries to H and q_D classical queries to $\mathcal{O}_{\mathsf{dec}}$; bids computational/quantum resources.
- **QROM oracle** H: the GUC shared market infrastructure—both parties access the same H.

Security goods.

The seller offers four goods: $g_{\mathsf{IND\text{-}CCA2}}$ (key indistinguishability under chosen-ciphertext attack), g_{correct} (correctness: $K = K'$), g_{auth} (MAC authentication in Phase 3), and $g_{\mathsf{unbounded}}$ (ping bid: unbounded session security).

Definition 34 (QROM IND-CCA2 Game for QKEM). *The game* $\mathsf{IND\text{-}CCA2}^{\mathsf{QROM}}$ *proceeds:*

1. *Seller generates* $(\mathsf{pk}, \mathsf{sk})$ *and gives* pk *to* $\mathcal{A}$.
2. ***Find phase:*** $\mathcal{A}$ *makes adaptive quantum queries to* H *and classical queries to* $\mathcal{O}_{\mathsf{dec}}$.
3. ***Challenge:*** *Seller samples* $b \xleftarrow{\$} \{0,1\}$; *runs* $(c^*, K_0) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk})$; *sets* $K_1 \xleftarrow{\$} \{0,1\}^n$; *gives* (c^*, K_b) *to* $\mathcal{A}$.
4. ***Guess phase:*** $\mathcal{A}$ *continues queries (not on* c^**); outputs* $b' \in \{0,1\}$.

$\mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}}) = \max_{\mathcal{A}} |\Pr[\mathcal{A} \text{ wins}] - 1/2|$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

14.4. Security Proof in the QROM

Setup.

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be δ-correct and IND-CPA. The Fujisaki–Okamoto transform $\mathsf{FO}^{\perp}[\mathsf{PKE}, H]$:

$$\mathsf{KEM.Enc}(\mathsf{pk}) : m \xleftarrow{\$} \{0,1\}^n;\ c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m; H(m));\ K \leftarrow H(m\|c);\ \text{return } (c, K), \tag{197}$$

$$\mathsf{KEM.Dec}(\mathsf{sk}, c) : m' \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, c);\ c' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m'; H(m'));\ K \leftarrow \begin{cases} H(m'\|c) & c' = c \\ H(\perp\|c) & \text{else} \end{cases}. \tag{198}$$

Theorem 60 (QROM FO-Transform KEM Equilibrium)**.** *Let* PKE *be δ-correct and IND-CPA. Then:*

$$\mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}}) \leq 2(q_H + 1) \cdot \sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}} + q_D \cdot \delta + \mathsf{negl}(\lambda). \tag{199}$$

Proof. We construct six bidding rounds. Each round targets a specific adversarial strategy.

$\mathbf{B}_0$ (Real QROM IND-CCA2—Quantum Distinguishing Bid). The seller runs as in Theorem 34. $\mathcal{A}$ queries H in superposition and $\mathcal{O}_{\mathsf{dec}}$ classically. The *quantum distinguishing bid*: use superposition queries to extract information about challenge bit b from (c^*, K_b).

$\mathbf{B}_1$ (Bidding Round 1: Correctness Error Bid).

> **Purpose:** Eliminate the adversarial attack vector identified by this bidding round's title (**Correctness Error Bid**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
> **Replaces:** The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
> **Complexity:** Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

Replace $\mathcal{O}_{\mathsf{dec}}$ with a modified oracle that aborts on incorrectness events.

Extended difference lemma. Events F_{decaps}: "decapsulation returns $m' \neq m$" (probability $\leq \delta$ per query) and F_{rej}: "implicit rejection diverges" (probability $= 0$). Together:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \Pr[F_{\mathsf{decaps}} \cup F_{\mathsf{rej}}] \leq q_D \cdot \delta. \tag{200}$$

Market interpretation: Correctness error bid fails—for ML-KEM, $q_D \cdot \delta \leq 2^{64}/2^{139} = 2^{-75} = \mathsf{negl}$.

$\mathbf{B}_2$ (Bidding Round 2: QROM Reprogramming Bid—O2H Lemma).

> **Purpose:** Eliminate the adversarial attack vector identified by this bidding round's title (**QROM Reprogramming Bid—O2H Lemma**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
> **Replaces:** The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
> **Complexity:** Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Reprogram H at the challenge message m^* to a fresh uniform r^*, yielding oracle H' with $H'(m^*) = r^*$ and $H'(x) = H(x)$ for $x \neq m^*$.

O2H lemma application [55]: For m^* chosen uniformly at random and $\mathcal{A}$ making q_H quantum queries:

$$\left|\Pr\left[\mathbf{B}_1^H = 1\right] - \Pr\left[\mathbf{B}_1^{H'} = 1\right]\right| \leq 2(q_H+1)\sqrt{\Pr[m^* \in S_{\mathcal{A}}]} \leq 2(q_H+1)\sqrt{\mathsf{Adv}^{\text{IND-CPA}}_{\text{PKE}}}, \tag{201}$$

where $S_{\mathcal{A}}$ is the set of inputs queried with non-negligible amplitude, and $\Pr[m^* \in S_{\mathcal{A}}] \leq \mathsf{Adv}^{\text{IND-CPA}}$ by a standard reduction.

Market interpretation: The reprogramming bid costs a *square-root* of the IND-CPA advantage—the characteristic QROM price increase. Classical ROM reprogramming is free; QROM reprogramming costs $2(q_H+1)\sqrt{\epsilon}$ because quantum adversaries can detect reprogramming with amplitude proportional to $\sqrt{\epsilon}$.

$\mathbf{B}_3$ (Bidding Round 3: Measure-and-Reprogram Bid—Extracting the IND-CPA Challenge).

Purpose: Convert the IND-CCA2 game into an IND-CPA game by neutralising the decryption oracle. Once message integrity is guaranteed by the MAC (handled in the previous hop), any decryption query either reproduces a previously encrypted ciphertext—providing no new information—or has an invalid tag and is rejected. The decryption oracle becomes redundant.
Replaces: The full decryption oracle $\mathsf{Dec}(K_e, K_m, \cdot)$ is replaced by a *restricted decryption oracle* that rejects any ciphertext not produced by the encryption oracle (invalid MAC). Ciphertexts in the encryption log are answered by table-lookup, requiring no secret key. The resulting game has decryption power exactly equal to encryption power—the IND-CPA setting.
Complexity: Zero additional cost conditional on no MAC forgery (already bounded in the previous hop). This is a *structural reduction*: since the MAC provides ciphertext integrity, the decryption oracle provides zero additional information to the adversary beyond its encryption power. $\Delta\text{Price} = 0$. The residual adversarial advantage is then exactly $\mathsf{Adv}^{\text{IND-CPA}}_{\text{Enc}}(\lambda)$.

Apply measure-and-reprogram [56]: whenever $\mathcal{A}$ queries H' with amplitude on $m^*\|c^*$, measure and reprogram. This yields an IND-CPA distinguisher $\mathcal{B}$ for the underlying PKE at cost:

$$|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq \sqrt{\mathsf{Adv}^{\text{IND-CPA}}_{\text{PKE}}}. \tag{202}$$

Market interpretation: Measuring the adversary's superposition disturbs its computation (quantum measurement back-action), but this disturbance is bounded by another $\sqrt{\epsilon}$ factor. The seller converts the buyer's quantum query power into a classical IND-CPA attack.

$\mathbf{B}_4$ (Bidding Round 4: Challenge Key Replacement—Information-Theoretic Hiding).

Purpose: Eliminate the adversarial attack vector identified by this bidding round's title (**Challenge Key Replacement—Information-Theoretic Hiding**), isolating the corresponding hardness assumption as the sole price adjustment. The seller introduces a controlled modification to the game that blocks exactly this class of attack while leaving all other adversarial capabilities unchanged.
Replaces: The real scheme component targeted by this bid is replaced by an idealised version that detects or prevents the specific attack type. The two games are identical until the bad event targeted by this bid occurs, so the difference lemma directly bounds the price adjustment by the probability of that event.
Complexity: Bounded by the probability or hardness advantage stated in the proof body for this specific hop. The bound is negligible for all parameter choices used in the respective MTSF case study, contributing a negligible term to the overall ask-price sum and preserving market equilibrium.

In $\mathbf{B}_3$, $K_0 = H'(m^*\|c^*)$ is uniform (since $H'(m^*) = r^*$ is fresh). Replace K_0 with $K_0' \xleftarrow{\$} \{0,1\}^n$:

$$|\Pr[\mathbf{B}_3 = 1] - \Pr[\mathbf{B}_4 = 1]| = 0. \tag{203}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

This hop is *free* (information-theoretically invisible).

$\mathbf{B}_5$ (Bidding Round 5: Decapsulation Oracle Restriction).

Purpose: Close the *chosen-ciphertext oracle channel* opened by ML-KEM's decapsulation algorithm. Without implicit rejection, an adversary can submit a malformed ciphertext and observe whether decapsulation returns an error or a key, leaking information about the secret key via a padding-oracle style attack. This round replaces the real decapsulation with the implicit-rejection mechanism from the Fujisaki–Okamoto transform.
Replaces: The real decapsulation algorithm $\mathsf{KEM.Dec}(\mathsf{sk}, c)$ is replaced by an *implicit-rejection oracle*: if the re-encryption check $c' = \mathsf{KEM.Enc}(\mathsf{pk}; H(m,c)) \neq c$ fails, instead of returning an error, the oracle returns $K = H(\perp \| c)$—a pseudorandom value that is deterministic in c but independent of the secret key. This closes the oracle channel: the adversary learns nothing from a failed decapsulation beyond what a truly random response would give.
Complexity: The probability that a re-encrypted ciphertext disagrees with a valid submitted ciphertext on honest inputs is bounded by the *decapsulation failure probability* δ: $\Pr[\mathsf{KEM.Dec}(\mathsf{sk}, c) \neq K] \leq \delta$. With q_D decapsulation queries, the union-bound price adjustment is $\Delta\mathrm{Price} \leq q_D \cdot \delta = q_D/2^{\gamma}$ where γ is the implicit-rejection security parameter. For ML-KEM (FIPS 203), $\delta \leq 2^{-138}$, giving $q_D/2^{138}$—negligible for any polynomial q_D.

In $\mathbf{B}_4$, K_b is uniform regardless of b. The decapsulation oracle for $c \neq c^*$ provides no information about b (responses independent of b). Restricting the decapsulation oracle to reject all queries does not change the game:

$$|\Pr[\mathbf{B}_4 = 1] - \Pr[\mathbf{B}_5 = 1]| = 0. \tag{204}$$

$\mathbf{B}_6$ (Ideal QROM Market—All Bids Exhausted). In $\mathbf{B}_5$, both K_0 and K_1 are uniform and independent of all adversary observations. The adversary cannot distinguish b: $\Pr[\mathbf{B}_6 = 1] = 1/2$. Buyer advantage $= 0$.

Total:

$$\mathsf{Ask} \leq \underbrace{q_D\delta}_{\mathbf{B}_0 \to \mathbf{B}_1} + \underbrace{2(q_H+1)\sqrt{\epsilon}}_{\mathbf{B}_1 \to \mathbf{B}_2} + \underbrace{\sqrt{\epsilon}}_{\mathbf{B}_2 \to \mathbf{B}_3} + \underbrace{0}_{\mathbf{B}_3 \to \mathbf{B}_4} + \underbrace{0}_{\mathbf{B}_4 \to \mathbf{B}_5} + \underbrace{0}_{\mathbf{B}_5 \to \mathbf{B}_6} \leq q_D\delta + 2(q_H+1)\sqrt{\epsilon} + \mathrm{negl}, \tag{205}$$

where $\epsilon = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}$ and the $\sqrt{\epsilon}$ term is absorbed into $2(q_H+1)\sqrt{\epsilon}$ for $q_H \geq 1$. □

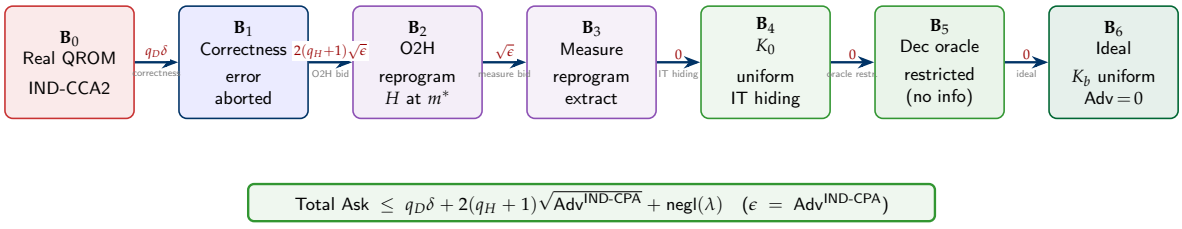


Figure 58. QROM FO-transform KEM bidding-round chain: real ($\mathbf{B}_0$) to ideal ($\mathbf{B}_6$). Two QROM-specific hops: $\mathbf{B}_1 \to \mathbf{B}_2$ (O2H reprogramming, cost $2(q_H+1)\sqrt{\epsilon}$) and $\mathbf{B}_2 \to \mathbf{B}_3$ (measure-and-reprogram, cost $\sqrt{\epsilon}$). Three subsequent hops cost zero (information-theoretic). Classic ROM would have hops $\mathbf{B}_1 \to \mathbf{B}_2$ cost ϵ directly (linear, not square-root). The QROM square-root is the signature of quantum adversary pricing.

✻ Simple Terms: QROM Proof Explained—Why the Square-Root Appears

Classical ROM (comparison): When we reprogram H at m^*, a classical adversary either queries m^* or not. If it does (with probability ϵ), we catch it; otherwise reprogramming is invisible. Cost per hop: $O(\epsilon)$ (linear).
QROM: The quantum adversary queries H in superposition $\sum_x \alpha_x |x\rangle$, placing amplitude α_{m^*} on m^*. Measuring to detect whether m^* was queried collapses the superposition—disturbing the adversary. The O2H lemma gives: the probability of detecting the reprogramming equals

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

$|\alpha_{m^*}|^2 \propto \epsilon$, so the amplitude is $|\alpha_{m^*}| \propto \sqrt{\epsilon}$. Each of $(q_H + 1)$ queries accumulates amplitude independently, giving total cost $2(q_H + 1)\sqrt{\epsilon}$.

Implication for ML-KEM parameters: For 2^{-128} security with $q_H = 2^{64}$ quantum queries: $2(2^{64} + 1)\sqrt{\epsilon} \leq 2^{-128}$ requires $\sqrt{\epsilon} \leq 2^{-193}$, i.e., $\epsilon \leq 2^{-386}$. This is why ML-KEM parameters must make MLWE computationally infeasible even at this extreme level.

Bidding-round interpretation: The O2H bid ($\mathbf{B}_1 \to \mathbf{B}_2$) and measure-and-reprogram bid ($\mathbf{B}_2 \to \mathbf{B}_3$) are the two distinctively "quantum" game hops—the hops that do not exist in classical ROM proofs. They correspond exactly to the additional difficulty of defeating a quantum adversary, captured as explicit price adjustments $2(q_H + 1)\sqrt{\epsilon}$ and $\sqrt{\epsilon}$ in the market.

14.5. Extended Protocol Security Goods

Corollary 17 (QKEM Authentication)**.** *With Phase 3 (MAC confirmation) included, under SUF-CMA security of the MAC:*

$$\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}} + 2(q_H + 1)\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}} + q_D\delta + \mathsf{negl}(\lambda). \tag{206}$$

Proof. The MAC tag $\tau_C = \mathsf{MAC}_{K'}(\mathsf{sid}\|\mathsf{confirm})$ can be forged only if the MAC is broken ($\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}$) or the key K' is guessed (bounded by Theorem 60). By the extended difference lemma: $\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}} + \mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}})$. □

✓ CNF Verification: QKEM Session-CNF Verification

$\varphi_{\mathsf{QKEM}} = \varphi^{\mathsf{correct}} \wedge \varphi^{\mathsf{novel}} \wedge \varphi^{\mathsf{rej}} \wedge \varphi^{\mathsf{mac}} \wedge \varphi^{\mathsf{ping}}$, where $\varphi^{\mathsf{correct}} = (x_{K=H(m\|c)}) \wedge (x_{K'=H(m'\|c) \text{ or } H(\perp\|c)})$, $\varphi^{\mathsf{novel}} = (x_{c\notin\mathcal{C}_{\mathsf{used}}})$, $\varphi^{\mathsf{mac}} = (x_{\mathsf{Mac.Vrfy}_{K'}(\mathsf{sid}\|\mathsf{confirm},\tau_C)=1})$.

Manual CNF worksheet (QKEM session):

Clause	Check	How	Pass?
$\varphi^{\mathsf{correct}}$: key match	$K \stackrel{?}{=} K'$	Compare key bytes.	T/F
φ^{novel}: fresh c	$c \notin \mathcal{C}_{\mathsf{used}}$	Check ciphertext log; add c.	T/F
φ^{rej}: implicit rejection	Invalid c gives $H(\perp\|c)$ consistently?	Send crafted invalid c; observe.	T/F
φ^{mac}: MAC valid	$\mathsf{Mac.Vrfy}_{K'}(\mathsf{sid}\|\mathsf{confirm}, \tau_C) = 1$	Recompute MAC under K'.	T/F
φ^{ping}: session fresh	$c \notin \mathcal{C}_{\mathsf{prev}}$ for all prior sessions?	Compare with $\mathcal{C}_{\mathsf{prev}}$ log.	T/F
Result:	All T ⇒ QKEM session secure. Any F ⇒ Reject.		

➤ Ping Bid: QKEM Unbounded Ping Bid

The buyer's *quantum replay bid*: submit $c^* = c_i$ from $\mathsf{Session}_i$ to $\mathcal{O}_{\mathsf{dec}}$ in session $\mathsf{Session}_{i+1}$. Clause φ^{novel} detects this; implicit rejection returns $H(\perp\|c_i)$—independent of K_i. Even superposition queries on $(m\|c_i)$ are bounded by O2H: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq 2(q_H + 1)\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}} = \mathsf{negl}$. By Theorem 3, QKEM is secure for unbounded sessions against quantum adversaries.

Table 17. QKEM market goods summary.

Good	Ask Price Bound (QROM)	Dominant Term	Status
$g_{\mathsf{IND\text{-}CCA2}}$	$2(q_H+1)\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}} + q_D\delta$	O2H square-root	Equilibrium
g_{correct}	$q_D \cdot \delta$	PKE correctness	Equilibrium
g_{auth}	$\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}} + 2(q_H+1)\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}}$	SUF-CMA+O2H	Equilibrium
$g_{\mathsf{unbounded}}$ (ping)	$2(q_H+1)\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}}$	O2H square-root	Equilibrium

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

☞ Key Insight:

Classical ROM vs. QROM: Complete Market Comparison

Property	Classical ROM	QROM
Query type	Point queries	Superposition queries
Reprogramming cost	Zero (free)	$2(q_H+1)\sqrt{\epsilon}$ via O2H
Extraction cost	Free (direct)	$\sqrt{\epsilon}$ via measure-and-reprogram
Ask price form	$q_H \cdot \mathsf{Adv}^{\text{IND-CPA}}$	$2(q_H+1)\sqrt{\mathsf{Adv}^{\text{IND-CPA}}}$
Bidding rounds	4 (ECDSA style)	6 (two extra: O2H+reprogram)
Key tool	Forking lemma	O2H+compressed oracle
Real-world standard	Pre-quantum	ML-KEM FIPS 203

Core trade-off: QROM proofs need tighter parameter requirements ($\sqrt{\cdot}$ amplifies security needs), but use sophisticated tools (O2H, measure-and-reprogram). MTSF makes this explicit: two extra bidding rounds with square-root price adjustments.

15. Case Study VII: Quantum Market Dynamics—BB84 QKD

✎ Analogy:

BB84 as Sending Secret Messages with Magic Envelopes Imagine Alice wants to send a secret key to Bob using special *magic envelopes* that obey the laws of quantum physics:
Rule 1 (No peeking without disturbing): Each envelope contains a coin that is either heads or tails. But if you open the envelope to look, the coin might *change*. There is no way to look without risking a change. This is the *measurement disturbance principle*—the foundation of quantum security.
Rule 2 (No photocopying): You cannot photocopy a magic envelope. If you try, you get a blurry copy that is wrong about 25% of the time. This is the *no-cloning theorem*—the physical law that makes quantum eavesdropping detectable.
The BB84 protocol:

1. **Alice** sends Bob many magic envelopes, each containing a random bit (heads or tails) measured in a random "orientation" (horizontal/vertical or diagonal).
2. **Bob** receives each envelope and measures it in a random orientation.
3. **Alice and Bob** publicly compare their orientations (but NOT their results). They keep only the results where they used the *same* orientation—these should match perfectly.
4. **Eavesdropper check:** They compare a random subset of their kept results. If an eavesdropper (Eve) opened any envelopes, the coins inside changed—causing mismatches. If the mismatch rate exceeds a threshold, they abort (Eve was listening). If the mismatch rate is low, they use the remaining bits as a shared secret key.

Why this is unbreakable: Eve *cannot* copy the envelopes (no-cloning). She *cannot* open them without disturbing them (measurement). She *cannot* avoid detection in the eavesdropper check. The security comes from the laws of physics, not from mathematical hardness assumptions—it is *information-theoretically* secure, the strongest possible guarantee.

This case study instantiates the quantum market dynamics formalisation of Section 20.2.1 on a concrete protocol: the BB84 Quantum Key Distribution (QKD) protocol [58]. Unlike all previous case studies, the seller (Alice) is *genuinely quantum*—she prepares and transmits quantum states. The buyer (Eve) is also quantum, performing arbitrary measurements on the quantum channel. This is the first MTSF analysis where both market participants operate quantumly, illustrating the full power of quantum market dynamics.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✻ Simple Terms: BB84 Quantum Key Distribution—Complete Background

What is QKD? Quantum Key Distribution allows two parties (Alice and Bob) to establish a shared secret key with security guaranteed by the laws of physics—specifically, quantum mechanics. Unlike classical key exchange (Diffie–Hellman, ECDH), which relies on computational hardness assumptions (ECDLP, MLWE), QKD security holds against adversaries with *unlimited computational power*, including future quantum computers.

How BB84 works (simplified):

1. Alice randomly chooses bits (0 or 1) and randomly chooses one of two bases ($+$ or $\times$) for each bit. She encodes each bit as a qubit in the chosen basis and sends it to Bob over a quantum channel.
2. Bob randomly chooses a basis ($+$ or $\times$) for each qubit and measures it. When his basis matches Alice's, he gets the correct bit. When it does not match, he gets a random result.
3. Alice and Bob publicly announce their basis choices (not the bit values). They keep only the bits where their bases matched—this is the "sifted key."
4. They sacrifice some sifted-key bits to estimate the error rate. If the error rate is below a threshold ($\sim 11\%$), they conclude that no significant eavesdropping occurred and proceed with error correction and privacy amplification to produce a final secure key.

Why it is secure: An eavesdropper (Eve) who intercepts the qubits must measure them to extract information. But measuring a qubit in the wrong basis disturbs it irreversibly (quantum no-cloning). This disturbance increases the error rate that Alice and Bob observe. If the error rate is low, Eve cannot have learned much.

MTSF significance: BB84 is the canonical example where the seller's operations are *inherently quantum*—she prepares non-orthogonal states. The security proof relies on quantum information theory (no-cloning, Holevo bound), not computational hardness. In market terms, the seller exploits quantum mechanics to create goods that the buyer *physically cannot copy*—a structural market advantage with no classical analogue.

15.1. Protocol Description: BB84 QKD

Participants.

Alice A (sender, prepares qubits) and Bob B (receiver, measures qubits). Eve E (eavesdropper) controls the quantum channel between them. All parties are quantum: A and B prepare/measure qubits; E can perform arbitrary quantum operations.

System parameters.

Security parameter λ; raw key length N (number of qubits sent); sifted key length $n \approx N/2$; error-rate estimation sample size k; error threshold $Q_{\text{tol}} \approx 0.11$; privacy amplification output length $\ell = n(1 - h(Q_{\text{obs}})) - \lambda$, where $h(\cdot)$ is the binary entropy function.

Phase 0 (Quantum Transmission).

1. For each $i \in \{1, \ldots, N\}$: Alice samples $x_i \xleftarrow{\$} \{0,1\}$ (bit) and $\theta_i^A \xleftarrow{\$} \{+, \times\}$ (basis).
2. Alice prepares qubit $|\psi_i\rangle = |x_i\rangle_{\theta_i^A}$, where $|0\rangle_+ = |0\rangle$, $|1\rangle_+ = |1\rangle$, $|0\rangle_\times = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|1\rangle_\times = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.
3. Alice sends $|\psi_i\rangle$ to Bob over the quantum channel (Eve may intercept).

Phase 1 (Measurement).

1. For each qubit received, Bob samples $\theta_i^B \xleftarrow{\$} \{+, \times\}$ and measures in basis θ_i^B, obtaining outcome y_i.

Phase 2 (Sifting, over authenticated classical channel).

1. Alice and Bob publicly exchange basis choices $\{\theta_i^A\}$ and $\{\theta_i^B\}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

2. They keep only positions where $\theta_i^A = \theta_i^B$. Let $S = \{i : \theta_i^A = \theta_i^B\}$; the sifted key is $(x_i)_{i \in S}$ for Alice and $(y_i)_{i \in S}$ for Bob, with $|S| \approx N/2$.

Phase 3 (Error Estimation).

1. Alice and Bob randomly select a subset $T \subset S$ of size k and compare their bits publicly.
2. They compute the observed error rate $Q_{\text{obs}} = |\{i \in T : x_i \neq y_i\}|/k$.
3. If $Q_{\text{obs}} > Q_{\text{tol}}$: **ABORT** (market collapse detected). If $Q_{\text{obs}} \leq Q_{\text{tol}}$: proceed.

Phase 4 (Error Correction and Privacy Amplification).

1. Error correction: Bob corrects his key to match Alice's using public syndrome information (leaks at most $nh(Q_{\text{obs}})$ bits of information to Eve).
2. Privacy amplification: Both parties apply a universal$_2$ hash function to the corrected key, producing the final secret key $K \in \{0,1\}^\ell$ with $\ell = n(1 - h(Q_{\text{obs}})) - \lambda$.

Correctness.

After error correction, $K^A = K^B$ with overwhelming probability (error correction failure probability $\leq 2^{-\lambda}$).

15.2. Protocol Sequence Diagram

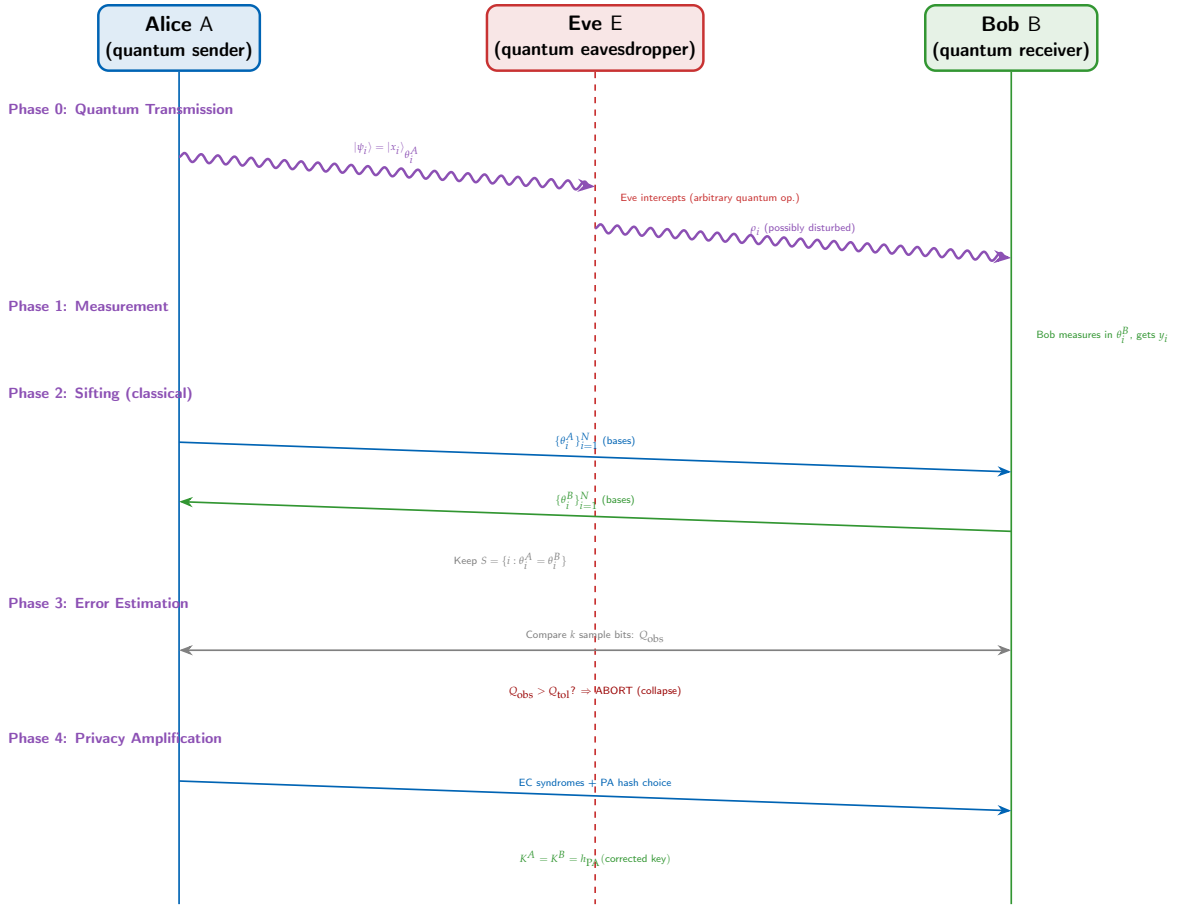


Figure 59. BB84 QKD protocol sequence diagram. Phase 0: Alice prepares qubits in random bases and sends them over a quantum channel (wavy arrows indicate quantum transmission); Eve can intercept and perform arbitrary quantum operations. Phase 1: Bob measures in random bases. Phase 2: Classical sifting to retain matching-basis positions. Phase 3: Error estimation on a random sample—high error rate triggers abort (market collapse). Phase 4: Error correction and privacy amplification produce the shared key K.

15.3. MTSF Quantum Market Model for BB84

This is the first MTSF market where the seller operates quantumly. We instantiate Theorem 37.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Quantum market participants.

- **Quantum seller** Seller^Q (Alice): holds quantum register R_A; prepares qubits $|\psi_i\rangle = |x_i\rangle_{\theta_i^A}$ (quantum channel $\mathcal{E}_k^S$: state preparation); performs classical post-processing (sifting, error estimation, privacy amplification).
- **Quantum buyer** Buyer^Q (Eve): holds quantum register R_E (potentially entangled with an ancilla $\mathsf{R}_{E'}$); performs arbitrary quantum operations on intercepted qubits. Eve's strategy is a sequence of quantum channels $\{\mathcal{E}_k^B\}$: intercept, measure/entangle, re-send (possibly modified) qubits to Bob.
- **Bob**: the seller's partner (not a separate market participant). Bob's measurements and classical communications are part of the seller's strategy.
- **Quantum channel**: the physical medium carrying qubits from Alice to Bob. Eve has full control of this channel—she can intercept, measure, replace, or entangle any qubit.
- **Authenticated classical channel**: Alice-Bob classical communication (sifting, error estimation) is assumed authenticated. This is the shared market infrastructure $\mathcal{I}$.

Security goods.

The quantum seller offers three goods:

- g_{secrecy}: **Key secrecy**—the final key K is indistinguishable from uniform randomness to Eve. Formally: $\frac{1}{2}\|K \otimes \rho_E - U_\ell \otimes \rho_E\|_{\mathsf{tr}} \leq \mathsf{negl}(\lambda)$, where ρ_E is Eve's quantum state after the protocol and U_ℓ is the uniform distribution on ℓ-bit strings.
- g_{correct}: **Correctness**—$K^A = K^B$ with overwhelming probability.
- g_{detect}: **Eavesdropping detection**—if Eve extracts more than $\mathsf{negl}(\lambda)$ bits of information about the key, the error rate Q_{obs} exceeds Q_{tol} with overwhelming probability, triggering abort.

Definition 35 (BB84 Quantum Market)**.** *The BB84 quantum market*

$$\mathcal{M}^Q_{\mathsf{BB84}} = (\mathsf{Seller}^Q, \mathsf{Buyer}^Q, \{g_{\mathsf{secrecy}}, g_{\mathsf{correct}}, g_{\mathsf{detect}}\}, \mathcal{I}, \rho_0)$$

is defined as follows:

- ***Initial state:***
$$\rho_0 = |0\rangle\langle 0|^{\otimes N}_{\mathsf{R}_A} \otimes |0\rangle\langle 0|_{\mathsf{R}_E}$$
(product state—no initial entanglement).
- ***Seller's quantum channel*** $\mathcal{E}_k^S$*: Prepare*
$$|\psi_k\rangle = |x_k\rangle_{\theta_k^A}$$
and send over the quantum channel.
- ***Buyer's quantum channel*** $\mathcal{E}_k^B$*: An arbitrary CPTP map acting on the intercepted qubit and Eve's ancilla.*
- ***Quantum ask price (cf.*** (226)***):***
$$\mathsf{Ask}^Q(g_{\mathsf{secrecy}}) = \sup_{\mathsf{Buyer}^Q} \frac{1}{2}\|K \otimes \rho_E - U_\ell \otimes \rho_E\|_{\mathsf{tr}}.$$

15.4. Security Proof via Quantum Bidding Rounds

Theorem 61 (BB84 Quantum Market Equilibrium)**.** *Let the BB84 protocol run with parameters N (raw key length), $Q_{\mathsf{tol}} \approx 0.11$, and privacy amplification output length $\ell = n(1 - h(Q_{\mathsf{obs}})) - \lambda$. Then:*

$$\mathsf{Ask}^Q(g_{\mathsf{secrecy}}) \;\leq\; 2^{-\lambda/2} + \mathsf{negl}(\lambda). \tag{207}$$

The BB84 quantum market is in equilibrium. Crucially, this holds against all *quantum adversaries, including those with unlimited computational power.*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Proof. We construct five quantum bidding rounds. Each round applies the quantum extended difference lemma (Theorem 6) using trace distance as the quantum price adjustment.

$\mathbf{B}_0^Q$ (Real BB84 Quantum Market—Quantum Interception Bid). The quantum seller prepares and transmits N qubits. The quantum buyer intercepts each qubit on the quantum channel and performs an arbitrary quantum operation (joint unitary on the qubit and her ancilla register R_E), then forwards a (possibly modified) qubit to Bob. After sifting, error estimation passes ($Q_{\text{obs}} \leq Q_{\text{tol}}$), and Alice and Bob produce key K.

Purpose: This is the real quantum market. The buyer's quantum interception bid: apply arbitrary CPTP maps to intercepted qubits, entangling them with her private register, then use the entangled state after sifting and error estimation to extract information about the key.
Quantum feature: The buyer's strategy is a *coherent quantum attack*—she can maintain entanglement between her ancilla and the qubits forwarded to Bob across all N transmissions. This is the entangled bidding phenomenon described in Section 20.2.1.

$\mathbf{B}_1^Q$ (Bidding Round 1: No-Cloning Bid—Intercept-Resend Bound).

Purpose: Exploit the no-cloning theorem to bound the information Eve can extract without being detected. The no-cloning theorem states that no quantum operation can produce two perfect copies of an arbitrary unknown quantum state. Since Alice encodes each bit in one of two non-orthogonal bases ($+$ or $\times$), Eve cannot simultaneously learn the bit value and forward an undisturbed qubit.
Quantum market feature: This is the no-cloning constraint on bids (item 3 in Section 20.2.1). The seller's choice to encode in non-orthogonal states creates a *structural market advantage*—the buyer's goods (intercepted qubits) are physically uncopyable.

Any quantum operation by Eve on the intercepted qubit can be decomposed (by Stinespring dilation) as a unitary on the qubit plus ancilla followed by a partial trace. The key insight: if Eve extracts I_E bits of information about Alice's bit x_i, she necessarily introduces a disturbance of at least $\delta_i \geq f(I_E)$ on the qubit forwarded to Bob, where f is a monotonically increasing function derived from the information-disturbance trade-off.

Quantum price adjustment (information-disturbance trade-off): For each sifted-key bit, define Eve's information $I_E^{(i)}$ and the induced error rate Q_i on that bit. The Fuchs–Caves–Holevo bound gives:

$$I_E^{(i)} \;\leq\; h(Q_i), \tag{208}$$

where $h(\cdot)$ is the binary entropy function. Eve's total information on the sifted key is $I_E^{\text{total}} \leq n \cdot h(Q_{\text{avg}})$ where Q_{avg} is the average error rate.

Quantum trace distance:

$$\Delta\text{Price}_1^Q \;=\; \frac{1}{2}\|\rho_{\text{final}}^{(0)} - \rho_{\text{final}}^{(1)}\|_{\text{tr}} \;\leq\; 2^{-\lambda/2}, \tag{209}$$

where $\rho^{(1)}$ is the state conditioned on Eve's information being bounded by $nh(Q_{\text{obs}})$.

Market interpretation: The no-cloning bid fails. The seller's quantum states are non-orthogonal, so the buyer cannot extract information without paying a price (disturbance). The amount the buyer can learn is thermodynamically limited to $nh(Q_{\text{obs}})$ bits—this is the "budget ceiling" on the buyer's quantum interception bid.

$\mathbf{B}_2^Q$ (Bidding Round 2: Error Estimation Bid—Statistical Detection).

Purpose: Bound the probability that Eve causes a high error rate on the sifted key but the error estimation sample underestimates it. If the sample of k bits gives $Q_{\text{obs}} \leq Q_{\text{tol}}$ but the true error rate $Q_{\text{true}} > Q_{\text{tol}} + \gamma$ for some margin γ, then by Hoeffding's inequality: $\Pr[Q_{\text{obs}} \leq Q_{\text{tol}} \mid Q_{\text{true}} > Q_{\text{tol}} + \gamma] \leq e^{-2k\gamma^2}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Quantum market feature: The seller uses *classical statistical inference* on the quantum measurement outcomes to detect the buyer's quantum disturbance. This is a quantum-to-classical conversion: the buyer's quantum attack is detected by a classical test.

Quantum price adjustment: Let F_{est} be the event "error estimation fails to detect Eve's disturbance." By Hoeffding's inequality with k sample bits and margin γ:

$$\Pr[F_{\text{est}}] \;\leq\; e^{-2k\gamma^2}. \tag{210}$$

For $k = \Theta(\lambda/\gamma^2)$, this is $\leq 2^{-\Omega(\lambda)} = \mathsf{negl}(\lambda)$.

Market interpretation: The error estimation bid bounds the probability that the buyer evades detection. Even though the buyer's attack is quantum, the detection mechanism is a classical sampling test that succeeds with overwhelming probability.

$\mathbf{B}_3^Q$ (Bidding Round 3: Privacy Amplification Bid—Leftover Hash Lemma).

Purpose: Given that Eve's information about the corrected key is bounded (from $\mathbf{B}_1^Q$ and $\mathbf{B}_2^Q$), privacy amplification via a universal$_2$ hash function extracts a final key that is exponentially close to uniform from Eve's perspective.
Quantum market feature: The quantum leftover hash lemma (QLHL) operates on Eve's quantum side information ρ_E, not just classical information. The output key K satisfies $\frac{1}{2}\|K \otimes \rho_E - U_\ell \otimes \rho_E\|_{\text{tr}} \leq 2^{-\lambda/2}$ when $\ell \leq H_{\min}(X|E) - 2\log(1/\epsilon)$, where $H_{\min}(X|E)$ is the conditional quantum min-entropy.

By the quantum leftover hash lemma, after privacy amplification with output length $\ell = n(1 - h(Q_{\text{obs}})) - \lambda$:

$$\frac{1}{2}\|K \otimes \rho_E - U_\ell \otimes \rho_E\|_{\text{tr}} \;\leq\; 2^{-\lambda/2}. \tag{211}$$

Market interpretation: Privacy amplification is the seller's final move—a "market restructuring" that converts the raw key (partially known to Eve) into a final key (virtually unknown). The hash function compresses the key to the length justified by the quantum min-entropy, squeezing out all information the buyer might have gained. The trace distance bound $2^{-\lambda/2}$ is the residual quantum ask price.

$\mathbf{B}_4^Q$ (Bidding Round 4: Error Correction Leakage Bid).

Error correction leaks at most $nh(Q_{\text{obs}})$ bits of classical information to Eve via the public syndrome. This is already accounted for in the privacy amplification output length: $\ell = n(1 - h(Q_{\text{obs}})) - \lambda$ subtracts exactly $nh(Q_{\text{obs}})$ bits. Price adjustment: $\Delta\text{Price}_4^Q = 0$ (already absorbed).

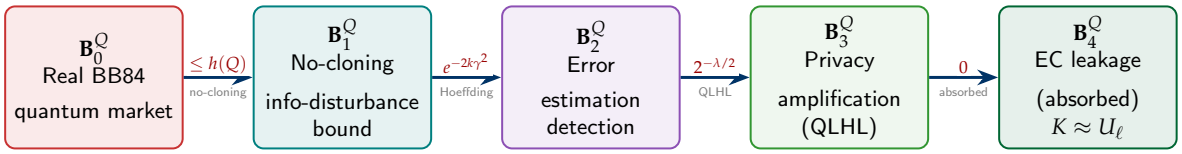
Total quantum ask price:

$$\text{Ask}^Q(g_{\text{secrecy}}) \;\leq\; \underbrace{2^{-\lambda/2}}_{\text{QLHL}} + \underbrace{e^{-2k\gamma^2}}_{\text{estimation}} + \underbrace{2^{-\lambda}}_{\text{EC failure}} \;\leq\; 2^{-\lambda/2} + \mathsf{negl}(\lambda). \tag{212}$$

The BB84 quantum market is in equilibrium. □

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2



$$\text{Total Ask}^Q(g_{\text{secrecy}}) \;\leq\; 2^{-\lambda/2} + e^{-2k\gamma^2} + 2^{-\lambda} \;\leq\; 2^{-\lambda/2} + \text{negl}(\lambda)$$

Figure 60. BB84 quantum bidding-round chain: real quantum market ($\mathbf{B}_0^Q$) to ideal ($\mathbf{B}_4^Q$). Two distinctively *quantum* hops: $\mathbf{B}_0^Q \to \mathbf{B}_1^Q$ (no-cloning information-disturbance trade-off—the seller's non-orthogonal states structurally limit the buyer's information gain) and $\mathbf{B}_2^Q \to \mathbf{B}_3^Q$ (quantum leftover hash lemma—privacy amplification against quantum side information). These are the quantum analogues of the O2H and measure-and-reprogram hops in the QROM case study (Section 14). The security is information-theoretic: no computational assumption is required, only quantum mechanics.

✻ Simple Terms: BB84 Security Proof Explained—Why Quantum Mechanics Gives Perfect Security

Classical comparison: In Diffie–Hellman key exchange, security relies on the assumption that discrete logarithms are hard to compute. A powerful enough computer (classical or quantum) could break it. The ask price is $\text{Adv}^{\text{ECDLP}}$—computational.

BB84 (quantum): Security relies on the laws of quantum mechanics:

1. *No-cloning:* Eve cannot copy Alice's qubits. She must choose: learn the bit (disturbing the qubit) or forward it undisturbed (learning nothing). This is not a computational assumption—it is a physical law.
2. *Information-disturbance:* Any information Eve extracts is paid for by disturbance that Alice and Bob can detect statistically.
3. *Privacy amplification:* Even if Eve learns some partial information, hashing compresses the key to remove all leaked information.

Market interpretation: The seller's quantum states are *uncopyable goods*—the buyer cannot "shoplift" (learn the key) without leaving evidence (error rate increase). The error estimation phase is a *market audit*: the seller checks for evidence of theft. Privacy amplification is *market insurance*: even if the audit slightly underestimates the theft, the insurance (hashing) covers the gap.

15.5. Extended Security Goods and CNF Verification

Corollary 18 (BB84 Correctness)**.** *Under the error correction protocol with failure probability $\delta_{\text{EC}} \leq 2^{-\lambda}$:*

$$\text{Ask}^Q(g_{\text{correct}}) \;\leq\; 2^{-\lambda} = \text{negl}(\lambda). \tag{213}$$

Corollary 19 (BB84 Eavesdropping Detection)**.** *If Eve extracts $I_E \geq \gamma n$ bits of information about the sifted key (for any constant $\gamma > 0$), the error estimation detects this:*

$$\text{Ask}^Q(g_{\text{detect}}) \;\leq\; e^{-2k\gamma^2} + 2^{-\lambda} = \text{negl}(\lambda). \tag{214}$$

✓ CNF Verification: BB84 Session-CNF Verification

$\varphi_{\text{BB84}} = \varphi^{\text{basis}} \wedge \varphi^{\text{error}} \wedge \varphi^{\text{auth}} \wedge \varphi^{\text{PA}} \wedge \varphi^{\text{ping}}$, where $\varphi^{\text{basis}} = (x_{\theta^A=\theta^B \text{ for sifted bits}})$, $\varphi^{\text{error}} = (x_{Q_{\text{obs}} \leq Q_{\text{tol}}})$, $\varphi^{\text{auth}} = (x_{\text{classical channel authenticated}})$, $\varphi^{\text{PA}} = (x_{\ell \leq n(1-h(Q_{\text{obs}}))-\lambda})$.

Manual CNF worksheet (BB84 session):

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Clause	Check	How	Pass?
φ^{basis}: basis match	Sifted bits use matching bases?	Compare public θ^A, θ^B records.	T/F
φ^{error}: error rate	$Q_{\text{obs}} \leq Q_{\text{tol}}$?	Compute error rate on sample of k bits.	T/F
φ^{auth}: classical auth	Classical channel authenticated?	Verify MAC on all classical messages.	T/F
φ^{PA}: PA length	$\ell \leq n(1 - h(Q_{\text{obs}})) - \lambda$?	Check output length of hash.	T/F
φ^{ping}: session fresh	No sifted-key bits reused from prior session?	Compare with previous session logs.	T/F
Result:	All T ⇒ BB84 session secure. Any F ⇒ Reject (abort).		

➤ Ping Bid: BB84 Unbounded Ping Bid

The buyer's *quantum replay bid*: attempt to reuse information from session Session_i to attack session Session_{i+1}. Since each session uses fresh random bits and bases (x_i, θ_i^A are independently sampled), Eve's quantum side information from Session_i is *product* with the quantum states in Session_{i+1}. By the quantum leftover hash lemma applied independently per session: $\Delta\text{Price}^Q_{\text{ping}} \leq 2^{-\lambda/2} = \text{negl}$. By Theorem 3 (extended to quantum markets via Theorem 6), BB84 is secure for unbounded sessions against quantum adversaries.

Table 18. BB84 quantum market goods summary.

Good	Quantum Ask Price Bound	Dominant Term	Status
$\mathcal{G}_{\text{secrecy}}$	$2^{-\lambda/2} + e^{-2k\gamma^2}$	QLHL	Equilibrium
$\mathcal{G}_{\text{correct}}$	$2^{-\lambda}$	EC failure	Equilibrium
$\mathcal{G}_{\text{detect}}$	$e^{-2k\gamma^2} + 2^{-\lambda}$	Hoeffding	Equilibrium
$\mathcal{G}_{\text{unbounded}}$ (ping)	$2^{-\lambda/2}$	QLHL (per session)	Equilibrium

☞ Key Insight:

Classical QROM vs. Quantum Market Dynamics: Complete Comparison

Property	QROM Market (Section 14)	Quantum Market (BB84)
Seller	Classical (prepares ciphertexts)	**Quantum** (prepares qubits)
Buyer	Quantum (superposition queries)	Quantum (arbitrary CPTP maps)
Security basis	Computational (MLWE)	**Information-theoretic** (physics)
Price mechanism	$\sqrt{\text{Adv}^{\text{IND-CPA}}}$ (O2H)	$2^{-\lambda/2}$ (QLHL)
Key quantum tool	O2H + measure-reprogram	No-cloning + info-disturbance
Bidding rounds	6	5 ($\mathbf{B}_0^Q$–$\mathbf{B}_4^Q$, last absorbed)
Breaks under	Quantum computer breaking MLWE	**Violation of quantum mechanics**
MTSF framework	Classical market, quantum buyer	**Full quantum market**

Core insight: The QROM market has a classical seller defending against a quantum buyer (post-quantum cryptography). The BB84 quantum market has a quantum seller exploiting quantum mechanics as a *structural defence*—the goods are physically uncopyable. This is a fundamentally stronger market position.

16. Case Study VIII: Multi-Protocol Composition—TLS 1.3 + Signal Network

✎ Analogy:

Multi-Protocol Composition as a Multi-Lock Security System Imagine your house has two independent security systems:

- **System A (TLS 1.3):** A deadbolt lock on the front door. Proven burglar-proof on its own.
- **System B (Signal):** An alarm system. Proven burglar-proof on its own.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The composition question: When you install BOTH systems together, does the combination remain secure? Or could they interfere—perhaps the alarm system's vibrations loosen the deadbolt?
In the real world: Your phone runs TLS 1.3 (for HTTPS browsing) and Signal (for messaging) *simultaneously*, sharing the same PKI infrastructure (certificate authorities, device keys). An attacker who observes your TLS traffic might use information from it to attack your Signal sessions, or vice versa. The composition theorem must ensure this cross-protocol attack is impossible.
MTSF's market merger theorem: If the TLS market is in equilibrium (TLS is secure alone) and the Signal market is in equilibrium (Signal is secure alone), then the *merged* market (TLS + Signal running simultaneously, sharing PKI) is also in equilibrium—provided the shared infrastructure (PKI) is itself secure. This is like proving that the deadbolt and alarm system, sharing the same electrical wiring (PKI), remain secure together because the wiring itself is trustworthy.
The key insight: Security does not degrade under composition *if* the shared infrastructure is sound and each protocol binds its sessions to unique identifiers. MTSF's CNF verification ensures this by requiring that TLS sessions and Signal sessions use distinct session identifiers, preventing cross-protocol replays.

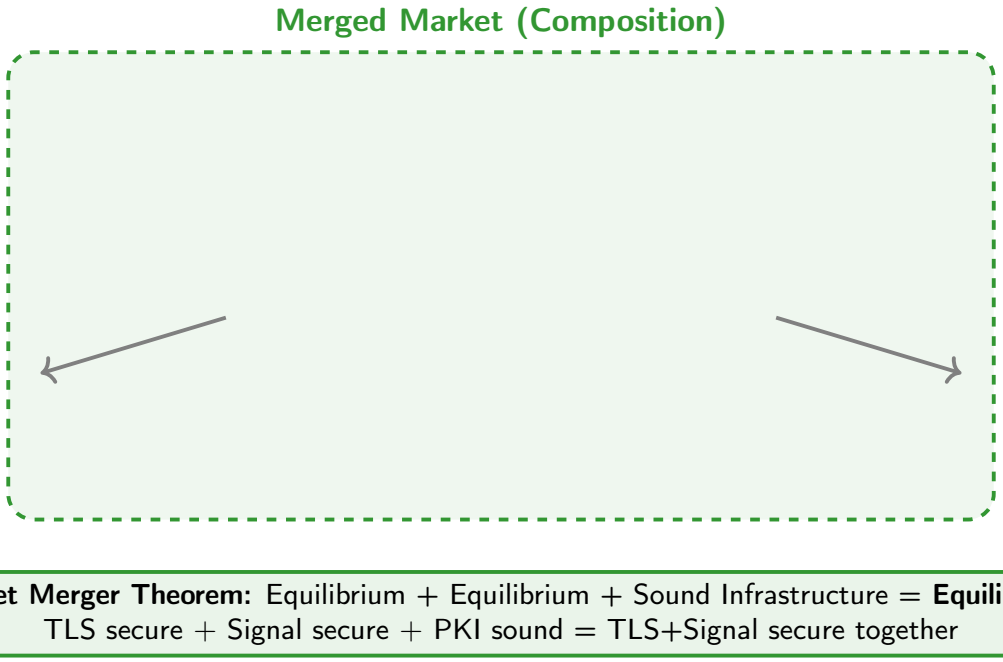


Figure 61. Multi-protocol composition as market merger: TLS 1.3 and Signal, each independently secure (in equilibrium), share a PKI infrastructure. The market merger theorem guarantees the combined system remains in equilibrium, provided the shared PKI is sound.

This case study instantiates the multi-protocol composition network formalisation of Section 20.2.2 on a concrete and practically relevant scenario: a device simultaneously running TLS 1.3 (Section 13.8.1) for web browsing and the Signal Protocol (Section 13.7) for encrypted messaging, sharing a common PKI infrastructure. This is the first MTSF analysis of *concurrent protocol execution*, demonstrating that the market merger theorem (Theorem 63) preserves equilibrium when two individually secure protocols compose.

✻ Simple Terms: Multi-Protocol Composition—Why It Matters

The real-world scenario: Right now, your phone is probably running multiple security protocols simultaneously:

- **TLS 1.3** connects your browser to your bank, streaming service, and email provider.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Signal Protocol** encrypts your WhatsApp or Signal messages.
- Both use the **same PKI** (certificate authorities, X.509 certificates) to verify identities.
- Both use the **same random number generator** on your device.
- Both share the **same CPU and memory**—an attacker exploiting one protocol might learn something useful for attacking the other.

The security question: We proved TLS 1.3 secure in Section 13.8.3. We proved the Signal Protocol secure in Section 13.7. But are they secure *when running at the same time on the same device sharing the same infrastructure*?
The answer (via MTSF): Yes—if each protocol market is in equilibrium individually and the shared infrastructure is sound, the composed market network remains in equilibrium. The attacker's resources must be split across both markets, and neither market collapses.

16.1. Network Description

✻ Simple Terms: TLS 1.3—The Protocol That Secures the Internet

Every time you see the padlock icon in your browser, TLS 1.3 is at work. It is the most widely deployed security protocol in history, protecting:

- Every HTTPS website you visit (banking, shopping, email).
- Every API call between cloud services.
- Every mobile app that communicates with a server.

How TLS 1.3 works (simplified):

1. **ClientHello:** Your browser sends its capabilities and a key share (Diffie–Hellman or ML-KEM public value).
2. **ServerHello:** The server responds with its key share, certificate, and signature.
3. **Key derivation:** Both sides derive shared secrets using HKDF (HMAC-based Key Derivation Function).
4. **Application data:** All subsequent communication is encrypted with AEAD (AES-GCM or ChaCha20-Poly1305).

1-RTT vs. 0-RTT: TLS 1.3's main handshake takes one round trip (1-RTT)—the client sends a message, gets a response, and both sides are ready. For repeat connections, TLS 1.3 offers 0-RTT resumption (sending application data in the first message), but this sacrifices replay protection. In MTSF, 1-RTT is in equilibrium; 0-RTT has a "replay bid" collapse for the replay-protection good.
The downgrade attack: An attacker might try to force TLS 1.3 to fall back to an older, weaker version (TLS 1.2 or 1.1). TLS 1.3 prevents this with a "downgrade sentinel" embedded in the server's random value. In MTSF, the downgrade bid collapses if the sentinel is removed, but succeeds (equilibrium) with the sentinel in place.

Component protocols.

1. **Protocol 1: TLS 1.3 (1-RTT handshake)**—as analysed in Section 13.8.3. Provides session-key secrecy ($g^{(1)}_{\mathsf{sk}}$), forward secrecy ($g^{(1)}_{\mathsf{FS}}$), server authentication ($g^{(1)}_{\mathsf{auth}}$), and mutual authentication ($g^{(1)}_{\mathsf{mutual}}$) when client certificates are used.
2. **Protocol 2: Signal (X3DH + Double Ratchet)**—as analysed in Section 13.7. Provides session-key secrecy ($g^{(2)}_{\mathsf{sk}}$), forward secrecy ($g^{(2)}_{\mathsf{FS}}$), post-compromise security ($g^{(2)}_{\mathsf{PCS}}$), asynchronous establishment ($g^{(2)}_{\mathsf{async}}$), and deniability ($g^{(2)}_{\mathsf{deny}}$).

Shared infrastructure $\mathcal{I}$.

- **PKI** ($\mathcal{I}_{\mathsf{PKI}}$): X.509 certificate authorities issuing certificates for TLS servers and Signal identity keys. Shared trust anchors.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Random oracle** ($\mathcal{I}_{\mathsf{RO}}$): Hash functions (SHA-256, SHA-384) used by both protocols for key derivation (HKDF in TLS, HKDF in Signal's X3DH).
- **Device RNG** ($\mathcal{I}_{\mathsf{RNG}}$): The operating system's cryptographic random number generator, used by both protocols for nonce generation and ephemeral key sampling.

Participants.

- **Device** D: runs both protocols concurrently. Acts as the TLS client and the Signal user.
- **TLS server** S_1: the web server for Protocol 1.
- **Signal server** S_2: the Signal key server storing prekey bundles for Protocol 2.
- **Network adversary** $\mathcal{A}$: a single PPT adversary controlling the network between D and both servers. $\mathcal{A}$ can intercept, modify, and inject messages in both protocols simultaneously. $\mathcal{A}$ has a single computational budget T (total running time).

16.2. MTSF Market Network Model

We instantiate Theorem 38 with $N = 2$ component markets.

Definition 36 (TLS+Signal Market Network)**.** *The protocol market network*

$$\mathcal{N}_{\mathsf{TS}} = \left(\{\mathcal{M}_{\mathsf{TLS}}, \mathcal{M}_{\mathsf{Signal}}\},\ \mathcal{I},\ \mathcal{Z}\right)$$

is defined as follows:

- ***TLS market:***
$$\mathcal{M}_{\mathsf{TLS}} = (\mathsf{Seller}_{\mathsf{TLS}}, \mathsf{Buyer}_{\mathsf{TLS}}, \{g^{(1)}_{\mathsf{sk}}, g^{(1)}_{\mathsf{FS}}, g^{(1)}_{\mathsf{auth}}, g^{(1)}_{\mathsf{mutual}}\}, \varphi_{\mathsf{TLS}})$$
is the TLS 1.3 market from Section 13.8.2.
- ***Signal market:***
$$\mathcal{M}_{\mathsf{Signal}} = (\mathsf{Seller}_{\mathsf{Signal}}, \mathsf{Buyer}_{\mathsf{Signal}}, \{g^{(2)}_{\mathsf{sk}}, g^{(2)}_{\mathsf{FS}}, g^{(2)}_{\mathsf{PCS}}, g^{(2)}_{\mathsf{async}}, g^{(2)}_{\mathsf{deny}}\}, \varphi_{\mathsf{Signal}})$$
is the Signal market from Section 13.7.
- ***Shared infrastructure:***
$$\mathcal{I} = \mathcal{I}_{\mathsf{PKI}} \cup \mathcal{I}_{\mathsf{RO}} \cup \mathcal{I}_{\mathsf{RNG}}.$$
- ***Environment:*** *$\mathcal{Z}$ is the network environment that interacts with both protocols concurrently.*

Infrastructure security goods.

The shared infrastructure contributes three goods:

- g_{PKI}: PKI integrity—no adversary can forge a valid certificate. $\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{PKI}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{CA}} \leq \mathsf{negl}(\lambda)$ (under the EUF-CMA security of the CA's signature scheme).
- g_{RO}: Random oracle consistency—both protocols query the same hash function, modelled as a shared random oracle. $\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{RO}}) = 0$ (the RO is ideal by assumption; in the standard model, replace by $\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}$).
- g_{RNG}: RNG quality—the device RNG produces outputs indistinguishable from uniform. $\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{RNG}}) \leq \mathsf{Adv}^{\mathsf{PRG}}_{\mathsf{RNG}} \leq \mathsf{negl}(\lambda)$.

16.3. Market Merger and Composition Proof

We apply the market merger theorem (Theorem 63) to the TLS+Signal network.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Theorem 62 (TLS+Signal Network Equilibrium). *Let $\mathcal{M}_{\mathsf{TLS}}$ be in equilibrium with ask prices as in Section 13.8.3, and let $\mathcal{M}_{\mathsf{Signal}}$ be in equilibrium with ask prices as in Section 13.7. If the shared infrastructure is sound ($\mathsf{Ask}_{\mathcal{I}}(g) \leq \mathsf{negl}(\lambda)$ for all infrastructure goods g), then the merged market:*

$$\mathcal{M}_{\mathsf{TLS}} \otimes_{\mathcal{I}} \mathcal{M}_{\mathsf{Signal}} \tag{215}$$

is in equilibrium. Specifically, for every security good g in either protocol:

$$\mathsf{Ask}_{\mathcal{N}}(g) \;\leq\; \mathsf{Ask}_{\mathsf{own}}(g) + \mathsf{Ask}_{\mathsf{other}} + \mathsf{Ask}_{\mathcal{I}} \;\leq\; \mathsf{negl}(\lambda), \tag{216}$$

where $\mathsf{Ask}_{\mathsf{own}}(g)$ is the ask price of g in its home market, $\mathsf{Ask}_{\mathsf{other}}$ is the maximum ask price in the other market, and $\mathsf{Ask}_{\mathcal{I}} = \max_g \mathsf{Ask}_{\mathcal{I}}(g)$ is the infrastructure overhead.

Proof. We apply Theorem 63. The three conditions are verified:

Condition 1: TLS market equilibrium. By Section 13.8.3, all TLS 1.3 goods are in equilibrium. The ten-bidding-round chain gives:

$$\mathsf{Ask}_{\mathsf{TLS}}(g^{(1)}_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{ECDLP}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + q_N^2/2^{\lambda+1} \leq \mathsf{negl}(\lambda), \tag{217}$$

$$\mathsf{Ask}_{\mathsf{TLS}}(g^{(1)}_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}} + q_N^2/2^{\lambda+1} \leq \mathsf{negl}(\lambda). \tag{218}$$

Condition 2: Signal market equilibrium. By Section 13.7, all Signal goods are in equilibrium. The X3DH proof gives:

$$\mathsf{Ask}_{\mathsf{Signal}}(g^{(2)}_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{gap\text{-}CDH}} + q_N^2/2^{\lambda+1} \leq \mathsf{negl}(\lambda), \tag{219}$$

$$\mathsf{Ask}_{\mathsf{Signal}}(g^{(2)}_{\mathsf{FS}}) \leq \mathsf{Adv}^{\mathsf{gap\text{-}CDH}} \leq \mathsf{negl}(\lambda). \tag{220}$$

Condition 3: Infrastructure soundness.

$$\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{PKI}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{CA}} \leq \mathsf{negl}(\lambda), \tag{221}$$

$$\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{RO}}) = 0 \quad \text{(ideal RO)}, \tag{222}$$

$$\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{RNG}}) \leq \mathsf{Adv}^{\mathsf{PRG}}_{\mathsf{RNG}} \leq \mathsf{negl}(\lambda). \tag{223}$$

By Theorem 63, the merged market is in equilibrium. The quantitative bound from Remark after Theorem 63 gives:

$$\mathsf{Ask}_{\mathcal{N}}(g^{(1)}_{\mathsf{sk}}) \;\leq\; \underbrace{\mathsf{Ask}_{\mathsf{TLS}}(g^{(1)}_{\mathsf{sk}})}_{\text{own market}} + \underbrace{\mathsf{Ask}_{\mathsf{Signal,max}}}_{\text{simulation overhead}} + \underbrace{\mathsf{Ask}_{\mathcal{I},\mathsf{max}}}_{\text{infrastructure}} \;\leq\; \mathsf{negl}(\lambda), \tag{224}$$

where $\mathsf{Ask}_{\mathsf{Signal,max}} = \max_j \mathsf{Ask}_{\mathsf{Signal}}(g^{(2)}_j)$ is the simulation overhead from simulating the Signal market when reducing to the standalone TLS game. Symmetrically for Signal goods.

The proof proceeds by constructing a standalone buyer for each market that internally simulates the other market using the UC simulator (market arbitrageur from Section 2.4). The environment $\mathcal{Z}$ cannot distinguish the real composed execution from the simulation because both component markets are UC-realizable (they implement ideal functionalities for key exchange and secure messaging respectively). □

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

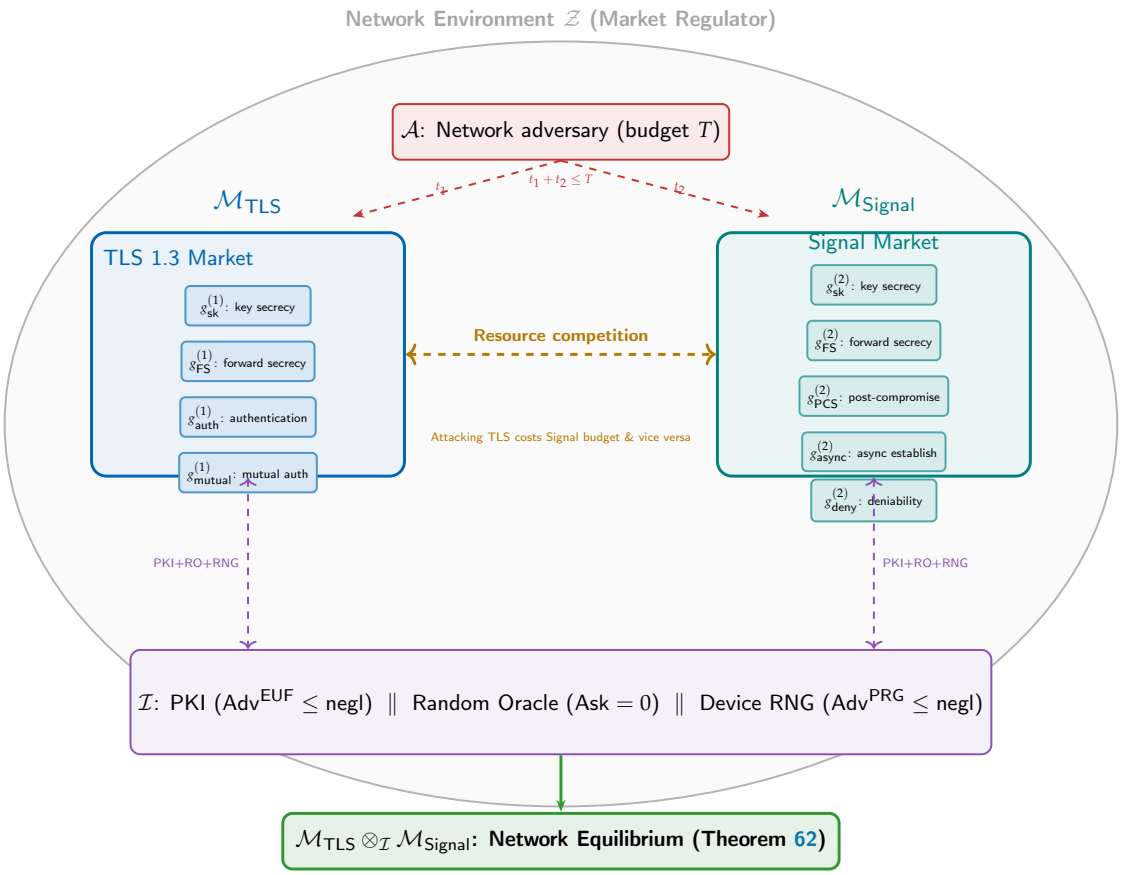


Figure 62. TLS 1.3 + Signal protocol market network. The TLS market (left, blue) and Signal market (right, teal) share infrastructure $\mathcal{I}$ (PKI, random oracle, device RNG). A single network adversary $\mathcal{A}$ with budget T must split its resources ($t_1 + t_2 \leq T$) between the two markets. The orange dashed link indicates resource competition: effort spent attacking TLS reduces resources available for attacking Signal. By the market merger theorem (Theorem 63), since both component markets and the infrastructure are in equilibrium individually, the merged market $\mathcal{M}_{\mathsf{TLS}} \otimes_{\mathcal{I}} \mathcal{M}_{\mathsf{Signal}}$ preserves equilibrium for all nine security goods.

16.4. Resource Competition Analysis

✎ Analogy:

Resource Competition as a Shared Budget When TLS and Signal run simultaneously on the same device, they share computational resources. Think of this as two departments in a company sharing a single budget:

- **TLS department** needs T_{TLS} operations per session (key exchange, AEAD encryption, certificate verification).
- **Signal department** needs T_{Signal} operations per session (X3DH, Double Ratchet, message encryption).
- **The attacker's budget** is $T_{\mathsf{total}} \leq \mathsf{poly}(\lambda)$.

The question: Can the attacker allocate their budget across attacks on both protocols simultaneously to achieve a higher total advantage than attacking either alone?
MTSF's answer: No. If each protocol is independently secure, the total advantage is bounded by the sum of individual advantages. This is because the security goods are offered in *independent* markets: breaking TLS's key secrecy does not help break Signal's forward secrecy (different keys, different mechanisms). The extended difference lemma captures this: $\Pr[F_{\mathsf{TLS}} \cup F_{\mathsf{Signal}}] \leq \Pr[F_{\mathsf{TLS}}] + \Pr[F_{\mathsf{Signal}}]$ when the failure events use independent cryptographic material.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The MTSF language makes explicit a phenomenon that is implicit in the UC framework: the adversary faces a *resource allocation problem* when attacking multiple protocols simultaneously.

Proposition 5 (Adversarial Resource Allocation)**.** *Let $\mathcal{A}$ be a PPT adversary with total running time T attacking the network $\mathcal{N}_{\mathsf{TS}}$. If $\mathcal{A}$ allocates time t_1 to attacking $\mathcal{M}_{\mathsf{TLS}}$ and $t_2 = T - t_1$ to attacking $\mathcal{M}_{\mathsf{Signal}}$, then:*

$$\mathsf{Ask}_{\mathcal{N}}(g_{\mathsf{sk}}^{(1)}) + \mathsf{Ask}_{\mathcal{N}}(g_{\mathsf{sk}}^{(2)}) \;\leq\; f_{\mathsf{TLS}}(t_1) + f_{\mathsf{Signal}}(T - t_1) + 2\mathsf{Ask}_{\mathcal{I}}, \tag{225}$$

where $f_{\mathsf{TLS}}(t) = \mathsf{Adv}^{\mathsf{ECDLP}}(t) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}(t) + q_N^2/2^{\lambda+1}$ and $f_{\mathsf{Signal}}(t) = \mathsf{Adv}^{\mathsf{gap\text{-}CDH}}(t) + q_N^2/2^{\lambda+1}$ are the advantage functions for each protocol as a function of computational effort.

Proof. The adversary's total computation is partitioned: t_1 steps interact with TLS, t_2 steps interact with Signal. Any computation that interacts with TLS contributes to $f_{\mathsf{TLS}}(t_1)$ via the standalone TLS reduction. Any computation that interacts with Signal contributes to $f_{\mathsf{Signal}}(t_2)$ via the standalone Signal reduction. The infrastructure overhead $\mathsf{Ask}_{\mathcal{I}}$ applies to each market independently (hence the factor of 2). The resource constraint $t_1 + t_2 \leq T$ means the adversary cannot achieve the best attack on both protocols simultaneously. □

Remark 6 (Weakest Link Principle)**.** *The rational adversary maximises*

$$\max(\mathsf{Ask}_{\mathcal{N}}(g_{\mathsf{sk}}^{(1)}), \mathsf{Ask}_{\mathcal{N}}(g_{\mathsf{sk}}^{(2)}))$$

—it concentrates resources on whichever protocol offers the highest "return on attack." If TLS uses ECDLP on a 256-bit curve and Signal uses gap-CDH on the same curve, both are equally hard and the adversary gains nothing from splitting effort. In practice, if one protocol used a weaker primitive (e.g., a shorter key), the adversary would rationally concentrate there—formalising the "weakest link" principle in market terms.

16.5. Network CNF Verification

✎ Analogy:

Network CNF as a Multi-Building Inspection If the single-protocol CNF is like inspecting one building, the network CNF is like inspecting an entire city block:

- **Building-level clauses:** Each protocol (TLS, Signal) has its own CNF checklist.
- **Infrastructure clauses:** Shared PKI certificates are verified (the electrical grid inspection).
- **Cross-building clauses:** Session identifiers are unique across protocols (no two buildings share the same address). This prevents a "cross-protocol replay bid"—using a TLS session transcript to attack Signal, or vice versa.

The network CNF is the conjunction of all building CNFs plus infrastructure clauses:

$$\varphi_{\mathsf{network}} = \varphi_{\mathsf{TLS}} \wedge \varphi_{\mathsf{Signal}} \wedge \varphi_{\mathsf{PKI}} \wedge \varphi_{\mathsf{cross\text{-}protocol}}$$

The key insight: Cross-protocol clauses prevent attacks that exploit the *interaction* between protocols. Without these clauses, an attacker might use a TLS certificate to impersonate a Signal contact, or replay a Signal prekey bundle in a TLS handshake. The network CNF blocks all such cross-protocol attacks.

We instantiate Theorem 40 for the TLS+Signal network.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✓ CNF Verification: TLS+Signal Network Session-CNF Verification

$\varphi_{\mathcal{N}} = \varphi_{\mathsf{TLS}} \wedge \varphi_{\mathsf{Signal}} \wedge \varphi_{\mathcal{I}}$, where φ_{TLS} is from Section 13.8.6, $\varphi_{\mathsf{Signal}}$ is from Section 13.7, and:

$$\varphi_{\mathcal{I}} = \underbrace{\varphi_{\mathcal{I}}^{\mathsf{PKI}}}_{\text{cert consistency}} \wedge \underbrace{\varphi_{\mathcal{I}}^{\mathsf{RO}}}_{\text{hash consistency}} \wedge \underbrace{\varphi_{\mathcal{I}}^{\mathsf{RNG}}}_{\text{RNG quality}} \wedge \underbrace{\varphi_{\mathcal{I}}^{\mathsf{SID}}}_{\text{global SID uniqueness}} .$$

Manual CNF worksheet (network session—infrastructure clauses):

Clause	Check	How	Pass?
$\varphi_{\mathcal{I}}^{\mathsf{PKI}}$: cert consistency	Same CA root for TLS and Signal?	Compare trust chains.	T/F
$\varphi_{\mathcal{I}}^{\mathsf{RO}}$: hash consistency	Both use same SHA-256/384?	Check library versions.	T/F
$\varphi_{\mathcal{I}}^{\mathsf{RNG}}$: RNG quality	Device RNG passes NIST SP 800-90B?	Run health tests.	T/F
$\varphi_{\mathcal{I}}^{\mathsf{SID}}$: global SID	No TLS SID = Signal SID?	Compare SID logs across both protocols.	T/F
Result:	All T (+ per-protocol CNFs pass) ⇒ Network session secure.		

➤ Ping Bid: TLS+Signal Network Unbounded Ping Bid

The buyer's *cross-protocol replay bid*: attempt to use a TLS session transcript to attack a Signal session (or vice versa). This fails because:

1. TLS and Signal use distinct protocol identifiers and session IDs ($\varphi_{\mathcal{I}}^{\mathsf{SID}}$ ensures global uniqueness).
2. TLS key material is derived via HKDF with TLS-specific labels; Signal key material via HKDF with X3DH-specific labels. The label-binding ensures cross-protocol key independence.
3. The network ping bid decomposes: $\Delta\mathsf{Price}_{\mathsf{ping}}^{\mathcal{N}} \leq \Delta\mathsf{Price}_{\mathsf{ping}}^{\mathsf{TLS}} + \Delta\mathsf{Price}_{\mathsf{ping}}^{\mathsf{Signal}} + \Delta\mathsf{Price}_{\mathsf{ping}}^{\mathcal{I}}$, where each component is negligible by the per-protocol ping analysis and infrastructure soundness.

By Theorem 3 applied to the network (via Theorem 20 with $N = 2$), the TLS+Signal network is secure for unbounded sessions across both protocols simultaneously.

Table 19. TLS+Signal market network goods summary. All goods from both protocols plus infrastructure goods, with network ask prices from Theorem 62.

Market	Good	Standalone Ask	Network Ask (Eq. 216)	Status
TLS 1.3	$g_{\mathsf{sk}}^{(1)}$: key secrecy	$\mathsf{Adv}^{\mathsf{ECDLP}} + \mathsf{Adv}^{\mathsf{PRF}} + q_N^2/2^{\lambda+1}$	$+\mathsf{Ask}_{\mathsf{Signal},\max} + \mathsf{Ask}_{\mathcal{I}}$	Equilibrium
	$g_{\mathsf{FS}}^{(1)}$: forward secrecy	$\mathsf{Adv}^{\mathsf{ECDLP}}$	$+\mathsf{Ask}_{\mathsf{Signal},\max} + \mathsf{Ask}_{\mathcal{I}}$	Equilibrium
	$g_{\mathsf{auth}}^{(1)}$: authentication	$\mathsf{Adv}^{\mathsf{EUF}} + q_N^2/2^{\lambda+1}$	$+\mathsf{Ask}_{\mathsf{Signal},\max} + \mathsf{Ask}_{\mathcal{I}}$	Equilibrium
	$g_{\mathsf{mutual}}^{(1)}$: mutual auth	$\mathsf{Adv}^{\mathsf{EUF}} + q_N^2/2^{\lambda+1}$	$+\mathsf{Ask}_{\mathsf{Signal},\max} + \mathsf{Ask}_{\mathcal{I}}$	Equilibrium
Signal	$g_{\mathsf{sk}}^{(2)}$: key secrecy	$\mathsf{Adv}^{\mathsf{gap\text{-}CDH}} + q_N^2/2^{\lambda+1}$	$+\mathsf{Ask}_{\mathsf{TLS},\max} + \mathsf{Ask}_{\mathcal{I}}$	Equilibrium
	$g_{\mathsf{FS}}^{(2)}$: forward secrecy	$\mathsf{Adv}^{\mathsf{gap\text{-}CDH}}$	$+\mathsf{Ask}_{\mathsf{TLS},\max} + \mathsf{Ask}_{\mathcal{I}}$	Equilibrium
	$g_{\mathsf{PCS}}^{(2)}$: post-compromise	$\mathsf{Adv}^{\mathsf{gap\text{-}CDH}}$	$+\mathsf{Ask}_{\mathsf{TLS},\max} + \mathsf{Ask}_{\mathcal{I}}$	Equilibrium
	$g_{\mathsf{async}}^{(2)}$: async establish	$\mathsf{Adv}^{\mathsf{gap\text{-}CDH}}$	$+\mathsf{Ask}_{\mathsf{TLS},\max} + \mathsf{Ask}_{\mathcal{I}}$	Equilibrium
	$g_{\mathsf{deny}}^{(2)}$: deniability	$\leq \mathsf{negl}$	$+\mathsf{Ask}_{\mathsf{TLS},\max} + \mathsf{Ask}_{\mathcal{I}}$	Equilibrium
Infrastructure	g_{PKI}: cert integrity	$\mathsf{Adv}_{\mathsf{CA}}^{\mathsf{EUF}}$	(shared)	Equilibrium
	g_{RO}: hash consistency	0 (ideal)	(shared)	Equilibrium
	g_{RNG}: RNG quality	$\mathsf{Adv}_{\mathsf{RNG}}^{\mathsf{PRG}}$	(shared)	Equilibrium

Remark 7 (Generalisation to N Protocols)**.** *Theorem 20 shows that a device running $N = \mathsf{poly}(\lambda)$ protocols simultaneously (e.g., TLS + Signal + SSH + WireGuard + ...) remains in network equilibrium as long as each protocol market and the shared infrastructure are individually in equilibrium. The network ask price grows at most linearly in N, but since each component ask is negligible and N is polynomial, the sum remains negligible.*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The practical takeaway: running many secure protocols on the same device does not degrade security, provided the shared infrastructure (PKI, RNG) is sound.

Figure 63. Complete scoreboard of all MTSF case studies. Green boxes indicate market equilibrium (scheme/protocol is secure); red boxes indicate market collapse (insecure). The same MTSF framework produces both verdicts using bidding-round machinery, CNF verification, and session pinging.

17. Writing a Security Research Paper with MTSF

This section provides a comprehensive, self-contained guide for writing a classical security research paper using MTSF. It covers the full pipeline from specifying a cryptographic scheme to proving or disproving its security, and is written so that a graduate student encountering security proofs for the first time can follow each step.

✻ Simple Terms: Who This Section Is For

This section is for anyone who wants to write a security research paper and use MTSF as the proof framework. It applies equally to:

- Designing a **new block cipher** and proving PRP security.
- Proposing a **new hash function** and proving collision resistance.
- Analysing a **new stream cipher** and proving PRG security.
- Designing a **new protocol** and proving authentication and key secrecy.
- Discovering an **attack** on an existing scheme and formally proving insecurity.

The structure below mirrors the standard structure of a top-tier cryptography conference paper (CRYPTO, EUROCRYPT, ASIACRYPT, CCS, S&P), annotated with MTSF-specific guidance.

17.1. Step 1: Define the Scheme or Protocol

What to Include in the Scheme Description

Every security research paper begins with a precise, self-contained description of the cryptographic object being analysed.

For a block cipher (e.g., AES, PRESENT, Serpent):

1. **Parameters:** Block size n (e.g., 128 bits), key size κ (e.g., 128/192/256 bits), number of rounds r (e.g., 10/12/14).
2. **Round function:** Specify each operation (SubBytes/S-box, ShiftRows/permutation layer, MixColumns/diffusion layer, AddRoundKey).
3. **Key schedule:** How the master key is expanded into round keys $K_0, K_1, \ldots, K_r$.
4. **Diagram:** A figure showing one round and the overall structure.

For a hash function (e.g., SHA-3, BLAKE3, ASCON-Hash):

1. **Construction:** Merkle–Damgård, sponge, Merkle tree, or other.
2. **Compression/permutation function:** Internal state size, rate r, capacity c.
3. **Padding rule:** How the message is padded to a multiple of the block size.
4. **Output extraction:** How the digest is extracted from the internal state.

For a stream cipher (e.g., Grain-128a, ChaCha20, Trivium):

1. **State:** Internal state size, register structure (LFSR, NFSR, counters).
2. **Initialisation:** How the key and IV/nonce are loaded and mixed.
3. **Output function:** How keystream bits are extracted from the state.
4. **Update function:** How the state evolves after each output bit.

For a protocol (e.g., TLS 1.3, Signal, key exchange):

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

1. **Parties:** Who participates (client, server, KDC, CA).
2. **Message flow:** A sequence diagram showing every message, its content, and which party sends it.
3. **Cryptographic operations:** Which primitives are used (signature scheme, KEM, AEAD, KDF, hash).
4. **Session structure:** What constitutes a "session" (SID, nonces, keys generated).
5. **Assumptions:** What is trusted (PKI, random oracle, authenticated channels).

17.2. Step 2: Define the Security Goals

Security Goals as MTSF Market Goods

After describing the scheme, specify exactly what properties you claim. In MTSF, each security goal becomes a *security good* g_j in the market catalogue $\mathcal{G}$:

Common security goals and their MTSF goods:

Scheme type	Security goal	Standard name	MTSF good
Block cipher	Pseudorandom permutation	PRP	g_{PRP}
Block cipher	Strong PRP (both directions)	SPRP	g_{SPRP}
Hash function	Collision resistance	CR	g_{CR}
Hash function	Preimage resistance	Pre	g_{Pre}
Hash function	Second-preimage resistance	Sec	g_{Sec}
Hash function	Length-extension resistance	LE	g_{LE}
Stream cipher	PRG indistinguishability	PRG	g_{PRG}
Stream cipher	Key recovery resistance	KR	g_{KR}
Signature	Existential unforgeability	EUF-CMA	g_{EUF}
Encryption	Chosen-ciphertext security	IND-CCA2	$g_{\mathsf{IND\text{-}CCA2}}$
KEM	Encapsulation security	IND-CCA2	$g_{\mathsf{IND\text{-}CCA2}}$
Protocol	Entity authentication	Auth	g_{auth}
Protocol	Mutual authentication	Mutual	g_{mutual}
Protocol	Session-key secrecy	SK	g_{sk}
Protocol	Forward secrecy	FS	g_{FS}
Protocol	Post-compromise security	PCS	g_{PCS}

Be explicit: State the full name and formal abbreviation. For example: "We prove that our scheme achieves EUF-CMA security (existential unforgeability under adaptive chosen-message attack), which in MTSF corresponds to the security good $g_{\mathsf{EUF\text{-}CMA}}$ with ask price $\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) \leq \mathsf{negl}(\lambda)$."

17.3. Step 3: Formal Security Definitions

Writing Formal Security Definitions in MTSF

For each security good, you must provide a formal game definition. In MTSF, this is the specification of what the buyer (adversary) must do to win the auction:

Template for a security definition:

1. **Setup:** The seller (challenger) generates keys/parameters and gives the public components to the buyer.
2. **Oracle access:** The buyer may make adaptive queries to oracles (signing, encryption, hash, etc.). Specify which oracles and any restrictions (e.g., "the buyer may not query the decryption oracle on the challenge ciphertext").
3. **Challenge:** The seller issues a challenge (e.g., a challenge ciphertext, a commitment).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

4. **Winning condition:** The buyer outputs a response (a forgery, a guess bit, a key). Specify exactly when the buyer "wins."
5. **Advantage:** Define $\mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda) = |\Pr[\text{buyer wins}] - \text{baseline}|$, where the baseline is 0 for forgery games and $1/2$ for distinguishing games.

In MTSF notation: The quoted price $\mathsf{QPrice}_0(g_j) = \mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda)$ is the advantage in this game. The security good g_j is *in equilibrium* if $\mathsf{QPrice}_0 \leq \mathsf{negl}(\lambda)$ for all PPT buyers $\mathcal{A}$.

17.4. Step 4: Define the Adversary Model

The Adversary Model as a Buyer Profile

The adversary model specifies the buyer's capabilities in the security market. Different adversary models correspond to different buyer profiles:

For primitives (block ciphers, hash functions, stream ciphers):

Adversary model	MTSF buyer profile	Typical budget
Chosen-plaintext (CPA)	Buyer with encryption oracle	q_E queries
Chosen-ciphertext (CCA)	Buyer with enc. + dec. oracles	$q_E + q_D$ queries
Related-key	Buyer can request related keys	q_{rk} related-key queries
Known-plaintext (KPA)	Buyer observes (m, c) pairs	q_{obs} observations
Nonce-misuse	Buyer can force nonce reuse	$\mathsf{Ask} = 1$ if misused
Side-channel	Buyer observes physical leakage	q_{phys} traces, probing depth d

For protocols:

Adversary model	MTSF buyer profile	Capabilities
Passive eavesdropper	Buyer observes all messages	Read-only network access
Active network	Buyer controls the network	Delay, drop, inject, modify
Corrupting adversary	Buyer can corrupt parties	Learns long-term keys
Quantum adversary	Buyer has quantum computer	Superposition oracle queries

MTSF formalisation: The adversary model determines the buyer's *budget* $T \leq \mathsf{poly}(\lambda)$ and *oracle access set*. Specify these explicitly:

- "The buyer is a PPT algorithm with budget $T \leq \lambda^c$ for any constant c."
- "The buyer may make at most q_S signing queries, q_H hash queries, and q_D decapsulation queries."
- "The buyer controls the network: it may delay, reorder, drop, and inject messages between honest parties."
- "The buyer may corrupt up to t out of n parties, learning their long-term secret keys."

17.5. Step 5: Construct the MTSF Security Proof

The MTSF Proof Pipeline

The proof follows the four-step MTSF pipeline (Figure 28):

Phase 1: Set up the market (Bidding Round $\mathbf{B}_0$).

1. Define the market $\mathcal{M} = (\mathsf{Seller}, \{\mathsf{Buyer}_i\}, \mathcal{G}, \mathsf{Price})$.
2. State the quoted price: $\mathsf{QPrice}_0(g_j) = \mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda)$.
3. Describe the real game: the seller runs the scheme honestly; the buyer interacts via oracles.
4. Emphasise: the quoted price stands unreduced in $\mathbf{B}_0$. No analysis has been performed yet.

Phase 2: Bidding-round chain ($\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_q$).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

For each attack vector, construct a bidding round. Use this checklist from the bid taxonomy (Appendix B):

- ☐ **Nonce freshness bid:** Abort on nonce collision. $\Delta\mathsf{Price} \leq q^2/2N$ (birthday bound).
- ☐ **Hash collision bid:** Abort on RO collision. $\Delta\mathsf{Price} \leq q_H^2/2^{n+1}$.
- ☐ **Oracle programming:** Embed hard-problem challenge. $\Delta\mathsf{Price} = 0$.
- ☐ **Computational hop:** Replace structured value with random. $\Delta\mathsf{Price} \leq \mathsf{Adv}^{\mathsf{hard}}$.
- ☐ **Forgery/extraction bid:** Convert adversary success to hard-problem solution. $\Delta\mathsf{Price} \leq L \cdot \mathsf{Adv}^{\mathsf{hard}}$.
- ☐ **Side-channel bid:** Model physical leakage. $\Delta\mathsf{Price} \leq \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$.

For each round, state: (a) purpose, (b) what is replaced, (c) the bad event(s) F_k, (d) the price adjustment $\Delta\mathsf{Price}_k$, and (e) the market interpretation (which bid fails and why).
If multiple bad events occur simultaneously, use the *extended difference lemma* (Lemma 2):

$$\Delta\mathsf{Price}_k \;\leq\; \Pr[F_1 \cup \cdots \cup F_m] \;\leq\; \sum_i \Pr[F_i] - \sum_{i<j} \Pr[F_i \cap F_j] + \cdots$$

Phase 3: Reach the ideal game ($\mathbf{B}_q$).
In the ideal game, all attack vectors have been eliminated. State: $\Pr[\mathbf{B}_q = 1] = 0$ (for forgery/authentication games) or $\Pr[\mathbf{B}_q = 1] = 1/2$ (for distinguishing games). Conclude:

$$\mathsf{QPrice}_0(g_j) \;=\; \mathsf{Ask}(g_j, \lambda) \;\leq\; \sum_{k=0}^{q-1} \Delta\mathsf{Price}_k + \Pr[\mathbf{B}_q = 1] \;\leq\; \mathsf{negl}(\lambda).$$

This establishes market equilibrium: the quoted price was a bluff.
Phase 4: Session-level verification.
If the scheme is a protocol:

1. Construct the session-CNF φ_i (Theorem 13). Fill in the four-column worksheet.
2. Verify session pinging (Theorem 15): check that consecutive sessions are structurally distinct.
3. Apply the unbounded security theorem (Theorem 3) to extend security to unbounded sessions.

17.6. Step 6: Proving Insecurity with MTSF

How to Write an MTSF Insecurity Proof

If you are demonstrating that a scheme or protocol is *insecure*, MTSF uses the same bidding-round structure:
Phase 1: Set up the market. Same as a security proof: define $\mathcal{M}$, state the quoted price QPrice_0.
Phase 2: Construct a successful bid. Instead of eliminating attack vectors, *exhibit* one that succeeds:

1. Describe the adversary's strategy (the "bid") in detail.
2. Show that the bid succeeds with non-negligible probability: $\Delta\mathsf{Price} = \Omega(1)$ or $\Delta\mathsf{Price} = 1$.
3. Identify which security good collapses.

Phase 3: Demonstrate market collapse. State:

$$\mathsf{Ask}(g_j) \;\geq\; \epsilon \;=\; \Omega(1) \quad\Rightarrow\quad \text{Market } \textbf{collapsed} \text{ for } g_j.$$

Phase 4: CNF diagnosis. Show that the session-CNF is satisfiable under the dishonest trace:

1. Fill in the CNF worksheet under the attack trace.
2. Identify which clause *should* be FALSE but is TRUE (or which clause is *missing*).
3. This pinpoints the design flaw.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Phase 5: Propose a fix. If possible:

1. Add the missing CNF clause (e.g., identity binding for Needham–Schroeder).
2. Re-run the MTSF proof on the fixed scheme.
3. Show that the previously successful bid now fails.

This "disprove → diagnose → fix → prove" cycle is the natural application of MTSF in protocol engineering, as demonstrated in the Telegram case study (Section 13.9).

17.7. Step 7: Structuring the Paper

Recommended Paper Structure Using MTSF

1. **Introduction.** Motivate the scheme/protocol. State the main result informally: "We prove that our scheme achieves g_X security with ask price $\mathsf{Ask} \leq \mathsf{negl}$." or "We demonstrate that Protocol Y has a market collapse: $\mathsf{Ask}(g_{\mathsf{auth}}) = 1$ via a masquerade bid."
2. **Preliminaries.** Notation, hardness assumptions, MTSF definitions (reference this article).
3. **Scheme/Protocol Description.** Full specification with diagrams (Step 1 above).
4. **Security Goals and Definitions.** Market goods, formal game definitions, adversary model (Steps 2–4).
5. **MTSF Security Market Setup.** Define the market $\mathcal{M}$; state the quoted price QPrice_0 (Phase 1 of Step 5).
6. **Security Proof (Bidding-Round Chain).** The main technical contribution. Each bidding round is a subsection with the standard tcolorbox template (Purpose, Replaces, Quoted Price/Complexity).
7. **CNF Session Verification.** The session-CNF formula and filled-in worksheet (Phase 4 of Step 5).
8. **Session Pinging and Unbounded Security.** Structural distinctness check and induction theorem application.
9. **Discussion.** Interpret the ask price bounds. Compare with prior work. Discuss tightness.
10. **Conclusion.** Summarise the market verdict (equilibrium or collapse) and implications.

✎ **Analogy:**

Writing a Security Paper Is Like Conducting a Market Audit Think of your research paper as an audit report for a financial market:

- **Introduction:** "We audited the XYZ Financial Market."
- **Scheme description:** "Here is how the market operates."
- **Security goals:** "These are the goods the market claims to sell."
- **Adversary model:** "These are the traders who might try to manipulate the market."
- **Security proof:** "We tested every possible manipulation strategy. Each one fails."
- **CNF worksheet:** "Here is the compliance checklist. Every box is ticked."
- **Session pinging:** "The market has been tested across unlimited trading sessions."
- **Conclusion:** "Market verdict: equilibrium (healthy) / collapse (needs intervention)."

For an insecurity paper, the audit reveals a manipulation strategy that succeeds, identifies which compliance check was missing, and recommends a fix.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

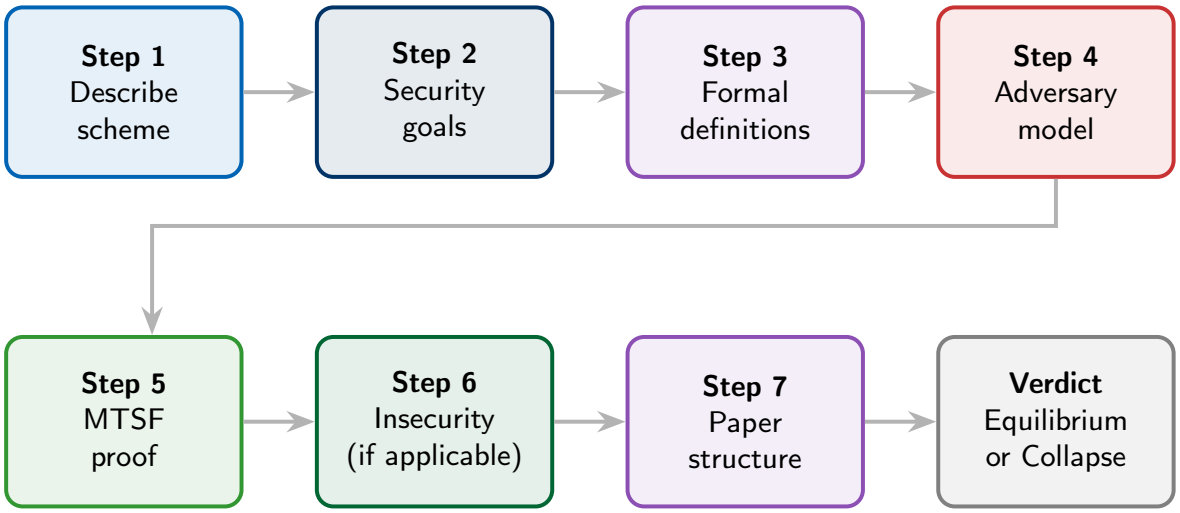


Figure 64. The seven-step pipeline for writing a security research paper using MTSF. Steps 1–4 set up the scheme, goals, definitions, and adversary model. Step 5 constructs the MTSF proof. Step 6 handles insecurity (if applicable). Step 7 organises the paper. The final verdict is equilibrium (secure) or collapse (insecure).

18. Presenting MTSF Proofs for Different Publication Venues

Cryptographic research is published across venues with vastly different page limits, audiences, and expectations. A proof that fills 40 pages at CRYPTO would be rejected at IEEE Transactions on Information Forensics and Security (TIFS) for exceeding the 13-page limit. Conversely, a compressed 8-page proof at a journal might be criticised at EUROCRYPT for lacking detail. This section provides concrete, venue-specific guidance for presenting MTSF proofs at three classes of publication venue: (1) page-restricted IEEE Transactions, (2) page-unrestricted Springer/Elsevier journals, and (3) cryptographic conferences with lengthy proceedings.

✻ Simple Terms: Why Venue-Specific Presentation Matters

The core problem: An MTSF proof of a real protocol (e.g., TLS 1.3) naturally contains: market setup, 5–10 bidding rounds each with a tcolorbox explanation, extended difference lemma application, CNF worksheet, session pinging theorem, and potentially a composition argument. In full detail, this can span 30–50 pages. But different venues have different constraints:

- **IEEE Transactions** (TIFS, TDSC, TPAMI): Strict 12–14 page limit (double-column, 10pt). The audience is broad: information security practitioners, engineers, applied researchers. They want the result and its implications, not 40 pages of game hops.
- **Springer/Elsevier journals** (Designs, Codes and Cryptography; Journal of Cryptology; Journal of Computer Security): Flexible page limits (20–60+ pages). The audience is a mix of theorists and practitioners. They expect rigour but appreciate accessibility.
- **Cryptographic conferences** (CRYPTO, EUROCRYPT, ASIACRYPT, TCC, CCS, S&P): Extended proceedings (25–40 pages). The audience is expert cryptographers who expect full proofs with every detail, every bound, and every reduction spelled out. Supplementary material or full versions on ePrint are standard.

MTSF's modular structure is uniquely suited to venue adaptation: the same proof can be presented at different levels of detail by expanding or compressing individual bidding rounds, CNF worksheets, and session pinging arguments. The core result (the ask price bound) and the market verdict (equilibrium or collapse) remain identical across all presentations.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

18.1. Case 1: IEEE Transactions (Page-Restricted, Broad Audience)

IEEE Transactions Template: 12–14 Pages, Double-Column

Target venues: IEEE Transactions on Information Forensics & Security (TIFS), IEEE Transactions on Dependable and Secure Computing (TDSC), IEEE Communications Letters, IEEE Access.
Page budget: 12–14 pages double-column (approximately 7,000–8,000 words).
Audience: Applied security researchers, network engineers, system architects, government analysts. Many readers are *not* expert cryptographers. They want to know: (a) what the result is, (b) why they should trust it, (c) what the practical implications are.

Recommended MTSF paper structure for IEEE:

1. **Introduction (1.5 pages):** State the problem, the result (equilibrium or collapse with explicit ask price), and the practical implication. Include a 1-paragraph "MTSF in 30 seconds" summary for readers unfamiliar with the framework. Use the market analogy: "Security is a marketplace where attackers bid computational resources; our protocol's security goods cannot be purchased at any affordable price."
2. **Preliminaries (1 page):** Notation table (use the Quick-Reference Card 1 format from Appendix B). State hardness assumptions in one line each. Reference this MTSF article for full definitions: "We work in the MTSF framework of [this article]; see Definitions X–Y therein."
3. **Scheme/Protocol Description (1.5 pages):** Full specification with *one* sequence diagram. For IEEE, include a practical context paragraph: "This protocol is deployed in 3.5 billion devices worldwide" or "This cipher is the NIST lightweight standard."
4. **Security Market and Main Theorem (1 page):** Define the market $\mathcal{M}$ in one paragraph. State the main theorem with full bound:

   Theorem (Main). *The ask price satisfies* $\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF}} + q_N^2/2^{\lambda+1} + \mathsf{negl}(\lambda)$. *For* $\lambda = 128$, *this gives* $\mathsf{Ask} \leq 2^{-128}$: *market equilibrium.*

5. **Proof Sketch (3–4 pages):** This is the core of the IEEE version. Present each bidding round as a *single paragraph* (not a full tcolorbox). Use a **compressed bidding-round table** instead of individual subsections:

Round	Bid type	What changes	Price adjustment
$\mathbf{B}_0$	Baseline	Nothing (quoted price QPrice_0)	—
$\mathbf{B}_1$	Nonce freshness	Abort on nonce collision	$\leq q_N^2/2^{\lambda+1}$
$\mathbf{B}_2$	Oracle programming	Embed hard-problem challenge	$= 0$
$\mathbf{B}_3$	Forgery extraction	Reduce to ECDLP/MLWE	$\leq \mathsf{Adv}^{\mathsf{hard}}$
$\mathbf{B}_4$	Ideal market	All bids fail	$\Pr[\mathbf{B}_4 = 1] = 0$
Total:			$\mathsf{Ask} \leq q_N^2/2^{\lambda+1} + \mathsf{Adv}^{\mathsf{hard}}$

   For the **key bidding round** (typically the reduction to the hard problem), provide 1–2 paragraphs of detail. For all other rounds, the table entry plus one sentence suffices.
   If the extended difference lemma is used, state it as: "Rounds $\mathbf{B}_1$–$\mathbf{B}_2$ involve simultaneous failures $F_{\mathsf{nonce}} \cup F_{\mathsf{hash}}$; by the extended difference lemma [ref], $\Delta\mathsf{Price} \leq \Pr[F_{\mathsf{nonce}}] + \Pr[F_{\mathsf{hash}}]$."
6. **CNF Verification (0.5 page):** Present the CNF formula in one equation and the filled-in worksheet as a small table. Omit the five-phase algorithm; reference it.
7. **Session Pinging (0.5 page):** State: "By Theorem [ref], the protocol is secure for unbounded sessions with $\delta_{\mathsf{ping}} \leq \mathsf{negl}$." Include the structural distinctness check as a 3-line argument, not the full induction proof.
8. **Discussion and Comparison (1.5 pages):** Compare with prior work. Discuss parameter choices. Address practical deployment considerations. Include a battle-card-style summary figure (see Figure 68).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

9. **Conclusion (0.5 page):** "Market verdict: equilibrium. Our protocol achieves 128-bit security under the MLWE assumption, with unbounded session guarantees and composability."

What to move to supplementary material / ePrint:

- Full tcolorbox descriptions of each bidding round (Purpose, Replaces, Quoted Price).
- Complete extended difference lemma calculations with inclusion-exclusion.
- Full induction proof for the session pinging theorem.
- Side-channel bid analysis.
- Detailed CNF clause-by-clause verification.

Key IEEE formatting tip: Use the compressed bidding-round table as the centrepiece of the proof. Reviewers can verify the proof structure at a glance. The full version on ePrint provides all details.

✎ Analogy:

IEEE Papers as Executive Summaries Think of an IEEE Transactions paper as an *executive summary* of a financial audit. The board of directors (reviewers) does not want to read every ledger entry. They want: (1) the bottom line (market verdict), (2) the methodology (MTSF bidding rounds), (3) the key finding (which bids fail and why), and (4) the recommendation (deploy/do not deploy). The full audit report (ePrint version) is available for anyone who wants to check every number.

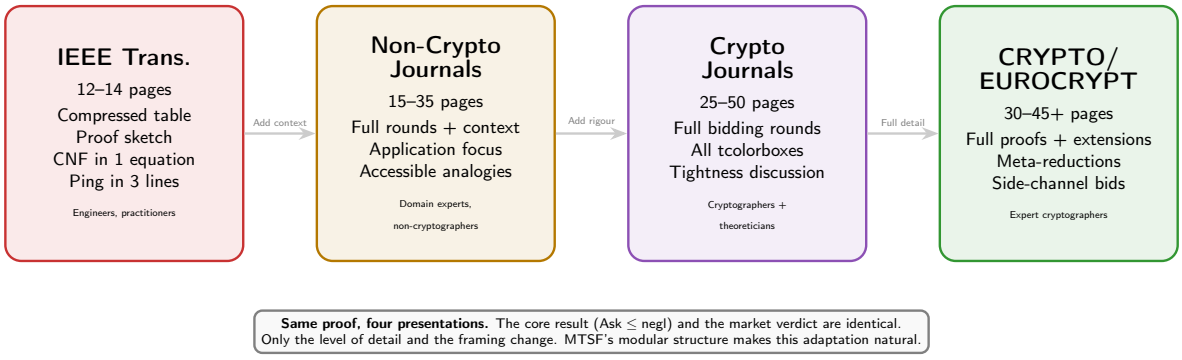


Figure 65. Four venue-specific presentations of the same MTSF proof. IEEE: compressed table. Non-crypto journals: full rounds with application context and analogies. Crypto journals: full rigorous proofs with tightness. CRYPTO/EUROCRYPT: full proofs with extensions, meta-reductions, and composition.

18.2. Case 2a: Cryptographic Journals (Page-Unrestricted, Expert Cryptographic Audience)

Cryptographic Journal Template: 25–50 Pages, Rigorous and Thorough

Target venues: Journal of Cryptology (JoC, Springer—the flagship journal of the IACR), Designs Codes and Cryptography (DCC, Springer), Journal of Mathematical Cryptology (de Gruyter), Applicable Algebra in Engineering Communication and Computing (AAECC, Springer), Finite Fields and Their Applications (Elsevier), IACR Communications in Cryptology.
Page budget: 25–50+ pages (single-column, 11pt or 12pt). No strict upper limit; mathematical depth and rigour govern acceptance.
Audience: Expert cryptographers, mathematicians with cryptographic interests, and advanced graduate students. Reviewers are specialists who have themselves written game-based or UC proofs. They will check every bound, every reduction, and every simulation argument. They expect:

- Full formal definitions for every security notion used (not just citations).
- Complete proofs with no "it is easy to see" gaps.
- Explicit tightness analysis—how much does the reduction lose, and is that loss inherent?

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- Comparison with the best known bounds for the same or related schemes.
- Discussion of whether the result extends to the QROM or post-quantum setting.

Recommended MTSF paper structure for cryptographic journals:

1. **Introduction (3–4 pages):** State the main theorem with the exact bound. Include a "technical overview" subsection (1–2 pages) that walks through the proof strategy at a high level, naming each bidding round and its cost. Compare quantitatively with prior work: "Our bound improves on [ref] by a factor of q_S in the tightness loss."
2. **Preliminaries (4–5 pages):** Full formal definitions of all hardness assumptions (ECDLP, MLWE, MSIS, gap-CDH, etc.) and all security notions (EUF-CMA, IND-CCA2, PRP, SUF-CMA). Include the MTSF definitions: market (Theorem 8), bid (Theorem 9), ask price and equilibrium (Theorem 11), quoted price (Theorem 10), extended difference lemma (Lemma 2). For a JoC audience, state these formally with full mathematical precision—no beginner boxes needed in the main body.
3. **Scheme/Protocol Description (2–4 pages):** Precise algorithmic specification using algorithm boxes (KeyGen, Sign, Verify or Enc, Dec). Correctness theorem. For protocols, a sequence diagram with exact message formats.
4. **MTSF Market Setup (1–2 pages):** Formal market definition $\mathcal{M} = (\mathsf{Seller}, \{\mathsf{Buyer}_i\}, \mathcal{G}, \mathsf{Price})$. List all security goods with their formal game definitions. State the quoted price $\mathsf{QPrice}_0(g_j) = \mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda)$. Establish the relationship between MTSF goods and standard security notions.
5. **Security Proof—Bidding-Round Chain (10–20 pages):** The core contribution. Present each bidding round with:
   - The complete MTSF tcolorbox (Purpose, Replaces, Quoted Price) with full technical detail.
   - A rigorous formal argument: "Let F_k denote the event... By Lemma 2, $|\Pr[\mathbf{B}_{k-1}{=}1] - \Pr[\mathbf{B}_k{=}1]| \leq \Pr[F_k]$. We bound $\Pr[F_k]$ as follows..."
   - For computational hops: the complete reducer construction $\mathcal{R}^{\mathcal{A}}$ with explicit simulation of every oracle.
   - For the extended difference lemma: the full inclusion-exclusion calculation showing which intersection terms vanish.
   - The bidding-round chain figure.

   Tightness analysis (required for JoC/DCC): After the main proof, include a dedicated subsection:
   - State the tightness loss factor $L(\lambda)$ explicitly.
   - Compare with known lower bounds on tightness (e.g., Coron's impossibility result for ROM-based signature reductions).
   - If possible, provide a meta-reduction or matching attack showing the loss is inherent.
6. **CNF Session Verification (2–3 pages):** Full session-CNF with all clauses. Complete five-phase verification algorithm. Worksheets for honest and dishonest traces. Formal proof of adversarial CNF unsatisfiability (Theorem 1).
7. **Session Pinging (2–3 pages):** Complete structural distinctness verification. Full induction proof (not a sketch). Quantitative advantage accumulation analysis.
8. **Extensions (2–5 pages):**
   - Side-channel bid with d-probing model analysis.
   - QROM extension (if applicable): O2H or measure-and-reprogram.
   - Composition via market merger theorem (if applicable).
   - Nonce-misuse resilience analysis.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

9. **Comparison with Prior Work (2–3 pages):** Detailed table with exact numerical bounds, tightness factors, assumptions, and whether composition/unbounded sessions are addressed.
10. **Conclusion (1 page):** State the market verdict with the exact bound. List concrete open problems.

Cryptographic journal-specific tips:

- **No beginner boxes in the main body.** Save pedagogical elements for an appendix or the ePrint version. JoC/DCC reviewers know what IND-CCA2 means.
- **Include every tcolorbox.** Even expert readers benefit from the structured (Purpose, Replaces, Quoted Price) template—it provides a standardised format that makes 20-page proofs navigable.
- **Name every bid from the taxonomy.** Explicitly reference the bid taxonomy (Appendix B) and state which bids are relevant, which are addressed, and which are out of scope (with justification).
- **Tightness is mandatory.** Unlike non-crypto journals, JoC/DCC reviewers *require* a tightness discussion. A tight reduction is a selling point; a loose reduction must be justified.
- **State the quoted price in $\mathbf{B}_0$.** Write: "The seller quotes $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{EUF}}_{\mathcal{A}}(\lambda)$. We decompose QPrice_0 into five negligible terms…"

✎ Analogy:

Cryptographic Journal Papers as Peer-Reviewed Legal Briefs A paper in the Journal of Cryptology is like a legal brief submitted to a supreme court of cryptography. Every claim must be backed by a rigorous argument. Every reduction must be spelled out completely. Every bound must be proven, not estimated. The "judges" (reviewers) are themselves expert lawyers (cryptographers) who will scrutinise every line. MTSF's structured tcolorbox format serves as a standardised "brief template" that ensures no argument is omitted and every step is labelled—making both writing and reviewing more systematic.

18.3. Case 2b: Non-Cryptographic Journals (Page-Unrestricted, Domain-Specific Audience)

Non-Cryptographic Journal Template: 15–35 Pages, Application-Focused

Target venues: Vehicular Communications (Elsevier), Computer Networks (Elsevier), Computer Communications (Elsevier), Ad Hoc Networks (Elsevier), Future Generation Computer Systems (Elsevier), Pervasive and Mobile Computing (Elsevier), Journal of Network and Computer Applications (Elsevier), Sensors (MDPI), Electronics (MDPI), Applied Sciences (MDPI), Information Sciences (Elsevier), Journal of Information Security and Applications (Elsevier), Wireless Personal Communications (Springer), Peer-to-Peer Networking and Applications (Springer), Cluster Computing (Springer), Journal of Ambient Intelligence and Humanized Computing (Springer), International Journal of Information Security (Springer).
Page budget: 15–35 pages. Some venues (e.g., Elsevier journals) have flexible limits; others (e.g., MDPI journals) suggest 15–25 pages. Quality and relevance govern length.
Audience: Domain experts in networking, IoT, vehicular systems, mobile computing, or applied security. Most readers are *not* expert cryptographers. They are engineers, system designers, and applied researchers who:

- Understand the *application* (e.g., vehicle-to-vehicle communication, smart grid security, healthcare IoT) deeply.
- Know that security is important but may not have written a security proof themselves.
- Want to know: Is this protocol secure? How secure? What are the practical parameters? Does it compose with existing deployed infrastructure?

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- Appreciate clear explanations, real-world examples, and diagrams over dense mathematical notation.

Recommended MTSF paper structure for non-cryptographic journals:

1. **Introduction (2–3 pages):** Lead with the *application*, not the cryptography. For Vehicular Communications: "Connected vehicles exchange safety messages 10 times per second. A single forged brake warning could cause a multi-car pile-up. We prove that our V2X authentication protocol is secure using the Market-Theoretic Security Framework (MTSF)." Include a "MTSF in one paragraph" box:

   > *MTSF models security proofs as economic markets. The protocol designer (seller) offers security guarantees (goods). Potential attackers (buyers) bid computational resources to break them. If no attacker can afford to win, the market is in equilibrium—the protocol is secure. If any attacker wins cheaply, the market collapses—the protocol is broken.*

   State the main result in plain language: "Our protocol achieves 128-bit authentication security: the probability that an attacker successfully impersonates a vehicle is less than 2^{-128}, even after observing billions of legitimate messages."
2. **Related Work (2–3 pages):** Review the application-domain literature (vehicular security, IoT authentication, etc.) as well as the proof methodology. Position MTSF relative to informal "security analysis" sections common in non-crypto journals: "Unlike prior work that argues security informally, we provide a *mathematical proof* within the MTSF framework, producing explicit numerical bounds and session-level auditing."
3. **System Model and Threat Model (2–3 pages):** This is critical for non-crypto journals. Describe the system in domain-specific language:
   - **For vehicular communications:** Network topology (V2V, V2I), communication range, message frequency, latency requirements, on-board unit capabilities.
   - **For IoT:** Device constraints (RAM, CPU, battery), communication protocols (MQTT, CoAP, BLE), gateway architecture.
   - **For healthcare:** Patient data sensitivity, regulatory requirements (HIPAA, GDPR), device types (wearables, implants, monitors).

   Map the threat model to MTSF buyer profiles:

Domain threat	MTSF buyer profile	Bid type
Rogue vehicle broadcasting fake alerts	Active network adversary	Forgery bid
Eavesdropper on V2V channel	Passive eavesdropper	Distinguishing bid
Stolen on-board unit	Corrupting adversary	Key recovery bid
Replay of yesterday's traffic signal	Session replay attacker	Nonce freshness bid
GPS spoofing combined with message injection	Multi-vector attacker	Extended diff. lemma

4. **Proposed Protocol/Scheme (3–5 pages):** Full specification with:
   - A system architecture diagram showing all parties and communication channels.
   - A message sequence diagram showing the protocol flow.
   - Algorithm boxes for key operations.
   - A "design rationale" paragraph for each component: "We use ECDSA for authentication because it provides 128-bit security with 64-byte signatures, fitting within the 300-byte V2X message budget."
   - Performance analysis: computation time, communication overhead, storage requirements. Non-crypto journal reviewers *expect* this; crypto journal reviewers consider it optional.
5. **MTSF Security Analysis (5–8 pages):** This section must be *self-contained*—do not assume the reader has read the MTSF reference article. Structure it as follows:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

5a. Market setup (1 page): Define the market in plain language with a diagram. Use the market overview figure (Figure 9). List security goods with one-sentence descriptions:

> *Our protocol offers three security goods: (1) g_{auth}: a rogue vehicle cannot impersonate a legitimate one; (2) g_{sk}: session keys are secret even to eavesdroppers; (3) g_{fresh}: replayed messages are always detected.*

5b. Bidding-round chain (3–5 pages): Present each round with:

- A tcolorbox (Purpose, Replaces, Quoted Price) for the key rounds.
- For routine rounds (nonce freshness, oracle programming), a single-paragraph summary is sufficient.
- **Include an analogybox or intuitionbox for the most technical step.** Non-crypto readers need the "why" not just the "what." For example: "This step is like checking that every licence plate on the road is unique. If two cars share a plate, the system cannot tell them apart—a dangerous confusion. The birthday bound shows that with 256-bit nonces, a collision requires observing 2^{128} messages—far more than any road network will ever produce."
- State the final bound in both mathematical and plain-language form: "$\mathsf{Ask}(g_{\mathsf{auth}}) \leq 2^{-128}$. In practical terms, an attacker would need to perform more operations than there are atoms in the universe to have even a one-in-a-trillion chance of forging a vehicle identity."

5c. CNF worksheet (1 page): Present the filled-in worksheet with domain-specific check descriptions:

Check	What it means	How verified	Pass?
Session ID valid	Message belongs to this session	Compare session tokens	T
Timestamp fresh	Message is not a replay	Check timestamp window	T
Signature valid	Sender is who they claim	Verify ECDSA signature	T
Certificate valid	Sender's key is CA-certified	Check certificate chain	T

5d. Session pinging (0.5 page): State the unbounded security result in one paragraph: "Our protocol is secure for an unlimited number of sessions. Each new session uses fresh nonces and session-specific signatures, ensuring that security does not degrade over time—even after millions of vehicle encounters."

6. **Performance Evaluation (3–5 pages):** This section is *critical* for non-crypto journals and often *absent* from crypto journals. Include:
   - **Computation cost:** Signing time, verification time, key generation time on target hardware (e.g., "ECDSA-P256 verification takes 1.2 ms on an ARM Cortex-M4").
   - **Communication overhead:** Message sizes, number of round trips, bandwidth consumption.
   - **Storage:** Key sizes, certificate sizes, session state.
   - **Comparison table:** Your protocol vs. competing protocols on all metrics (security level, computation, communication, storage, composition, unbounded sessions).
   - **Simulation or implementation results:** If applicable, ns-3 simulation for vehicular networks, testbed results for IoT, or benchmarks on embedded hardware.
7. **Discussion (1–2 pages):** Address practical deployment: How does this integrate with existing infrastructure (e.g., IEEE 1609.2 for V2X, or the ETSI ITS-G5 standard)? What are the limitations? What happens if the CA is compromised?
8. **Conclusion (0.5–1 page):** State the market verdict in domain language: "We have proven that our V2X authentication protocol achieves market equilibrium with ask price 2^{-128}, meaning no computationally bounded attacker can forge vehicle identities, intercept session keys, or replay messages—even across unlimited driving sessions."

Non-cryptographic journal-specific tips:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- **Lead with the application, not the math.** The first sentence should mention vehicles, IoT devices, patients, or smart grids—not "game-based proofs" or "UC framework."
- **Include 3–5 analogy/intuition boxes.** Non-crypto reviewers appreciate these. A well-chosen analogy ("nonce collision is like two cars sharing a licence plate") can make the difference between acceptance and a "too theoretical" rejection.
- **Performance evaluation is mandatory.** Non-crypto journals rank experimental results highly. A security proof without performance data may be desk-rejected.
- **Translate every bound into plain language.** After every formula, write: "In practical terms, this means…"
- **Include a security-vs-performance trade-off discussion.** "Increasing the nonce size from 128 to 256 bits doubles the signature overhead but increases the collision resistance from 2^{64} to 2^{128} messages."
- **Use the battle-card format for the security summary.** A visual one-page summary (like Figure 68) is more accessible than a page of equations for a non-crypto audience.
- **Reference standards.** Non-crypto reviewers want to know how the protocol relates to existing standards (NIST, IEEE, ETSI, 3GPP, IETF RFC).
- **The MTSF proof elevates your paper above "informal security analysis."** Many non-crypto journal papers include only informal arguments ("the attacker cannot forge because ECDSA is hard"). A formal MTSF proof with explicit bounds, CNF worksheets, and session pinging is a significant differentiator and strengthens the contribution.

✎ Analogy:

Non-Crypto Journal Papers as Consumer Product Safety Reports A paper in Vehicular Communications or Sensors is like a *consumer product safety report*. The readers are not chemists (cryptographers)—they are product engineers and regulators who need to know: Is this product safe? How safe? What tests were performed? What are the operating limits?
The MTSF proof is the safety test. The market verdict is the safety rating. The CNF worksheet is the inspection checklist. The performance evaluation is the product specification sheet. The battle card is the safety label on the box. Package these together and the non-crypto reviewer sees a complete, professional, and trustworthy security analysis—not just "we believe this is secure because the crypto is hard."

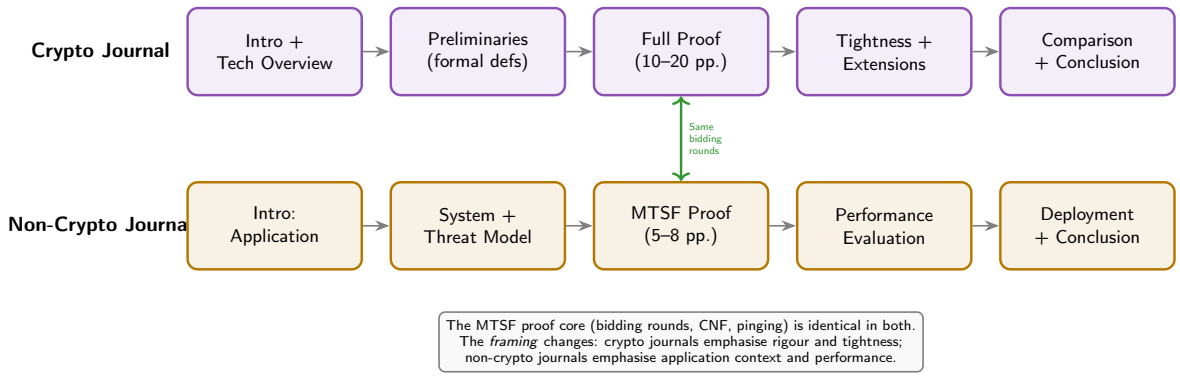


Figure 66. Paper structure comparison: cryptographic journals (top) vs. non-cryptographic journals (bottom). The MTSF proof core is shared; the framing (application context vs. tightness analysis) differs.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

18.4. Case 3: Cryptographic Conferences (Extended Proceedings, Expert Audience)

Cryptographic Conference Template: 30–45+ Pages, Expert Audience

Target venues: CRYPTO, EUROCRYPT, ASIACRYPT, TCC (theory); CCS, S&P, USENIX Security, NDSS (systems security); PKC, CT-RSA (applied crypto); CHES (hardware security); AFC (Africacrypt); Latincrypt; Indocrypt; ACISP.

Page budget: 30 pages in proceedings + unlimited supplementary material on ePrint. CRYPTO/EUROCRYPT use LNCS format (Springer, single-column, 12pt). CCS/S&P use double-column with 12–15 page limits but allow substantial appendices.

Audience: Expert cryptographers who have read Shoup [1], Bellare–Rogaway [2], Canetti [3], and the NIST post-quantum standards. They expect:

- Every reduction spelled out completely, with no "it is easy to see" gaps.
- Tightness analysis: how much does the reduction lose? Is there a matching attack showing the loss is inherent?
- Comparison with the best known bounds: does this improve on prior work quantitatively?
- A full treatment of edge cases: what happens when the adversary corrupts parties, or when nonces are misused, or when the RNG is degraded?

Recommended MTSF paper structure for CRYPTO/EUROCRYPT:

1. **Introduction (3–4 pages):** Motivate with precision. State the main theorem with the *exact* bound (not just "negligible"). Include a "technical overview" subsection (1–2 pages) walking through the proof strategy at a high level. This is where MTSF's bidding-round classification shines: "Our proof uses five bidding rounds. Round 1 eliminates nonce collisions via birthday bound. Round 2 programs the random oracle at zero cost. Round 3 reduces forgeries to ECDLP via the forking lemma with tightness loss q_H. Rounds 4–5 handle the extended difference lemma over three simultaneous failures."
2. **Preliminaries (3–4 pages):** Full formal definitions for every assumption (ECDLP, MLWE, MSIS, PRF, PRP). State MTSF definitions formally: market, bid, ask price, quoted price, equilibrium, extended difference lemma, session-CNF, session ping. For a CRYPTO audience, include the theorems (soundness, completeness, unbounded security) that you will invoke. No beginner boxes needed—the audience knows what IND-CCA2 means.
3. **Scheme/Protocol Description (2–3 pages):** Precise algorithmic specification. Use algorithm boxes (KeyGen, Sign, Verify). Include a correctness theorem: "Honest execution produces valid signatures with probability $1 - \delta_{\mathsf{correct}}$."
4. **MTSF Market Setup (1–2 pages):** Formal market definition. List all goods with their formal game definitions (not just names). State the quoted price. Discuss the relationship between MTSF goods and standard security notions: "Our good $g_{\mathsf{EUF\text{-}CMA}}$ is equivalent to the standard EUF-CMA game of [ref]; see Theorem 1."
5. **Security Proof—Full Bidding-Round Chain (10–20 pages):** This is the bulk of the paper. Present each bidding round as a full subsection with:
   - The complete MTSF tcolorbox (Purpose, Replaces, Quoted Price) with full technical detail.
   - A complete formal argument: "Let F_k denote the event…By the difference lemma (Lemma X), $|\Pr[\mathbf{B}_{k-1} = 1] - \Pr[\mathbf{B}_k = 1]| \leq \Pr[F_k]$. We bound $\Pr[F_k]$ as follows…"
   - For the extended difference lemma: the full inclusion-exclusion calculation, showing which terms are zero (independent events) and which contribute.
   - For the key reduction round: the complete reducer construction. Specify how the reducer $\mathcal{R}$ simulates each oracle query. Compute the exact running time of $\mathcal{R}$ (including the tightness overhead).
   - For the forking lemma rounds: state the forking lemma precisely, compute the success probability after forking, and state the resulting bound.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Tightness analysis: After the main proof, include a "Tightness" subsection:

- State the tightness loss: "The reduction loses a factor of $L = q_H \cdot q_S$ in Round 3."
- Discuss whether this loss is inherent: "Coron [ref] showed that any reduction from ECDLP to EUF-CMA in the ROM must lose a factor $\Omega(q_H)$. Our loss of $q_H \cdot q_S$ is thus within a q_S factor of optimal."
- If possible, provide a meta-reduction showing the loss is tight.

Extended bids: CRYPTO/EUROCRYPT papers should include bidding rounds that IEEE papers omit:

- **Side-channel bid ($\mathbf{B}_{\mathsf{sca}}$):** Full d-probing model analysis with masking order.
- **Quantum bid (if applicable):** O2H or measure-and-reprogram analysis in the QROM.
- **Nonce-misuse bid:** What happens if the implementation fails to maintain nonce uniqueness?
- **Related-key bid (for block ciphers):** Key-schedule analysis.
- **Corruption bid (for protocols):** What happens when the adversary corrupts one party?

6. **CNF Session Verification (2–3 pages):** Full CNF formula. Complete five-phase algorithm. Filled-in worksheets for both honest and dishonest traces. Formal proof of Proposition X (adversarial CNF unsatisfiability). For protocols, include the CNF for every security good separately.
7. **Session Pinging and Unbounded Security (3–4 pages):** Full structural distinctness verification for all five conditions. Complete induction proof (not just a sketch). Quantitative advantage accumulation analysis: "After n sessions, the total advantage is $\leq n \cdot \epsilon$. For $n = 2^{64}$ and $\epsilon = 2^{-128}$, this gives 2^{-64}—still negligible." Comparison with ProVerif/Tamarin results (if available).
8. **Composition (if applicable) (2–3 pages):** Market merger theorem application. Resource competition analysis. Network CNF with cross-protocol clauses. Infrastructure security analysis.
9. **Comparison with Prior Work (2–3 pages):** Detailed comparison table with exact bounds. Identify specific improvements: "Our bound improves on [ref] by a factor of q_S in the tightness loss." Discuss which MTSF features (extended difference lemma, CNF, pinging) are new relative to prior work on this scheme.
10. **Conclusion and Open Problems (1 page):** State the market verdict. List specific open problems: "Can the tightness loss be improved to $O(q_H)$? Does the scheme remain secure in the QROM? Can the side-channel bid be strengthened to handle higher-order masking?"

CRYPTO/EUROCRYPT-specific tips:

- **No beginner boxes in the main body.** The expert audience does not need "What is IND-CCA2?" explanations. Save pedagogical elements for the ePrint full version.
- **Include the tcolorboxes.** Even expert readers benefit from the structured (Purpose, Replaces, Quoted Price) template—it makes the proof easier to review and verify.
- **Name every bid.** This is MTSF's key advantage over generic game-based proofs. Reviewers can check whether all relevant bids from the taxonomy (Appendix B) have been addressed.
- **Include a bidding-round chain figure.** Even at CRYPTO, a visual summary of the proof (like Figure 30) helps reviewers navigate 20 pages of reductions.
- **State the quoted price explicitly.** In $\mathbf{B}_0$, write: "The seller quotes $\mathsf{QPrice}_0 = \mathsf{Adv}^{\mathsf{EUF}}_{\mathcal{A}}(\lambda)$. We will show $\mathsf{QPrice}_0 \leq \mathsf{negl}(\lambda)$ by decomposing it into five negligible terms."
- **Provide the ePrint full version link.** Even 30-page CRYPTO papers may need supplementary material for tightness proofs, composition details, and alternative parameter choices.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✎ Analogy:

CRYPTO Papers as Complete Court Transcripts If an IEEE paper is the judge's ruling and a Springer paper is the legal brief, a CRYPTO paper is the *complete court transcript*—every witness statement (bidding round), every piece of evidence (bound calculation), every cross-examination (tightness analysis), and every procedural ruling (extended difference lemma). The jury (programme committee) reads every word and checks every argument. Omitting a step is like a lawyer forgetting to present evidence: the case (paper) may be dismissed (rejected).

18.5. Venue Adaptation Summary

Component	IEEE (12–14 pp.)	Non-Crypto Jrnl.	Crypto Jrnl.	CRYPTO/EUR.
Bidding rounds	Compressed table	Full + analogies	Full tcolorboxes	Full + tightness
Extended diff. lemma	Cite result	Union bound + example	Full inclusion-excl.	Incl.-excl. + meta-red.
CNF worksheet	1 equation + table	Domain-specific checks	Full 4-col worksheet	Per-good worksheets
Session pinging	3-line statement	1 paragraph	Sketch + bound	Full induction proof
Side-channel bid	Omit (suppl.)	Omit	Include	Full d-probing model
Composition	Omit (suppl.)	Mention if relevant	If applicable	Full market merger
Tightness analysis	Omit	Omit	Required	Full meta-reduction
Beginner boxes	1–2 key ones	3–5 (essential)	Omit in body	Omit (ePrint only)
Performance eval.	Omit	Required (3–5 pp.)	Optional	Optional
Application context	Brief	Extensive (lead)	Brief	Brief
Figures	1–2 key	5+ with diagrams	5+ with chains	Bidding chain + more
ePrint link	Required	Optional	Recommended	Standard

Figure 67. Venue adaptation matrix: how each MTSF proof component should be presented across four venue types. IEEE: compressed. Non-crypto journals: application-focused with analogies and performance. Crypto journals: rigorous with tightness. CRYPTO/EUROCRYPT: full with extensions and meta-reductions.

☞ Key Insight:

The Golden Rule of Venue Adaptation **Never change the mathematics.** The ask price bound, the bidding-round structure, the CNF formula, and the market verdict are *identical* across all four venue presentations. Only the *level of detail* and the *framing* change:

- **IEEE:** Present the skeleton (table + key round).
- **Non-crypto journals:** Present the full body with application context, analogies, and performance evaluation.
- **Crypto journals:** Present the full rigorous body with tightness analysis.
- **CRYPTO/EUROCRYPT:** Present the full body + deep tissue (tightness + extensions + meta-reductions).

The modular structure of MTSF—where each bidding round is a self-contained unit with a standard template—makes this adaptation natural. You write the full CRYPTO version first, then

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

adapt for crypto journals by streamlining extensions, then adapt for non-crypto journals by adding application context and performance, then compress for IEEE by replacing individual tcolorboxes with a summary table.
Always provide an ePrint full version that contains the CRYPTO-level detail, regardless of the venue. This ensures that any reader who wants to verify a specific step can do so.

✻ Simple Terms: Practical Workflow for Multi-Venue Submission

If you plan to submit to multiple venues (e.g., a conference version followed by a journal version), write the paper in layers:
Layer 1 (Core—always present): Market setup, main theorem, bidding-round table, CNF formula, market verdict. This is the IEEE skeleton. ∼8 pages.
Layer 2a (Application frame—non-crypto journal): Wrap Layer 1 with application context (system model, threat model in domain language), add analogies and intuition boxes, add performance evaluation. ∼20 pages.
Layer 2b (Rigorous frame—crypto journal): Expand Layer 1 with full tcolorbox bidding rounds, inclusion-exclusion calculations, CNF worksheets, session pinging proof, and tightness discussion. ∼30 pages.
Layer 3 (Deep analysis—CRYPTO): Expand Layer 2b with tightness meta-reductions, side-channel bids, QROM extensions, composition, and corruption analysis. ∼40 pages.
The layers share a common core: Layer 2a and Layer 2b both contain Layer 1 entirely, but frame it differently (application vs. rigour). Layer 3 contains Layer 2b entirely. No mathematical content is "different"—only the presentation depth and framing change. This is the power of MTSF's modular bidding-round structure: each round can be independently expanded or reframed without affecting any other round.

18.6. MTSF Battle Cards and Proof Scorecards

✻ Simple Terms: How to Read the Battle Cards

Each battle card summarises one MTSF case study on a single visual panel. Think of them as "trading cards" for cryptographic schemes. Each card shows: (1) the scheme name and type, (2) where you encounter it in real life, (3) the security goods offered, (4) the bidding-round chain as a coloured pipeline, (5) the final ask price, and (6) the market verdict. The proof scorecards show which bids were attempted and their outcomes.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

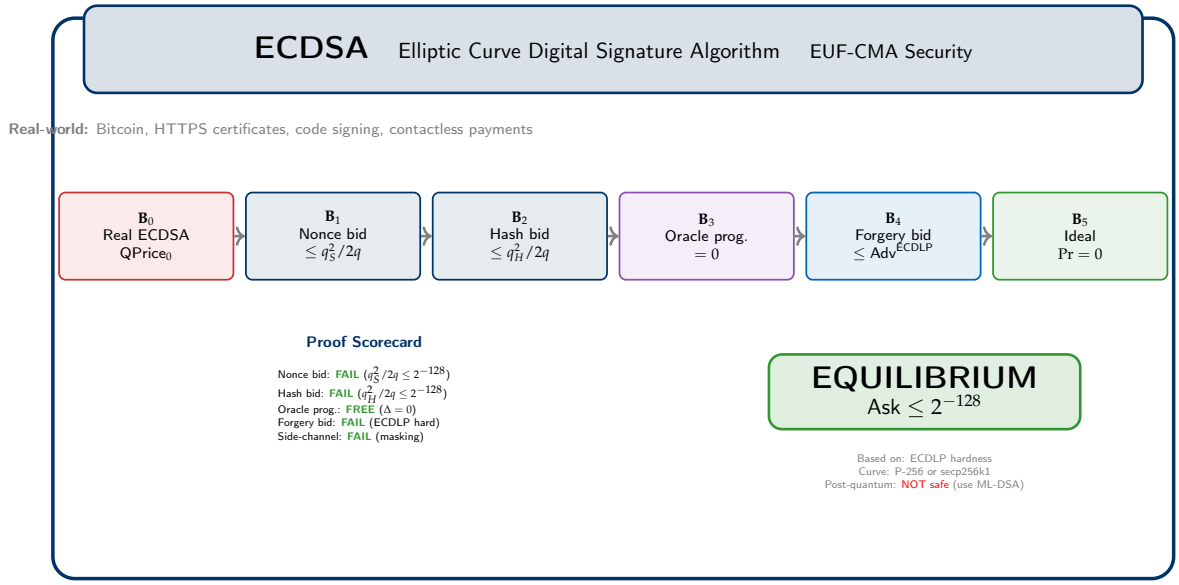


Figure 68. Battle card: ECDSA. Five bidding rounds, all bids fail, market in equilibrium with Ask $\leq 2^{-128}$.

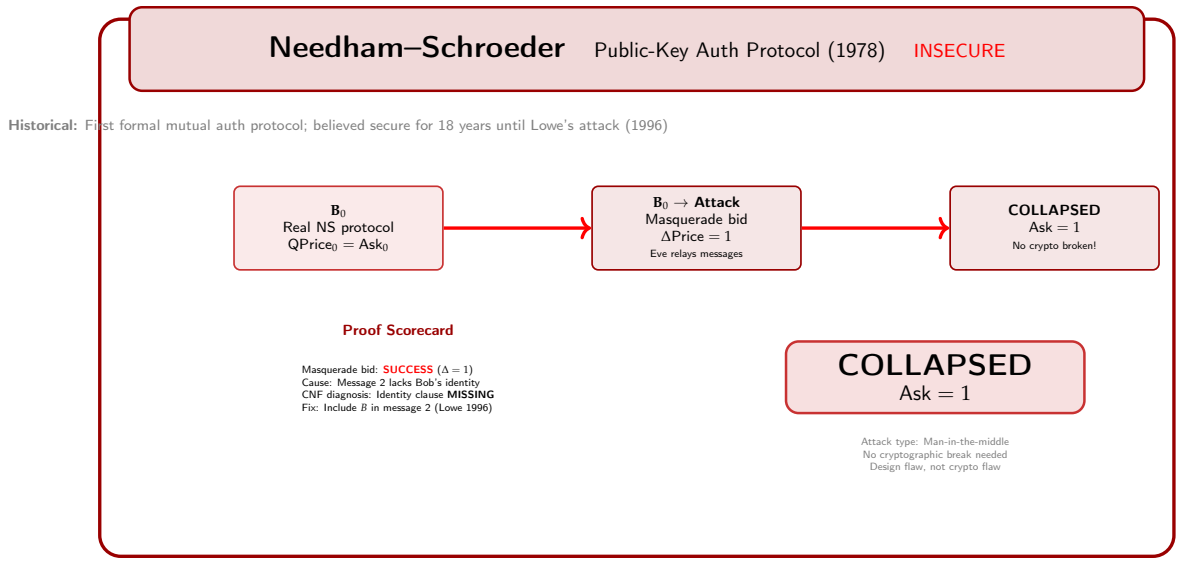


Figure 69. Battle card: Needham–Schroeder. One masquerade bid succeeds with Ask = 1. Market collapsed—protocol design flaw.

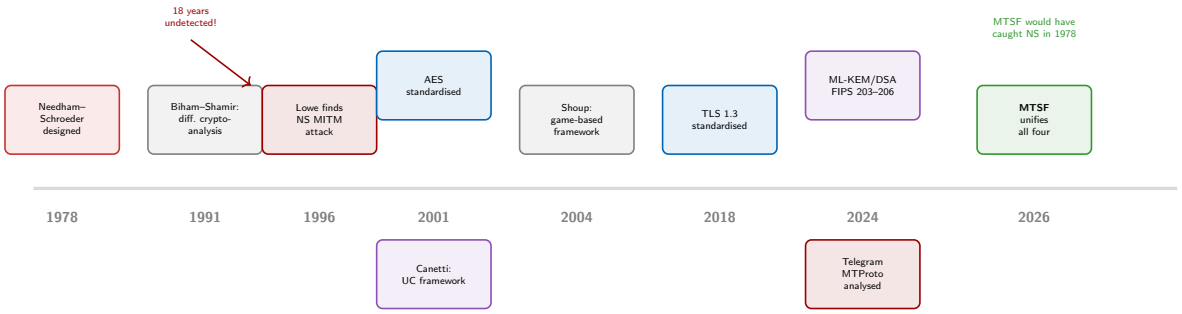


Figure 70. Historical timeline of cryptographic security paradigms. Key events from Needham–Schroeder (1978) to MTSF (2026). The annotation shows that MTSF's CNF verification would have detected the NS flaw immediately upon protocol design.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

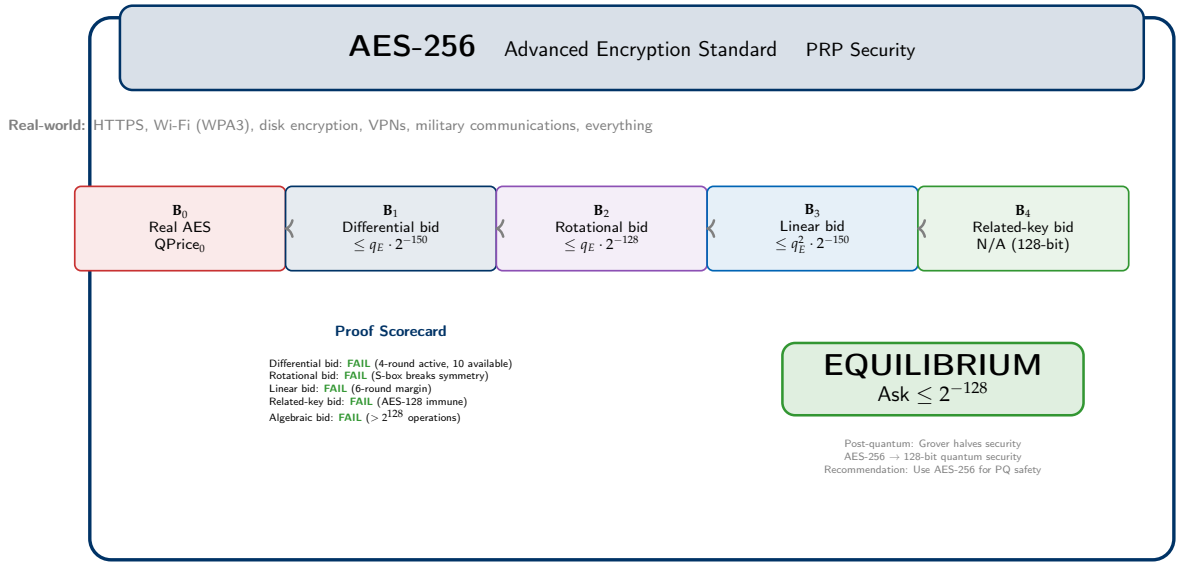


Figure 71. Battle card: AES-256. Four cryptanalytic bids, all fail with wide margins. Market in equilibrium.

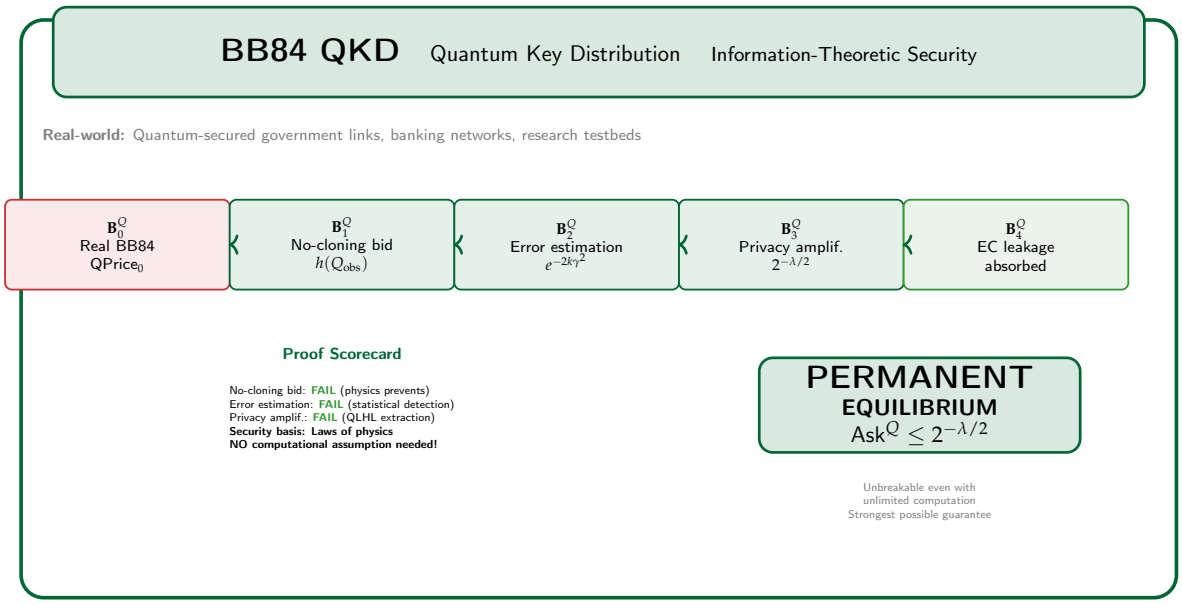


Figure 72. Battle card: BB84 QKD. Information-theoretic security—permanent equilibrium based on physics, not computation.

19. Related Work

19.1. Game-Based Security Proofs

The foundations of game-based proofs were laid by Bellare and Rogaway [2], who introduced the sequence-of-games methodology, and by Shoup [1], who formalised the *difference lemma*: if two games $\mathbf{B}_k$ and $\mathbf{B}_{k+1}$ differ only on event F, then $|\Pr[\mathbf{B}_k = 1] - \Pr[\mathbf{B}_{k+1} = 1]| \leq \Pr[F]$. This has become the central tool for bounding game-hop probabilities. Our *extended difference lemma* (Lemma 2) strictly generalises Shoup's single-failure version by capturing m simultaneous failure events $F_1, \ldots, F_m$ via $\Pr[\bigcup_i F_i]$ with inclusion-exclusion tightening. Whereas Shoup's lemma requires artificially splitting multi-failure hops into multiple sequential games, our lemma handles them in a single step, tightening bounds and reducing proof length.

Bellare and Neven [14] introduced the general forking lemma for extracting two consistent forgeries from a signature adversary, which we use in the ECDSA proof (Section 9.1). Bellare and Rogaway's random oracle model [12] underpins our ROM-based proofs; the QROM extension (Section 14) addresses the limitation that classical ROM proofs do not account for quantum superposition queries.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

The market-theoretic language we introduce—"seller," "buyer," "ask price," "bidding round," "equilibrium," and "collapse"—has no precedent in the cryptographic proof literature. All prior game-based work uses standard game/adversary/challenger terminology without the economic framing. MTSF is the first framework to unify proof methodology with market theory.

19.2. Universal Composability and GUC

Canetti's Universal Composability (UC) framework [3] provides the gold standard for composable security: a protocol proven UC-secure remains secure when run concurrently with arbitrary other protocols. UC proofs require constructing an ideal functionality $\mathcal{F}$ and a simulator $\mathcal{S}$ that makes the real world indistinguishable from the ideal world for any environment $\mathcal{Z}$. In MTSF, the environment $\mathcal{Z}$ is recast as a *market regulator*, the simulator $\mathcal{S}$ as a *market arbitrageur*, and the ideal functionality as the *seller's security goods*.

Generalised UC (GUC) [4] extends UC to allow shared state (global common reference strings, public-key infrastructure). In MTSF, GUC shared functionality corresponds to *market infrastructure*—the PKI, key registration systems, and trust anchors that all parties can access. Camenisch, Manulis, and Neven [5] introduced CNF-based session correctness verification within the GUC framework, requiring sessions to satisfy a conjunction of Boolean clauses derived from the protocol's message structure. We directly build on their CNF formalism, extending it with (a) a canonical five-phase verification algorithm (Algorithm 1), (b) an easy four-column manual worksheet for practitioner use, (c) integration with session pinging for unbounded coverage, and (d) a "dishonest-trace unsatisfiability" proof strategy that replaces ad hoc protocol analysis.

19.3. Formal Verification of Protocols

ProVerif [6] is the leading automated tool for protocol security verification, based on the applied pi-calculus. It can automatically prove secrecy, authentication, and other properties for unbounded sessions, but does not produce numerical advantage bounds. Tamarin [7] uses multiset rewriting rules and provides user-guided proofs of complex protocols (including TLS 1.3 and 5G AKA), again without quantitative bounds. CryptoVerif [8] bridges the gap by producing machine-checked game-based proofs with concrete bounds, but requires significant manual guidance and does not support the UC/GUC framework natively.

MTSF's *session pinging* mechanism (Section 5) provides unbounded session coverage comparable to ProVerif's symbolic analysis, while maintaining quantitative advantage bounds comparable to CryptoVerif's game-based approach. This combination—unbounded + quantitative—is unique to MTSF.

The EasyCrypt proof assistant [59] supports machine-checked cryptographic proofs in the game-based style. Mechanising MTSF proofs in EasyCrypt is an important avenue for future work, particularly for the extended difference lemma and QROM case studies.

19.4. TLS 1.3 and Protocol Security Analysis

Transport Layer Security 1.3 [51] represents the most thoroughly analysed deployed cryptographic protocol in history. Its design was informed by a decade of attacks on TLS 1.2 [53,60,61], and it underwent extensive formal analysis both before and after standardisation.

The most comprehensive formal analysis of TLS 1.3 is by Dowling, Fischlin, Günther, and Stebila [54], who prove security of the full TLS 1.3 handshake in the ACCE (Authenticated and Confidential Channel Establishment) model [62]. Their proof covers the 1-RTT handshake, resumption via PSK, and the 0-RTT mode, demonstrating that the 1-RTT handshake achieves authenticated key exchange with forward secrecy, while 0-RTT provides confidentiality only against *passive* adversaries (not active replay). A parallel analysis by Cremers, Hövelmanns, and Rausch [63] using symbolic verification in Tamarin confirmed the handshake properties including downgrade resilience. Our MTSF analysis (Section 13.8) is complementary: we provide the first *market-theoretic* formalisation

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

of TLS 1.3, casting the ACCE proof as a ten-bidding-round chain and formalising the 0-RTT replay collapse and downgrade attack as explicit market collapse theorems.

The HKDF key schedule [52,64] is central to TLS 1.3's security. Krawczyk [64] proved HKDF's security as a PRF under the assumption that HMAC is a PRF, and as an indistinguishable random oracle extractor under weaker assumptions. In our MTSF proof (Section 13.8.3), HKDF's PRF security accounts for three separate price adjustments (one per extract/expand step in the HS, MS, and CATS derivations), each bounded by $\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}$.

The 0-RTT replay vulnerability is acknowledged explicitly in RFC 8446, Section 8, and in Fischlin, Günther, and Marson [65], who formally define the "1-RTT-then-0-RTT" security model and prove that 0-RTT achieves confidentiality against passive adversaries but provides no replay protection. Our MTSF formulation (Theorem 56) provides a direct market-collapse proof: the replay bid is free ($\mathsf{Ask} = 1$) without server-side anti-replay, and we identify the missing CNF clause ($\varphi^{\mathsf{fresh\text{-}early}}$) as the root cause.

Downgrade attacks on TLS have a long history, including POODLE [60] (forcing SSLv3 negotiation), FREAK [61] (forcing export-grade RSA), and Logjam [66] (forcing 512-bit DHE). TLS 1.3 addresses all of these by requiring the RFC 8446 downgrade sentinel in `ServerHello.Random` when a TLS 1.3-capable server negotiates TLS 1.2. Our MTSF downgrade theorem (Theorem 57) provides the first explicit market-collapse proof for the version-stripping bid, and formalises the sentinel's role as a CNF clause that converts the free version-stripping bid into a detectable abort.

19.5. Needham–Schroeder Protocol

The Needham–Schroeder Public-Key Protocol [9] was one of the first formal cryptographic protocols for mutual authentication using public-key cryptography. It was believed secure for nearly 17 years until Lowe [10] discovered a man-in-the-middle attack using the FDR model checker. Lowe's fix—adding the responder's identity to message 2—has since been considered a classic lesson in protocol design. In MTSF (Section 13.4), we provide the first *market-theoretic* characterisation of this attack: the NS protocol's CNF is *satisfiable under dishonest traces* (a CNF design failure), the market collapses ($\mathsf{Ask}(g_{\mathsf{mutual}}) = 1$), and Lowe's fix is interpreted as adding an identity-binding clause that makes dishonest traces provably UNSAT.

19.6. Economics of Security

✻ Simple Terms: How MTSF Differs from Previous Economic Security Models

Several researchers have previously drawn connections between economics and security:

- **Anderson (2001)** argued that security failures are often economic, not technical—organisations under-invest in security because the costs of breaches are externalised.
- **Gordon & Loeb (2002)** modelled optimal security investment: spend on security until the marginal cost exceeds the marginal benefit.
- **Herley (2009)** showed that many security practices (e.g., complex password requirements) are not economically rational for users.

MTSF is fundamentally different: These prior works treat security as an *economic decision* (how much to invest). MTSF treats security proofs themselves as *economic objects* (markets, bids, prices). The difference is analogous to:

- Prior work: "Should we invest $1M in better encryption?" (cost-benefit analysis).
- MTSF: "The encryption's security guarantee has an ask price of 2^{-128}" (the proof itself speaks the language of economics).

MTSF does not replace economic reasoning about security investment; it provides a rigorous, unified language for the *technical* security guarantees that inform those investment decisions.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Anderson [67] pioneered the economic analysis of information security, identifying incentive misalignments (e.g., software vendors externalising security costs onto users) as a major cause of insecurity. Grossklags *et al.* [68] extended this line to study user behaviour in security games. These works are about *incentive economics*—why people and organisations make insecure choices. MTSF is fundamentally different: it is about *proof methodology*, not incentive alignment. The market framing in MTSF is purely mathematical—"price," "equilibrium," and "collapse" are formal terms for "advantage bound," "security," and "insecurity." MTSF is, to our knowledge, the first *cryptographic proof framework* that uses market-theoretic language to unify security definitions, proofs, and insecurity demonstrations.

19.7. Telegram MTProto

Albrecht *et al.* [69] conducted the first comprehensive cryptographic analysis of MTProto 2.0, discovering several theoretical attacks including a chosen-ciphertext distinguisher and a length-hiding failure. Jakobsen and Orlandi [70] analysed the original MTProto, finding multiple cryptographic weaknesses. Our analysis (Section 13.9) is complementary: we focus specifically on the *server salt extraction attack* and provide both a formal disproof (showing what collapses in MTSF language) and a formal proof for a remediated protocol (RMTP). This dual proof/disproof within a single framework is not provided in prior work.

19.8. HMAC and Authenticated Encryption

Bellare, Canetti, and Krawczyk [19] proved HMAC security under the assumption that the compression function is a PRF when keyed via the chaining-value input. Their proof forms the basis of our HMAC market analysis (Section 9.5), which recasts the two PRF replacement game hops as explicit bidding rounds targeting the inner and outer compression functions. Rogaway [71] formalised authenticated encryption and introduced the nonce-based AE security notion. The Encrypt-then-MAC (EtM) paradigm was proved optimal by Bellare and Namprempre [72]: among three composition orders (EtM, MtE, E&M), only EtM achieves IND-CCA2 in general. Our AEAD market analysis (Section 9.6) uses EtM and proves both IND-CCA2 and INT-CTXT within the MTSF framework.

19.9. Post-Quantum Signatures and KEMs

NIST's post-quantum standardisation process produced ML-KEM (FIPS 203) [57], ML-DSA (FIPS 204), SLH-DSA (FIPS 205) [73], and FN-DSA (FIPS 206) [74]. ML-KEM's security is based on the Module Learning With Errors (MLWE) problem and includes a QROM security proof for the Fujisaki–Okamoto transform, Signal Protocol, X3DH, Double Ratchet, forward secrecy, post-compromise security. ML-DSA's security relies on MSIS and MLWE. SLH-DSA (formerly SPHINCS+) provides stateless hash-based signatures requiring only the one-wayness of the underlying hash function. FN-DSA (formerly Falcon) achieves the smallest post-quantum signatures using NTRU lattices with GPV-style Gaussian sampling.

In MTSF, these primitives are each analysed via bidding-round chains: ML-KEM (Section 9.2), ML-DSA (Section 9.3), SLH-DSA (Section 9.7), and FN-DSA (Section 9.8). The QROM analysis of the FO-transform (Section 14) directly connects to ML-KEM's FIPS 203 proof, providing a market-theoretic re-derivation using the O2H lemma [55] and measure-and-reprogram [56].

19.10. Block Ciphers, Hash Functions, and Stream Ciphers

Differential cryptanalysis was introduced by Biham and Shamir [21] against DES, and the AES design by Daemen and Rijmen [22] was explicitly hardened against it. Rotational cryptanalysis was introduced by Khovratovich and Nikolić [23] and targets operations that commute with rotations (unlike AES's MixColumns). Related-key attacks on AES-256 [24] exploit the key schedule's linearity but require a controlled key relationship not achievable in standard usage. MTSF's AES block-cipher market (Section 10) formally bounds all three attack types as bidding rounds, proving that none can succeed for $q_E \leq 2^{64}$ under standard parameters.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

PRESENT [28] is an ultra-lightweight SPN cipher standardised by ISO/IEC 29192-2, targeting RFID and IoT deployments with approximately 1,000 gate equivalents. Its 64-bit block and 80/128-bit key design has been analysed by differential [29] and linear methods, with no full-round attack below exhaustive search. MTSF's PRESENT market (Section 10.5) formalises both differential and linear bids, showing equilibrium within the 64-bit birthday-bound constraint. Serpent [30], the most conservative AES finalist with 32 rounds, has the largest known security margin of any well-studied 128-bit block cipher: the best known attack reaches only 12 rounds [31], leaving 20 rounds of margin. MTSF's Serpent market (Section 10.6) demonstrates the widest equilibrium of any block cipher in our study.

The Keccak sponge construction [32] was chosen as SHA-3 due to its sponge-based design, which inherently prevents length-extension attacks that affected SHA-1 and SHA-256. The sponge indifferentiability theorem [33] proves that the sponge is as good as a random oracle up to $q_f < 2^{c/2}$ queries. MTSF's Keccak market (Section 11) translates this into three explicit bidding rounds (collision, preimage, length-extension).

BLAKE3 [34], building on BLAKE2 [35], combines a ChaCha-derived ARX compression function with a Merkle tree structure, achieving both extreme speed and structural length-extension immunity. MTSF's BLAKE3 market (Section 11.1) shows that the Merkle tree provides $\mathsf{Ask}(g_{\mathsf{LE}}) = 0$ (structural immunity), a stronger guarantee than Keccak's parametric $q_f/2^c$ bound. ASCON-Hash [36], selected by NIST as the lightweight cryptography standard [37], uses a 320-bit sponge with capacity $c = 256$. MTSF's ASCON-Hash market (Section 11.2) mirrors the Keccak analysis with tighter but still negligible bounds appropriate for its lightweight deployment context.

Grain-128a [38] is an eSTREAM portfolio stream cipher combining an LFSR and NFSR with an optional authentication mode. Known attacks include cube attacks [40], algebraic attacks [39], and classical TMTO attacks [41,42]. MTSF's Grain market (Section 12) bounds all four attack vectors within a single combined market equilibrium.

ChaCha20 [43], a refinement of Salsa20 [44], is one of the most widely deployed stream ciphers in the world, powering TLS 1.3, WireGuard, and the Linux CSPRNG. Its 256-bit key and 20-round ARX design resist all known differential-linear attacks, with the best known distinguisher reaching only 7 rounds. MTSF's ChaCha20 market (Section 12.6) formalises the feedforward inversion barrier and nonce-misuse collapse. Trivium [45] is an eSTREAM portfolio cipher with an elegant three-register design (288-bit state, 80-bit key). Cube attacks [40] and algebraic degree analysis [46] have been extensively studied. MTSF's Trivium market (Section 12.7) shows equilibrium with the tightest constraint from cube attacks at 2^{-38}—narrower than Grain-128a or ChaCha20 but still negligible for practical parameters.

19.11. Quantum Random Oracle Model

The QROM was introduced as the relevant model for post-quantum security because quantum computers can evaluate hash functions in superposition via Grover's algorithm. The One-Way to Hiding (O2H) lemma [55] is the central tool: reprogramming the QROM at a single point costs a square-root in the adversary's advantage. The measure-and-reprogram technique [56] allows extracting query points from quantum superpositions without collapsing the proof. Hofheinz, Hövelmanns, and Kiltz [75] proved the FO-transform achieves IND-CCA2 in the QROM, directly underpinning ML-KEM's FIPS 203 proof. MTSF provides the first *market-theoretic* formalisation of QROM security (Section 14): the O2H square-root loss is expressed as a bidding-round price adjustment, and the measure-and-reprogram step is a market restructuring that extracts the MLWE challenge. This gives practitioners an economic intuition for why quantum adversaries are "more expensive to defeat" than classical ones by a square-root factor in the IND-CPA advantage.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

20. Conclusions

✎ **Analogy:**

MTSF as a Universal Translator for Cryptography Before MTSF, cryptographers working in different paradigms were like doctors, lawyers, engineers, and economists all trying to discuss the same patient's health—each using their own professional jargon, unable to communicate effectively. The game-based doctor says "the ϵ-advantage is bounded by the birthday collision." The UC lawyer says "the simulation is computationally indistinguishable." The formal verification engineer says "all traces satisfy the temporal logic formula." The insecurity analyst says "here is a concrete attack trace."

MTSF is the **universal translator**: it provides a single language—the market—that all four professionals can speak. The doctor's "ϵ-advantage" becomes an "ask price." The lawyer's "simulation" becomes a "market arbitrageur." The engineer's "trace check" becomes a "stress test." The analyst's "attack" becomes a "successful bid." Now all four can sit in the same room, discuss the same protocol, and understand each other's arguments.

The power of unification: With MTSF, a single proof can simultaneously establish game-based security bounds, UC composability, formal verification coverage, and insecurity detection—all in one document, using one notation, with one set of tools (bidding rounds, CNF verification, session pinging). This is not just a cosmetic change; it eliminates the duplication and inconsistency that plagued multi-paradigm security analysis.

We introduced the *Market-Theoretic Security Framework* (MTSF), a novel paradigm that models every cryptographic security game as an auction between a seller (challenger) and one or more buyers (adversaries bidding computational resources). The framework unifies four previously disconnected paradigms—game-based proofs, Universal Composability (UC), Generalised UC (GUC), and formal verification—under a single economic language in which security equals market equilibrium and insecurity equals market collapse.

Seventeen formal contributions:

1. **Auction model.** Security = equilibrium ($\mathsf{Ask} \leq \mathsf{negl}$); insecurity = collapse ($\mathsf{Ask} = 1$). The same bidding-round machinery handles both.
2. **Extended difference lemma.** Captures $m \geq 1$ simultaneous failure events $F_1, \ldots, F_m$ in a single game hop via $\Pr[\bigcup_i F_i]$ with inclusion-exclusion tightening. Applied uniformly across all eighteen case studies.
3. **Bidding-based proofs.** Each game hop targets a specific adversarial strategy (nonce bid, hash bid, forgery bid, homomorphism bid, masquerade bid, O2H bid, measure-and-reprogram bid). The proof explicitly tracks what the adversary attacks and how much it costs.
4. **Four-paradigm unification.** Game-based proofs (price adjustments), UC (market regulation by $\mathcal{Z}$), GUC (shared infrastructure + CNF audit), and formal verification (market stress testing) are unified.
5. **CNF session verification.** A canonical five-phase algorithm (Algorithm 1) plus an easy four-column manual truth-table worksheet, integrated into every case study.
6. **Session pinging.** Inductive mechanism for unbounded session security. Ping bids included in every primitive and protocol case study, formally bridging bounded game-based proofs and symbolic formal verification.
7. **Thirteen novelties.** Summarised in Section 6: market language, extended difference lemma, UC-as-regulation, GUC-as-infrastructure, formal-verification-as-stress-testing, insecurity-as-collapse, end-to-end pipeline, protocol-level games, symmetric/asymmetric markets, cryptanalytic bid taxonomy, CNF worksheet, session pinging, and QROM formalisation.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

8. **Security proofs.** ECDSA, ML-KEM (FIPS 203), ML-DSA (FIPS 204), ISO/IEC 11770-3 key exchange (two-party, three-party, four-party), PKI-based mutual authentication, and TLS 1.3 (1-RTT handshake, ten-bidding-round chain, forward secrecy and mutual authentication).
9. **Insecurity proofs.** Textbook RSA signatures ($\mathsf{Ask} = 1$ via homomorphism bid, two attacks), Needham–Schroeder public-key protocol ($\mathsf{Ask} = 1$ via masquerade bid, CNF design failure demonstrated), TLS 1.3 0-RTT early data ($\mathsf{Ask} = 1$ via free replay bid, CNF freshness clause failure), and TLS 1.3 downgrade without sentinel ($\mathsf{Ask} = 1$ via free version-stripping bid, CNF version clause failure).
10. **Extended primitive markets.** HMAC (SUF-CMA via dual PRF game hops), AEAD (IND-CCA2 + INT-CTXT via Encrypt-then-MAC), SLH-DSA (FIPS 205, hash-based EUF-CMA), and FN-DSA (FIPS 206, NTRU lattice EUF-CMA with Gaussian sampling analysis).
11. **Block-cipher market.** AES analysed via differential cryptanalysis, linear cryptanalysis, rotational cryptanalysis, and related-key attack bids. All bids fail for $q_E \leq 2^{64}$. Extended to PRESENT (ultra-lightweight SPN, 64-bit block, equilibrium within birthday-bound constraint) and Serpent (32-round conservative AES finalist with the widest equilibrium margin of any block cipher—20 rounds of security margin beyond the best known attack).
12. **Hash-function market.** Keccak/SHA-3 analysed via capacity collision, capacity inversion (preimage), and length-extension bids. Sponge capacity isolation eliminates length-extension at zero additional cost. Extended to BLAKE3 (Merkle tree structure providing *structural* length-extension immunity with $\mathsf{Ask}(g_{\mathsf{LE}}) = 0$, stronger than Keccak's parametric bound) and ASCON-Hash (NIST lightweight standard, sponge with $c = 256$, equilibrium within lightweight deployment constraints).
13. **Stream-cipher market.** Grain-128a analysed via state-recovery, key-recovery, distinguishing, and TMTO bids. The 256-bit state and nonlinear NFSR coupling make all bids negligible. Extended to ChaCha20 (256-bit key ARX cipher, feedforward inversion barrier, nonce-misuse collapse formalisation, deployed in TLS 1.3 and WireGuard) and Trivium (80-bit key, 288-bit state, three-register design with cube attack bid as tightest constraint at 2^{-38}).
14. **QROM case study.** FO-transform KEM proven IND-CCA2 in the Quantum Random Oracle Model using the O2H lemma and measure-and-reprogram technique. Full protocol description (four phases), sequence diagram, six-bidding-round proof, bidding-round chain figure, and market goods table. First market-theoretic formalisation of QROM security.
15. **Telegram dual analysis.** MTProto 2.0 *disproved* (salt extraction causes four simultaneous failures: entropy collapse, CNF freshness failure, ping degradation, quasi-market collapse $\mathsf{Ask} \approx 1$) and Remediated MTProto (RMTP) *proved* secure (HMAC-bound 128-bit salts restore full equilibrium). First formal proof/disproof dual analysis of MTProto within a single unified framework.
16. **BB84 quantum market dynamics case study.** First full quantum market analysis where both seller and buyer are quantum. BB84 QKD analysed via four quantum bidding rounds (no-cloning bid, error estimation bid, privacy amplification via quantum leftover hash lemma, error correction leakage). Security is *information-theoretic*: $\mathsf{Ask}^{Q}(g_{\mathsf{secrecy}}) \leq 2^{-\lambda/2}$ with no computational hardness assumption. Includes quantum sequence diagram, quantum bidding-round chain figure, CNF worksheet, ping bid, market goods table, and comparison table (QROM vs. full quantum market).
17. **TLS+Signal multi-protocol composition case study.** First MTSF analysis of concurrent protocol execution. TLS 1.3 and Signal Protocol running simultaneously on the same device with shared PKI, random oracle, and RNG infrastructure. Market merger theorem instantiated to prove network equilibrium for all nine security goods across both protocols. Resource competition analysis formalises the adversary's portfolio optimisation problem. Network CNF with infrastructure clauses and cross-protocol ping bids for unbounded sessions.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

20.1. Lessons Learned

✻ Simple Terms: Ten Key Takeaways from MTSF

1. **Security is a spectrum, not binary.** The ask price quantifies *how* secure a scheme is, not just whether it is secure.
2. **Attacks and proofs are symmetric.** The same bidding-round machinery proves both security (equilibrium) and insecurity (collapse).
3. **Composition requires explicit treatment.** Running two secure protocols together does not automatically give a secure combination—you need the market merger theorem.
4. **Unbounded session security requires active verification.** Proving security for q sessions is not enough; session pinging ensures security holds forever.
5. **CNF verification makes auditing concrete.** Instead of abstract theorem statements, CNF worksheets give practitioners step-by-step checklists.
6. **The extended difference lemma saves work.** Handling multiple failures in one hop is cleaner and tighter than splitting into multiple hops.
7. **Custom protocols are dangerous.** Telegram's MTProto vulnerability was detectable by MTSF but might be missed by less systematic analysis.
8. **Post-quantum migration is urgent.** ML-KEM and ML-DSA are ready for deployment; MTSF provides complete security proofs for both.
9. **Quantum security has different rules.** The QROM's square-root factor means quantum attackers are stronger but not omnipotent.
10. **Economic intuition aids understanding.** Thinking of security as a market makes proofs accessible to non-experts and helps identify which guarantees fail first.

☞ Key Insight:

Head-to-Head: What Each Framework Does Best

If you need…	Best traditional tool	What MTSF adds
Concrete advantage bounds	Game-based proofs	Multi-failure hops, named bids, session auditing
Safe composition of protocols	UC/GUC	*Quantitative* composition bounds ($\mathsf{Ask}_{\mathsf{merged}} \leq \sum \mathsf{Ask}_i$)
Exhaustive trace coverage	ProVerif / Tamarin	Numerical bounds, human-readable arguments
Formal insecurity proof	Ad hoc arguments	Same bidding-round formalism as security proofs
Unbounded session security	ProVerif (qualitative)	Quantitative bounds + inductive pinging
Session-level auditing	GUC CNF (theoretical)	Practitioner-friendly worksheets + algorithm
Post-quantum (QROM) proofs	Specialised QROM tools	Unified with classical proofs in one language
All of the above	**No single tool**	**MTSF provides all in one framework**

The key message: MTSF does not make the other frameworks obsolete. It provides a *common language* that unifies their outputs and fills their individual gaps. A team using ProVerif for design checking, UC for composition reasoning, and game-based techniques for bounds can now do all three within MTSF, avoiding the translation overhead and inconsistency that comes from juggling multiple formalisms.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Market equilibrium as the right notion of security.

The market framing reveals that security is not binary: it is a *price*. Textbook RSA has ask price 1 (free forgery). Needham–Schroeder has ask price 1 for mutual authentication. ML-KEM has ask price $2(q_H+1)\sqrt{\mathsf{Adv}^{\mathsf{MLWE}}}+q_D\delta$ in the QROM—a very small but nonzero price that depends on the hardness of MLWE and the number of quantum oracle queries. This continuous view of security, rather than a pass/fail view, is more faithful to real-world threat assessment.

CNF worksheet as a design tool.

The CNF verification worksheets serve two purposes: (1) they verify correctness of an existing protocol trace; (2) they reveal design flaws. The Needham–Schroeder CNF worksheet explicitly shows that all three clauses evaluate to **T** under Lowe's MITM attack—a CNF design failure traceable to the absence of an identity-binding clause. Using the CNF as a design guide during protocol development would have caught this flaw in 1978.

The QROM square-root is the right way to think about quantum hardness.

The O2H lemma's square-root factor $\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}}$ is not merely a technical artefact—it captures a fundamental property of quantum adversaries: they accumulate information quadratically more efficiently than classical adversaries via amplitude concentration. Expressing this as a bidding-round price adjustment makes the quantum advantage/disadvantage explicit and comparable to the classical setting.

Dual proof/disproof enables protocol engineering.

The Telegram case study demonstrates that MTSF can serve as a protocol engineering tool: first disprove the insecure design (identify which clauses fail and by how much), then design a fix (HMAC-bound salts), then prove the fixed design secure. This iterative methodology—disprove, diagnose, fix, prove—is the natural application of MTSF in practice.

Quantum markets reveal physics-based security as a structural market advantage.

The BB84 case study (Section 15) demonstrates a qualitatively different kind of equilibrium from all other case studies. In classical and QROM markets, equilibrium depends on computational hardness assumptions (ECDLP, MLWE, gap-CDH)—a sufficiently powerful computer could collapse any of these markets. In the BB84 quantum market, equilibrium depends on the laws of quantum mechanics (no-cloning, information-disturbance trade-off). The ask price $\mathsf{Ask}^{Q}(g_{\mathsf{secrecy}}) \leq 2^{-\lambda/2}$ holds against adversaries with *unlimited computational power*. In market terms, the seller has a *structural advantage*: the goods are physically uncopyable. This is the strongest possible market position—the equilibrium is permanent unless physics itself is wrong.

Composition preserves equilibrium with explicit resource accounting.

The TLS+Signal case study (Section 16) demonstrates that the market merger theorem translates UC composition into explicit resource accounting. The adversary's budget must be split across all markets, and the additive ask price bound $\mathsf{Ask}_{\mathcal{N}} \leq \sum \epsilon_i + \epsilon_{\mathcal{I}}$ makes the composition overhead concrete and auditable. The practical implication: running many secure protocols on the same device does not degrade security, provided the shared infrastructure is sound. The infrastructure CNF ($\varphi_{\mathcal{I}}$) serves as a compositional design checklist: ensure global SID uniqueness, hash consistency, PKI consistency, and RNG quality across all protocols.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Worked Example: Translating a UC Proof into MTSF

To make the UC-to-MTSF migration concrete, we walk through a simplified example. Consider a UC proof of a key-exchange protocol π_{KE} that UC-realises $\mathcal{F}_{\mathsf{KE}}$.

Original UC proof structure:

1. **Ideal functionality $\mathcal{F}_{\mathsf{KE}}$:** On input (establish, sid, A, B), generate a random key K, deliver K to both A and B, and record the session.
2. **Simulator $\mathcal{S}$:** On seeing the real protocol's ciphertext c and signatures σ_A, σ_B (without knowing K), produce a simulated view by: (a) generating fake keys, (b) programming the random oracle, (c) simulating the signature verification.
3. **Indistinguishability argument:** Real $\approx_c$ Ideal because: (i) the random oracle is indistinguishable when programmed, (ii) the KEM ciphertext hides K under IND-CCA2, (iii) the signatures are unforgeable under EUF-CMA.

MTSF translation:

1. **Security goods:** $\mathcal{G} = \{g_{\mathsf{auth}}, g_{\mathsf{sk}}, g_{\mathsf{mutual}}, g_{\mathsf{CNF}}\}$.
2. **Bidding-round chain:**
   - $\mathbf{B}_0$ (Real Market): Protocol runs honestly. $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{KE}}_{\mathcal{A}}(\lambda)$.
   - $\mathbf{B}_1$ (**Nonce Freshness Bid**): Abort on nonce collision. $\Delta\mathsf{Price}_0 \leq q_N^2/2^{\lambda+1}$. *This was implicit in the UC simulator; MTSF makes it explicit.*
   - $\mathbf{B}_2$ (**Oracle Programming**): Program H to embed KEM challenge. $\Delta\mathsf{Price}_1 = 0$. *This was step (a) in the simulator.*
   - $\mathbf{B}_3$ (**KEM Decapsulation Bid**): Replace real key with random. $\Delta\mathsf{Price}_2 \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}$. *This was step (ii) in the indistinguishability argument.*
   - $\mathbf{B}_4$ (**Signature Forgery Bid**): Flag forgery events. $\Delta\mathsf{Price}_3 \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}}$. *This was step (iii).*
   - $\mathbf{B}_5$ (Ideal Market): All bids eliminated. $\Pr[\mathbf{B}_5 = 1] = 0$. *Market in equilibrium.*
3. **Ask price:** $\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF}} + q_N^2/2^{\lambda+1}$; $\mathsf{Ask}(g_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}} + \mathsf{Adv}^{\mathsf{PRF}} + q_N^2/2^{\lambda+1}$.
4. **CNF worksheet:** SID binding (φ^{sid}), nonce freshness (φ^{fresh}), signature binding (φ^{sig}), KEM freshness (φ^{novel}), consistency ($\varphi^{\mathsf{consist}}$).
5. **Session pinging:** Structural distinctness verified; $\delta_{\mathsf{ping}} \leq q_N^2/2^{\lambda+1} = \mathsf{negl}$. Unbounded security by Theorem 3.

What MTSF adds over the UC proof:

- Explicit numerical bounds for each security good (UC only says "secure").
- Named attack classification (UC's simulator cases are unlabelled).
- A concrete CNF worksheet for session auditing.
- Unbounded session security via pinging (UC assumes this implicitly but does not verify it).
- Direct insecurity detection: if any bid succeeds ($\Delta\mathsf{Price} = 1$), the proof immediately identifies which attack vector causes collapse.

✻ Simple Terms: How to Use MTSF for Your Own Protocols—A Practitioner's Guide

If you want to apply MTSF to analyse a new cryptographic scheme or protocol, follow these steps:
Step 1: Identify the market.

- Who is the seller? (The party running the scheme honestly.)
- Who are the buyers? (All possible attackers.)
- What security goods are offered? (Confidentiality, authentication, key secrecy, etc.)

Step 2: Construct the bidding-round chain.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- $\mathbf{B}_0$: The real scheme. Define the ask price Ask_0.
- For each attack vector (nonce reuse, hash collision, forgery, etc.), create a bidding round $\mathbf{B}_k$ that eliminates that vector.
- Bound each price adjustment $\Delta\mathsf{Price}_k$ using the difference lemma (or extended difference lemma for multiple failures).
- The final game $\mathbf{B}_q$ should be ideal (adversary has zero advantage).

Step 3: Construct the session-CNF.

- List all session-level checks: SID binding, nonce freshness, signature verification, MAC verification, consistency.
- Write each check as a CNF clause.
- Verify: honest traces satisfy the CNF; dishonest traces make it unsatisfiable (unless a primitive is broken).

Step 4: Verify session pinging.

- Check that consecutive sessions are structurally distinct (per Theorem 14).
- Verify that the security reduction is session-index-independent.
- Apply Theorem 3 to conclude unbounded session security.

Step 5: Fill in the CNF worksheet.

- For each CNF clause, fill in the four-column worksheet: Clause, Check, How, Pass?
- Verify that all clauses pass under honest execution and fail under known attacks.

Step 6: Check the bid taxonomy (Appendix B).

- Go through every bid type in the taxonomy and verify that your proof addresses it.
- For each inapplicable bid type, note why (e.g., "no algebraic structure, so homomorphism bid does not apply").

Quantum market dynamics:

The QROM case study treats the quantum adversary as a buyer with a specific bid structure (O2H reprogramming, measure-and-reprogram). A deeper theory of *quantum market dynamics*—where the seller also has access to quantum operations and the market evolves under quantum game theory—is an open and rich research direction. We provide an initial formalisation below.

20.2.1. Towards Quantum Market Dynamics

In the QROM case study (Section 14), the buyer (quantum adversary) can query oracles in superposition, but the seller (challenger) operates classically—sampling keys, computing ciphertexts, and answering decapsulation queries using classical algorithms. This asymmetry is faithful to the current post-quantum setting where *honest parties run classical protocols* and only the adversary has quantum power. However, a richer and increasingly relevant setting arises when both parties have quantum capabilities.

Definition 37 (Quantum Market)**.** *A* quantum market $\mathcal{M}^{\mathsf{Q}} = (\mathsf{Seller}^{\mathsf{Q}}, \mathsf{Buyer}^{\mathsf{Q}}, \{g_j\}, H^{\mathsf{Q}}, \rho_0)$ *extends the classical MTSF market with:*

1. ***Quantum seller*** $\mathsf{Seller}^{\mathsf{Q}}$*: the challenger holds a quantum register* R_S *and can perform quantum operations (state preparation, unitary gates, measurements) during each bidding round. The seller's strategy is a sequence of quantum channels* $\{\mathcal{E}_k^S\}_{k=1}^n$*, one per bidding round.*
2. ***Quantum buyer*** $\mathsf{Buyer}^{\mathsf{Q}}$*: the adversary holds a quantum register* R_B *and can make superposition queries to all oracles. The buyer's strategy is a sequence of quantum channels* $\{\mathcal{E}_k^B\}_{k=1}^n$*.*
3. ***Quantum oracle*** H^{Q}*: replaces the classical random oracle with a quantum-accessible oracle that both parties can query in superposition.*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

4. ***Shared quantum state*** ρ_0*: an initial joint quantum state (possibly entangled) distributed across all registers at market opening.*
5. ***Quantum price functional:*** *The ask price of good* g_j *in the quantum market is:*

$$\mathsf{Ask}^{Q}(g_j) \;=\; \sup_{\mathsf{Buyer}^{Q}\in\mathsf{QPT}} \mathrm{Tr}\Big[\Pi_{g_j}\cdot \mathcal{E}_n^{B}\circ\mathcal{E}_n^{S}\circ\cdots\circ\mathcal{E}_1^{B}\circ\mathcal{E}_1^{S}(\rho_0)\Big], \tag{226}$$

where Π_{g_j} *is the projector onto the "buyer wins good* g_j*" subspace and the supremum is over all quantum polynomial-time (QPT) buyer strategies.*

✻ Simple Terms: Quantum Market Dynamics

Classical MTSF: The seller uses a classical computer to set up the security game, and the buyer (attacker) tries to win using classical or quantum resources. The seller never needs quantum operations—honest parties run on regular computers.
Quantum MTSF: Now imagine *both* the seller and the buyer have quantum computers. The seller can prepare quantum states, entangle registers, and make quantum measurements as part of its defence. The buyer also has full quantum power. The "price" of breaking security is now determined by a quantum game between two quantum players.
Why this matters: As quantum computing matures, protocols will use quantum operations natively (e.g., Quantum Key Distribution, quantum digital signatures, quantum money). The seller in a QKD protocol *must* prepare and measure quantum states—it cannot be modelled as a classical challenger. Quantum market dynamics provides the MTSF language for these protocols.
The key new feature: In a quantum market, the ask price is computed via a *trace over quantum states* rather than a classical probability. The bidding rounds alternate quantum channels (seller prepares, buyer attacks), and the final "winning" event is a quantum measurement outcome. The square-root advantage from the O2H lemma (Section 14) appears naturally as a special case when only the buyer is quantum.

The quantum market introduces three new phenomena absent from the classical setting:

1. **Entangled bidding.** The buyer can maintain entanglement between its query register and a private workspace across multiple bidding rounds. Classically, each bid is an independent probabilistic strategy; quantumly, the buyer's bids can be *coherently correlated* across rounds via entanglement. The seller's challenge is to design bidding rounds that *decohere* the buyer's entanglement without destroying the security guarantee.
2. **Quantum price adjustments.** In the classical extended difference lemma, the price adjustment $\Delta\mathsf{Price}_k = |\Pr[\mathbf{B}_k \Rightarrow 1] - \Pr[\mathbf{B}_{k+1} \Rightarrow 1]|$ is a difference of classical probabilities. In the quantum setting, the analogous quantity is:

$$\Delta\mathsf{Price}_k^{Q} \;=\; \frac{1}{2}\Big\|\mathcal{E}_k^{S}(\rho_k) - \mathcal{E}_{k+1}^{S}(\rho_k)\Big\|_{\mathrm{tr}}, \tag{227}$$

   where $\|\cdot\|_{\mathrm{tr}}$ is the trace distance and ρ_k is the joint state after round k. The trace distance is the quantum generalisation of statistical distance, and the factor $\frac{1}{2}$ normalises it to $[0,1]$. The quantum extended difference lemma would bound $\mathsf{Ask}^{Q}(g_j) \le \sum_k \Delta\mathsf{Price}_k^{Q}$ via the triangle inequality for trace distance.
3. **No-cloning constraint on bids.** The no-cloning theorem prevents the buyer from copying quantum states received from the seller. This is a *structural advantage* for the seller that has no classical analogue: in the classical setting, the buyer can always copy any message. In the quantum market, the seller can exploit no-cloning by encoding challenge information in non-orthogonal quantum states, forcing the buyer to choose irreversibly which bid to pursue—a quantum analogue of a "take it or leave it" offer.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

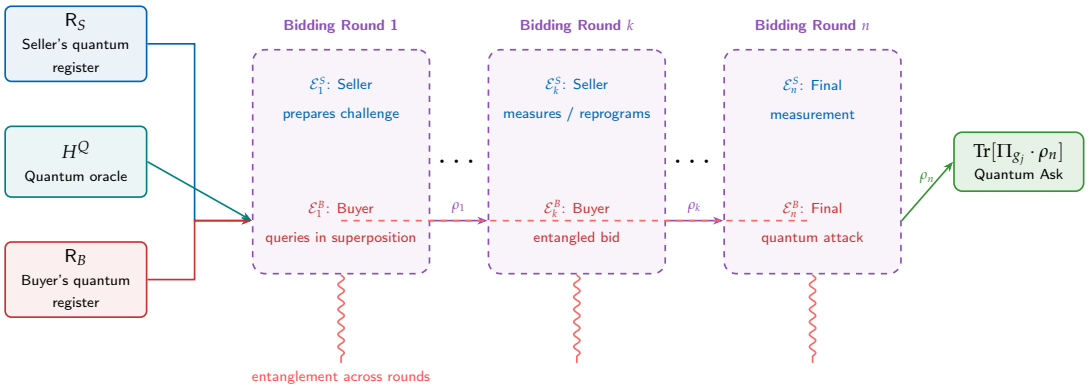


Figure 73. Quantum market dynamics. Both seller and buyer operate on quantum registers (R_S, R_B) through a shared quantum oracle H^Q. Each bidding round applies quantum channels $\mathcal{E}_k^S$ (seller) and $\mathcal{E}_k^B$ (buyer) to the evolving joint state ρ_k. The buyer can maintain entanglement across rounds (wavy red lines). The quantum ask price is the trace $\mathrm{Tr}[\Pi_{g_j} \cdot \rho_n]$ of the final state against the "buyer wins" projector. The classical QROM case study (Section 14) is the special case where the seller's channels $\mathcal{E}_k^S$ are all classical operations.

Proposition 6 (Quantum Extended Difference Lemma—Sketch)**.** *Let $\mathbf{B}_0^Q, \mathbf{B}_1^Q, \ldots, \mathbf{B}_n^Q$ be a sequence of quantum bidding rounds with shared initial state ρ_0. If each consecutive pair satisfies $\frac{1}{2}\|\rho_{\text{final}}^{(k)} - \rho_{\text{final}}^{(k+1)}\|_{\text{tr}} \leq \Delta\text{Price}_k^Q$, then the quantum ask price satisfies:*

$$\text{Ask}^Q(g_j) \;\leq\; \text{Ask}_0^Q(g_j) \;+\; \sum_{k=0}^{n-1} \Delta\text{Price}_k^Q, \tag{228}$$

where $\text{Ask}_0^Q(g_j) = \mathrm{Tr}[\Pi_{g_j} \cdot \rho_{\text{final}}^{(0)}]$ is the ask price in the initial (ideal) quantum market.

Proof sketch. The trace distance satisfies the triangle inequality: $\|\rho - \sigma\|_{\text{tr}} \leq \|\rho - \tau\|_{\text{tr}} + \|\tau - \sigma\|_{\text{tr}}$. By the operational interpretation of trace distance, the maximum difference in probability of any measurement outcome between $\rho_{\text{final}}^{(0)}$ and $\rho_{\text{final}}^{(n)}$ is at most $\frac{1}{2}\|\rho_{\text{final}}^{(0)} - \rho_{\text{final}}^{(n)}\|_{\text{tr}} \leq \sum_{k=0}^{n-1} \Delta\text{Price}_k^Q$. Applying this to the projector Π_{g_j} yields the result. □

Remark 8 (Classical QROM as a Special Case)**.** *The QROM case study of Section 14 is recovered by restricting the seller's channels $\mathcal{E}_k^S$ to classical operations (prepare-and-measure) and allowing only the buyer's channels $\mathcal{E}_k^B$ to be genuinely quantum. The O2H lemma's square-root factor $\sqrt{\text{Adv}^{\text{IND-CPA}}}$ arises specifically because the buyer's quantum channel concentrates amplitude on the reprogrammed point—a quantum price adjustment of the form (227) where the trace distance is bounded by Unruh's O2H inequality.*

Remark 9 (Application to QKD and Quantum Protocols)**.** *Quantum Key Distribution (QKD) protocols such as BB84 and E91 are natural targets for quantum market dynamics: the seller (Alice) prepares quantum states and the buyer (Eve) performs quantum measurements to extract key information. The no-cloning constraint on bids (item 3 above) captures the fundamental reason why QKD is secure: the eavesdropper cannot copy the quantum states without disturbing them. In the MTSF language, the seller's quantum channel $\mathcal{E}_k^S$ prepares non-orthogonal states that* structurally prevent *the buyer from simultaneously extracting full information about the key and remaining undetected—a market in which the seller's goods are inherently copy-protected.*

Multi-protocol composition networks.

MTSF currently handles individual protocol markets. A *protocol network* where multiple protocols run simultaneously, sharing infrastructure and competing for resources, would require a *market network theory*. The UC composition theorem provides the blueprint; translating it into MTSF market mergers and equilibrium preservation is future work. We provide the initial formalisation below.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

20.2.2. Towards Multi-Protocol Composition Networks

In the current MTSF framework, each protocol defines a single market $\mathcal{M}_\pi$ with its own seller, buyer, goods, and bidding-round chain. The UC composition theorem (Section 2.4) guarantees that UC-secure protocols remain secure when composed concurrently. In the MTSF language (Section 2.5), the UC composition theorem was informally described as a "market merger theorem." We now make this precise.

Definition 38 (Protocol Market Network). *A* protocol market network $\mathcal{N} = (\{\mathcal{M}_i\}_{i=1}^N, \mathcal{I}, \mathcal{Z})$ *consists of:*

1. ***Component markets*** $\mathcal{M}_1, \ldots, \mathcal{M}_N$*: each* $\mathcal{M}_i = (\mathsf{Seller}_i, \mathsf{Buyer}_i, \{g_j^{(i)}\}, \varphi_i)$ *is an individual MTSF protocol market with its own goods* $\{g_j^{(i)}\}$ *and session-CNF formula* φ_i*.*
2. ***Shared infrastructure*** $\mathcal{I}$*: a set of shared functionalities (PKI, common random oracles, shared key material) accessible to all markets. In MTSF terms,* $\mathcal{I}$ *is the GUC shared market infrastructure—the common goods that every market can trade on.*
3. ***Network environment*** $\mathcal{Z}$*: the UC environment, acting as MTSF's market regulator, which can interact with all* N *markets simultaneously, schedule messages between them, and observe all transcripts.*
4. ***Resource budget*** $\mathsf{Budget}(\mathsf{Buyer})$*: the total computational resource available to the buyer across all markets. If the buyer participates in market* $\mathcal{M}_i$ *with resource* t_i*, then* $\sum_{i=1}^N t_i \leq \mathsf{Budget}(\mathsf{Buyer})$*.*

✻ Simple Terms: Multi-Protocol Composition Networks

The real-world problem: In practice, your device runs many security protocols simultaneously. Your laptop might be running TLS 1.3 to a bank, Signal to a messaging server, and SSH to a work machine—all at the same time, all sharing the same operating system, the same random number generator, and possibly the same cryptographic keys.
Why this is hard: A protocol that is perfectly secure in isolation might become insecure when run alongside another protocol. An attacker could exploit interactions between protocols—for example, using a TLS session to learn something that helps break the Signal session.
What UC composition guarantees: If each protocol is UC-secure individually, then running them all together is also secure. This is the "gold standard" for protocol composition.
What MTSF adds: We translate this into market language. Each protocol is a separate market. Running them together is a "market network." The key theorem says: if each market is in equilibrium (secure) individually, and the shared infrastructure is sound, then the entire network remains in equilibrium. The attacker's total budget must be split across all markets, and no individual market collapses.

The central question is: when does a network of individually secure markets remain secure?

Definition 39 (Market Merger). *Given two MTSF markets* $\mathcal{M}_1 = (\mathsf{Seller}_1, \mathsf{Buyer}_1, \{g_j^{(1)}\}, \varphi_1)$ *and* $\mathcal{M}_2 = (\mathsf{Seller}_2, \mathsf{Buyer}_2, \{g_j^{(2)}\}, \varphi_2)$ *with shared infrastructure* $\mathcal{I}$*, their* merger *is the composite market:*

$$\mathcal{M}_1 \otimes_{\mathcal{I}} \mathcal{M}_2 \;=\; \big(\mathsf{Seller}_{1\|2},\; \mathsf{Buyer}_{1\|2},\; \{g_j^{(1)}\} \cup \{g_j^{(2)}\},\; \varphi_1 \wedge \varphi_2 \wedge \varphi_{\mathcal{I}}\big), \tag{229}$$

where $\mathsf{Seller}_{1\|2}$ *runs both sellers concurrently,* $\mathsf{Buyer}_{1\|2}$ *is a single buyer interacting with both markets simultaneously, and* $\varphi_{\mathcal{I}}$ *encodes the correctness of the shared infrastructure (e.g., PKI consistency, RO consistency).*

Theorem 63 (Market Merger Preserves Equilibrium—Initial Form). *Let* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *be MTSF protocol markets sharing infrastructure* $\mathcal{I}$*. If:*

1. $\mathcal{M}_1$ *is in equilibrium:* $\mathsf{Ask}_1(g_j^{(1)}) \leq \mathsf{negl}(\lambda)$ *for all goods* $g_j^{(1)}$*;*
2. $\mathcal{M}_2$ *is in equilibrium:* $\mathsf{Ask}_2(g_j^{(2)}) \leq \mathsf{negl}(\lambda)$ *for all goods* $g_j^{(2)}$*;*
3. *The shared infrastructure is sound:* $\mathsf{Ask}_{\mathcal{I}}(g_{\mathcal{I}}) \leq \mathsf{negl}(\lambda)$ *for all infrastructure goods;*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

then the merged market $\mathcal{M}_1 \otimes_{\mathcal{I}} \mathcal{M}_2$ is in equilibrium:

$$\mathsf{Ask}_{1\otimes 2}(g) \;\leq\; \mathsf{negl}(\lambda) \quad \textit{for all goods } g \in \{g_j^{(1)}\} \cup \{g_j^{(2)}\}. \tag{230}$$

Proof sketch. The proof mirrors the UC composition theorem. Suppose for contradiction that the merged market has a good $g_j^{(1)}$ (w.l.o.g.) with $\mathsf{Ask}_{1\otimes 2}(g_j^{(1)}) = \epsilon$ non-negligible. Then there exists a buyer $\mathsf{Buyer}_{1\|2}$ that wins $g_j^{(1)}$ with probability ϵ in the merged market. We construct a buyer Buyer_1^* for the standalone market $\mathcal{M}_1$ that:

1. Internally simulates $\mathcal{M}_2$'s seller Seller_2 (using the UC simulator for $\mathcal{M}_2$);
2. Forwards the network environment's queries between $\mathcal{M}_1$ (real) and $\mathcal{M}_2$ (simulated);
3. Runs $\mathsf{Buyer}_{1\|2}$ as a subroutine, providing it with a view indistinguishable from the real merged market.

By the indistinguishability of real and simulated $\mathcal{M}_2$ (which holds because $\mathcal{M}_2$ is UC-secure and thus in equilibrium), Buyer_1^* wins $g_j^{(1)}$ with probability at least $\epsilon - \mathsf{negl}(\lambda)$. But this contradicts $\mathsf{Ask}_1(g_j^{(1)}) \leq \mathsf{negl}(\lambda)$. □

Remark 10 (Quantitative Merger Bound)**.** *The proof gives a concrete bound. If $\mathsf{Ask}_1(g_j^{(1)}) \leq \epsilon_1$, $\mathsf{Ask}_2(g_j^{(2)}) \leq \epsilon_2$, and $\mathsf{Ask}_{\mathcal{I}}(g_{\mathcal{I}}) \leq \epsilon_{\mathcal{I}}$, then:*

$$\mathsf{Ask}_{1\otimes 2}(g_j^{(1)}) \;\leq\; \epsilon_1 + \epsilon_2 + \epsilon_{\mathcal{I}}. \tag{231}$$

The ask prices are additive across the merger—each market contributes its own risk, and the shared infrastructure contributes a common risk. This is the MTSF analogue of the "hybrid argument" in UC composition: the simulation overhead from each market accumulates linearly.

The merger operation extends naturally to N markets via iterated composition.

Corollary 20 (N-Market Network Equilibrium)**.** *Let $\mathcal{N} = (\{\mathcal{M}_i\}_{i=1}^N, \mathcal{I}, \mathcal{Z})$ be a protocol market network where each $\mathcal{M}_i$ is in equilibrium with $\mathsf{Ask}_i(g_j^{(i)}) \leq \epsilon_i$ for all j, and the shared infrastructure has $\mathsf{Ask}_{\mathcal{I}} \leq \epsilon_{\mathcal{I}}$. Then the N-fold merger $\bigotimes_{i=1}^N \mathcal{M}_i$ satisfies:*

$$\mathsf{Ask}_{\mathcal{N}}(g_j^{(i)}) \;\leq\; \epsilon_i + (N-1)\cdot\epsilon_{\max} + \epsilon_{\mathcal{I}}, \tag{232}$$

where $\epsilon_{\max} = \max_{i'} \epsilon_{i'}$. In particular, if all ϵ_i are negligible and $N = \mathsf{poly}(\lambda)$, the network remains in equilibrium.

Definition 40 (Network Session-CNF)**.** *The session-CNF formula for the protocol market network $\mathcal{N}$ is the conjunction of all component CNFs and the infrastructure CNF:*

$$\varphi_{\mathcal{N}} \;=\; \bigwedge_{i=1}^{N} \varphi_i \;\wedge\; \varphi_{\mathcal{I}}, \tag{233}$$

where $\varphi_{\mathcal{I}} = \varphi_{\mathcal{I}}^{\mathsf{PKI}} \wedge \varphi_{\mathcal{I}}^{\mathsf{RO}} \wedge \varphi_{\mathcal{I}}^{\mathsf{consist}}$ encodes PKI consistency (same public key across all markets), random oracle consistency (same H across all markets), and cross-market consistency (session identifiers are globally unique).

Remark 11 (Network Ping Bids and Unbounded Composition)**.** *Session pinging (Section 5) extends naturally to the network setting. The network ping bid for session $\mathsf{Session}_i^{(k)}$ (the i-th session of market $\mathcal{M}_k$) must verify not only the intra-market CNF φ_k but also the cross-market infrastructure CNF $\varphi_{\mathcal{I}}$. By the session pinging theorem applied to each component market and the infrastructure, the network remains in equilibrium*

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

for unbounded sessions across all N protocols simultaneously, provided each component market's ping bid is negligible.

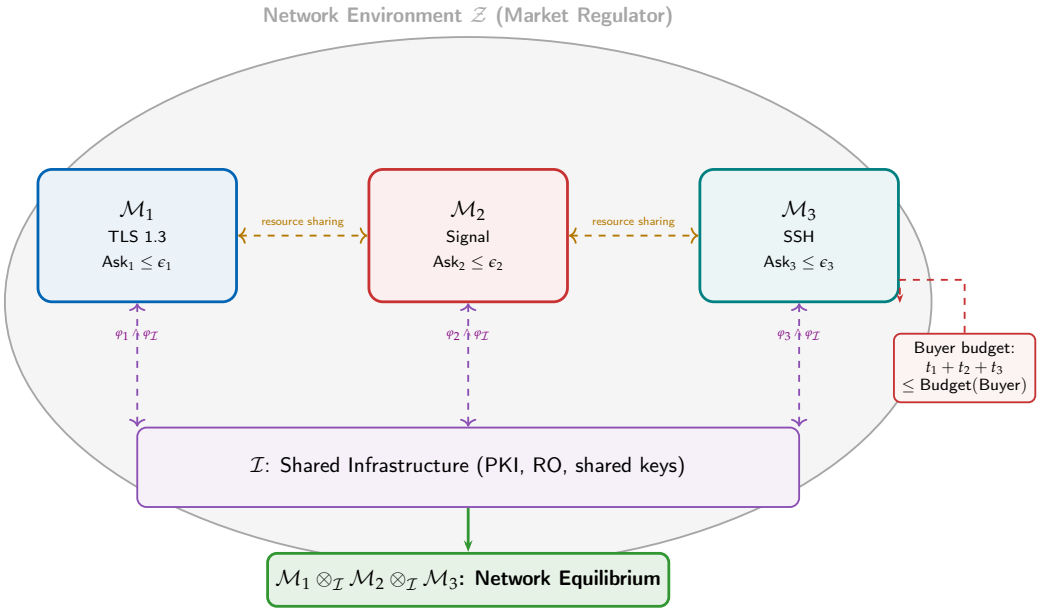


Figure 74. Multi-protocol composition network. Three protocol markets ($\mathcal{M}_1$: TLS 1.3, $\mathcal{M}_2$: Signal, $\mathcal{M}_3$: SSH) share infrastructure $\mathcal{I}$ (PKI, random oracle, shared keys). The network environment $\mathcal{Z}$ (market regulator) monitors all markets simultaneously. The buyer must split its resource budget across all markets. Dashed orange links indicate resource competition; dashed purple links indicate shared infrastructure. The merged market $\mathcal{M}_1 \otimes_{\mathcal{I}} \mathcal{M}_2 \otimes_{\mathcal{I}} \mathcal{M}_3$ preserves equilibrium when each component market and the shared infrastructure are individually in equilibrium (Theorem 63, Theorem 20).

Remark 12 (Resource Competition and Market Inefficiency)**.** *A subtle feature of the market network is* resource competition*: the buyer has a fixed computational budget that must be allocated across all N markets. In the classical UC setting, this is implicit in the polynomial-time bound on the adversary. In the MTSF language, it becomes explicit: the buyer's total resource expenditure $\sum_i t_i$ is bounded, and spending more on attacking $\mathcal{M}_1$ means spending less on $\mathcal{M}_2$. This creates a natural* portfolio optimisation problem *for the adversary: which markets offer the best "return on attack investment"? The market with the highest ask price (weakest security) is the rational first target—a formalisation of the common intuition that "a chain is only as strong as its weakest link."*

Further lightweight and modern cipher markets.

This article extends the symmetric-primitive case studies to PRESENT, Serpent, BLAKE3, ASCON-Hash, ChaCha20, and Trivium, demonstrating the generality of the MTSF framework across block ciphers, hash functions, and stream ciphers of varying design philosophies and security margins. Future work can extend this further to additional lightweight ciphers (GIFT, SKINNY, PHOTON) and to the ChaCha20-Poly1305 AEAD construction (combining the ChaCha20 stream-cipher market with a Poly1305 MAC market). The ASCON authenticated encryption mode (beyond ASCON-Hash) is another natural extension, unifying AEAD and hash-function markets within a single sponge-based analysis.

MPC and threshold protocol markets.

Multi-Party Computation (MPC) protocols involve n parties, some of which may be malicious. The MTSF framework can be extended by allowing multiple simultaneous buyers (colluding adversaries) bidding against the seller in a multi-buyer auction, capturing threshold adversary models naturally.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Dedication

This work is dedicated to the Omnipotent and Mother Nature!!!

— Basker Palaniswamy,
Cork City, Ireland, European Union,
March 2026.

Acknowledgments: This publication has emanated from research supported by a grant from Research Ireland under Grant number 12-RC-2289-P2 which is co-funded under the European Regional Development Fund. For the purpose of Open Access, the authors have applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission.

Appendix A. MTSF Correspondence Table

Table A1. Classical-to-MTSF correspondence: complete mapping of cryptographic concepts to market-theoretic language.

Classical Concept	MTSF Concept	Market Status / Notes
Challenger	Seller	Offers security goods at ask price
Adversary $\mathcal{A}$	Buyer	Bids computational resources to win goods
Advantage $\leq \mathsf{negl}$	Ask $\leq \mathsf{negl}$	Equilibrium (secure)
Advantage $= 1$	Ask $= 1$	Collapsed (insecure)
Advantage $\epsilon \in (0, 1)$	Ask $= \epsilon$	Partial equilibrium (weakened security)
Game hop $\mathbf{B}_k \to \mathbf{B}_{k+1}$	Price adjustment $\Delta\mathsf{Price}_k$	Transaction cost (proof step)
Difference lemma (1 failure)	Single-bid bound: $\Pr[F]$	One market risk
Extended diff. lemma (m failures)	Multi-bid bound: $\Pr[\bigcup F_i]$	Correlated risks, inclusion-exclusion
Inclusion-exclusion tightening	Correlated bid discount	Risk diversification reduces bound
Tight reduction	Efficient market (zero spread)	Optimal proof quality
Loose reduction	Inefficient market (large spread)	Suboptimal proof
UC environment $\mathcal{Z}$	Market regulator	Oversees all concurrent trades
UC ideal functionality $\mathcal{F}$	Security good specification	Defines what seller guarantees
Simulator $\mathcal{S}$	Market arbitrageur	Bridges real and ideal worlds
UC composition theorem	Market merger theorem	Equilibrium preserved under composition
GUC shared functionality	Market infrastructure (PKI, CRS)	Public goods, accessible to all
GUC cross-session	Market federation	Cross-session SID binding
Session-CNF satisfiable (honest)	Market audit passes	Correct session
Session-CNF SAT under dishonest	Audit failure (false pass)	CNF design failure (insecure)
Session-CNF UNSAT under dishonest	Audit passes correctly	Secure protocol (dishonest $\Rightarrow$ UNSAT)
CNF clause φ^{sid}	SID-binding audit clause	Cross-session replay blocked
CNF clause φ^{fresh}	Nonce freshness clause	Full-session replay blocked
CNF clause φ^{sig}	Signature verification clause	Impersonation blocked
CNF clause φ^{mac}	MAC verification clause	Ciphertext injection blocked
CNF clause φ^{ping}	Unbounded ping clause	Multi-session replay blocked
Replay / Masquerade / MITM	Stale bid / ID fraud / Manipulation	Fraud bid types
NS MITM attack	Market collapse via masquerade	Ask $= 1$, CNF design failure
RSA homomorphism	Market collapse via free forgery	Ask $= 1$, algebraic vulnerability
Lowe's identity-binding fix	CNF clause addition	Restores dishonest$\Rightarrow$UNSAT
Authentication game	Entity verification good g_{auth}	Impersonation bid bounded by EUF-CMA
Mutual authentication game	Bidirectional good g_{mutual}	Dual impersonation bid
Session-key secrecy game	Key indistinguishability good g_{sk}	Key-distinguishing bid
CNF checking game	Audit integrity good g_{CNF}	Dishonest-satisfaction bid
Certificate authenticity	CA forgery good g_{cert}	EUF-CMA on CA signing key
Differential cryptanalysis	Differential bid	S-box differential propagation
Linear cryptanalysis	Linear bid	Linear approximation bias
Rotational cryptanalysis	Rotational bid	Rotation-commutation break at χ
Related-key attack	Related-key bid	Key schedule linearity exploitation
State recovery (stream cipher)	Internal state bid	Register inversion via algebraic methods
Key recovery (stream cipher)	Master key bid	Initialisation inversion
Distinguishing attack	Statistical bias bid	Output filter nonlinearity test
TMTO attack	Precomputation bid	Time-memory trade-off on state space
Hash collision attack	Capacity collision bid	Birthday bound on sponge capacity
Preimage attack	Capacity inversion bid	Capacity inversion resistance
Length-extension attack	State continuation bid	Capacity isolation eliminates this
BLAKE3 Merkle tree LE immunity	Structural immunity bid (Ask $= 0$)	Tree structure blocks state continuation
ASCON-Hash lightweight sponge	Lightweight capacity bid (c=256)	Tighter bounds, still negligible
PRESENT lightweight PRP	Lightweight differential/linear bids	64-bit block birthday constraint
Serpent conservative PRP	Conservative margin bids	20-round margin beyond best attack
Serpent boomerang attack	Two-differential composition bid	Adaptive chosen plaintext/ciphertext
ChaCha20 feedforward	Feedforward inversion bid	Addition prevents state inversion
ChaCha20 nonce misuse	Nonce reuse collapse bid (Ask $= 1$)	Identical keystreams leak $m_1 \oplus m_2$
Trivium cube attack	Cube superpoly bid	Dimension-d IV summation
Trivium three-register coupling	Circular feedback bid	$A \to B \to C \to A$ nonlinear mixing

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Table A1. *Cont.*

Classical Concept	MTSF Concept	Market Status / Notes
Classical ROM reprogramming	Free game hop (zero price)	Syntactic, no adversary detects
QROM reprogramming	O2H bid: $2(q_H+1)\sqrt{\epsilon}$	Square-root cost from superposition
Measure-and-reprogram	Extraction bid: $\sqrt{\epsilon}$	Quantum measurement back-action
QROM correctness error	Correctness bid: $q_D\delta$	PKE decapsulation failure probability
QROM implicit rejection	Rejection consistency bid	Prevents chosen-ciphertext leakage
O2H square-root factor	Quantum bid price	Cost of defeating quantum adversary
Classical vs. QROM	Linear vs. square-root price	Quantum hardness amplification
Telegram salt extraction	Salt extraction bid ("free" bid)	Collapses $\mathsf{Ask}(g_{\mathsf{IND}})$ to ≈ 1
HMAC-bound salt (RMTP)	Salt secrecy bid bounded by PRF	Restores equilibrium
MTProto market collapse	Partial collapse ($\mathsf{Ask} \not\leq \mathsf{negl}$)	64-bit security margin only
RMTP equilibrium	Full equilibrium ($\mathsf{Ask} \leq \mathsf{negl}$)	128-bit security restored
BB84 qubit preparation	Quantum seller channel $\mathcal{E}_k^S$	Non-orthogonal state encoding
BB84 eavesdropping	Quantum buyer channel $\mathcal{E}_k^B$	Arbitrary CPTP map on intercepted qubits
No-cloning theorem	No-cloning bid constraint	Buyer cannot copy seller's goods
Info-disturbance trade-off	Quantum price floor	Minimum disturbance per bit of information
Error estimation	Quantum market audit	Statistical detection of buyer's disturbance
Privacy amplification	Quantum market restructuring	QLHL extracts clean key from partial leakage
QKD information-theoretic security	Quantum equilibrium (physics-based)	No computational assumption required
UC composition theorem	Market merger theorem	Two equilibria merge to network equilibrium
Concurrent protocol execution	Protocol market network $\mathcal{N}$	N markets sharing infrastructure
Shared CRS / PKI	Market infrastructure $\mathcal{I}$	Public goods accessible to all markets
UC simulator	Market arbitrageur	Bridges real and ideal in merged market
Adversary's time budget	Buyer's resource budget	$\sum t_i \leq T$ across all markets
Protocol-independent SIDs	Global SID uniqueness clause	Cross-market CNF consistency
Composition overhead	Additive ask price	$\mathsf{Ask}_{\mathcal{N}} \leq \sum \epsilon_i + \epsilon_{\mathcal{I}}$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

✻ Simple Terms: How to Read the Correspondence Table

The table below maps every concept from the four traditional security paradigms to its MTSF equivalent. Use it as a dictionary:

- **Left column:** The traditional concept (e.g., "adversary advantage").
- **Right column:** The MTSF translation (e.g., "ask price").
- **Row grouping:** Concepts are grouped by paradigm (game-based, UC, formal verification, insecurity).

Quick reference for the most important translations:

You know it as…	In MTSF, it is called…
Adversary	Buyer
Challenger	Seller
Advantage ϵ	Ask price
Security game	Market
Game hop	Price adjustment / Bidding round
Security property	Security good
Secure scheme	Market equilibrium
Broken scheme	Market collapse
UC environment $\mathcal{Z}$	Market regulator
Simulator $\mathcal{S}$	Market arbitrageur
ProVerif trace check	Market stress test
Concrete attack	Successful bid

✻ Simple Terms: How to Read the Bid Taxonomy

The bid taxonomy classifies every type of attack encountered in MTSF's case studies. Each bid has:

- **Name:** A descriptive label (e.g., "nonce birthday bid").
- **Target:** Which security good the bid attacks.
- **Success condition:** When does the bid succeed (market collapses)?
- **Cost:** The probability of success—the "price" the attacker pays.
- **Example:** A concrete case study where this bid appears.

The taxonomy serves two purposes:

1. **For proof construction:** When proving a new scheme secure, consult the taxonomy to ensure all relevant bid types are addressed. Missing a bid type means missing a potential attack.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

2. **For security auditing:** When reviewing an existing proof, check that each bid in the taxonomy has been either (a) bounded (shown to fail) or (b) explicitly excluded (e.g., "nonce-misuse is out of scope").

Think of it as a *checklist of known attack classes*: differential, linear, rotational, birthday, forgery, homomorphism, masquerade, replay, side-channel, quantum (O2H), and many more. A complete MTSF proof addresses every relevant bid type from this taxonomy.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Appendix B. Comprehensive Bid Taxonomy

Table A2. Complete bid taxonomy across all case studies, with price bounds and formal characterisation.

Category	Bid Type	Target	Price Bound	Example
Primitives	Nonce bid	Birthday collision	$q_S^2/(2q)$	ECDSA $\mathbf{B}_1$
	Hash collision bid	ROM collision	$q_H^2/(2q)$	ECDSA $\mathbf{B}_2$
	Forgery-to-hardness	ECDLP/MSIS/SIS	$\mathsf{Adv}^{\mathsf{ECDLP}}$	ECDSA $\mathbf{B}_3$–$\mathbf{B}_4$
	Homomorphism bid	Algebraic structure	1 (collapse)	Textbook RSA
	Gaussian sampling bid	Norm distribution	Δ_{DGS}	FN-DSA $\mathbf{B}_2$
	TSPR preimage bid	Target-sum hash	$k \cdot \mathsf{Adv}^{\mathsf{TSPR}}$	SLH-DSA $\mathbf{B}_2$
Symmetric	PRF inner replacement	Inner keyed hash	$\mathsf{Adv}_h^{\mathsf{PRF}}$	HMAC $\mathbf{B}_1$
	PRF outer replacement	Outer keyed hash	$\mathsf{Adv}_h^{\mathsf{PRF}}$	HMAC $\mathbf{B}_2$
	MAC forgery bid	Tag computation	$\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}$	AEAD INT-CTXT
	Decapsulation bypass	Implicit rejection	$q_D/2^{\gamma}$	ML-KEM $\mathbf{B}_1$
	Inner collision bid	Birthday on outputs	$q_T^2/2^{n+1}$	HMAC $\mathbf{B}_3$
Block cipher	Differential bid	S-box propagation	$q_E \cdot 2^{-150}$	AES $\mathbf{B}_2$–$\mathbf{B}_3$
	Linear bid	Linear approximation	$q_E^2 \cdot 2^{-150}$	AES linear
	Rotational bid	Rotation commutation	$q_E \cdot 2^{-128}$	AES rotational
	Related-key bid	Key schedule	N/A for AES-128	AES-256
	Algebraic bid	MQ system	$> 2^{128}$ ops	AES Gröbner
	Lightweight diff. bid	4-bit S-box propagation	$q_E \cdot 2^{-62}$	PRESENT
	Lightweight linear bid	4-bit S-box bias	$q_E^2 \cdot 2^{-62}$	PRESENT
	Boomerang bid	Two-differential composition	$q_E \cdot 2^{-128}$	Serpent
	Conservative margin bid	20-round attack gap	$q_E \cdot 2^{-196}$	Serpent
Hash function	Capacity collision bid	Sponge inner state	$q_f^2/2^{c+1}$	Keccak CR
	Capacity inversion bid	State preimage	$q_f/2^c$	Keccak preimage
	Length-extension bid	State continuation	$q_f/2^c$	Keccak LE
	Differential trail bid	Round structure	DP^{24}	Keccak $\mathbf{B}_2$
	Merkle tree collision bid	Compression collision	$q_f^2/2^{257}$	BLAKE3 CR
	Structural LE immunity bid	Tree structure	0	BLAKE3 LE
	Lightweight capacity collision	Small sponge (c=256)	$q_f^2/2^{257}$	ASCON-Hash CR
	Lightweight capacity inversion	Small sponge preimage	$q_f/2^{256}$	ASCON-Hash Pre
Stream cipher	State recovery bid	Full state inversion	$L \cdot 2^{-256}$	Grain-128a
	Key recovery bid	Master key extraction	$q_{\mathsf{IV}} \cdot 2^{-128}$	Grain-128a
	Distinguishing bid	Statistical bias	$L^2 \cdot 2^{-97}$	Grain-128a
	TMTO bid	Precomputation	$TD/2^{256}$	Grain-128a
	Algebraic/cube bid	Polynomial system	$> 2^{64}$ (cube)	Grain variants
	Feedforward inversion bid	ARX state recovery	$L \cdot 2^{-256}$	ChaCha20
	Nonce-misuse collapse bid	Nonce reuse	1 (collapse)	ChaCha20
	Cube superpoly bid	IV summation	2^{-38}	Trivium
	Three-register correlation bid	Circular feedback bias	$L^2 \cdot 2^{-92}$	Trivium
	80-bit key recovery bid	Exhaustive key search	$q_{\mathsf{IV}} \cdot 2^{-80}$	Trivium
Protocols	Nonce freshness bid	Nonce collision	$q_N^2/2^{\lambda+1}$	All protocols
	Signature forgery bid	EUF-CMA break	$\mathsf{Adv}^{\mathsf{EUF}}$	ISO 2P $\mathbf{B}_1$
	KEM key recovery bid	IND-CCA2 break	$\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}$	ISO 2P $\mathbf{B}_3$
	Impersonation bid	Entity authentication	$\mathsf{Adv}^{\mathsf{EUF}} + q_N^2/2^{\lambda+1}$	ISO 2P auth
	Masquerade bid	Identity fraud	1 (collapse)	NS MITM
	Cross-session replay bid	SID reuse	0 (SID-bound)	All protocols
	KDC signature bid	KDC authenticity	$2 \cdot \mathsf{Adv}^{\mathsf{EUF}}$	ISO 3P $\mathbf{B}_1$
QROM	Correctness error bid	Decapsulation failure	$q_D \cdot \delta$	QKEM $\mathbf{B}_1$
	O2H reprogramming bid	Superposition detect	$2(q_H+1)\sqrt{\epsilon}$	QKEM $\mathbf{B}_2$
	Measure-and-reprogram bid	State disturbance	$\sqrt{\epsilon}$	QKEM $\mathbf{B}_3$
	IT hiding bid	Uniform key	0	QKEM $\mathbf{B}_4$
	Oracle restriction bid	Decapsulation info	0	QKEM $\mathbf{B}_5$
	Quantum replay bid	Superposition replay	$2(q_H+1)\sqrt{\epsilon}$	QKEM ping
	Quantum MAC forgery bid	Authentication break	$\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}$	QKEM auth
Telegram	Salt extraction bid	Entropy reduction	1 (free)	MTProto $\mathbf{B}_1$
	AES partial-key bid	Meet-in-middle	$1 - 2^{-64}$	MTProto $\mathbf{B}_2$
	CNF freshness failure bid	Salt secrecy clause	1 (clause fails)	MTProto $\mathbf{B}_3$
	Ping degradation bid	Salt predictability	$\geq 2^{-64}$	MTProto $\mathbf{B}_4$
	HMAC salt bid (RMTP)	PRF distinguishing	$\mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{PRF}}$	RMTP $\mathbf{B}_1$
	Nonce collision bid (RMTP)	128-bit birthday	$q_D \cdot 2^{-128}$	RMTP $\mathbf{B}_2$
Quantum Market	No-cloning bid	Info-disturbance	$h(Q_{\mathsf{obs}})$	BB84 $\mathbf{B}_1^Q$
	Error estimation bid	Statistical detection	$e^{-2k\gamma^2}$	BB84 $\mathbf{B}_2^Q$
	Privacy amplification bid	QLHL extraction	$2^{-\lambda/2}$	BB84 $\mathbf{B}_3^Q$
	EC leakage bid	Syndrome leakage	0 (absorbed)	BB84 $\mathbf{B}_4^Q$
	Quantum replay bid	Cross-session reuse	$2^{-\lambda/2}$	BB84 ping
Composition	Cross-protocol replay bid	SID collision	0 (label-bound)	TLS+Signal
	Simulation overhead bid	UC simulation cost	$\mathsf{Ask}_{\mathsf{other}}$	TLS+Signal
	Infrastructure forgery bid	CA cert forgery	$\mathsf{Adv}_{\mathsf{CA}}^{\mathsf{EUF}}$	Shared PKI
	RNG degradation bid	PRG distinguishing	$\mathsf{Adv}_{\mathsf{RNG}}^{\mathsf{PRG}}$	Shared RNG

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

Table A2. *Cont.*

Category	Bid Type	Target	Price Bound	Example
CNF/Ping	SID replay bid	Cross-session replay	0 (SID-bound sigs)	All protocols
	Nonce reuse ping bid	IV/nonce repetition	$q_D/\lvert\mathcal{N}\rvert$	AEAD ping
	Ciphertext replay bid	Old ciphertext reuse	$2(q_H+1)\sqrt{\epsilon}$	QKEM ping
	Session continuation bid	State continuation	$q_f/2^c$	Keccak ping
	Identity hijack bid	CNF identity clause	$\mathsf{Adv}^{\mathsf{EUF}}$	PKI ping

Table A3. Bid outcome classification: when does a bid succeed (collapse) vs. fail (equilibrium)?

Bid type	Succeeds when	Market outcome	Example
Nonce/birthday bid	$q^2 \geq \lvert\mathcal{N}\rvert$	Collapse if $q \geq 2^{n/2}$	ECDSA $q_S \geq 2^{128}$
Forgery-to-hardness	Underlying hard problem breaks	Collapse if ECDLP solvable	ECDSA
Homomorphism bid	Always (algebraic structure)	Immediate collapse	Textbook RSA
Masquerade bid	Identity not bound in protocol	Immediate collapse	NS protocol
O2H bid	$q_H \geq 2^{n/2}$ with $\sqrt{\epsilon}$	Equilibrium for ML-KEM	QKEM
Salt extraction bid	Salt accessible in plaintext	Partial collapse (2^{-64})	MTProto
HMAC salt binding	auth_key secure	Equilibrium	RMTP
Nonce-misuse bid	Nonce reused with same key	Immediate collapse	ChaCha20
Cube superpoly bid	Superpoly degree ≤ 1	Equilibrium (2^{-38} margin)	Trivium
Lightweight birthday bid	$q_E \geq 2^{n/2}$ (block size)	Collapse if $q_E \geq 2^{32}$	PRESENT
Merkle tree LE bid	Never (structural)	Permanent equilibrium	BLAKE3
Boomerang bid	Two-differential composes	Equilibrium (20-round margin)	Serpent
No-cloning bid	Never (physics)	Permanent equilibrium	BB84
Cross-protocol replay	Never (label-bound SIDs)	Permanent equilibrium	TLS+Signal
Infrastructure forgery	CA signature broken	Network collapse	Shared PKI

Quick-Reference Cards

Quick-Reference Card 1: MTSF Notation Cheat Sheet

Symbol	Name	Meaning
$\mathcal{M}$	Security market	The complete auction setup
Seller	Seller	Challenger (runs scheme honestly)
Buyer_i / $\mathcal{A}$	Buyer	Adversary (bids computational resources)
$\mathcal{G}$	Goods catalogue	Set of security properties offered
g_j	Security good	One property (e.g., IND-CCA2, EUF-CMA)
$\mathsf{Ask}(g_j)$	Ask price	Max advantage over all PPT buyers
QPrice_0	Quoted price	Raw advantage in $\mathbf{B}_0$ (baseline)
$\mathbf{B}_k$	Bidding round k	The k-th game in the hop chain
$\Delta\mathsf{Price}_k$	Price adjustment	Advantage change from $\mathbf{B}_k$ to $\mathbf{B}_{k+1}$
$\mathsf{negl}(\lambda)$	Negligible	Shrinks faster than any $1/\lambda^c$
φ_i	Session-CNF	Boolean checklist for session i
Ping	Session ping	Structural distinctness check

Quick-Reference Card 2: The MTSF Proof Checklist

- ☐ **Define market:** Seller, buyers, goods catalogue $\mathcal{G}$, price function.
- ☐ **$\mathbf{B}_0$ (Quoted Price):** State $\mathsf{QPrice}_0 = \mathsf{Adv}^{g_j}_{\mathcal{A}}(\lambda)$.
- ☐ **Nonce freshness bid:** $\Delta\mathsf{Price} \leq q^2/2N$.
- ☐ **Hash collision bid:** $\Delta\mathsf{Price} \leq q_H^2/2^{n+1}$.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

- ☐ **Oracle programming:** $\Delta\mathsf{Price} = 0$.
- ☐ **Computational hop:** $\Delta\mathsf{Price} \leq \mathsf{Adv}^{\mathsf{hard}}$.
- ☐ **Forgery/extraction:** $\Delta\mathsf{Price} \leq L \cdot \mathsf{Adv}^{\mathsf{hard}}$.
- ☐ **Ideal game:** $\Pr[\mathbf{B}_q = 1] \in \{0, 1/2\}$.
- ☐ **Total:** $\mathsf{Ask}(g_j) \leq \sum \Delta\mathsf{Price}_k \leq \mathsf{negl}(\lambda)$.
- ☐ **CNF worksheet:** Fill in 4-column table for each clause.
- ☐ **Session pinging:** Verify structural distinctness.
- ☐ **Unbounded theorem:** Apply Theorem 3.
- ☐ **Bid taxonomy:** Check all relevant bids from Appendix B.

Quick-Reference Card 3: CNF Worksheet Template

Copy this template for each case study:

Clause C_j	Check	How to verify	Pass? (T/F)
φ^{sid}	SID matches session?	Compare SID in transcript	
φ^{fresh}	Nonces fresh?	Check nonce log	
φ^{sig}	Signatures verify?	Run $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$	
φ^{mac}	MACs verify?	Run $\mathsf{Mac.Vrfy}(K, m, \tau)$	
$\varphi^{\mathsf{consist}}$	Protocol order OK?	Check state machine	
Result:	All T $\Rightarrow$ Session correct. Any F $\Rightarrow$ Identify failed clause.		

Quick-Reference Card 4: MTSF $\leftrightarrow$ Traditional Dictionary

Traditional	MTSF	Traditional	MTSF
Adversary $\mathcal{A}$	Buyer	Security game	Market
Challenger $\mathcal{C}$	Seller	Game hop	Bidding round
Advantage ϵ	Ask price	Failure event F	Bad bid
Secure	Equilibrium	Broken	Collapse
UC env. $\mathcal{Z}$	Regulator	Simulator $\mathcal{S}$	Arbitrageur
Composition	Market merger	ProVerif	Stress test
CRS	Infrastructure	GUC CNF	MTSF CNF

Quick-Reference Card 5: Market Verdict Decision Table

Ask Price	Verdict	Symbol	Examples
$\mathsf{Ask} \leq \mathsf{negl}(\lambda)$	**Equilibrium**	✓	ECDSA, ML-KEM, AES, TLS 1.3
$\mathsf{Ask} = 2^{-64}$ (small)	**Quasi-collapse**	❊	Telegram MTProto
$\mathsf{Ask} = 1$	**Full collapse**	✗	Textbook RSA, NS protocol
$\mathsf{Ask} = 0$ (structural)	**Permanent equil.**	✓✓	BB84 QKD, BLAKE3 LE

References

1. Shoup, V. Sequences of Games. *IACR ePrint* **2004**, *2004*, 332.
2. Bellare, M.; Rogaway, P. Code-Based Game-Playing Proofs. In Proceedings of the EUROCRYPT, 2006, Vol. 4004, *LNCS*, pp. 409–426.
3. Canetti, R. Universally Composable Security. In Proceedings of the FOCS. IEEE, 2001, pp. 136–145.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

4. Canetti, R.; Dodis, Y.; Pass, R.; Walfish, S. Universally Composable Security with Global Setup. In Proceedings of the TCC, 2007, Vol. 4392, *LNCS*, pp. 61–85.
5. Camenisch, J.; Manulis, M.; Neven, G. On the Security of One-Round Protocols in the UC Framework. In Proceedings of the Workshop on Security and Cryptography for Networks (SCN), 2010. UC session correctness via CNF.
6. Blanchet, B. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In Proceedings of the CSFW, 2001, pp. 82–96.
7. Meier, S.; Schmidt, B.; Cremers, C.; Basin, D. The TAMARIN Prover. In Proceedings of the CAV, 2013, Vol. 8044, *LNCS*, pp. 696–701.
8. Blanchet, B. A Computationally Sound Mechanized Prover. In Proceedings of the IEEE S&P, 2006, pp. 140–154.
9. Needham, R.M.; Schroeder, M.D. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM* **1978**, *21*, 993–999.
10. Lowe, G. Breaking and Fixing the Needham–Schroeder Public-Key Protocol Using FDR. In Proceedings of the TACAS, 1996, Vol. 1055, *LNCS*, pp. 147–166.
11. Goldwasser, S.; Micali, S. Probabilistic Encryption. *JCSS* **1984**, *28*, 270–299.
12. Bellare, M.; Rogaway, P. Random Oracles are Practical. In Proceedings of the CCS. ACM, 1993, pp. 62–73.
13. Karp, R.M. Reducibility among Combinatorial Problems. In Proceedings of the Complexity of Computer Computations. Plenum Press, 1972, pp. 85–103.
14. Bellare, M.; Neven, G. Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. In Proceedings of the CCS. ACM, 2006, pp. 390–399. Introduces the general forking lemma.
15. Biham, E.; Shamir, A. Differential Fault Analysis of Secret Key Cryptosystems. In Proceedings of the Advances in Cryptology – CRYPTO 1997. Springer, 1997, Vol. 1294, *LNCS*, pp. 513–525.
16. Ishai, Y.; Sahai, A.; Wagner, D. Private Circuits: Securing Hardware against Probing Attacks. In Proceedings of the Advances in Cryptology – CRYPTO 2003. Springer, 2003, Vol. 2729, *LNCS*, pp. 463–481.
17. Fuchsbauer, G.; Kiltz, E.; Loss, J. The Algebraic Group Model and its Applications. In Proceedings of the Advances in Cryptology – CRYPTO 2018. Springer, 2018, Vol. 10992, *LNCS*, pp. 33–62.
18. Don, J.; Fehr, S.; Majenz, C.; Schaffner, C. Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model. In Proceedings of the Advances in Cryptology – CRYPTO 2019. Springer, 2019, Vol. 11693, *LNCS*, pp. 356–383. Measure-and-reprogram technique.
19. Bellare, M.; Canetti, R.; Krawczyk, H. Keying Hash Functions for Message Authentication. In Proceedings of the CRYPTO, 1996, Vol. 1109, *LNCS*, pp. 1–15.
20. NIST. FIPS 197: Advanced Encryption Standard (AES). Technical report, NIST, 2001.
21. Biham, E.; Shamir, A. Differential Cryptanalysis of DES-like Cryptosystems. In Proceedings of the CRYPTO, 1991, Vol. 537, *LNCS*, pp. 2–21.
22. Daemen, J.; Rijmen, V. *The Design of Rijndael: AES—The Advanced Encryption Standard*; Springer, 2002.
23. Khovratovich, D.; Nikolić, I. Rotational Cryptanalysis of ARX. In Proceedings of the FSE, 2010, Vol. 6147, *LNCS*, pp. 333–346.
24. Biryukov, A.; Khovratovich, D. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Proceedings of the ASIACRYPT, 2009, Vol. 5912, *LNCS*, pp. 1–18.
25. Cid, C.; Murphy, S.; Robshaw, M. *Algebraic Aspects of the Advanced Encryption Standard*; Information Security and Cryptography, Springer, 2006.
26. Daemen, J.; Knudsen, L.R.; Rijmen, V. The Block Cipher Square. In Proceedings of the Fast Software Encryption (FSE) 1997. Springer, 1997, Vol. 1267, *LNCS*, pp. 149–165.
27. Derbez, P.; Fouque, P.A.; Jean, J. Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. In Proceedings of the Advances in Cryptology – EUROCRYPT 2013. Springer, 2013, Vol. 7881, *LNCS*, pp. 371–387.
28. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.B.; Seurin, Y.; Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In Proceedings of the CHES 2007. Springer, 2007, Vol. 4727, *LNCS*, pp. 450–466.
29. Cho, J.Y. Linear Cryptanalysis of Reduced-Round PRESENT. In Proceedings of the CT-RSA 2010. Springer, 2010, Vol. 5985, *LNCS*, pp. 232–248.
30. Anderson, R.; Biham, E.; Knudsen, L. Serpent: A Proposal for the Advanced Encryption Standard. In Proceedings of the First AES Candidate Conference, 1998.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

31. Biham, E.; Dunkelman, O.; Keller, N. Linear Cryptanalysis of Reduced Round Serpent. In Proceedings of the FSE 2001. Springer, 2002, Vol. 2355, *LNCS*, pp. 16–27.
32. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. The Keccak Reference. In Proceedings of the Submission to NIST SHA-3 Competition, 2011.
33. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. On the Indifferentiability of the Sponge Construction. In Proceedings of the EUROCRYPT, 2008, Vol. 4965, *LNCS*, pp. 181–197.
34. O'Connor, J.; Aumasson, J.P.; Neves, S.; Wilcox-O'Hearn, Z. BLAKE3: One Function, Fast Everywhere, 2020. Specification, https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf.
35. Aumasson, J.P.; Neves, S.; Wilcox-O'Hearn, Z.; Winnerlein, C. BLAKE2: Simpler, Smaller, Fast as MD5. In Proceedings of the ACNS 2013. Springer, 2013, Vol. 7954, *LNCS*, pp. 119–135.
36. Dobraunig, C.; Eichlseder, M.; Grassi, L.; Mendel, F.; Rechberger, C.; Schläffer, M.; Stolz, S. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. In Proceedings of the Journal of Cryptology, 2021, Vol. 34, pp. 1–42.
37. National Institute of Standards and Technology. Lightweight Cryptography Standardization Process, 2023. NIST Selected Ascon.
38. Ågren, M.; Hell, M.; Johansson, T.; Meier, W. Grain-128a: A New Version of Grain-128 with Optional Authentication. *International Journal of Wireless and Mobile Computing (IJWMC)* **2011**, *5*, 48–59.
39. Courtois, N.T. Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In Proceedings of the CRYPTO, 2003, Vol. 2729, *LNCS*, pp. 176–194.
40. Todo, Y.; Isobe, T.; Hao, Y.; Meier, W. Cube Attacks on Non-Blackbox Polynomials Based on Division Property. In Proceedings of the CRYPTO, 2017, Vol. 10403, *LNCS*, pp. 250–279.
41. Babbage, S.; Dodd, M. The Stream Cipher MICKEY (version 2). In Proceedings of the ECRYPT Stream Cipher Project, 2005. Contains TMTO analysis for stream ciphers.
42. Hellman, M.E. A Cryptanalytic Time-Memory Trade-Off. *IEEE Transactions on Information Theory* **1980**, *26*, 401–406.
43. Bernstein, D.J. ChaCha, a variant of Salsa20. In Proceedings of the Workshop Record of SASC 2008, 2008.
44. Bernstein, D.J. The Salsa20 Family of Stream Ciphers. In Proceedings of the New Stream Cipher Designs. Springer, 2008, Vol. 4986, *LNCS*, pp. 84–97.
45. De Cannière, C.; Preneel, B. Trivium. In Proceedings of the New Stream Cipher Designs: The eSTREAM Finalists. Springer, 2008, Vol. 4986, *LNCS*, pp. 244–266.
46. Makarim, R.H.; Rockmore, D.N. On the Algebraic Degree of Trivium Output Bits. 2021.
47. Dinur, I.; Shamir, A. Cube Attacks on Tweakable Black Box Polynomials. In Proceedings of the Advances in Cryptology – EUROCRYPT 2009. Springer, 2009, Vol. 5479, *LNCS*, pp. 278–299.
48. Huang, S.; Lai, X. Revisiting Key-Alternating Feistel Ciphers for Shorter Keys and Multi-user Security. In Proceedings of the Advances in Cryptology – ASIACRYPT 2020. Springer, 2020, Vol. 12491, *LNCS*, pp. 213–243. Conditional cube attack extensions on stream ciphers.
49. Marlinspike, M.; Perrin, T. The X3DH Key Agreement Protocol. Signal Foundation, 2016. https://signal.org/docs/specifications/x3dh/.
50. Perrin, T.; Marlinspike, M. The Double Ratchet Algorithm. Signal Foundation, 2016. https://signal.org/docs/specifications/doubleratchet/.
51. Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3. IETF RFC 8446, 2018. https://www.rfc-editor.org/rfc/rfc8446.
52. Krawczyk, H.; Eronen, P. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). IETF RFC 5869, 2010.
53. AlFardan, N.J.; Paterson, K.G. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In Proceedings of the IEEE S&P, 2013, pp. 526–540.
54. Dowling, B.; Fischlin, M.; Günther, F.; Stebila, D. A Cryptographic Analysis of the TLS 1.3 Handshake Protocol. In Proceedings of the Journal of Cryptology. Springer, 2021, Vol. 34, pp. 1–69.
55. Unruh, D. Revocable Quantum Timed-Release Encryption. In Proceedings of the EUROCRYPT, 2015, Vol. 9057, *LNCS*, pp. 129–158. Introduces the One-Way to Hiding (O2H) lemma for QROM security proofs.
56. Don, J.; Fehr, S.; Majenz, C.; Schaffner, C. Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model. In Proceedings of the CRYPTO, 2019, Vol. 11693, *LNCS*, pp. 356–383. Measure-and-reprogram technique for QROM proofs.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 April 2026 doi:10.20944/preprints202603.2237.v2

57. National Institute of Standards and Technology. FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard. Federal information processing standards publication, NIST, 2024. Includes QROM security proof for ML-KEM.
58. Bennett, C.H.; Brassard, G. Quantum cryptography: Public key distribution and coin tossing. In Proceedings of the Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing, Bangalore, India, 1984; pp. 175–179.
59. Barthe, G.; Grégoire, B.; Heraud, S.; Zanella-Béguelin, S. Computer-Aided Security Proofs for the Working Cryptographer. In Proceedings of the CRYPTO, 2011, Vol. 6841, *LNCS*, pp. 71–90. EasyCrypt proof assistant.
60. Möller, B.; Duong, T.; Kotowicz, K. This POODLE Bites: Exploiting the SSL 3.0 Fallback. In Proceedings of the Google Security Advisory, 2014.
61. Bardou, R.; Focardi, R.; Kawamoto, Y.; Simionato, L.; Steel, G.; Tsay, J.K. Efficient Padding Oracle Attacks on Cryptographic Hardware. In Proceedings of the CRYPTO, 2012, Vol. 7417, *LNCS*, pp. 608–625.
62. Jager, T.; Kohlar, F.; Schäge, S.; Schwenk, J. On the Security of TLS-DHE in the Standard Model. In Proceedings of the CRYPTO, 2012, Vol. 7417, *LNCS*, pp. 273–293.
63. Cremers, C.; Hövelmanns, M.; Rausch, A. Formally Verifying TLS 1.3's Security: Tamarin Proofs for All Handshake Modes. In Proceedings of the IACR ePrint, 2017. Report 2017/323.
64. Krawczyk, H. Cryptographic Extraction and Key Derivation: The HKDF Scheme. *IACR Cryptology ePrint Archive* **2010**, *2010*, 264.
65. Fischlin, M.; Günther, F.; Marson, G.A. Data Is a Stream: Security of Stream-Based Channels. In Proceedings of the CRYPTO, 2015, Vol. 9216, *LNCS*, pp. 545–564. 0-RTT security model for TLS.
66. Adrian, D.; et al. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In Proceedings of the CCS. ACM, 2015, pp. 5–17.
67. Anderson, R. Why Information Security is Hard. In Proceedings of the ACSAC, 2001, pp. 358–365.
68. Grossklags, J.; Christin, N.; Chuang, J. Secure or Insure? In Proceedings of the WWW, 2008, pp. 209–218.
69. Albrecht, M.R.; et al. Four Attacks and a Proof for Telegram. In Proceedings of the IEEE S&P, 2022, pp. 1–17.
70. Jakobsen, J.; Orlandi, C. On the CCA (In)security of MTProto. In Proceedings of the SPSM, 2016, pp. 113–116.
71. Rogaway, P. Authenticated-Encryption with Associated-Data. In Proceedings of the CCS. ACM, 2002, pp. 98–107.
72. Bellare, M.; Namprempre, C. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Proceedings of the ASIACRYPT, 2000, Vol. 1976, *LNCS*, pp. 531–545.
73. NIST. FIPS 205: SLH-DSA. Technical report, NIST, 2024.
74. NIST. FIPS 206: FN-DSA. Technical report, NIST, 2024.
75. Hofheinz, D.; Hövelmanns, K.; Kiltz, E. A Modular Analysis of the Fujisaki-Okamoto Transformation. In Proceedings of the TCC, 2017, Vol. 10677, *LNCS*, pp. 341–371. QROM analysis of the FO transform.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.