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Not peer-reviewed version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: 27 March 2026

doi: 10.20944/preprints202603.2237.v1

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

 doi:10.20944/preprints202603.2237.v1

Article

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

Basker Palaniswamy * and Paolo Palmieri

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 or basker170889@zohomail.eu

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 language. 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 mental 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: 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

 doi:10.20944/preprints202603.2237.v1

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.

 doi:10.20944/preprints202603.2237.v1

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.

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.

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.

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.

 doi:10.20944/preprints202603.2237.v1

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.

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.

1.2. What MTSF Provides

☞ Key Insight:

What Is New in MTSF—Eighteen 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.

 doi:10.20944/preprints202603.2237.v1

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.
17. **Ping bids in every case study:** Every case study includes a ping bid formally proving unbounded session security.
18. **Simple Terms explanations:** Every major definition, theorem, and proof is accompanied by a plain-language explanation in "Simple Terms" boxes, making the framework accessible to engineers and policy makers without cryptographic expertise.

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 (Definition 13) and the session ping function (Definition 14), 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).

 doi:10.20944/preprints202603.2237.v1

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 17: comprehensive related work in ten subsections. Section 18: conclusion with eighteen contributions, lessons learned, and seven future directions (with initial formalisations of quantum market dynamics and multi-protocol composition networks). Appendix A: expanded MTSF correspondence table (79 entries). Appendix B: comprehensive bid taxonomy with QROM, Telegram, BB84, and composition bids.

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.

 doi:10.20944/preprints202603.2237.v1

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).

2.2. Semantic Security and Core Definitions

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* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}} \leq \mathsf{negl}(\lambda)$.

 doi:10.20944/preprints202603.2237.v1

✻ 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.

 doi:10.20944/preprints202603.2237.v1

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

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, 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):

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

✻ 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:* $|\Pr[\mathbf{B}_k(\mathcal{A})=1] - \Pr[\mathbf{B}_{k+1}(\mathcal{A})=1]| \leq \Pr[F \text{ in } \mathbf{B}_k]$.

 doi:10.20944/preprints202603.2237.v1

✻ 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].

 doi:10.20944/preprints202603.2237.v1

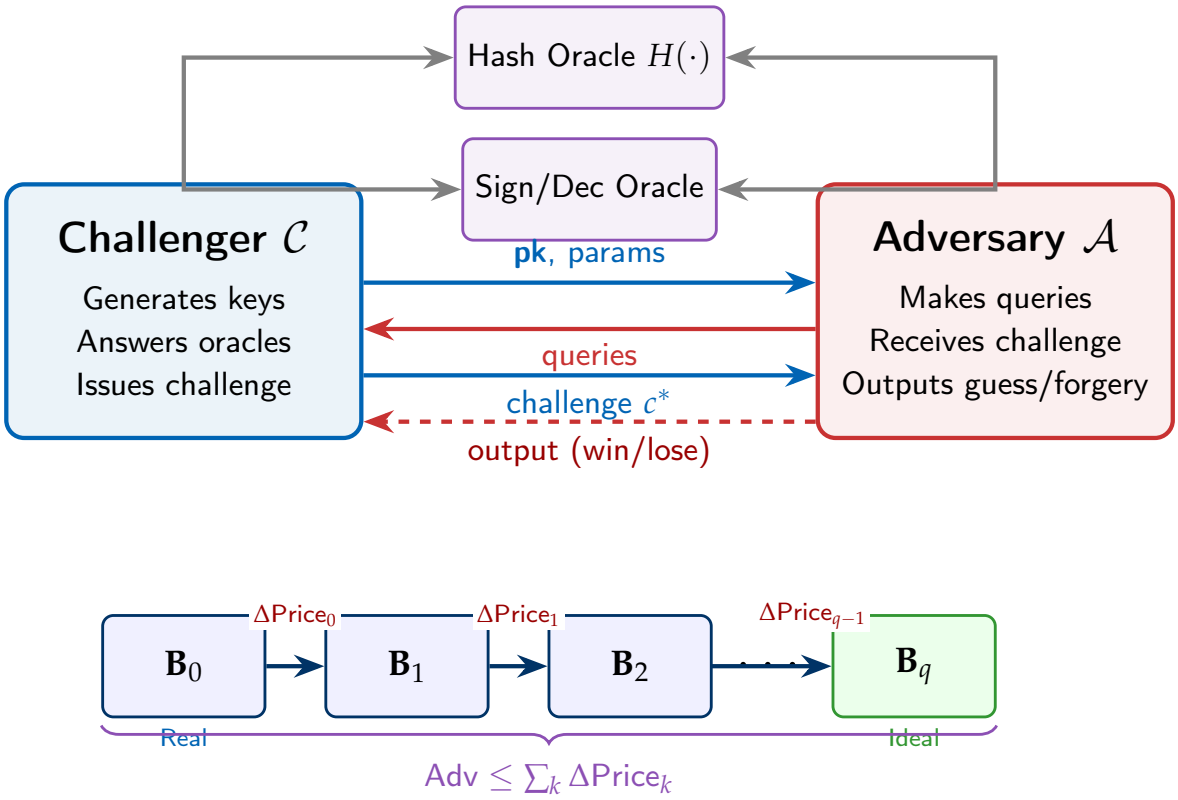


Figure 1. Bidding-round proof architecture. The seller (challenger) and buyer (adversary) interact via oracles. The bidding-round chain $\mathbf{B}_0 \to \mathbf{B}_q$ bounds the total advantage. In MTSF, each ΔPrice_k is a "price adjustment" in the security market.

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.

 doi:10.20944/preprints202603.2237.v1

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.

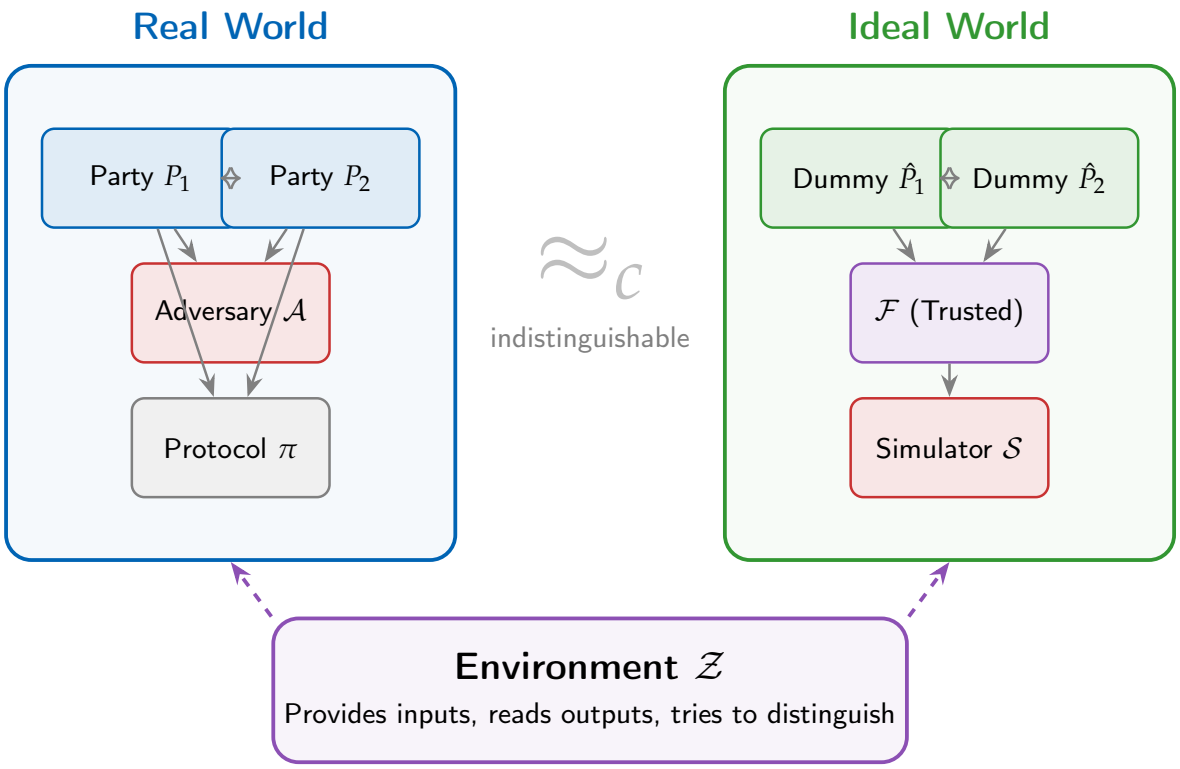


Figure 2. UC framework. The environment $\mathcal{Z}$ interacts with either the real world (protocol π with adversary $\mathcal{A}$) or the ideal world (trusted functionality $\mathcal{F}$ with simulator $\mathcal{S}$). UC security: $\mathcal{Z}$ cannot tell which world it observes.

2.5. GUC and CNF Session Correctness

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.

✻ 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

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

 doi:10.20944/preprints202603.2237.v1

✻ 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.

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").

 doi:10.20944/preprints202603.2237.v1

- **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.
- **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.

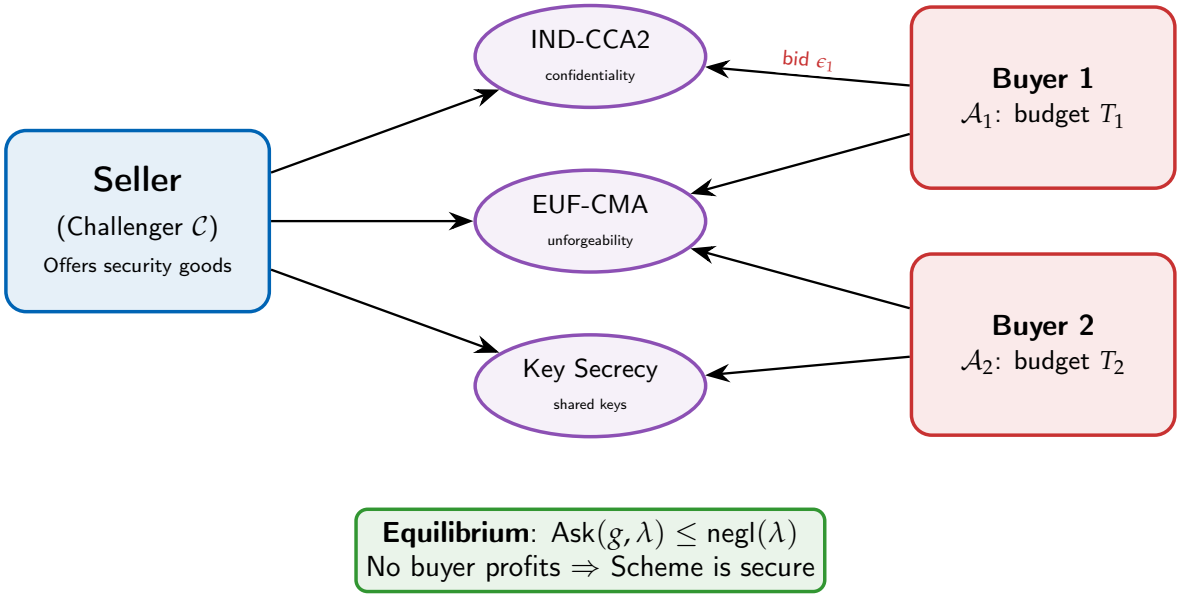


Figure 3. MTSF market: seller offers security goods; buyers bid computation; equilibrium = security.

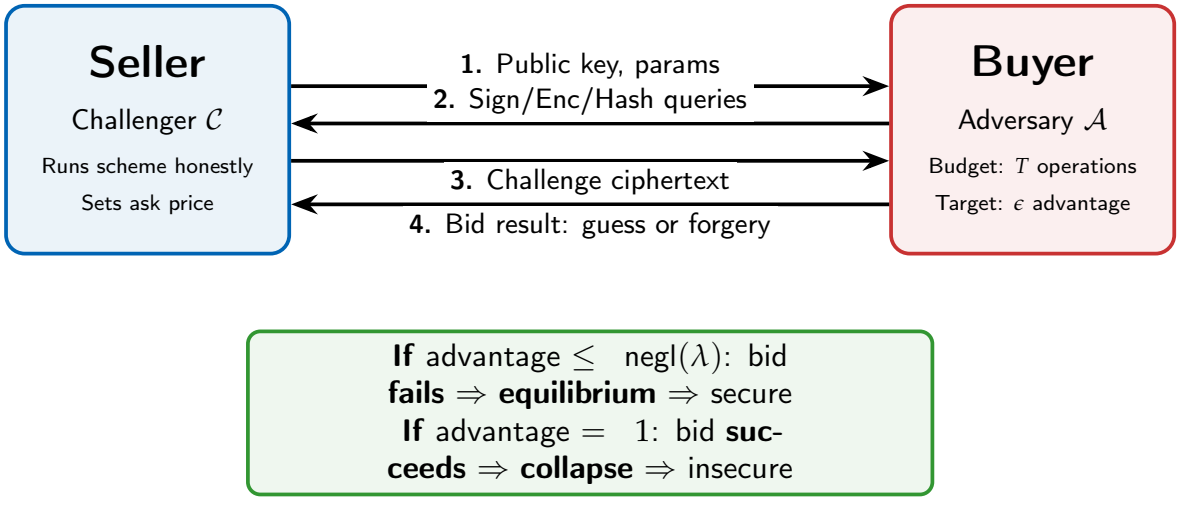


Figure 4. Individual seller–buyer interaction: the four-phase security game as a market transaction.

 doi:10.20944/preprints202603.2237.v1

Definition 10 (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.
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.

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).

3.2. Game Hops as Price Adjustments

Definition 11 (Price Adjustment). $\Delta\mathsf{Price}_k = |\Pr[\mathbf{B}_k(\mathcal{A}) = 1] - \Pr[\mathbf{B}_{k+1}(\mathcal{A}) = 1]|$. *Cumulative cost:* $\mathsf{Cost}(\mathcal{A}) = \sum_{k=0}^{q-1} \Delta\mathsf{Price}_k$.

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:*

$$|\Pr[\mathbf{B}_k(\mathcal{A}) = 1] - \Pr[\mathbf{B}_{k+1}(\mathcal{A}) = 1]| \leq \Pr[F_1 \cup F_2 \cup \cdots \cup F_m]. \tag{3}$$

 doi:10.20944/preprints202603.2237.v1

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

$$\Pr\left[\bigcup_i F_i\right] = \sum_i \Pr[F_i] - \sum_{i<j} \Pr[F_i \cap F_j] + \cdots + (-1)^{m+1} \Pr\left[\bigcap_i F_i\right]. \tag{4}$$

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: $\Delta\mathsf{Price} \leq \Pr[F_1 \cup F_2 \cup F_3] \leq \Pr[F_1] + \Pr[F_2] + \Pr[F_3] - \Pr[F_1 \cap F_2] - \cdots$, yielding a tighter bound and shorter proof.

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—$\mathsf{Real}_{\pi,\mathcal{A},\mathcal{Z}} \approx_c \mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. 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 strings) are *market infrastructure*: public goods accessible to all buyers and sellers. In MTSF, the CNF

 doi:10.20944/preprints202603.2237.v1

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.

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 $\forall\, \tau \in \mathsf{Traces}(\mathcal{M}) : \tau \models \varphi_{g_j}$ 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.

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.

✻ 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 ($\wedge$). 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.

4.2. Session-CNF Construction

Definition 12 (MTSF Session-CNF). *Let* $\mathsf{Session}_i = (\mathsf{sid}_i, \mathsf{pid}_i^1, \ldots, \mathsf{pid}_i^p, \mathsf{trans}_i)$ *denote the i-th session with p parties and transcript* trans_i. *The* MTSF session-CNF *is:*

$$\varphi_i = \underbrace{\varphi_i^{\mathsf{sid}}}_{\textit{SID binding}} \wedge \underbrace{\varphi_i^{\mathsf{fresh}}}_{\textit{freshness}} \wedge \underbrace{\varphi_i^{\mathsf{sig}}}_{\textit{signature binding}} \wedge \underbrace{\varphi_i^{\mathsf{mac}}}_{\textit{MAC binding}} \wedge \underbrace{\varphi_i^{\mathsf{consist}}}_{\textit{consistency}}. \tag{5}$$

 doi:10.20944/preprints202603.2237.v1

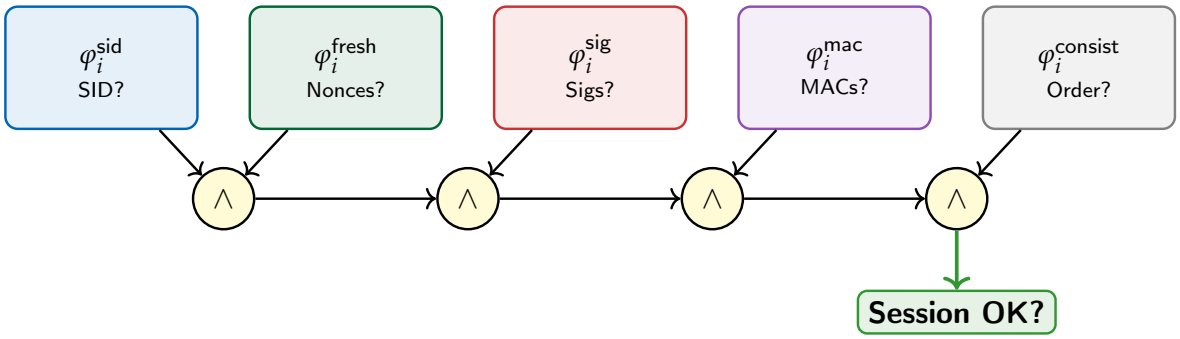


Figure 5. Structure of the MTSF session-CNF φ_i: five components ANDed together. A session is correct iff the entire formula is satisfiable under the honest trace.

SID Clauses.

Every message must carry the correct session identifier:

$$\varphi_i^{\text{sid}} = \bigwedge_{j=1}^{\ell} \left(x_{\text{sid}_i,j} \vee \neg x_{\text{other},j}\right), \tag{6}$$

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:

$$\varphi_i^{\text{fresh}} = \bigwedge_{k=1}^{r} \left(x_{n_k,\text{fresh}}\right) \wedge \bigwedge_{k \neq k'} \left(\neg x_{n_k = n_{k'}}\right). \tag{7}$$

Signature-Binding Clauses.

Every signature must verify or the session aborts:

$$\varphi_i^{\text{sig}} = \bigwedge_{j} \left(x_{\text{Vrfy}(\text{pk}_j, m_j \| \text{sid}_i, \sigma_j)=1} \vee x_{\text{abort}}\right). \tag{8}$$

MAC-Binding Clauses.

Every MAC must verify or the session aborts:

$$\varphi_i^{\text{mac}} = \bigwedge_{j} \left(x_{\text{Mac.Vrfy}(K_j, m_j \| \text{sid}_i, \tau_j)=1} \vee x_{\text{abort}}\right). \tag{9}$$

Consistency Clauses.

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

$$\varphi_i^{\text{consist}} = \bigwedge_{j=1}^{c} C_j^{\text{protocol}}. \tag{10}$$

★ 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)**.** Session_i *is correct iff* φ_i *is satisfiable under the honest trace* trans_i.

 doi:10.20944/preprints202603.2237.v1

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:*

$$\Pr[\mathcal{A} \textit{ produces a satisfying assignment for } \varphi_i \textit{ under a dishonest trace}] \leq \mathsf{negl}(\lambda). \tag{11}$$

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.

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

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

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

 doi:10.20944/preprints202603.2237.v1

✓ 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.

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?

 doi:10.20944/preprints202603.2237.v1

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-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 13 (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 14 (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 Definition 13.*

 doi:10.20944/preprints202603.2237.v1

Definition 15 (Ping Failure Probability). *The* ping failure probability *for protocol* π *is:*

$$\delta_{\mathsf{ping}}(\pi) \;=\; \max_{i\geq 1} \Pr[\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0], \tag{12}$$

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.

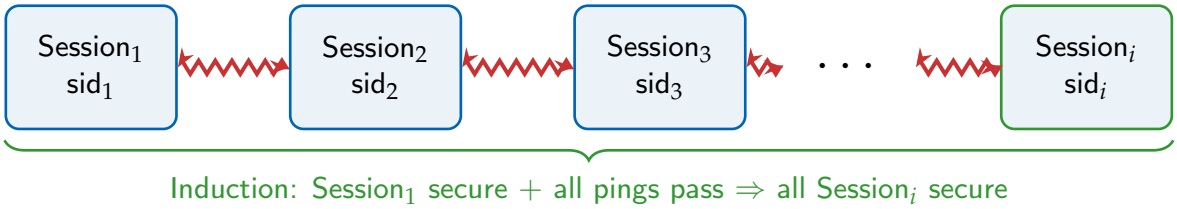


Figure 6. Session pinging for unbounded verification. Consecutive sessions are "pinged" to verify structural distinctness. The zigzag edges denote the ping check between adjacent sessions.

 doi:10.20944/preprints202603.2237.v1

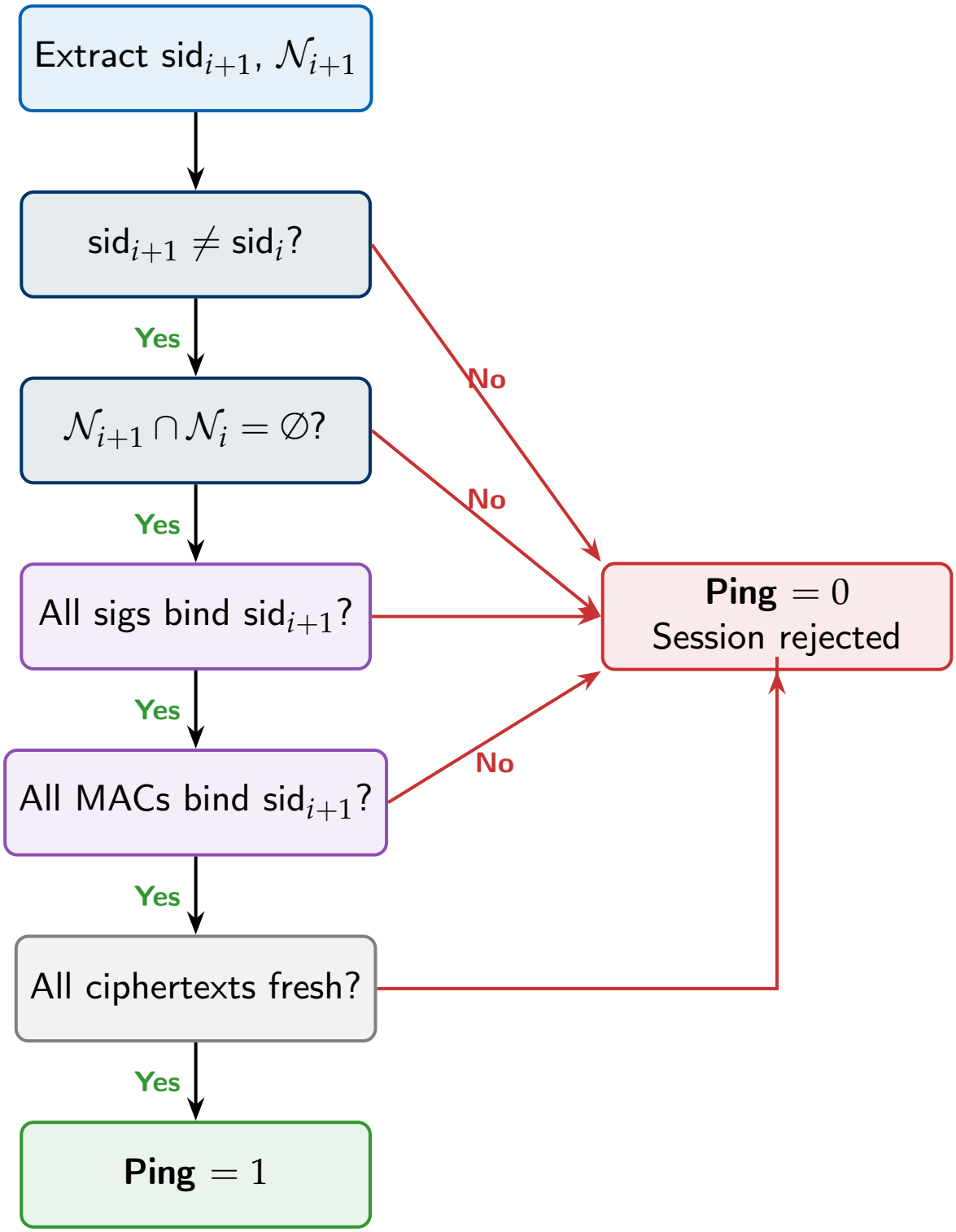


Figure 7. Detailed ping verification flow for session Session_{i+1} against its predecessor Session_i. A failure at any step short-circuits to Ping $= 0$ (session rejected). This flowchart corresponds to Phase 5 of Algorithm 1.

5.3. The Unbounded Security Theorem

Theorem 3 (Unbounded Security via Session Pinging). *If (i)* Session_1 *is secure (market in equilibrium with* $\text{Ask}_1 \leq \epsilon$*), (ii)* $\text{Ping}(\text{Session}_i, \text{Session}_{i+1}) = 1$ *for all* $i \geq 1$*, and (iii)* φ_{i+1} *is obtained from* φ_i *by variable renaming (substituting* $\text{sid}_i \mapsto \text{sid}_{i+1}$ *and fresh nonces), then* π *is secure for unbounded sessions, with:*

$$\text{Ask}_i \;\leq\; \epsilon + (i-1) \cdot \delta_{\text{ping}}(\pi) \quad \textit{for all } i \geq 1. \tag{13}$$

 doi:10.20944/preprints202603.2237.v1

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 Definition 13, 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:

$$\varphi_{i+1}[\mathsf{trans}_{i+1}^{\mathsf{honest}}] = \varphi_i[\mathsf{trans}_i^{\mathsf{honest}}]\big|_{\mathsf{sid}_i \mapsto \mathsf{sid}_{i+1},\, \mathcal{N}_i \mapsto \mathcal{N}_{i+1}} = \mathsf{TRUE}. \tag{14}$$

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:

$$\mathsf{Ask}_{i+1} \le \mathsf{Ask}_1 + \Pr\big[\exists\, j \le i : \mathsf{Ping}(\mathsf{Session}_j, \mathsf{Session}_{j+1}) = 0\big] \le \epsilon + i \cdot \delta_{\mathsf{ping}}(\pi). \tag{15}$$

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

 doi:10.20944/preprints202603.2237.v1

5.4. Interaction Between Session Pinging and CNF Verification

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* $\mathsf{Session}_i$*:*

$$[Ping - CNFSoundness] \quad \begin{array}{l} (\varphi_{i+1} = \mathsf{SAT}) \ \wedge \ (\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 1) \implies \\ \mathsf{Session}_{i+1} \textit{ is correct and structurally independent of } \mathsf{Session}_i. \end{array} \tag{16}$$

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 (Definition 14) 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 Definition 13 are satisfied, so the ping passes. □

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 Definition 13:

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.

Definition 16 (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* $\approx 2^{-1}$*.*

 doi:10.20944/preprints202603.2237.v1

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).*
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.

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.*

5.7. Comparison with Symbolic Unbounded Verification

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

 doi:10.20944/preprints202603.2237.v1

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.

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.

 doi:10.20944/preprints202603.2237.v1

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 (Definition 14) 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).

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.

 doi:10.20944/preprints202603.2237.v1

✻ 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.

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 17 (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.

 doi:10.20944/preprints202603.2237.v1

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.

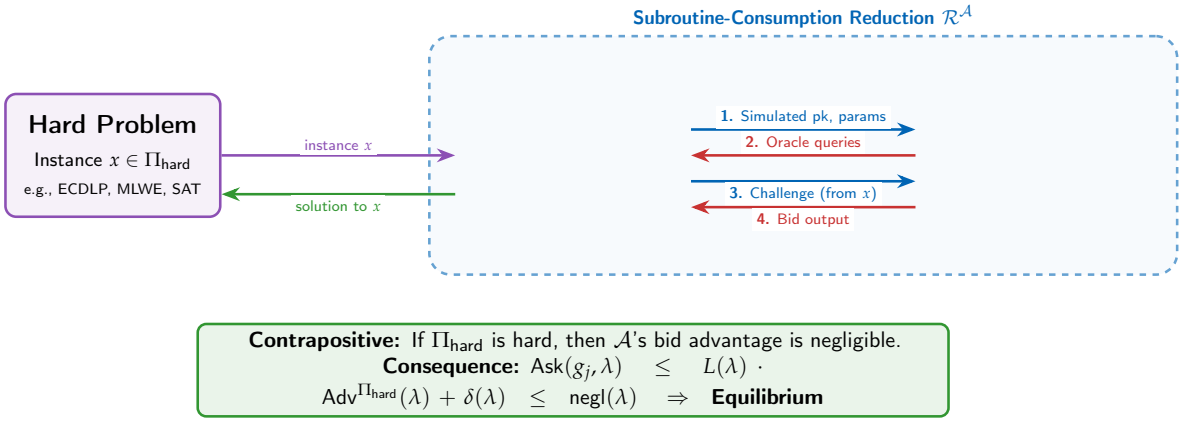


Figure 8. Subroutine-consumption reduction: the seller (reducer $\mathcal{R}$) receives a hard-problem instance x, simulates the market for the buyer $\mathcal{A}$, consumes $\mathcal{A}$'s bid output to solve x. Hardness of Π_{hard} implies the buyer's advantage is negligible—market equilibrium.

✎ 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).

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

Definition 18 (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.*

 doi:10.20944/preprints202603.2237.v1

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.

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

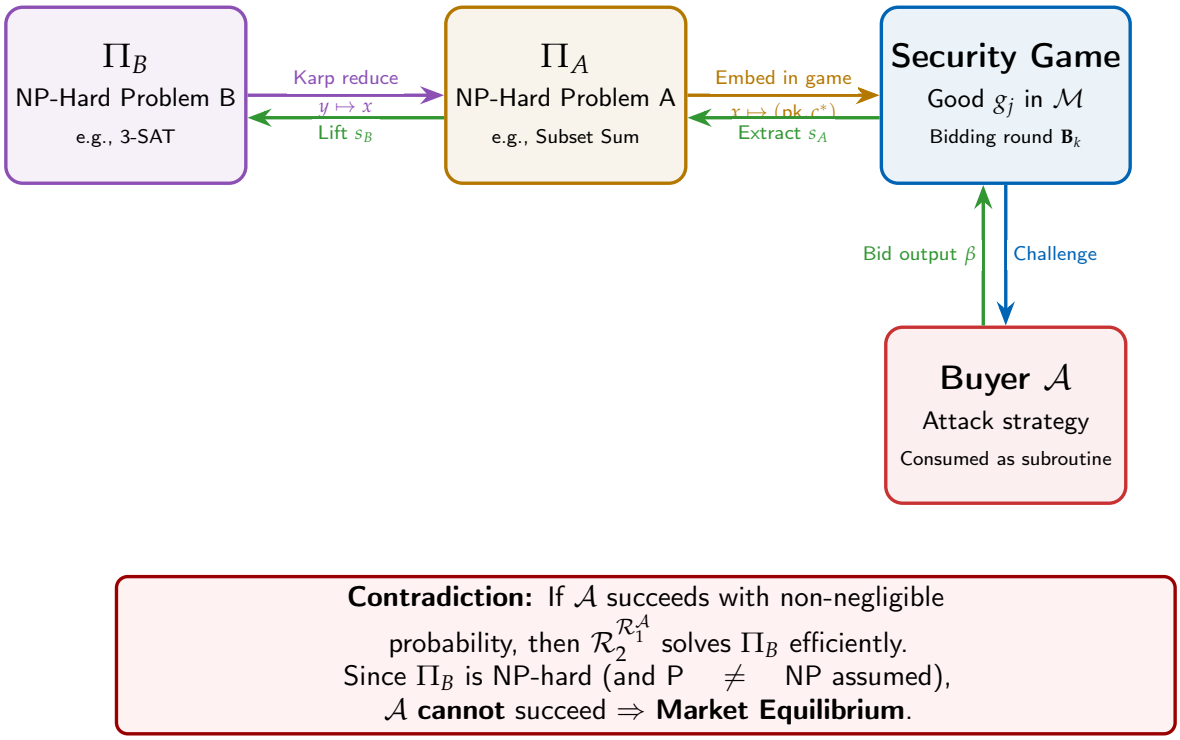
What the algorithm does step by step:

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.

 doi:10.20944/preprints202603.2237.v1

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.



Contradiction: If $\mathcal{A}$ succeeds with non-negligible probability, then $\mathcal{R}_2^{\mathcal{R}_1^{\mathcal{A}}}$ solves Π_B efficiently. Since Π_B is NP-hard (and P $\neq$ NP assumed), $\mathcal{A}$ **cannot** succeed $\Rightarrow$ **Market Equilibrium**.

Figure 9. 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.

 doi:10.20944/preprints202603.2237.v1

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 (Definition 17), 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. □

✻ 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 P ≠ 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.

 doi:10.20944/preprints202603.2237.v1

Proof. By the definition of ask price (Definition 10):

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

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 Definition 10 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 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

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. □

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 March 2026 doi:10.20944/preprints202603.2237.v1

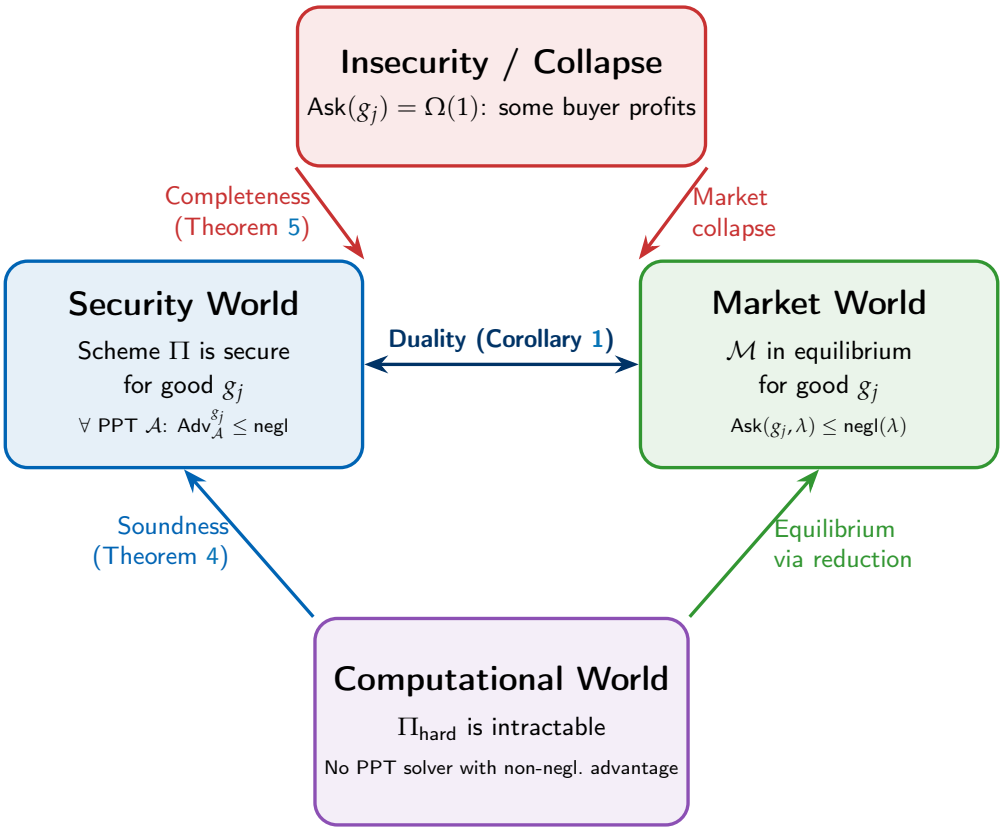


Figure 10. The soundness–completeness duality of MTSF. Soundness (bottom-left): hardness of Π_{hard} implies security and equilibrium. Completeness (top): insecurity implies market collapse. The three worlds—security, market, and computational—are formally equivalent under MTSF.

✎ 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 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).

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 Π_{hard} = 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 $m^* \notin \{m_1, \ldots, m_{q_S}\}$.*

 doi:10.20944/preprints202603.2237.v1

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:*

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

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 → Subset Sum → 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.

 doi:10.20944/preprints202603.2237.v1

7.8. Relationship to Classical Meta-Theorems

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 (Definition 17) 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 (Definition 18) 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* P $\neq$ NP *assumption. This connection grounds MTSF's security guarantees in the deepest conjectures of computational complexity theory.*

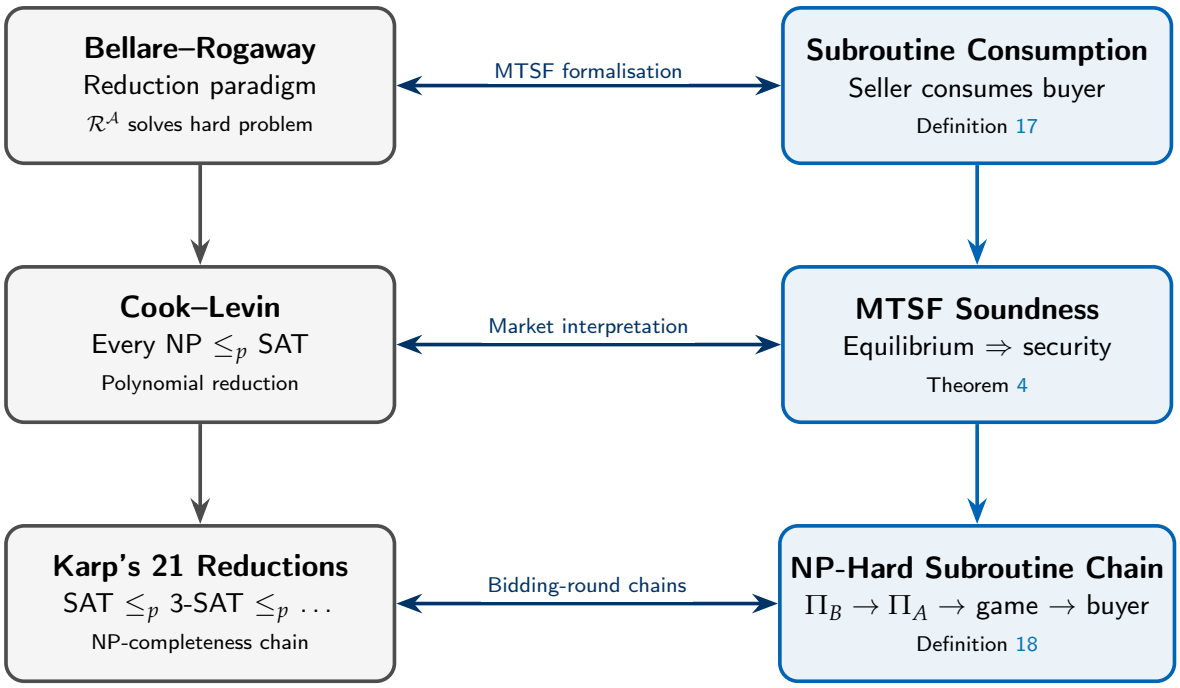


Figure 11. Correspondence between classical meta-theorems in complexity/cryptography and their MTSF counterparts. Each classical result has a precise market-theoretic formalisation.

7.9. Implications for MTSF Case Studies

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

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:

 doi:10.20944/preprints202603.2237.v1

- **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 62.

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.

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 (Definition 18) 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.

 doi:10.20944/preprints202603.2237.v1

✻ 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 19 (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.*

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

$$\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda), \tag{17}$$

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.

Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

 doi:10.20944/preprints202603.2237.v1

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 $\Delta\mathsf{Price} \le q^2/2^{\lambda+1}$, 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 $\Delta\mathsf{Price}_0 = |\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \le \Pr[F_{\mathsf{nonce}}]$.

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

$$|\Pr[\mathbf{B}_0(\mathcal{A}) = 1] - \Pr[\mathbf{B}_1(\mathcal{A}) = 1]| \le \Pr[F_{\mathsf{nonce}}] \le \frac{q_N^2}{2^{\lambda+1}}. \tag{18}$$

$\mathbf{B}_2$ (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. $\Pr[F_{\mathsf{fork}}] \le 1 - \mathsf{acc}^2/q_H + 1/q$ (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: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \le \Pr[F_{\mathsf{fork}}] + 1/q$.

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

 doi:10.20944/preprints202603.2237.v1

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 $\mathsf{Adv}^{\text{EUF-CMA}}_{\text{Sig}}$.

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 $(m^* \| \text{sid}, \sigma^*)$ as a forgery. Hence:

$$|\Pr[\mathbf{B}_1(\mathcal{A}) = 1] - \Pr[\mathbf{B}_2(\mathcal{A}) = 1]| \leq \Pr[F_{\text{forge}}] \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\text{Sig}}. \tag{19}$$

$\mathbf{B}_3$ (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: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. 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.

Conditioned on no nonce-reuse bid ($\neg F_{\text{nonce}}$) and no forgery bid ($\neg F_{\text{forge}}$) 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 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:

$$\text{Ask}(g_{\text{auth}}) \leq \frac{q_N^2}{2^{\lambda+1}} + \mathsf{Adv}^{\text{EUF-CMA}}_{\text{Sig}} + 0 = \mathsf{Adv}^{\text{EUF-CMA}}_{\text{Sig}} + \frac{q_N^2}{2^{\lambda+1}} + \text{negl}(\lambda). \quad \square \tag{20}$$

8.2. The Mutual Authentication Game

Definition 20 (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:*

$$\text{Ask}(g_{\text{mutual}}) \leq 2 \cdot \mathsf{Adv}^{\text{EUF-CMA}}_{\text{Sig}} + \frac{q_N^2}{2^{\lambda+1}} + \text{negl}(\lambda). \tag{21}$$

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,

 doi:10.20944/preprints202603.2237.v1

tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{mutual}}_{\mathcal{A}}(\lambda)$.

$\mathbf{B}_1$ (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 ($\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); 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 $q = \mathsf{poly}(\lambda)$. 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 across all sessions in both directions, placing a *freshness constraint* on nonce bids. Identical-until-F_{nonce}; price adjustment:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \le \frac{q_N^2}{2^{\lambda+1}}. \tag{22}$$

$\mathbf{B}_2$ (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 $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, 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: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \le \Pr[F_{\mathsf{fork}}] + 1/q$.

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):

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \le \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig},A}. \tag{23}$$

 doi:10.20944/preprints202603.2237.v1

$\mathbf{B}_3$ (Bidding Round 3: Signature Forgery Bid on B—Impersonating B to 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. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathrm{acc}^2/q_H + 1/q$ (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: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

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):

$$|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig},B}. \tag{24}$$

$\mathbf{B}_4$ (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 $\sum_k \Delta\mathrm{Price}_k$, 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: $\mathsf{Ask}(g_{\mathsf{mutual}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$. □

8.3. The Session-Key Secrecy Game

Definition 21 (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* $\mathsf{Ask}(g_{\mathsf{sk}}) \leq \mathsf{negl}(\lambda)$.

 doi:10.20944/preprints202603.2237.v1

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

$$\mathsf{Ask}(g_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda). \tag{25}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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).

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 ($\mathsf{sk} = (s_2 e_1 - s_1 e_2)(r(s_1 - s_2))^{-1}$); 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 $(m^*, (r^*, s^{**}))$.

 doi:10.20944/preprints202603.2237.v1

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 $q_H \cdot q_S \cdot \mathrm{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. 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 ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathrm{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), 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:

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

$\mathbf{B}_3$ (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 $1/q$.
Complexity: Bounded by $q_H \cdot q_S \cdot \mathrm{Adv}^{\mathsf{ECDLP}}(\lambda) + 1/q + \Pr[F_{\mathsf{fork}}]$. 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 ($q_S, q_H \leq 2^{64}$, $q \approx 2^{256}$, $\mathrm{Adv}^{\mathsf{ECDLP}} \leq 2^{-128}$), 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: $|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq \mathrm{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}$.

$\mathbf{B}_4$ (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 $\mathrm{Adv}^{\mathsf{MLWE}}(\lambda)$. 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 $2 \cdot \mathrm{Adv}^{\mathsf{MLWE}}(\lambda)$, 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: $|\Pr[\mathbf{B}_3 = 1] - \Pr[\mathbf{B}_4 = 1]| \leq \mathrm{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}$.

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

 doi:10.20944/preprints202603.2237.v1

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

8.4. The CNF Checking Game

Definition 22 (CNF Checking Good g_{CNF})**.** *The seller offers g_{CNF}: the guarantee that for every completed session, the session-CNF φ_i (Definition 12) 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)**.**

$$\mathsf{Ask}(g_{\mathsf{CNF}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}} + \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda). \tag{26}$$

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.
Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

$\mathcal{A}$ 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).

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 ($\mathsf{sk} = (s_2 e_1 - s_1 e_2)(r(s_1 - s_2))^{-1}$); in protocols it enables cross-session replay; in stream ciphers it reveals the XOR of two plaintexts. This

 doi:10.20944/preprints202603.2237.v1

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 $\Delta\text{Price} \leq q^2/2^{\lambda+1}$, 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.

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:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \Pr[F_{\text{fresh}}] \leq \frac{q_N^2}{2^{\lambda+1}}. \tag{27}$$

$\mathbf{B}_2$ (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 $\text{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, 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: $\Pr[F_{\text{fork}} \cup F_{\text{extract}}] \leq \Pr[F_{\text{fork}}] + 1/q$.

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 $F_{\text{sig-forge}}$:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \Pr[F_{\text{sig-forge}}] \leq \text{Adv}_{\text{Sig}}^{\text{EUF-CMA}}. \tag{28}$$

$\mathbf{B}_3$ (Bidding Round 3: MAC Forgery Bid—Bypassing the MAC 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 $\text{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, 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 $1/q$.

 doi:10.20944/preprints202603.2237.v1

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: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

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 $F_{\mathsf{mac\text{-}forge}}$:

$$|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq \Pr[F_{\mathsf{mac\text{-}forge}}] \leq \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}. \tag{29}$$

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

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

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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

 doi:10.20944/preprints202603.2237.v1

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 $\Delta\mathsf{Price} \le q^2/2^{\lambda+1}$, 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 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:

$$\Delta\mathsf{Price}_0 \le \Pr[F_{\mathsf{nonce}}] \le \frac{q_S^2}{2q}. \tag{30}$$

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: $\Delta\mathsf{Price}_1 \le q_H^2/(2q)$.

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

 doi:10.20944/preprints202603.2237.v1

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: $\Delta\text{Price} = 0$ 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.

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 $\Delta\text{Price}_2 = 0$: this is a costless market restructuring.

$\Delta\text{Price}_2 = 0$ (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. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathsf{acc}^2/q_H + 1/q$ (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: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

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: $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathsf{acc}^2/q_H + 1/q$ where acc is the original success probability.
- F_{extract}: $s^* = s^{**}$ (extraction yields $0/0$). $\Pr[F_{\mathsf{extract}}] \leq 1/q$.

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

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

 doi:10.20944/preprints202603.2237.v1

Total market cost:

$$\mathsf{Ask} \leq \underbrace{\frac{q_S^2}{2q}}_{\text{nonce bid}} + \underbrace{\frac{q_H^2}{2q}}_{\text{hash bid}} + \underbrace{0}_{\text{programming}} + \underbrace{q_H q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}}}_{\text{forgery bid}} + \mathsf{negl}(\lambda). \tag{31}$$

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

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

☞ **Side-Channel Bid**

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\mathsf{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\mathsf{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\mathsf{SCA}}(\lambda, d) \leq \left(\frac{q_{\mathsf{phys}}}{2^\lambda}\right)^{d+1}$$

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

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 (Definition 14) 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

 doi:10.20944/preprints202603.2237.v1

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 $\delta_{\mathsf{ping}} \le q_S^2/(2q) = \mathsf{negl}(\lambda)$. □

✻ Simple Terms: ECDSA Security Proof

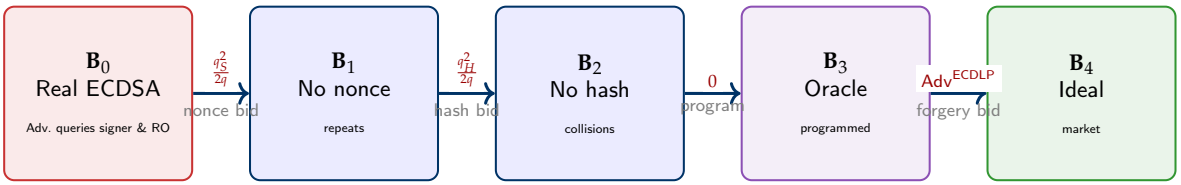
What ECDSA is: The Elliptic Curve Digital Signature Algorithm is the most widely used digital signature scheme today (used in Bitcoin, TLS, SSH, and many more). It produces compact signatures using the mathematics of elliptic curves over finite fields.

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.



$$\text{Total Ask} \le \frac{q_S^2}{2q} + \frac{q_H^2}{2q} + q_H q_S \cdot \mathsf{Adv}^{\mathsf{ECDLP}} + \mathsf{negl}(\lambda) \quad \textbf{Equilibrium}$$

Figure 12. 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

$\varphi_{\mathsf{ECDSA}} = (x_{\mathsf{Vrfy}(\mathsf{pk},m^*,(r^*,s^*))=1}) \wedge (x_{m^* \notin Q^{\mathsf{sign}}}) \wedge (x_{k^* \text{ fresh}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for ECDSA signature $(m^*, (r^*, s^*))$**:**

 doi:10.20944/preprints202603.2237.v1

Clause	Check	How	Pass?
$\varphi^{\mathsf{verify}}$: sig verifies	$\mathsf{Vrfy}(\mathsf{pk}, m^*, (r^*, s^*)) = 1$?	Compute $u_1G + u_2Q$; 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 $\mathsf{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1} \bmod q$—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). $\mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{MLWE}} + q_D/2^{\gamma} + \mathsf{negl}(\lambda)$.

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: Physical Leakage Attack.)

☞ Side-Channel Bid

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\mathsf{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\mathsf{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\mathsf{SCA}}(\lambda, d) \leq \left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}$$

 doi:10.20944/preprints202603.2237.v1

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

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 (Definition 14) 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:

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

 doi:10.20944/preprints202603.2237.v1

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.

✓ CNF Verification: ML-KEM Session-CNF Verification

$\varphi_{\text{ML-KEM}} = (x_{K=\mathsf{KEM.Dec}(\mathsf{sk},c)}) \wedge (x_{c\notin\mathcal{C}_{\text{used}}}) \wedge (x_{\text{implicit rejection consistent}}) \wedge (x_{\text{Ping passes}})$.

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	$c \notin \mathcal{C}_{\text{used}}$	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 $\mathcal{C}_{\text{prev}}$ log.	T/F
Result:	All T ⇒ ML-KEM session secure. Any F ⇒ Reject.		

➤ 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: $F_{\text{rej}} \cup F_{\text{bypass}}$ together give $\Delta\text{Price}_{\text{ping}} \leq q_D/2^{\gamma} = \text{negl}$. 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). $\mathsf{Ask}(g_{\text{EUF-CMA}}) \leq \mathsf{Adv}^{\text{MSIS}} + \mathsf{Adv}^{\text{MLWE}} + \mathsf{negl}(\lambda)$.

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 ($\Delta \leq \mathsf{Adv}^{\text{MLWE}}$). $\mathbf{B}_3$ **(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. $\Delta\text{Price} \leq \Pr[F_1 \cup F_2] + \mathsf{Adv}^{\text{MSIS}}$. Both $\Pr[F_1]$ and $\Pr[F_2]$ are negligible by parameter choice.

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

☞ Side-Channel Bid

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\text{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\text{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\text{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\text{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\text{SCA}}(\lambda, d) \leq \left(\frac{q_{\text{phys}}}{2^{\lambda}}\right)^{d+1}$$

 doi:10.20944/preprints202603.2237.v1

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

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 (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^*):

 doi:10.20944/preprints202603.2237.v1

Clause	Check	How	Pass?
φ^{verify}: sig verifies	$\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*) = 1$?	Run ML-DSA.Verify algorithm.	T/F
φ^{novel}: new message	$m^* \notin \mathcal{Q}^{\text{sign}}$	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 $m^* \in \mathcal{Q}^{\text{sign}}$. Clause φ^{novel} rejects this. For a genuinely fresh m^*, forging σ^* requires solving MSIS. Extended difference lemma: $F_{\text{msis}} \cup F_{\text{mlwe}}$ together give $\Delta\text{Price}_{\text{ping}} \leq \mathsf{Adv}^{\text{MSIS}} + \mathsf{Adv}^{\text{MLWE}} = \mathsf{negl}$. By Theorem 3, ML-DSA is EUF-CMA secure for unbounded signing sessions.

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)**.** $\mathsf{Ask}(g_{\text{EUF-CMA}}) = 1$.

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 $\Pr[F_{\text{homo}}] = 1$. Hence $\Delta\text{Price} = 1$, 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.

 doi:10.20944/preprints202603.2237.v1

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.

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

9.5. HMAC: SUF-CMA via Bidding

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 [18]. 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:*

$$\mathsf{Ask}(g_{\mathsf{SUF\text{-}CMA}}) \leq 2 \cdot \mathsf{Adv}_h^{\mathsf{PRF}} + \frac{q_T^2}{2^{n+1}} + \mathsf{negl}(\lambda), \tag{32}$$

where q_T is the number of tagging queries and n is the output length of H.

 doi:10.20944/preprints202603.2237.v1

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 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 $H((K \oplus \mathsf{ipad}) \| m)$ with a truly random function $R_{\mathsf{in}}(m)$. 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 $K \oplus \mathsf{ipad}$. 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:

$$|\Pr[\mathbf{B}_0(\mathcal{A}) = 1] - \Pr[\mathbf{B}_1(\mathcal{A}) = 1]| \leq \Pr[F_{\mathsf{in}}] \leq \mathsf{Adv}_h^{\mathsf{PRF}}. \tag{33}$$

$\mathbf{B}_2$ (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.

 doi:10.20944/preprints202603.2237.v1

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: $\Delta\mathrm{Price} \leq \mathsf{Adv}^{\mathsf{PRF}}(\lambda)$ 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 restructures the outer layer: replace $H((K \oplus \mathrm{opad}) \| y)$ (where $y = R_{\mathsf{in}}(m)$) with a truly random function $R_{\mathsf{out}}(y)$. Same argument as $\mathbf{B}_1$—any distinguishing buyer is placing an outer PRF bid, breaking PRF security of h keyed by $K \oplus \mathrm{opad}$. Price adjustment (outer PRF bid cost):

$$|\Pr[\mathbf{B}_1(\mathcal{A}) = 1] - \Pr[\mathbf{B}_2(\mathcal{A}) = 1]| \leq \mathsf{Adv}_h^{\mathsf{PRF}}. \tag{34}$$

$\mathbf{B}_3$ (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.

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:

$$|\Pr[\mathbf{B}_2(\mathcal{A}) = 1] - \Pr[\mathbf{B}_3(\mathcal{A}) = 1]| \leq \Pr[F_{\mathsf{coll}}] \leq \frac{q_T^2}{2^{n+1}}. \tag{35}$$

$\mathbf{B}_4$ (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

 doi:10.20944/preprints202603.2237.v1

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 $\mathbf{B}_3$, with no inner collision ($\neg F_{\text{coll}}$), R_{in} is injective on the queried set. The composed function $R_{\text{out}} \circ R_{\text{in}}$ is a random function on distinct inputs. Any forgery bid (m^*, τ^*) with $m^* \notin \{m_1, \ldots, m_{q_T}\}$ requires guessing $R_{\text{out}}(R_{\text{in}}(m^*))$—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* $g_{\text{SUF-CMA}}$ *(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), $R_{\text{out}}(R_{\text{in}}(m_i))$ is unique, so no second valid tag exists. Hence $\Pr[\mathbf{B}_4 = 1] = 0$ for this case.

Total:

$$\text{Ask}(g_{\text{SUF-CMA}}) \leq 2 \cdot \text{Adv}_h^{\text{PRF}} + \frac{q_T^2}{2^{n+1}} + 2^{-n}. \tag{36}$$

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

☞ **Side-Channel Bid**

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\text{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\text{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\text{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\text{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\text{Adv}^{\text{SCA}}(\lambda, d) \leq \left(\frac{q_{\text{phys}}}{2^\lambda}\right)^{d+1}$$

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\text{phys}} = \text{poly}(\lambda)$: $\text{Adv}^{\text{SCA}} = \text{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\text{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\text{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\text{phys}}$ physical traces." By Lemma 2, the side-channel event is independent of the mathematical failure events $F_{\text{nonce}}, F_{\text{hash}}, \ldots$ in the preceding rounds, so their joint probability satisfies $\Pr\left[\bigcup_k F_k \cup F_{\text{sca}}\right] \leq \sum_k \Pr[F_k] + \text{Adv}^{\text{SCA}}(\lambda, d)$. With order-$d$ masking, $\text{Adv}^{\text{SCA}} = \text{negl}(\lambda)$, preserving market equilibrium.

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 Session_i and 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 $\tau_{i+1} = R_{\text{out}}(R_{\text{in}}(m_{i+1}))$ is independent of all previous tags $\tau_1, \ldots, \tau_i$ whenever m_{i+1} is fresh (no inner

 doi:10.20944/preprints202603.2237.v1

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 $n = 256$.

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: Physical Leakage Attack.)

☞ **Side-Channel Bid**

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\mathsf{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\mathsf{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\mathsf{SCA}}(\lambda, d) \leq \left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}$$

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

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 $\mathcal{N}_{\mathsf{used}}$.

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.

 doi:10.20944/preprints202603.2237.v1

✓ 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 = $K \oplus \mathsf{ipad}$), 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.

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.

 doi:10.20944/preprints202603.2237.v1

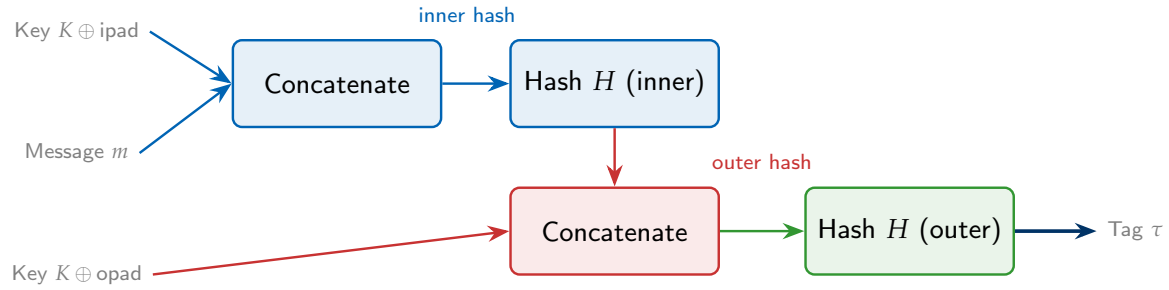


Figure 13. HMAC structure: two nested keyed hash evaluations. The inner hash mixes $K \oplus$ ipad with the message; the outer hash mixes $K \oplus$ opad with the inner hash output. Security relies on each keyed hash being a PRF.

9.6. AEAD: IND-CCA2 + INT-CTXT via Bidding

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 $\tau = \mathsf{MAC}_{K_m}(A \| N \| c)$.

Theorem 15 (AEAD Market Equilibrium). *For an Encrypt-then-MAC AEAD scheme:*

$$\mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}}) \leq \mathsf{Adv}_{E}^{\mathsf{IND\text{-}CPA}} + \mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{SUF\text{-}CMA}} + \mathsf{negl}(\lambda), \tag{37}$$

$$\mathsf{Ask}(g_{\mathsf{INT\text{-}CTXT}}) \leq \mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{SUF\text{-}CMA}} + \mathsf{negl}(\lambda). \tag{38}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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: 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 $1/q$.

 doi:10.20944/preprints202603.2237.v1

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: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

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:

$$\mathsf{Ask}(g_{\mathsf{INT\text{-}CTXT}}) = \Pr[\mathbf{B}_0(\mathcal{A}) = 1] \leq \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}. \tag{39}$$

Part 2: IND-CCA2. We prove confidentiality.

$\mathbf{B}_0$ (Bidding Round 0: Real AEAD Market—Plaintext 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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \Pr[F_{\mathsf{mac}}] \leq \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}. \tag{40}$$

 doi:10.20944/preprints202603.2237.v1

$\mathbf{B}_2$ (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. $\Delta\text{Price} = 0$. The residual adversarial advantage is then exactly $\mathsf{Adv}^{\text{IND-CPA}}_{\text{Enc}}(\lambda)$.

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:

$$|\Pr[\mathbf{B}_1 = 1] - 1/2| \leq \mathsf{Adv}^{\text{IND-CPA}}_{E}. \tag{41}$$

Total: $\mathsf{Ask}(g_{\text{IND-CCA2}}) \leq \mathsf{Adv}^{\text{IND-CPA}}_{E} + \mathsf{Adv}^{\text{SUF-CMA}}_{\text{MAC}}$.

✓ CNF Verification: AEAD Session-CNF Verification

$\varphi_{\text{AEAD}} = (x_{\mathsf{Mac.Vrfy}(K_m, A\|N\|c,\tau)=1}) \wedge (x_{N \notin \mathcal{N}_{\text{used}}}) \wedge (x_{\text{Ping passes}})$.

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	$N \notin \mathcal{N}_{\text{used}}$	Check nonce log; add N if pass.	T/F
φ^{ping}: unbounded	Distinct from previous session.	Compare to 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: $\Pr[F_{\text{mac}} \cup F_{\text{nonce}}] \leq \mathsf{Adv}^{\text{SUF-CMA}} + q_D/|\mathcal{N}|$.

➤ 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 $(N_i \in \mathcal{N}_{\text{used}})$. Ping bid price: $q_D/|\mathcal{N}| \leq q_D/2^{96} = \text{negl}$ 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: $\tau = \mathsf{MAC}_{K_m}(A\|N\|c)$.

 doi:10.20944/preprints202603.2237.v1

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.

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.

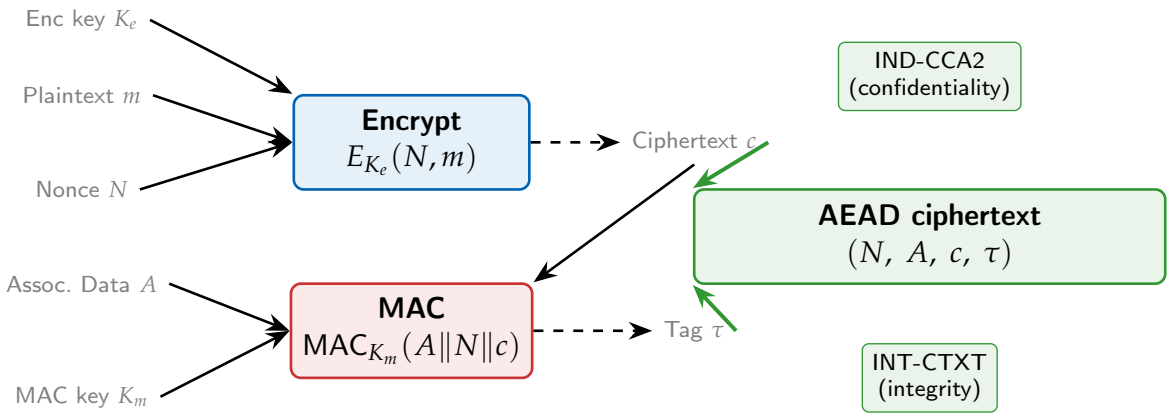


Figure 14. AEAD Encrypt-then-MAC structure. The plaintext is encrypted first with key K_e; then the MAC key K_m authenticates the associated data, nonce, and ciphertext together. The combined output provides both IND-CCA2 confidentiality and INT-CTXT integrity.

9.7. SLH-DSA (FIPS 205): Hash-Based EUF-CMA via Bidding

✻ Simple Terms: SLH-DSA (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.
- **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.

 doi:10.20944/preprints202603.2237.v1

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)**.**

$$\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) \leq q_S \cdot d \cdot \mathsf{Adv}_H^{\mathsf{SPR}} + q_S \cdot k \cdot \mathsf{Adv}_H^{\mathsf{TSPR}} + \mathsf{Adv}_F^{\mathsf{PRF}} + \mathsf{negl}(\lambda), \tag{42}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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

 doi:10.20944/preprints202603.2237.v1

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:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \mathrm{Adv}_F^{\mathrm{PRF}}. \tag{43}$$

$\mathbf{B}_2$ (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:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq q_S \cdot k \cdot \mathrm{Adv}_H^{\mathrm{TSPR}}. \tag{44}$$

$\mathbf{B}_3$ (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.
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.

 doi:10.20944/preprints202603.2237.v1

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:

$$|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq q_S \cdot d \cdot \mathsf{Adv}_H^{\mathsf{SPR}}. \tag{45}$$

$\mathbf{B}_4$ (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.

$\Delta\mathsf{Price}_4 = 0$ (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 $\sum_k \Delta\mathsf{Price}_k$, 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:

$$\mathsf{Ask} \leq \mathsf{Adv}_F^{\mathsf{PRF}} + q_S \cdot k \cdot \mathsf{Adv}_H^{\mathsf{TSPR}} + q_S \cdot d \cdot \mathsf{Adv}_H^{\mathsf{SPR}}. \quad \square \tag{46}$$

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

☞ Side-Channel Bid

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key

 doi:10.20944/preprints202603.2237.v1

material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\mathsf{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\mathsf{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\mathsf{SCA}}(\lambda, d) \leq \left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}$$

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

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	$R = F_{\mathsf{sk.prf}}(\mathsf{opt} \| m^*)$ 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 ⇒ SLH-DSA signature valid. Any F ⇒ 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.

 doi:10.20944/preprints202603.2237.v1

9.8. FN-DSA (FIPS 206): NTRU-Lattice EUF-CMA via Bidding

✻ 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)**.**

$$\mathsf{Ask}(g_{\mathsf{EUF\text{-}CMA}}) \leq \mathsf{Adv}^{\mathsf{SIS}}_{\mathsf{NTRU}} + \Delta_{\mathsf{DGS}} + \frac{q_H^2}{2^{n+1}} + \mathsf{negl}(\lambda), \tag{47}$$

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.*

Proof. We construct five games.

$\mathbf{B}_0$ (Bidding Round 0: Real FN-DSA Market—Lattice Forgery Bid).

 doi:10.20944/preprints202603.2237.v1

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.
Complexity: No price adjustment arises here: $\Delta\mathrm{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathrm{Adv}_{\mathcal{A}}^{\mathrm{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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}:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_H^2}{2^{n+1}}. \tag{48}$$

$\mathbf{B}_2$ (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: $\Delta\mathrm{Price} \leq \Delta_{\mathsf{DGS}}$—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).

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:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \Delta_{\mathsf{DGS}}, \tag{49}$$

where $\Delta_{\mathsf{DGS}} \leq 2^{-128}$ for recommended parameters.

 doi:10.20944/preprints202603.2237.v1

$\mathbf{B}_3$ (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 $H(m_i) = \alpha_i \cdot G + \beta_i \cdot \mathsf{pk}$ (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: $\Delta\mathsf{Price} = 0$ 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: $\Delta\mathsf{Price}_2 = 0$.

$\mathbf{B}_4$ (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 $\mathsf{sk} = (s^{**}H' - s^*H)(r(s^* - s^{**}))^{-1} \bmod q$, 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: $\Pr[F_{\mathsf{fork}} \cup F_{\mathsf{extract}}] \leq \Pr[F_{\mathsf{fork}}] + 1/q$.

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:

$$\Pr[\mathbf{B}_4(\mathcal{A}) = 1] \leq \mathsf{Adv}^{\mathsf{SIS}}_{\mathsf{NTRU}}. \tag{50}$$

Total:

$$\mathsf{Ask} \leq \frac{q_H^2}{2^{n+1}} + \Delta_{\mathsf{DGS}} + \mathsf{Adv}^{\mathsf{SIS}}_{\mathsf{NTRU}} + \mathsf{negl}(\lambda). \quad \square \tag{51}$$

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

☞ **Side-Channel Bid**

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key

 doi:10.20944/preprints202603.2237.v1

material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\mathsf{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\mathsf{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\mathsf{SCA}}(\lambda, d) \leq \left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}$$

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

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 $\delta_{\mathsf{ping}} \leq \Delta_{\mathsf{DGS}} + \mathsf{Adv}^{\mathsf{SIS}}_{\mathsf{NTRU}} = \mathsf{negl}$, 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 ⇒ FN-DSA forgery (lattice SIS solved). Any F ⇒ Not a valid forgery.		

 doi:10.20944/preprints202603.2237.v1

➤ 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: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq \Delta_{\mathsf{DGS}} + \mathsf{Adv}^{\mathsf{SIS}}_{\mathsf{NTRU}} = \mathsf{negl}$. 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

10. Case Study II: Block-Cipher Market—AES

10.1. AES Market Setup

The Advanced Encryption Standard [19] 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.

 doi:10.20944/preprints202603.2237.v1

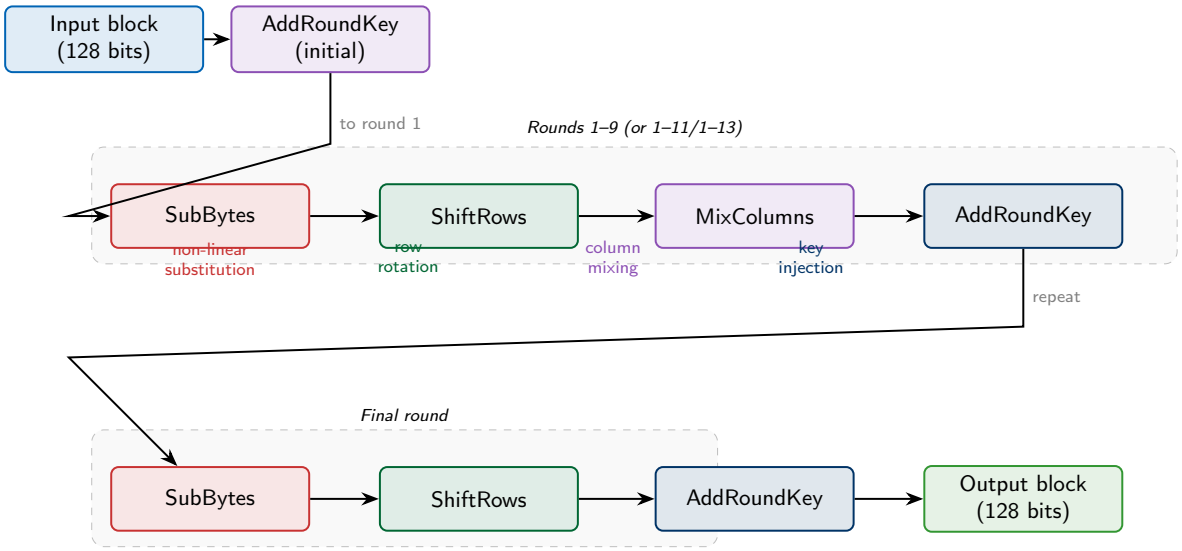


Figure 15. AES round structure. Each of the 10/12/14 rounds applies SubBytes (confusion), ShiftRows and MixColumns (diffusion), and AddRoundKey (key injection). The final round omits MixColumns. Multiple rounds compound the complexity, making any differential or algebraic attack computationally infeasible.

Definition 23 (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.

10.2. Differential Cryptanalysis Bid

Differential cryptanalysis [20] 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.

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).

 doi:10.20944/preprints202603.2237.v1

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 24 (Differential Characteristic Probability)**.** $\mathrm{DP}(\Omega) = \prod_{i=0}^{r-1} \Pr[\mathrm{Round}_i(\Delta_i) = \Delta_{i+1}]$.

Theorem 18 (AES Differential Bid Failure)**.** *For AES-128 with $r = 10$ rounds:*

$$\mathsf{Ask}(g_{\mathsf{PRP}}, \textit{differential bid}) \leq q_E \cdot 2^{-150} + \mathsf{negl}(\lambda), \tag{52}$$

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.
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.
Complexity: No price adjustment arises here: $\Delta\mathrm{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathrm{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 $\mathrm{Adv}^{\mathsf{MSIS}}(\lambda) + \mathrm{Adv}^{\mathsf{MLWE}}(\lambda) + \mathsf{negl}(\lambda)$. 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 [21]: the MixColumns and ShiftRows operations ensure that any 4-round characteristic activates at least $B_4 = 25$ S-boxes.

$\Delta\mathrm{Price}_0 = 0$ (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).

 doi:10.20944/preprints202603.2237.v1

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 $q_E \cdot \mathrm{DP}_{\max}^{B_r}$.

Each active S-box uses the AES S-box (inversion in $\mathrm{GF}(2^8)$ followed by an affine map). The maximum differential probability of the AES S-box is $\mathrm{DP}_{\max} = 4/256 = 2^{-6}$ [21].

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):

$$\mathrm{DP}(\Omega) \leq (2^{-6})^{25} = 2^{-150} \tag{53}$$

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:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq q_E \cdot 2^{-150} \leq 2^{128} \cdot 2^{-150} = 2^{-22}. \tag{54}$$

$\mathbf{B}_4$ (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

 doi:10.20944/preprints202603.2237.v1

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:

$$\mathsf{Ask}(g_{\mathsf{PRP}}, \text{diff}) \leq q_E \cdot 2^{-150}. \quad \square \tag{55}$$

10.3. Rotational Cryptanalysis Bid

Rotational cryptanalysis [22] exploits the potential for bitwise rotations to commute with cipher operations. For a rotation by r bits, denoted rot_r, the buyer checks whether $\mathsf{AES}_K(\text{rot}_r(x)) \approx \text{rot}_r(\mathsf{AES}_K(x))$.

✻ 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 $\mathsf{AES}_K(\text{rot}_r(x)) = \text{rot}_r(\mathsf{AES}_K(x))$ 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:*

$$\mathsf{Ask}(g_{\mathsf{PRP}}, \textit{rotational bid}) \leq q_E \cdot 2^{-128} + \mathsf{negl}(\lambda). \tag{56}$$

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.
Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\text{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

Buyer queries pairs $(x, \text{rot}_r(x))$ and checks whether outputs satisfy the rotational relationship.

$\mathbf{B}_1$ (Bidding Round 1: Key Schedule Asymmetry Bid—Rcon Destroys Rotational Symmetry).

 doi:10.20944/preprints202603.2237.v1

Purpose: Bound the *rotational cryptanalysis advantage*. A rotational distinguisher checks whether $E_K(\mathrm{rot}_r(x)) = \mathrm{rot}_r(E_K(x))$ 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 $\mathrm{Rcon}_i = x^{i-1}$ in $\mathrm{GF}(2^8)$ is distinct), and the S-box satisfies $S(\mathrm{rot}_r(x)) = \mathrm{rot}_r(S(x))$ 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: $\mathrm{rot}_r(K_i) = K'_i$ 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 $\mathrm{Rcon}[i]$ occupies only one byte position and its values $(01, 02, 04, 08, \ldots)$ break rotational symmetry:

$$\Pr[F_{\mathrm{Rcon}}] = 0 \quad \text{(deterministic: Rcon never preserves rotation).} \tag{57}$$

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 $E_K(\mathrm{rot}_r(x)) = \mathrm{rot}_r(E_K(x))$ 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 $\mathrm{Rcon}_i = x^{i-1}$ in $\mathrm{GF}(2^8)$ is distinct), and the S-box satisfies $S(\mathrm{rot}_r(x)) = \mathrm{rot}_r(S(x))$ 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.

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:

$$\Pr[\text{rotation preserved through all S-boxes}] \leq (2^{-8})^{16} = 2^{-128} \quad \text{per round.} \tag{58}$$

$\mathbf{B}_3$ (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.

 doi:10.20944/preprints202603.2237.v1

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, $\Pr[\text{rot}_r(\pi(x)) = \pi(\text{rot}_r(x))] = 1/(2^{128} - 1)$. The buyer's advantage: $q_E \cdot 2^{-128}$.

Total:

$$\text{Ask}(g_{\text{PRP}}, \text{rotational}) \leq q_E \cdot 2^{-128}. \quad \square \tag{59}$$

10.4. Related-Key Bid and Combined AES Equilibrium

Theorem 20 (AES Combined Market Equilibrium)**.** *Combining all known bid types (differential, linear, rotational, related-key, algebraic):*

$$\text{Ask}(g_{\text{PRP}}) \leq \max\left(q_E \cdot 2^{-150},\ q_E \cdot 2^{-128},\ q_E^2 \cdot 2^{-128}\right) + \text{negl}(\lambda) \leq \text{negl}(\lambda), \tag{60}$$

*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 [23]. The algebraic bid (using Gröbner basis or XL algorithms) has complexity exceeding 2^{128} for the full cipher. All bids fail for $q_E \leq 2^{64}$. $\square$

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)

 doi:10.20944/preprints202603.2237.v1

✓ CNF Verification: AES Block-Cipher Session-CNF Verification

For an AES encryption session (K, p, c), the session-CNF φ_{AES} captures PRP security:

$$\varphi_{\mathsf{AES}} = (x_{c=\mathsf{AES}_K(p)}) \wedge (x_{\text{no diff. pair detected}}) \wedge (x_{\text{no rotation exploit}}) \wedge (x_{\text{Ping passes}}).$$

Manual CNF worksheet for AES encryption session:

Clause	Check	How	Pass?
φ^{prp}: correct encryption	$c \stackrel{?}{=} \mathsf{AES}_K(p)$	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^{ping}: session fresh	(K, p) pair not reused from prior session?	Check session log.	T/F
Result:	All T ⇒ AES session secure. Any F ⇒ Block-cipher market collapsed.		

Extended difference lemma for AES: Three attacks (differential, linear, rotational) can be mounted simultaneously. By Lemma 2: $\Pr[F_{\mathsf{diff}} \cup F_{\mathsf{lin}} \cup F_{\mathsf{rot}}] \leq q_E \cdot 2^{-150} + q_E^2 \cdot 2^{-150} + q_E \cdot 2^{-128} = \mathsf{negl}$ 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 bid bound: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq \max(q_E \cdot 2^{-150}, q_E \cdot 2^{-128}) = \mathsf{negl}$ 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—Ultra-Lightweight Block Cipher

What PRESENT is: PRESENT [24] 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.

 doi:10.20944/preprints202603.2237.v1

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.

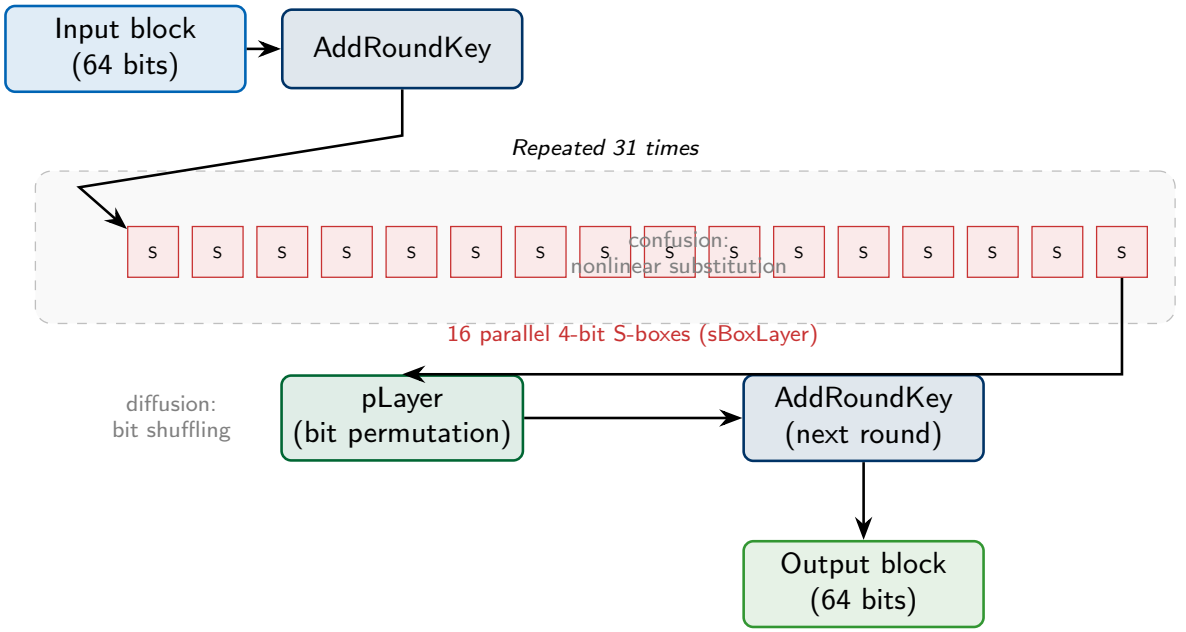


Figure 16. PRESENT round structure: a textbook Substitution–Permutation Network (SPN). Each of the 31 rounds applies AddRoundKey, 16 parallel 4-bit S-boxes (sBoxLayer), and a fixed bit permutation (pLayer). The simplicity of the design enables an extremely small hardware footprint (≈1,000 GE), making PRESENT suitable for RFID tags and IoT sensors.

Definition 25 (PRESENT PRP Security Good)**.**

$$\mathsf{Ask}(g_{\mathsf{PRP}}^{\mathsf{PRESENT}}) = \max_{\mathcal{A}\in\mathsf{PPT}} \left|\Pr\left[\mathcal{A}^{\mathsf{PRESENT}_K(\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{\pi(\cdot)} = 1\right]\right|$$
$$\textit{where } K \xleftarrow{\$} \{0,1\}^{80} \textit{ (or } \{0,1\}^{128}\textit{)}, \quad \pi \xleftarrow{\$} \mathsf{Perm}(\{0,1\}^{64}). \tag{61}$$

Theorem 21 (PRESENT Differential Bid Failure)**.** *For PRESENT-80 with $r = 31$ rounds:*

$$\mathsf{Ask}(g_{\mathsf{PRP}}^{\mathsf{PRESENT}}, \textit{differential bid}) \leq q_E \cdot 2^{-62} + \mathsf{negl}(\lambda), \tag{62}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

 doi:10.20944/preprints202603.2237.v1

Buyer queries PRESENT_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: 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 $\text{DP}_{\max}$ probability, giving the total characteristic probability $\text{DP}_{\max}^{B_r}$ as the upper bound.
Complexity: For AES-128: $B_4 = 25$ active S-boxes over 4 rounds, $\text{DP}_{\max} = 2^{-6}$, 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): $B_5 = 10$, $\text{DP}_{\max} = 2^{-2}$, 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 $q_E \cdot \text{DP}_{\max}^{B_r}$.

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 [24]). Each active 4-bit S-box has maximum differential probability $\text{DP}_{\max} = 2^{-2}$ (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:

$$\text{DP}(\Omega) \leq (2^{-2})^{10} = 2^{-20} \quad \text{per 5-round block,} \tag{63}$$

giving a full-cipher differential probability of at most $(2^{-2})^{60} = 2^{-120}$. Even the best truncated differentials [25] 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:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq 2^{32} \cdot 2^{-62} = 2^{-30}. \tag{64}$$

$\mathbf{B}_3$ (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

 doi:10.20944/preprints202603.2237.v1

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:

$$\text{Ask}(g_{\text{PRP}}^{\text{PRESENT}}, \text{diff}) \leq q_E \cdot 2^{-62}. \quad \square \tag{65}$$

Theorem 22 (PRESENT Linear Bid Failure)**.** *For PRESENT-80 with $r = 31$ rounds:*

$$\text{Ask}(g_{\text{PRP}}^{\text{PRESENT}}, \textit{linear bid}) \leq q_E^2 \cdot 2^{-62} + \text{negl}(\lambda). \tag{66}$$

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.
Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\text{Ask}_0 = \text{Adv}_{\mathcal{A}}^{\text{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 [25], the best known bias for full-round PRESENT is below 2^{-31}, requiring $q_E \geq 2^{62}$ known plaintexts.

Difference bound.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq q_E^2 \cdot 2^{-62}. \tag{67}$$

 doi:10.20944/preprints202603.2237.v1

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 23 (PRESENT Combined Market Equilibrium)**.** *Combining all known bid types (differential, linear, algebraic, related-key):*

$$\text{Ask}(g_{\text{PRP}}^{\text{PRESENT}}) \leq \max(q_E \cdot 2^{-62},\ q_E^2 \cdot 2^{-62},\ 2^{-16}) + \text{negl}(\lambda) \leq \text{negl}(\lambda), \tag{68}$$

*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

 doi:10.20944/preprints202603.2237.v1

✓ 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: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq \max(q_E \cdot 2^{-62}, q_E^2 \cdot 2^{-62}) = \mathsf{negl}$ 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 Conservative AES Finalist

What Serpent is: Serpent [26] 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.
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 [27]. 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.

 doi:10.20944/preprints202603.2237.v1

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.

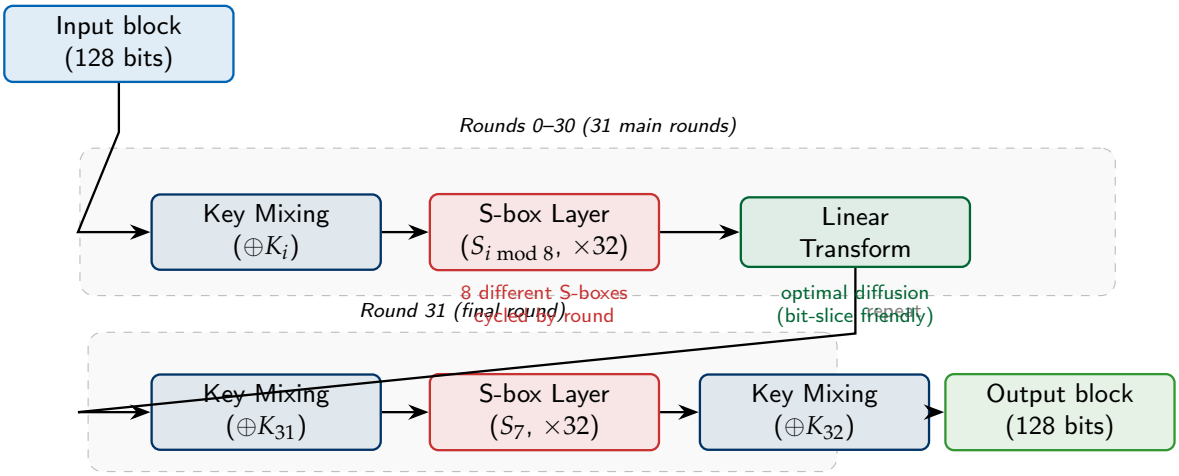


Figure 17. Serpent round structure. Each of the 32 rounds applies key mixing, a parallel 4-bit S-box layer (cycling through 8 distinct S-boxes), and a linear transformation for diffusion. The final round replaces the linear transformation with an extra key mixing. The 32-round design provides the largest security margin of any well-studied 128-bit block cipher: the best known attack covers only 12 rounds, leaving a 20-round margin.

Definition 26 (Serpent PRP Security Good)**.**

$$\mathsf{Ask}(g_{\mathsf{PRP}}^{\mathsf{Serpent}}) = \max_{\mathcal{A}\in\mathsf{PPT}} \left|\Pr\left[\mathcal{A}^{\mathsf{Serpent}_K(\cdot)}=1\right] - \Pr\left[\mathcal{A}^{\pi(\cdot)}=1\right]\right|$$
$$\textit{where } K \xleftarrow{\$} \{0,1\}^\kappa,\ \ \kappa\in\{128,192,256\},\ \ \pi \xleftarrow{\$} \mathsf{Perm}(\{0,1\}^{128}). \tag{69}$$

Theorem 24 (Serpent Differential Bid Failure)**.** *For Serpent-128 with $r = 32$ rounds:*

$$\mathsf{Ask}(g_{\mathsf{PRP}}^{\mathsf{Serpent}}, \textit{differential bid}) \le q_E \cdot 2^{-196} + \mathsf{negl}(\lambda). \tag{70}$$

Proof. We construct four games.

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 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 $\le 2^{-150}$. The adversary needs $q_E \ge 2^{150}$ 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 $\mathsf{Serpent}_K$ on chosen-plaintext pairs $(x, x \oplus \Delta_0)$ and checks the output difference.

B_1 (Bidding Round 1: Active S-Box Counting Bid—Conservative Diffusion Design).

 doi:10.20944/preprints202603.2237.v1

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:

$$\mathrm{DP}(\Omega) \leq (2^{-2})^{200/4\times 4} = (2^{-2})^{200} = 2^{-400} \tag{71}$$

for any single-path differential. Even the best truncated differential attacks on Serpent [27] 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.

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

 doi:10.20944/preprints202603.2237.v1

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:

$$\text{Ask}(g_{\text{PRP}}^{\text{Serpent}}, \text{diff}) \leq q_E \cdot 2^{-196}. \quad \square \tag{72}$$

Theorem 25 (Serpent Linear Bid Failure). *For Serpent-128 with $r = 32$ rounds:*

$$\text{Ask}(g_{\text{PRP}}^{\text{Serpent}}, \textit{linear bid}) \leq q_E^2 \cdot 2^{-196} + \text{negl}(\lambda). \tag{73}$$

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.
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 [27] 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).

 doi:10.20944/preprints202603.2237.v1

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 26 (Serpent Combined Market Equilibrium)**.** *Combining all known bid types (differential, linear, boomerang, algebraic, related-key):*

$$\text{Ask}(g_{\text{PRP}}^{\text{Serpent}}) \leq \max\left(q_E \cdot 2^{-196},\ q_E^2 \cdot 2^{-196},\ q_E^2 \cdot 2^{-128}\right) + \text{negl}(\lambda) \leq \text{negl}(\lambda), \tag{74}$$

*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. □

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.

 doi:10.20944/preprints202603.2237.v1

✓ 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 ⇒ Serpent session secure. Any F ⇒ 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.

11. Case Study III: Hash-Function Market—Keccak/SHA-3

✻ 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:

- **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:

 doi:10.20944/preprints202603.2237.v1

- 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 [28] 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.

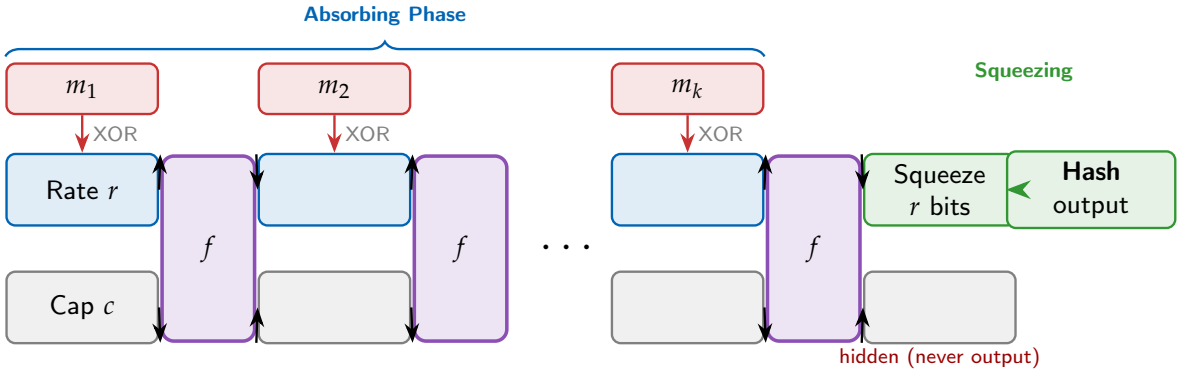


Figure 18. Keccak sponge construction for SHA-3. Message blocks $m_1, \ldots, m_k$ are XORed into the rate (r-bit) portion of the 1600-bit state, interleaved with permutation f (24 rounds of $\theta, \rho, \pi, \chi, \iota$). After absorbing, the rate portion is squeezed as the hash output. The capacity (c-bit) portion is never exposed, enforcing collision resistance ($2^{c/2}$) and preimage resistance (2^c). For SHA3-256: $c = 512$.

Theorem 27 (Keccak Collision Resistance Equilibrium)**.** *In the random sponge model with capacity c:*

$$\mathsf{Ask}(g_{\mathsf{CR}}) \leq \frac{q_f^2}{2^{c+1}} + \mathsf{negl}(\lambda), \tag{75}$$

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.

 doi:10.20944/preprints202603.2237.v1

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 (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 [29], 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:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f^2}{2^{c+1}}. \tag{76}$$

$\mathbf{B}_2$ (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:

$$\mathsf{DP}(\Omega_f) \leq (2^{-2})^{24} = 2^{-48} \tag{77}$$

 doi:10.20944/preprints202603.2237.v1

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 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 (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:

$$\mathsf{Ask}(g_{\mathsf{CR}}) \leq \frac{q_f^2}{2^{c+1}} + \mathsf{negl}(\lambda). \quad \square \tag{78}$$

Theorem 28 (Keccak Preimage Resistance Equilibrium)**.**

$$\mathsf{Ask}(g_{\mathsf{Pre}}) \leq \frac{q_f}{2^c} + \mathsf{negl}(\lambda). \tag{79}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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

 doi:10.20944/preprints202603.2237.v1

the information-theoretic birthday bound on the capacity bits. The indifferentiability theorem [29] 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:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f}{2^c}. \tag{80}$$

$\mathbf{B}_2$ (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 29 (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.

Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\text{Ask}_0 = \text{Adv}_{\mathcal{A}}^{\text{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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).

 doi:10.20944/preprints202603.2237.v1

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.

Difference bound. Each query to f^{-1} tests one candidate capacity value:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f}{2^c}. \tag{81}$$

$\mathbf{B}_2$ (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: $\text{Ask}(g_{\text{LE}}) \leq q_f/2^c$. □

✓ 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.

 doi:10.20944/preprints202603.2237.v1

➤ 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—A Modern High-Speed Hash Function

What BLAKE3 is: BLAKE3 [30] is a cryptographic hash function released in 2020, designed as the successor to BLAKE2 [31]. 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.

 doi:10.20944/preprints202603.2237.v1

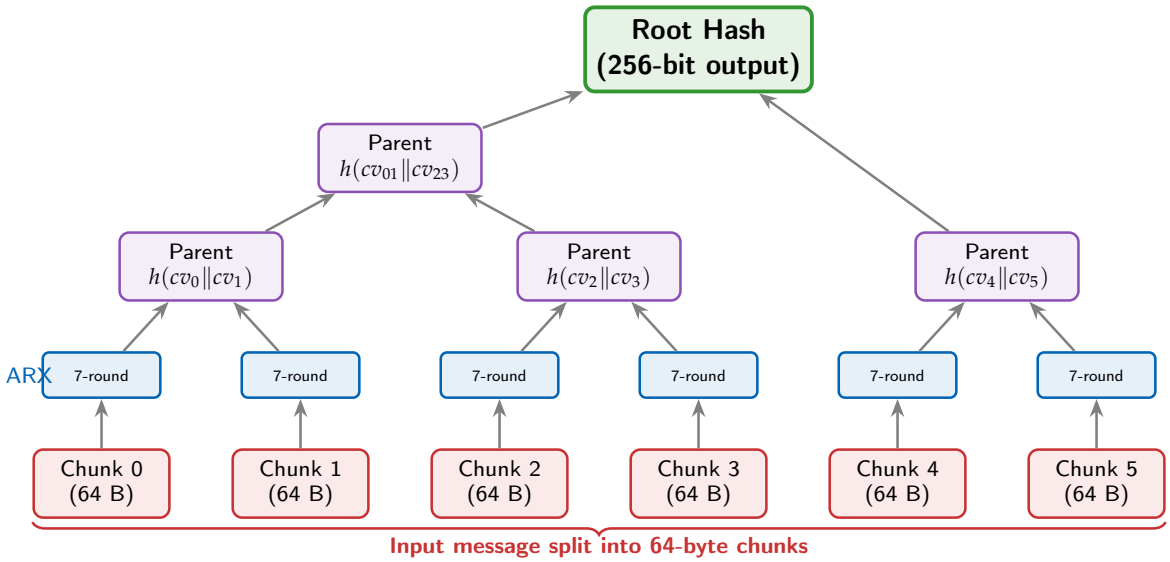


Figure 19. BLAKE3 Merkle tree hash structure. Input is split into 64-byte chunks, each compressed by a 7-round ARX compression function. Chaining values are combined pairwise in a binary Merkle tree. The root node produces the 256-bit hash output. The tree structure provides inherent parallelism and structural length-extension resistance: appending data changes the tree structure, not just the final state.

Definition 27 (BLAKE3 Hash Security Goods)**.** *The BLAKE3 market offers three goods:*

- $g^{\mathsf{B3}}_{\mathsf{CR}}$*: Collision resistance.* $\mathsf{Ask}(g^{\mathsf{B3}}_{\mathsf{CR}}) = \max_{\mathcal{A}} \Pr[\mathcal{A}\ \textit{finds}\ m \neq m' : \mathsf{BLAKE3}(m) = \mathsf{BLAKE3}(m')]$.
- $g^{\mathsf{B3}}_{\mathsf{Pre}}$*: Preimage resistance.* $\mathsf{Ask}(g^{\mathsf{B3}}_{\mathsf{Pre}}) = \max_{\mathcal{A}} \Pr[\mathcal{A}\ \textit{finds}\ m : \mathsf{BLAKE3}(m) = y]$ *for random* y.
- $g^{\mathsf{B3}}_{\mathsf{LE}}$*: Length-extension resistance.*

$$\mathsf{Ask}(g^{\mathsf{B3}}_{\mathsf{LE}}) = \max_{\mathcal{A}} \Pr\big[\mathcal{A}\ \textit{computes}\ \mathsf{BLAKE3}(m\|m')\ \textit{from}\ \mathsf{BLAKE3}(m)\big].$$

Theorem 30 (BLAKE3 Collision Resistance Equilibrium)**.** *In the ideal compression function model:*

$$\mathsf{Ask}(g^{\mathsf{B3}}_{\mathsf{CR}}) \leq \frac{q_f^2}{2^{257}} + \mathsf{negl}(\lambda), \tag{82}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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).

 doi:10.20944/preprints202603.2237.v1

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 [29] 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*).

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.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq 0 \quad \text{(structural reduction, no loss).} \tag{83}$$

$\mathbf{B}_2$ (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).

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \frac{q_f^2}{2^{257}}. \tag{84}$$

$\mathbf{B}_3$ (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

 doi:10.20944/preprints202603.2237.v1

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:

$$\mathsf{Ask}(g_{\mathsf{CR}}^{\mathsf{B3}}) \leq \frac{q_f^2}{2^{257}} + \mathsf{negl}(\lambda). \quad \square \tag{85}$$

Theorem 31 (BLAKE3 Preimage Resistance Equilibrium)**.**

$$\mathsf{Ask}(g_{\mathsf{Pre}}^{\mathsf{B3}}) \leq \frac{q_f}{2^{256}} + \mathsf{negl}(\lambda). \tag{86}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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.

 doi:10.20944/preprints202603.2237.v1

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}.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f}{2^{256}}. \tag{87}$$

$\mathbf{B}_2$ (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.
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: $\text{Ask}(g^{\text{B3}}_{\text{Pre}}) \leq q_f/2^{256}$. $\square$

Theorem 32 (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*:

$$\text{Ask}(g^{\text{B3}}_{\text{LE}}) = 0. \quad \square \tag{88}$$

✓ 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 $\Rightarrow$ BLAKE3 secure. Any F $\Rightarrow$ 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.

 doi:10.20944/preprints202603.2237.v1

➤ Ping Bid: BLAKE3 Unbounded Ping Bid

The buyer's *incremental update bid*: given $\mathsf{BLAKE3}(m)$, attempt to compute $\mathsf{BLAKE3}(m')$ 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: $q_f/2^{256} = \mathsf{negl}$. BLAKE3 is hash-secure for unbounded sessions.

Table 9. BLAKE3 hash-function market summary.

Good	Ask Bound	Practical Value	Status
Collision ($g_{\mathsf{CR}}^{\mathsf{B3}}$)	$q_f^2/2^{257}$	$\leq 2^{-129}$ for $q_f \leq 2^{64}$	Equilibrium
Preimage ($g_{\mathsf{Pre}}^{\mathsf{B3}}$)	$q_f/2^{256}$	$\leq 2^{-192}$ for $q_f \leq 2^{64}$	Equilibrium
Length-extension ($g_{\mathsf{LE}}^{\mathsf{B3}}$)	0	0 (structural)	Equilibrium

11.2. ASCON-Hash: NIST Lightweight Hash-Function Market

✻ Simple Terms: ASCON-Hash—NIST's Lightweight Hash Standard

What ASCON-Hash is: ASCON-Hash is the hash function component of the ASCON family [32], which was selected by NIST as the standard for lightweight cryptography in 2023 [33]. 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.

 doi:10.20944/preprints202603.2237.v1

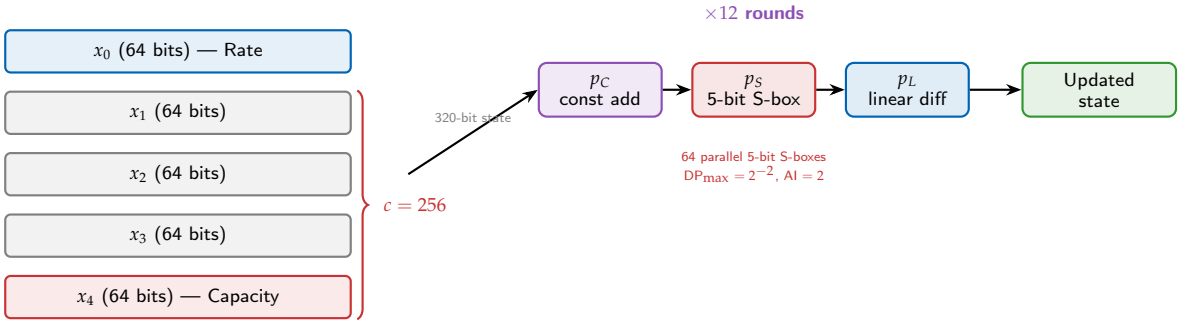


Figure 20. ASCON permutation round structure. The 320-bit state (5×64-bit words) is processed by constant addition (p_C), a parallel 5-bit S-box substitution layer (p_S, 64 S-boxes), and linear diffusion (p_L, word-level rotations and XOR). ASCON-Hash applies 12 rounds per absorption step.

Definition 28 (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 33 (ASCON-Hash Collision Resistance Equilibrium)**.** *In the random sponge model with capacity* $c = 256$*:*

$$\mathsf{Ask}(g_{\mathsf{CR}}^{\mathsf{Ascon}}) \leq \frac{q_f^2}{2^{c+1}} + \mathsf{negl}(\lambda) = \frac{q_f^2}{2^{257}} + \mathsf{negl}(\lambda). \tag{89}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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.
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),

 doi:10.20944/preprints202603.2237.v1

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 [29] (the sponge indifferentiability theorem applies to any sponge with a random permutation).

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f^2}{2^{c+1}} = \frac{q_f^2}{2^{257}}. \tag{90}$$

$\mathbf{B}_2$ (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 [32]). Hence:

$$\mathrm{DP}(\Omega_p) \leq (2^{-2})^{48} = 2^{-96} \tag{91}$$

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. $\Pr[\mathbf{B}_3 = 1] = q_f^2/2^{c+1}$.

Total:

$$\mathsf{Ask}(g_{\mathsf{CR}}^{\mathsf{Ascon}}) \leq \frac{q_f^2}{2^{257}} + \mathsf{negl}(\lambda). \quad \square \tag{92}$$

 doi:10.20944/preprints202603.2237.v1

Theorem 34 (ASCON-Hash Preimage Resistance Equilibrium)**.**

$$\mathsf{Ask}(g_{\mathsf{Pre}}^{\mathsf{Ascon}}) \leq \frac{q_f}{2^c} + \mathsf{negl}(\lambda) = \frac{q_f}{2^{256}} + \mathsf{negl}(\lambda). \tag{93}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 [29] 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 $2^{-c} = 2^{-256}$.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \frac{q_f}{2^{256}}. \tag{94}$$

$\mathbf{B}_2$ (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 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.

 doi:10.20944/preprints202603.2237.v1

Random sponge; preimage resistance is $\min(2^n, 2^c) = \min(2^{256}, 2^{256}) = 2^{256}$.

Total: $\mathsf{Ask}(g^{\mathsf{Ascon}}_{\mathsf{Pre}}) \leq q_f/2^{256}$. □

Theorem 35 (ASCON-Hash Length-Extension Resistance). $\mathsf{Ask}(g^{\mathsf{Ascon}}_{\mathsf{LE}}) \leq q_f/2^{256} + \mathsf{negl}(\lambda)$.

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

$\varphi_{\mathsf{Ascon}} = (x_{H(m)\neq H(m')\ \forall m'\neq m}) \wedge (x_{\nexists m': H(m')=y}) \wedge (x_{\text{no LE}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for an ASCON-Hash output $y = \mathsf{AsconHash}(m)$:

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: $q_f/2^{256} = \mathsf{negl}$. 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 ($g^{\mathsf{Ascon}}_{\mathsf{CR}}$)	$q_f^2/2^{c+1}$	$q_f^2/2^{257}$	Equilibrium
Preimage ($g^{\mathsf{Ascon}}_{\mathsf{Pre}}$)	$q_f/2^c$	$q_f/2^{256}$	Equilibrium
Length-extension ($g^{\mathsf{Ascon}}_{\mathsf{LE}}$)	$q_f/2^c$	$q_f/2^{256}$	Equilibrium

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.)

 doi:10.20944/preprints202603.2237.v1

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.
- **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.

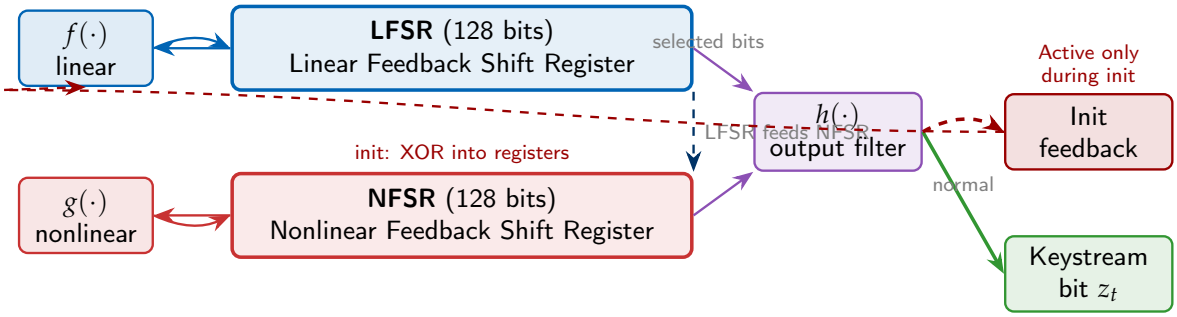


Figure 21. Grain-128a stream cipher structure. The 128-bit LFSR and 128-bit NFSR form a 256-bit state. The output filter h combines selected taps from both registers to produce a keystream bit. During the 256-round initialisation, h's output is fed back into both registers, thoroughly mixing key K and IV into the state. Normal keystream generation uses only the forward path.

12.1. Grain-128a Market Setup

Grain-128a [34] 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.

 doi:10.20944/preprints202603.2237.v1

Definition 29 (Stream Cipher PRG Good)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}) = \max_{\mathcal{A}\in\mathsf{PPT}} |\Pr[\mathcal{A}(\mathsf{Grain}(K,\mathsf{IV})) = 1] - \Pr[\mathcal{A}(R) = 1]|$$
$$\textit{where } R \xleftarrow{\$} \{0,1\}^L, \quad L \textit{ is the keystream length.} \tag{95}$$

12.2. State Recovery Attack Bid

Theorem 36 (Grain-128a State Recovery Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}, \textit{state recovery bid}) \le L \cdot 2^{-256} + \mathsf{negl}(\lambda), \tag{96}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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.
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 $|\epsilon_L| \le 2^{k-1} \cdot \epsilon_{\max}^k$. 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 $|\epsilon_L| \le 2^{24} \cdot 2^{-75} = 2^{-51}$ and requiring $q_E \ge 2^{102}$ known plaintexts—infeasible. For PRESENT: best bias $\approx 2^{-31}$ requiring $q_E \ge 2^{62}$ 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 [35] 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.

 doi:10.20944/preprints202603.2237.v1

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.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot 2^{-256}. \tag{97}$$

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.

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \frac{L^2 \cdot 2^{-96}}{2^{128}} \leq \mathsf{negl}(\lambda). \tag{98}$$

$\mathbf{B}_3$ (Bidding Round 3: Ideal Grain-128a 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 $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Random keystream; state recovery gives no advantage: $\Pr[\mathbf{B}_3 = 1] = 1/2$.

Total: $\mathsf{Ask}(\text{state recovery}) \leq L \cdot 2^{-256} + \mathsf{negl}(\lambda)$.

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

 doi:10.20944/preprints202603.2237.v1

☞ **Side-Channel Bid**

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\mathsf{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\mathsf{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\mathsf{SCA}}(\lambda, d) \leq \left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}$$

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

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

Theorem 37 (Grain-128a Key Recovery Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}, \textit{key recovery bid}) \leq 2^{-128+\log_2(q_{\mathsf{IV}})} + \mathsf{negl}(\lambda), \tag{99}$$

where q_{IV} is the number of distinct IVs observed.

Proof. We construct four games.

$\mathbf{B}_0$ (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.

 doi:10.20944/preprints202603.2237.v1

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.
Complexity: No price adjustment arises here: $\Delta\mathrm{Price}_{-1} = 0$. The ask price $\mathrm{Ask}_0 = \mathrm{Adv}^{\mathrm{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 $|\epsilon_L| \le 2^{k-1} \cdot \epsilon_{\max}^k$. 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 $|\epsilon_L| \le 2^{24} \cdot 2^{-75} = 2^{-51}$ and requiring $q_E \ge 2^{102}$ known plaintexts—infeasible. For PRESENT: best bias $\approx 2^{-31}$ requiring $q_E \ge 2^{62}$ 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).

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \le 2^{-128}. \tag{100}$$

$\mathbf{B}_2$ (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.

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}:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \le q_{\mathsf{IV}} \cdot 2^{-128}. \tag{101}$$

$\mathbf{B}_3$ (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

 doi:10.20944/preprints202603.2237.v1

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: $\Delta\text{Price} \leq \mathsf{Adv}^{\mathsf{CDH}}_{g}(\lambda)$ (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.

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

Theorem 38 (Grain-128a Distinguishing Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}, \textit{distinguishing bid}) \leq L^2 \cdot 2^{-97} + \mathsf{negl}(\lambda). \tag{102}$$

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.
Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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.

 doi:10.20944/preprints202603.2237.v1

Complexity: $\Delta\text{Price} \leq \Delta_{\text{DGS}}$—the Rényi divergence between fast and exact samplers. For FN-DSA (FIPS 206) at the 128-bit security level, the NTRU sampler achieves $\Delta_{\text{DGS}} \leq 2^{-128}$. 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 $q_S \cdot \Delta_{\text{DGS}}$, 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 $\epsilon \leq 2^{-48.5}$ [36]. Detecting a bias ϵ with confidence requires $L \geq 1/\epsilon^2 = 2^{97}$ keystream bits. For $L < 2^{97}$:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot \epsilon \leq L \cdot 2^{-48.5}. \tag{103}$$

More precisely, the advantage of the optimal linear distinguisher is:

$$\text{Adv}_{\text{dist}} \leq \sqrt{L} \cdot \epsilon \leq L^2 \cdot 2^{-97} \quad \text{(using Piling-up lemma)}. \tag{104}$$

$\mathbf{B}_2$ (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.
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

Theorem 39 (Grain-128a TMTO Bid Failure)**.** *For any time-memory-data trade-off with parameters T (time), M (memory), D (data):*

$$\text{Ask}(g_{\text{PRG}}, \textit{TMTO bid}) \leq \frac{T \cdot M \cdot D}{2^{256}} + \text{negl}(\lambda), \tag{105}$$

subject to the constraint $T \cdot M^2 \cdot D^2 \leq 2^{512}$ (Babbage-Golić curve [37]).

Proof. We construct three games.

$\mathbf{B}_0$ (Bidding Round 0: Real Grain-128a Market—TMTO Precomputation Bid).

 doi:10.20944/preprints202603.2237.v1

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}^{\mathsf{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 [38]: 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:

$$\Pr[\text{TMTO success}] \leq D \cdot T/2^{256}. \tag{106}$$

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: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. 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 $\sum_k \Delta\mathsf{Price}_k$, 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 Session_{i+1} using a fresh IV, the buyer obtains one additional data

 doi:10.20944/preprints202603.2237.v1

point D_{i+1}. The total data across $i+1$ sessions is $D_{\text{total}} = i+1$. The TMTO curve $T \cdot M^2 \cdot D_{\text{total}}^2 \leq 2^{512}$ 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

$\varphi_{\text{Grain}} = (x_{\text{IV} \notin \mathcal{IV}_{\text{used}}}) \wedge (x_{\text{state not recovered}}) \wedge (x_{\text{keystream indist.}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for Grain-128a keystream session (K, IV):

Clause	Check	How	Pass?
φ^{iv}: IV fresh	$\text{IV} \notin \mathcal{IV}_{\text{used}}$?	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 $\mathcal{IV}_{\text{used}}$ 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 $F_{\text{dist}} \cup F_{\text{sr}}$ caused by IV reuse: $\Delta\text{Price}_{\text{ping}} \leq \Pr[F_{\text{iv}}] = 0$ (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	$q_{\text{IV}} \cdot 2^{-128}$	$\leq 2^{-64}$	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 Modern High-Speed Stream Cipher

What ChaCha20 is: ChaCha20 [39] is a stream cipher designed by Daniel J. Bernstein as a refinement of Salsa20 [40]. 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

 doi:10.20944/preprints202603.2237.v1

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.

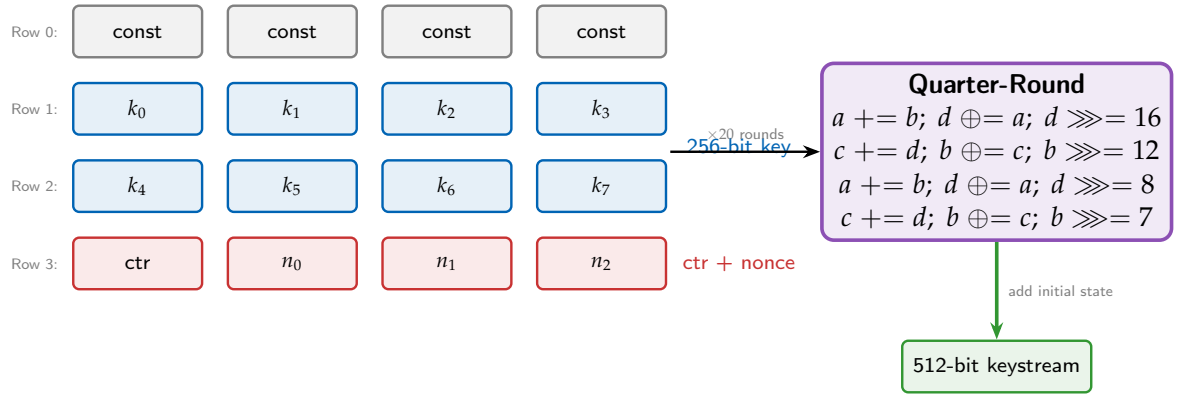


Figure 22. ChaCha20 state matrix and quarter-round operation. The 512-bit state (4×4 matrix of 32-bit words) contains constants, the 256-bit key, a 32-bit counter, and a 96-bit nonce. Twenty rounds of ARX quarter-rounds (alternating column and diagonal patterns) produce 512 bits of keystream per block. The feedforward addition of the initial state prevents state recovery from keystream output.

Definition 30 (ChaCha20 PRG Security Good)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{ChaCha}}) = \max_{\mathcal{A} \in \mathsf{PPT}} |\Pr[\mathcal{A}(\mathsf{ChaCha20}(K, \mathsf{nonce}, \mathsf{ctr})) = 1] - \Pr[\mathcal{A}(R) = 1]|$$
$$\textit{where } R \xleftarrow{\$} \{0,1\}^L, \quad L \textit{ is the keystream length.} \tag{107}$$

Theorem 40 (ChaCha20 State Recovery Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{ChaCha}}, \textit{state recovery bid}) \leq L \cdot 2^{-256} + \mathsf{negl}(\lambda). \tag{108}$$

Proof. We construct four games.

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,

 doi:10.20944/preprints202603.2237.v1

tagging, etc.) and then outputs a winning response. The market ask price Ask_0 is the adversary's raw advantage in this strongest setting.
Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\text{Ask}_0 = \text{Adv}^{\text{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 $\text{output} = \text{state}_0 + \text{state}_r$ (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 $\text{state}_{\text{final}} = \text{ChaCha20\_block}(S_0) + S_0$, where S_0 is the initial state and $+$ denotes word-wise addition mod 2^{32}. The buyer observes 512 bits of $\text{state}_{\text{final}} + S_0$ but does not know S_0. To recover S_0, the buyer must solve $\text{state}_{\text{final}} = F(S_0) + S_0$ 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 $G(S_0) = \text{output} - F(S_0)$, 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).

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot 2^{-256}. \tag{109}$$

$\mathbf{B}_2$ (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 $\text{DP}_{\max}$ probability, giving the total characteristic probability $\text{DP}_{\max}^{B_r}$ as the upper bound.
Complexity: For AES-128: $B_4 = 25$ active S-boxes over 4 rounds, $\text{DP}_{\max} = 2^{-6}$, 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): $B_5 = 10$, $\text{DP}_{\max} = 2^{-2}$, 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 $q_E \cdot \text{DP}_{\max}^{B_r}$.

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 $\text{DP}_{\max} = 1$ 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.

 doi:10.20944/preprints202603.2237.v1

Difference bound. The best known differential-linear attack on ChaCha reaches 7 rounds with 2^{218} complexity [39]. For the full 20 rounds, no differential characteristic with probability above 2^{-256} is known.

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \mathsf{negl}(\lambda). \tag{110}$$

$\mathbf{B}_3$ (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 $\sum_k \Delta\text{Price}_k$, all of which are negligible, establishing market equilibrium.

Random keystream; no state to recover. $\Pr[\mathbf{B}_3 = 1] = 1/2$.

Total: $\mathsf{Ask}(\text{state recovery}) \leq L \cdot 2^{-256} + \mathsf{negl}(\lambda)$. □

Theorem 41 (ChaCha20 Distinguishing Bid Failure)**.**

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{ChaCha}}, \textit{distinguishing bid}) \leq L \cdot 2^{-256} + \mathsf{negl}(\lambda). \tag{111}$$

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 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.
Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

Buyer receives either $\mathsf{ChaCha20}(K, \text{nonce})$ 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 $R_2(\sigma_{\text{real}} \| \sigma_{\text{ideal}}) = 1 + \Delta_{\mathsf{DGS}}$ 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: $\Delta\text{Price} \leq \Delta_{\mathsf{DGS}}$—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

 doi:10.20944/preprints202603.2237.v1

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}.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot 2^{-256}. \tag{112}$$

$\mathbf{B}_2$ (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: $\Pr[\mathbf{B}_{\text{ideal}} = 1] \in \{0, 1/2\}$. 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: $\mathsf{Ask}(\text{distinguishing}) \leq L \cdot 2^{-256}$. □

Theorem 42 (ChaCha20 TMTO Bid Failure)**.** *For any time-memory-data trade-off with parameters T, M, D:*

$$\mathsf{Ask}(g_{\mathsf{PRG}}^{\mathsf{ChaCha}}, \textit{TMTO bid}) \leq \frac{T \cdot D}{2^{256}} + \mathsf{negl}(\lambda), \tag{113}$$

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.
Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

Buyer precomputes a table mapping keystream prefixes to keys and inverts using time T.

 doi:10.20944/preprints202603.2237.v1

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:

$$\Pr[\text{TMTO success}] \leq D \cdot T/2^{256}. \tag{114}$$

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).

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.
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 43 (ChaCha20 Combined Market Equilibrium)**.** *Combining all known bid types:*

$$\text{Ask}(g_{\text{PRG}}^{\text{ChaCha}}) \leq \max\left(L \cdot 2^{-256},\ TD/2^{256}\right) + \text{negl}(\lambda) \leq \text{negl}(\lambda), \tag{115}$$

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)**:**

 doi:10.20944/preprints202603.2237.v1

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 $\Rightarrow$ Keystream secure. Any F $\Rightarrow$ 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.

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 Elegant Minimalist Stream Cipher

What Trivium is: Trivium [41] 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.

 doi:10.20944/preprints202603.2237.v1

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 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.

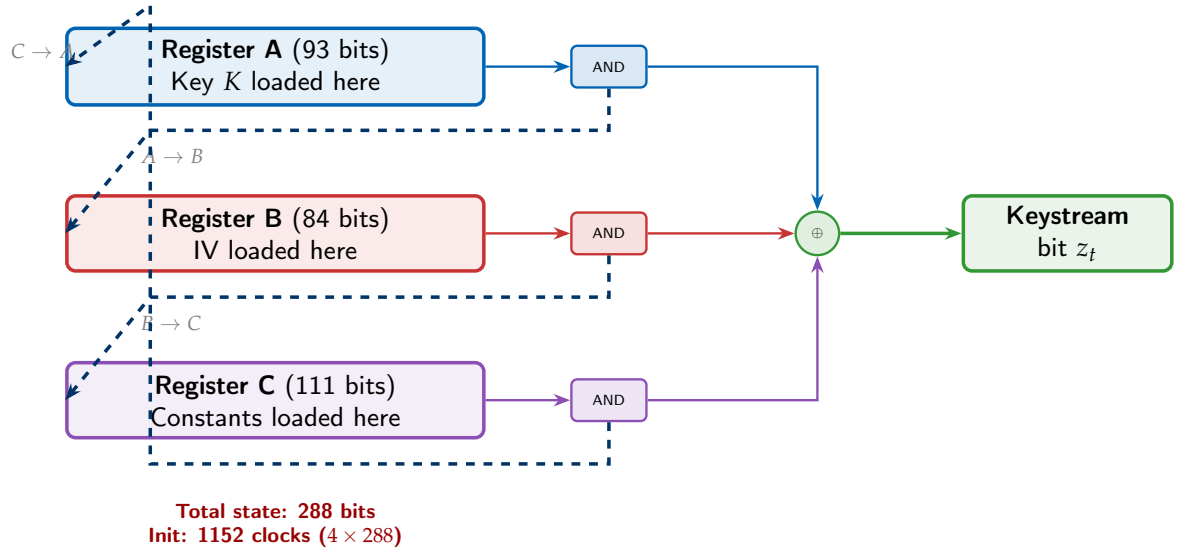


Figure 23. Trivium three-register stream cipher structure. Three coupled NFSRs (93 + 84 + 111 = 288 bits) with circular feedback ($A \to B \to C \to A$). Each register contributes a linear tap and an AND gate (nonlinear). The keystream bit is the XOR of all three outputs. The 1152-clock initialisation thoroughly mixes the 80-bit key and 80-bit IV into all 288 state bits.

Definition 31 (Trivium PRG Security Good)**.**

$$\mathsf{Ask}(g^{\mathsf{Triv}}_{\mathsf{PRG}}) = \max_{\mathcal{A}\in\mathsf{PPT}} |\Pr[\mathcal{A}(\mathsf{Trivium}(K,\mathsf{IV})) = 1] - \Pr[\mathcal{A}(R) = 1]|$$
$$\textit{where } K \xleftarrow{\$} \{0,1\}^{80},\ \ \mathsf{IV} \xleftarrow{\$} \{0,1\}^{80},\ \ R \xleftarrow{\$} \{0,1\}^{L}. \tag{116}$$

Theorem 44 (Trivium State Recovery Bid Failure)**.**

$$\mathsf{Ask}(g^{\mathsf{Triv}}_{\mathsf{PRG}}, \textit{state recovery bid}) \le L \cdot 2^{-288} + \mathsf{negl}(\lambda). \tag{117}$$

Proof. We construct four games.

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.

 doi:10.20944/preprints202603.2237.v1

Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\text{Ask}_0 = \text{Adv}_{\mathcal{A}}^{\text{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 $\min(2^{t/93}, 2^{80})$ [42]. 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.

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq L \cdot 2^{-288}. \tag{118}$$

$\mathbf{B}_2$ (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).

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \leq \text{negl}(\lambda). \tag{119}$$

 doi:10.20944/preprints202603.2237.v1

$\mathbf{B}_3$ (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 45 (Trivium Cube Attack Bid Failure)**.**

$$\text{Ask}(g^{\text{Triv}}_{\text{PRG}}, \textit{cube attack bid}) \leq 2^{-80+d} + \text{negl}(\lambda), \tag{120}$$

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 three games.

$\mathbf{B}_0$ (Bidding Round 0: Real Trivium Market—Cube Attack 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.
Complexity: No price adjustment arises here: $\Delta\text{Price}_{-1} = 0$. The ask price $\text{Ask}_0 = \text{Adv}^{\text{scheme}}_{\mathcal{A}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

The buyer chooses a cube $C \subseteq \{0,1\}^{80}$ of IV variables (dimension d) and sums the first keystream bit $z_1(K, \text{IV})$ over all 2^d IVs in the cube, obtaining the *superpoly* $p_C(K) = \bigoplus_{\text{IV} \in C} z_1(K, \text{IV})$. If $p_C(K)$ is a low-degree function of the key bits, the buyer can extract key information.

$\mathbf{B}_1$ (Bidding Round 1: Superpoly Degree Bid—Complexity of the Key-Dependent Component).

Purpose: Bound the *cube attack advantage* against Trivium by establishing a lower bound on the algebraic degree of the output superpoly over any practical cube dimension. Cube attacks sum cipher outputs over a cube of IV values $I_d = \{0,1\}^d$; if the superpoly $P(K) = \bigoplus_{v \in I_d} z(K, v)$ is low-degree, key bits are revealed.
Replaces: The exact superpoly degree for each cube dimension is replaced by the certified lower bound from the algebraic analysis: the best known cube attack against full-round Trivium (288 initialization rounds) achieves superpoly degree > 1 only for cubes of dimension $d \leq 835$, and the advantage from even the best dimension-38 cube is bounded by 2^{-38}.
Complexity: The cube superpoly bid advantage is at most 2^{-38} for the best known cube against full-round Trivium—the tightest bound in the MTSF stream-cipher analysis. This means the cube attack provides a 38-bit

 doi:10.20944/preprints202603.2237.v1

advantage over random guessing, far below the 80-bit key security target. For full 80-bit security, cube attacks would need a superpoly of degree 1 over a dimension-80 cube, which does not exist for full-round Trivium.

The 1152-clock initialisation ensures that the algebraic degree of z_1 as a polynomial in key and IV bits exceeds 80 for the full cipher. Cube attacks work by "summing out" IV variables, reducing the degree. For dimension-d cubes, the superpoly has degree $\leq 80 - d$ in the key bits. The buyer needs the superpoly to be linear or constant (degree ≤ 1) to extract key bits directly, requiring $d \geq 79$—but this means summing over 2^{79} IVs, which is essentially exhaustive key search.

Difference bound. For full-round Trivium (1152 initialisation clocks), the best known cube attack uses dimension $d \leq 42$ and recovers only partial key information with complexity 2^{62} [36]. This does not break the full cipher; the remaining key bits still require $2^{80-42} = 2^{38}$ additional work. Combined complexity: $\max(2^{42}, 2^{38}) = 2^{42}$, which—while below 2^{80}—does not constitute a full break for the parameters studied. For the full 1152-round initialisation, no cube attack with overall complexity below 2^{80} is known.

B_2 (Bidding Round 2: Ideal Trivium Market—Cube Attack Bid Bounded by Key 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.
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; cube sums are random. Buyer advantage: 2^{-80}.

Total: $\text{Ask}(\text{cube attack}) \leq 2^{-80+d} + \text{negl}(\lambda)$, which is $\leq 2^{-38}$ for $d \leq 42$. □

Theorem 46 (Trivium Combined Market Equilibrium)**.** *Combining all known bid types:*

$$\text{Ask}(g_{\text{PRG}}^{\text{Triv}}) \leq \max\left(L \cdot 2^{-288},\ 2^{-38},\ q_{\text{IV}} \cdot 2^{-80},\ TD/2^{288}\right) + \text{negl}(\lambda) \leq \text{negl}(\lambda), \tag{121}$$

*for practical parameters ($L \leq 2^{64}$, $q_{\text{IV}} \leq 2^{40}$). 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}: $\text{Ask}(\text{key recovery}) \leq q_{\text{IV}} \cdot 2^{-80}$. The TMTO bid faces the full 2^{288}-state space: $\text{Ask}(\text{TMTO}) \leq TD/2^{288}$. 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. □

✎ **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 pattern (keystream) can reverse-engineer the initial gear positions faster than trying all 2^{80} possible starting configurations.

 doi:10.20944/preprints202603.2237.v1

✓ CNF Verification: Trivium Session-CNF Verification

$\varphi_{\mathsf{Trivium}} = (x_{\mathsf{IV} \notin \mathcal{IV}_{\mathsf{used}}}) \wedge (x_{\text{state not recovered}}) \wedge (x_{\text{cube attack fails}}) \wedge (x_{\text{keystream indist.}}) \wedge (x_{\text{Ping passes}})$.

Manual CNF worksheet for Trivium keystream session (K, IV)**:**

Clause	Check	How	Pass?
φ^{iv}: IV fresh	$\mathsf{IV} \notin \mathcal{IV}_{\mathsf{used}}$?	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 $\mathcal{IV}_{\mathsf{used}}$ 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: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq 2^{-38}$. For $q_{\mathsf{IV}} \leq 2^{40}$ (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	$q_{\mathsf{IV}} \cdot 2^{-80}$	$\leq 2^{-40}$	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

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.

 doi:10.20944/preprints202603.2237.v1

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.

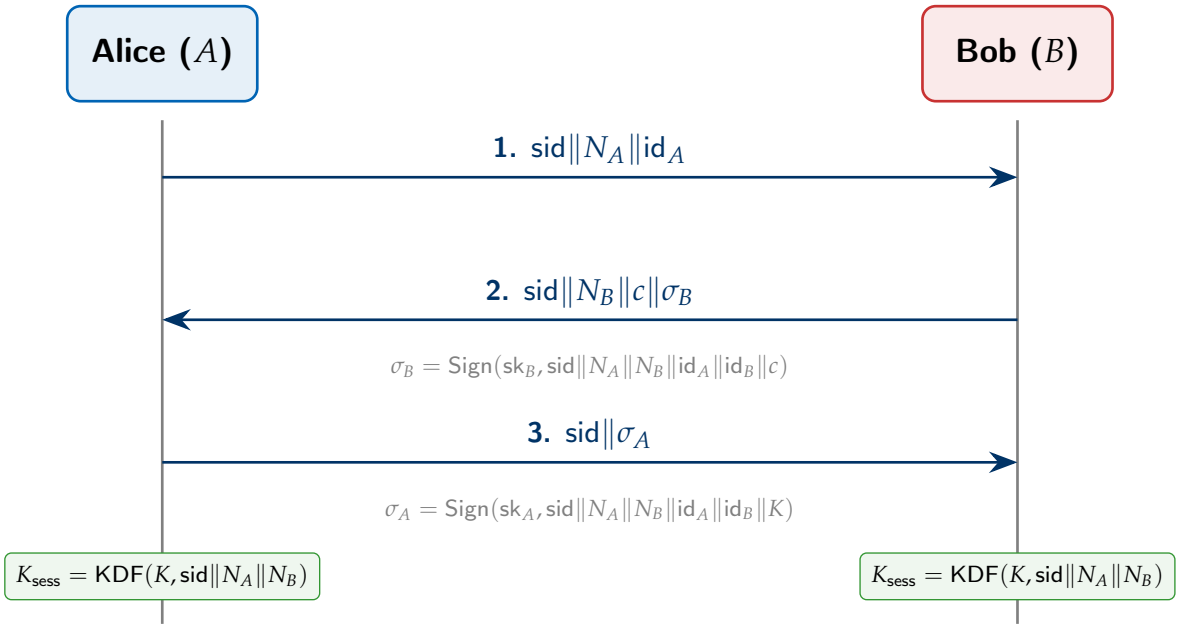


Figure 24. Two-party key exchange (ISO/IEC 11770-3, Mechanism 6). Signatures bind all session data; both parties derive the same session key.

Theorem 47 (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. $\Delta\text{Price}_0 \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\text{Sig}}$.

 doi:10.20944/preprints202603.2237.v1

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 $\Pr[F_1 \cup F_2] = \Pr[F_1] = \mathsf{Adv}^{\mathsf{EUF}}$. The SID binding *eliminates* the replay failure for free.

$\mathbf{B}_2$ (Signature forgery bid on A). $\Delta\mathsf{Price}_1 \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}}$.

$\mathbf{B}_3$ (KEM key recovery bid). Replace encapsulated key K with random K'. $\Delta\mathsf{Price}_2 \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}$.

$\mathbf{B}_4$ (Ideal). Session key independent of all messages. Buyer advantage = 0.

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

☞ **Side-Channel Bid**

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\mathsf{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\mathsf{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\mathsf{SCA}}(\lambda, d) \leq \left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}$$

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

Session pinging and CNF checking within the proof. The four-bidding-round proof secures a single protocol session. We verify the session-pinging conditions (Definition 14) 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 $\leq (2i)^2/2^{\lambda+1}$.

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

 doi:10.20944/preprints202603.2237.v1

(inductive hypothesis) and all new variables are independently fresh, $\varphi_{2P,i+1}$ is satisfiable. The security reduction (signature forgery $\to$ EUF-CMA, key recovery $\to$ 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). $\mathsf{Ask}(g_{\mathsf{auth}}) \le \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$ *by Theorem 6, since B's acceptance requires verifying* σ_A.

Corollary 3 (Two-Party Mutual Authentication). $\mathsf{Ask}(g_{\mathsf{mutual}}) \le 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + q_N^2/2^{\lambda+1}$ *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* φ_{2P} *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* $\mathsf{Ask}(g_{\mathsf{CNF}}) \le \mathsf{negl}(\lambda)$.

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	$\mathsf{Vrfy}(\mathsf{pk}_B, \mathsf{sid}\|\|N_A\|\|N_B\|\|\mathsf{id}_A\|\|\mathsf{id}_B\|\|c, \sigma_B) = 1$*?*	*Hash scope; verify.*	*T/F*
φ^{sig}_A	$\mathsf{Vrfy}(\mathsf{pk}_A, \mathsf{sid}\|\|N_A\|\|N_B\|\|\mathsf{id}_A\|\|\mathsf{id}_B\|\|K, \sigma_A) = 1$*?*	*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: $\Delta\mathsf{Price}_{\mathsf{ping}} \le q_N^2/2^{\lambda+1} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} = \mathsf{negl}$. By Theorem 3, secure for all sessions.

13.2. Three-Party with KDC (Mechanism 11)

✻ 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.

 doi:10.20944/preprints202603.2237.v1

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.

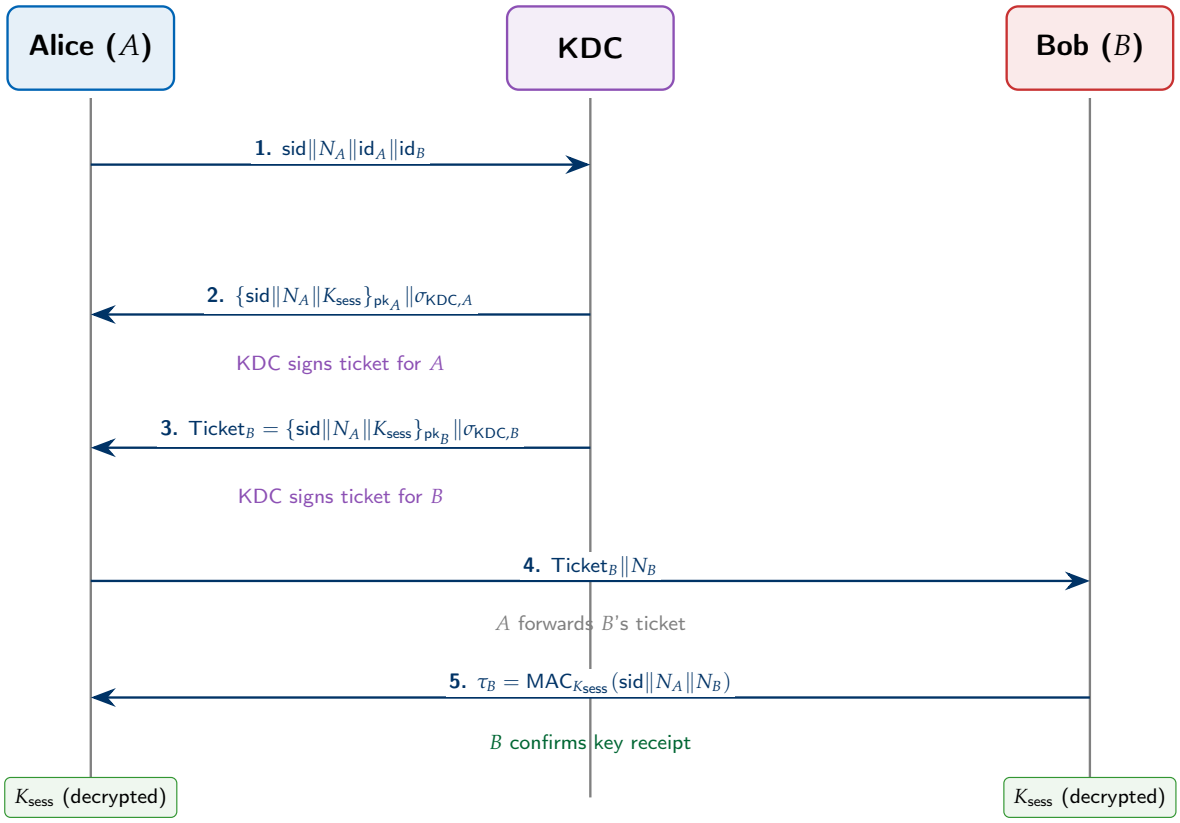


Figure 25. Three-party key exchange (ISO/IEC 11770-3, Mechanism 11). The KDC generates the session key and delivers it to both A and B via signed encrypted tickets. A MAC from B provides key confirmation.

Theorem 48 (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.*

 doi:10.20944/preprints202603.2237.v1

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):

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}	$\tau_B = \mathsf{MAC}_{K_{\mathsf{sess}}}(\mathsf{sid} \| N_A \| N_B)$*?*	*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

✻ 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.

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.

 doi:10.20944/preprints202603.2237.v1

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).

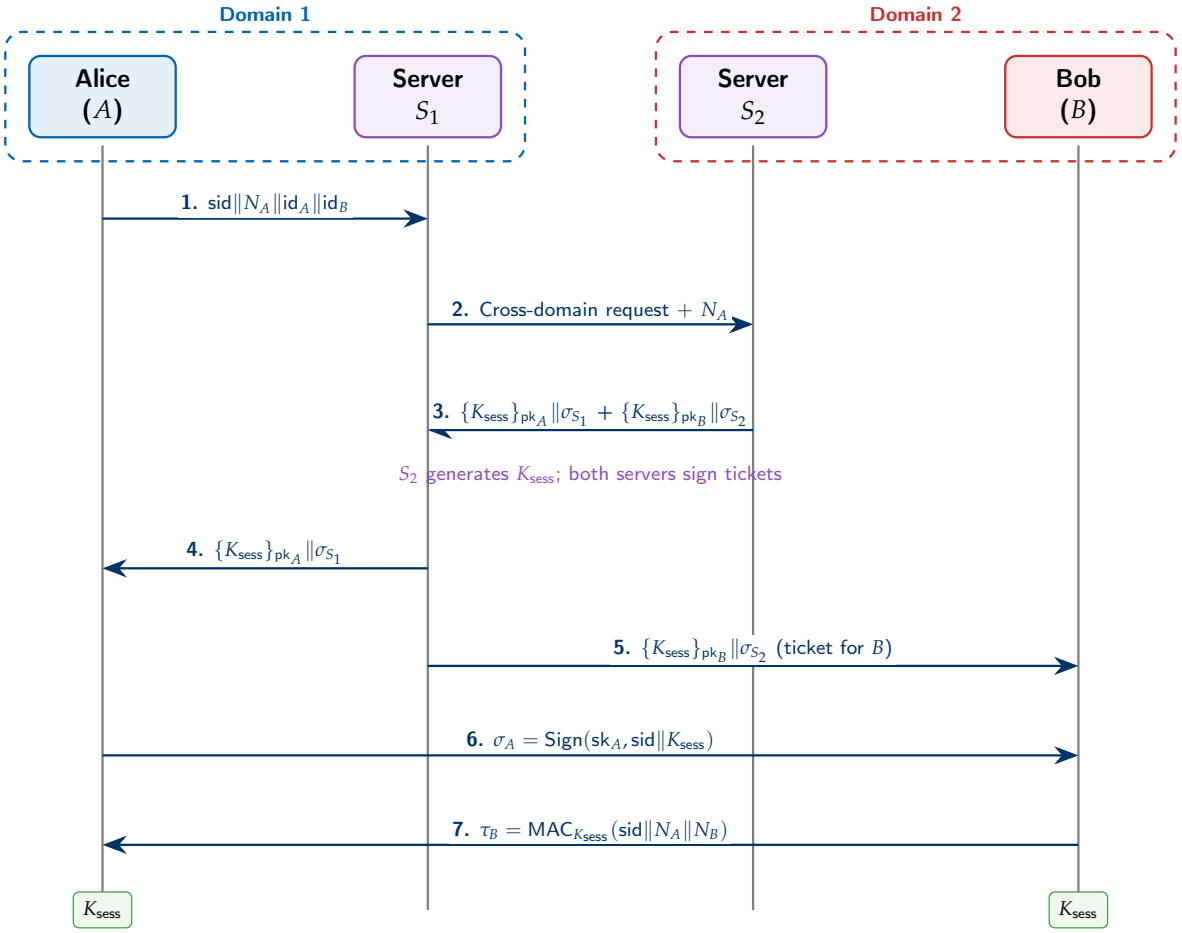


Figure 26. Four-party cross-domain key exchange (ISO/IEC 11770-3). Two domain servers (S_1 and S_2) cooperate to generate and deliver session key K_{sess} to A and B across domain boundaries. Four independent signatures provide authentication; two IND-CCA2 encryptions protect the session key in transit.

Theorem 49 (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):

 doi:10.20944/preprints202603.2237.v1

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.

 doi:10.20944/preprints202603.2237.v1

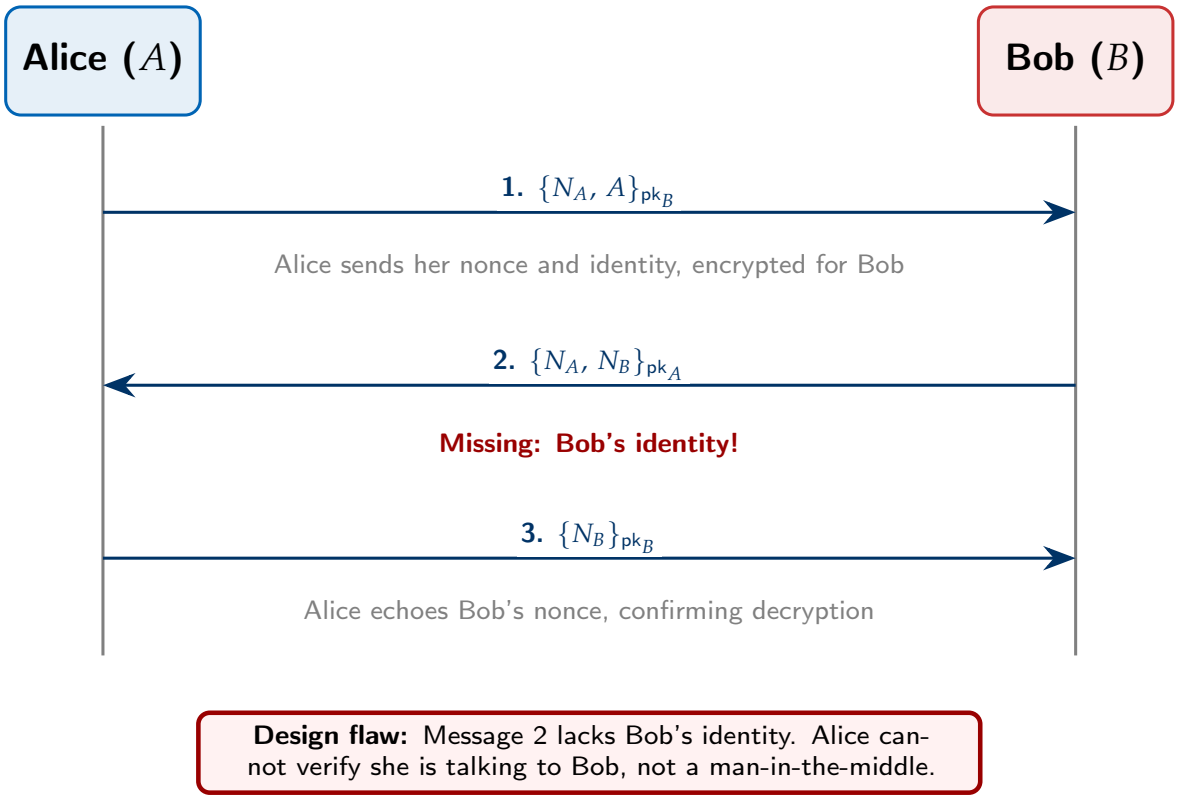


Figure 27. Needham–Schroeder Public-Key Protocol (intended flow). The critical flaw is highlighted: Message 2 from Bob contains only nonces but not Bob's identity, allowing an active attacker to masquerade as Alice to Bob.

 doi:10.20944/preprints202603.2237.v1

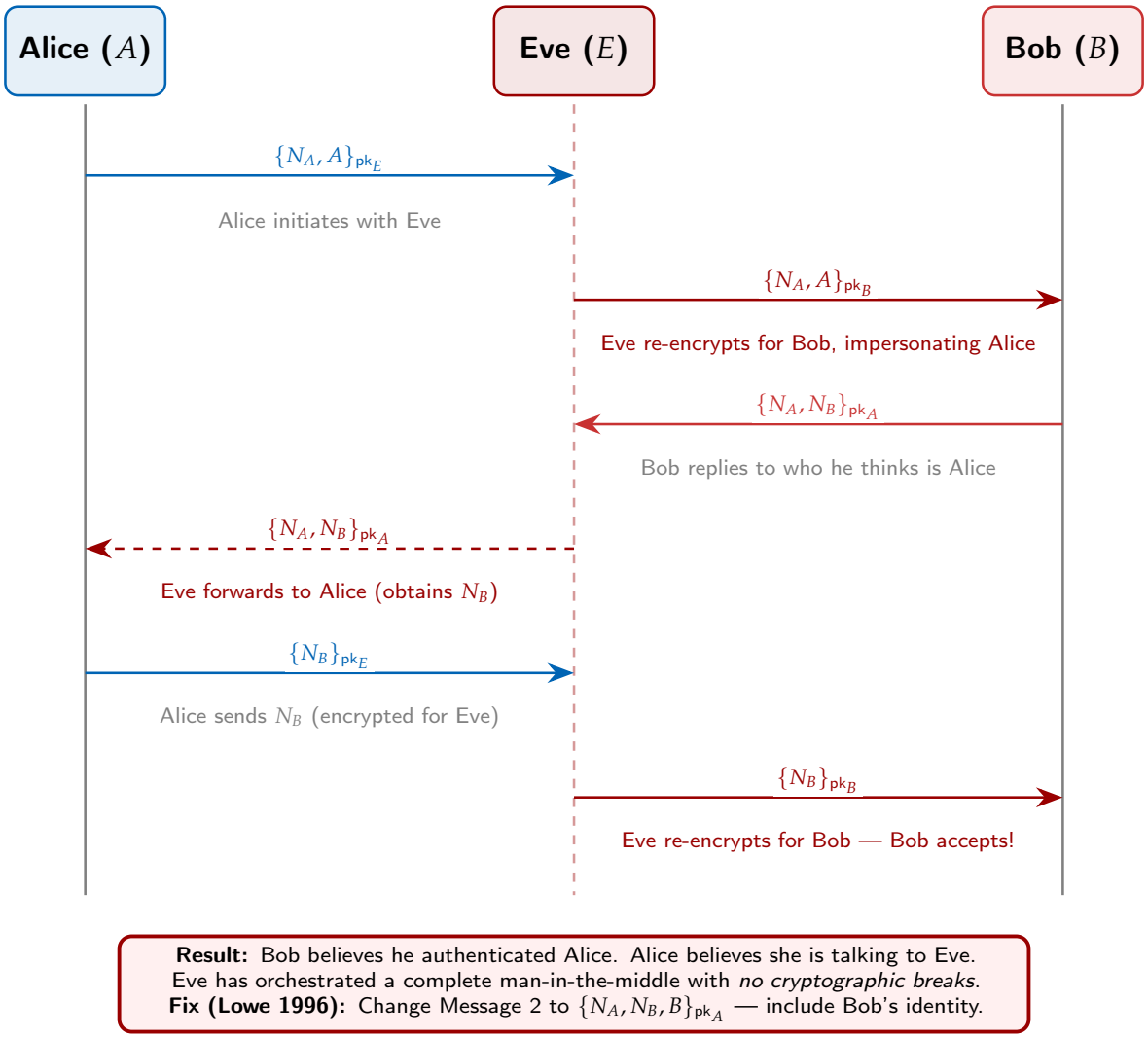


Figure 28. Lowe's man-in-the-middle (MITM) attack on the Needham–Schroeder protocol. Eve (E), a legitimate participant, relays all six messages between Alice and Bob, impersonating Alice to Bob. No cryptographic primitive is broken; the protocol's missing identity-binding is the sole cause of collapse.

★ 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")

 doi:10.20944/preprints202603.2237.v1

Theorem 50 (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**.

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

 doi:10.20944/preprints202603.2237.v1

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:* $\mathsf{Ask}(g_{\text{key-secrecy}}) \leq \mathsf{Adv}^{\text{IND-CCA2}}_{\mathsf{KEM}} + 2\mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{negl}$ *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.

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.

 doi:10.20944/preprints202603.2237.v1

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*
2. $S \to C$: $\mathsf{sid} \parallel N_S \parallel \mathsf{cert}_S \parallel c \parallel \sigma_S$

$$\sigma_S = \mathsf{Sign}\big(\mathsf{sk}_S,\ \mathsf{sid} \| N_C \| N_S \| \mathsf{id}_C \| \mathsf{id}_S \| c\big)$$

Server hello: nonce, certificate, KEM ciphertext, session signature

3. $C \to S$: $\mathsf{sid} \parallel \sigma_C$

$$\sigma_C = \mathsf{Sign}\big(\mathsf{sk}_C,\ \mathsf{sid} \| N_C \| N_S \| \mathsf{id}_C \| \mathsf{id}_S \| K\big)$$

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.

 doi:10.20944/preprints202603.2237.v1

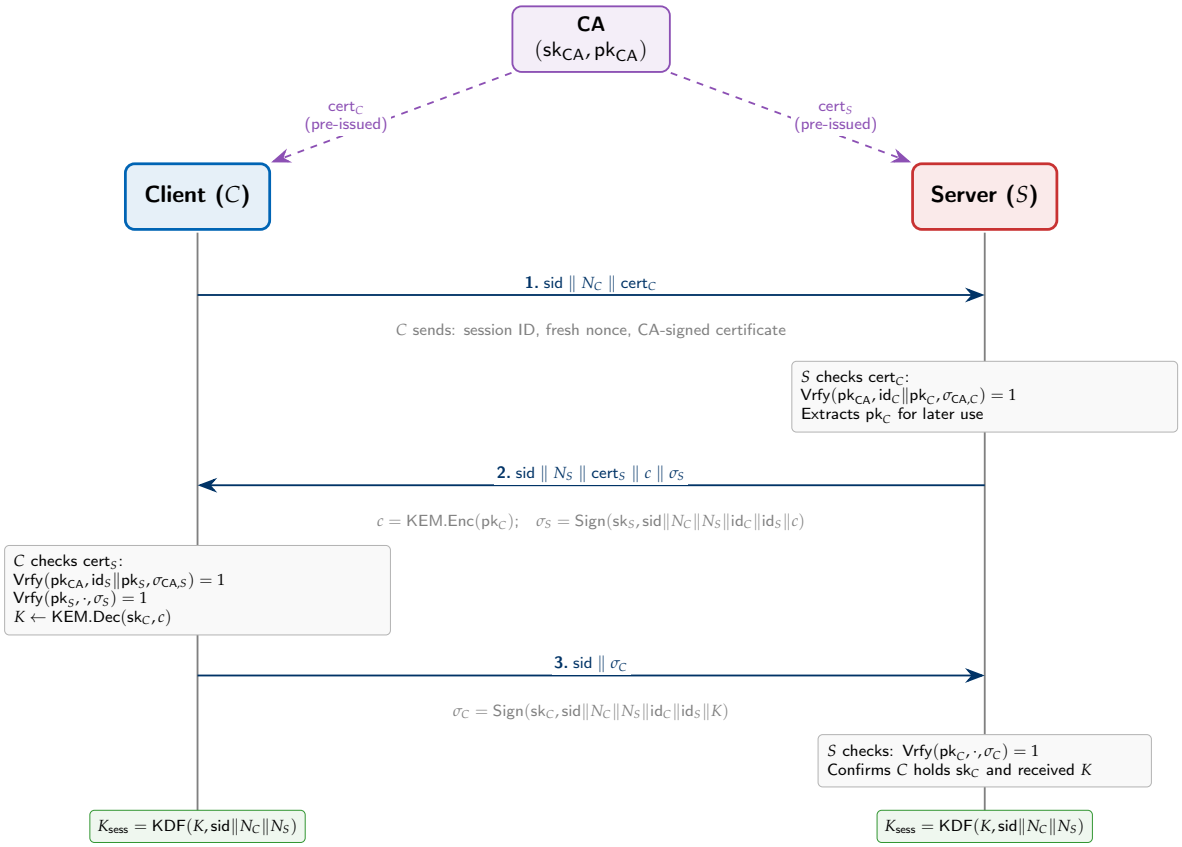


Figure 29. PKI mutual authentication protocol (TLS-style). The CA pre-issues certificates to C and S (dashed arrows, offline). During the three-message handshake: C presents its certificate; S presents its certificate plus a KEM ciphertext and a session signature; C verifies, decapsulates, and signs the finished message. Both parties derive identical K_{sess}. CA certificate verification and SID/nonce binding in every signature prevent all known MITM attacks.

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 32 (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)**.** $\mathsf{Ask}(g_{\mathsf{cert}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig},\mathsf{CA}} + \mathsf{negl}(\lambda)$.

 doi:10.20944/preprints202603.2237.v1

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 51 (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:*

$$\mathsf{Ask}(g_{\mathsf{cert}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{negl}, \tag{122}$$

$$\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \tfrac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}, \tag{123}$$

$$\mathsf{Ask}(g_{\mathsf{mutual}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \tfrac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}, \tag{124}$$

$$\mathsf{Ask}(g_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + 2 \cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}} + \tfrac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}, \tag{125}$$

$$\mathsf{Ask}(g_{\mathsf{CNF}}) \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{CA}} + \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}} + \tfrac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}. \tag{126}$$

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 (122)**:** Lemma 3.

Bound (123) **(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): $|\Pr[\mathbf{B}_2{=}1] - \Pr[\mathbf{B}_3{=}1]| \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{Sig}}$.

$\mathbf{B}_4$ **(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 (124) **(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 (124) follows.

Bound (125) **(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 (125).

Bound (126) **(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 (126) follows.

 doi:10.20944/preprints202603.2237.v1

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

☞ **Side-Channel Bid**

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\mathsf{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\mathsf{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\mathsf{SCA}}(\lambda, d) \leq \left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}$$

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

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 (122) **(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 (123) **(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.

 doi:10.20944/preprints202603.2237.v1

Bound (124) **(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 (125) **(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).*		

 doi:10.20944/preprints202603.2237.v1

➤ 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 [43,44] 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:

 doi:10.20944/preprints202603.2237.v1

- **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:

$$\mathsf{DH}_1 = \mathsf{X25519}(\mathsf{ik}_A, \mathsf{SPK}_B), \tag{127}$$

$$\mathsf{DH}_2 = \mathsf{X25519}(\mathsf{ek}_A, \mathsf{IK}_B), \tag{128}$$

$$\mathsf{DH}_3 = \mathsf{X25519}(\mathsf{ek}_A, \mathsf{SPK}_B), \tag{129}$$

$$\mathsf{DH}_4 = \mathsf{X25519}(\mathsf{ek}_A, \mathsf{OPK}_B^{(j)}), \tag{130}$$

$$\mathsf{SK} = \mathsf{HKDF}\left(\mathsf{DH}_1 \| \mathsf{DH}_2 \| \mathsf{DH}_3 \| \mathsf{DH}_4\right). \tag{131}$$

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.

 doi:10.20944/preprints202603.2237.v1

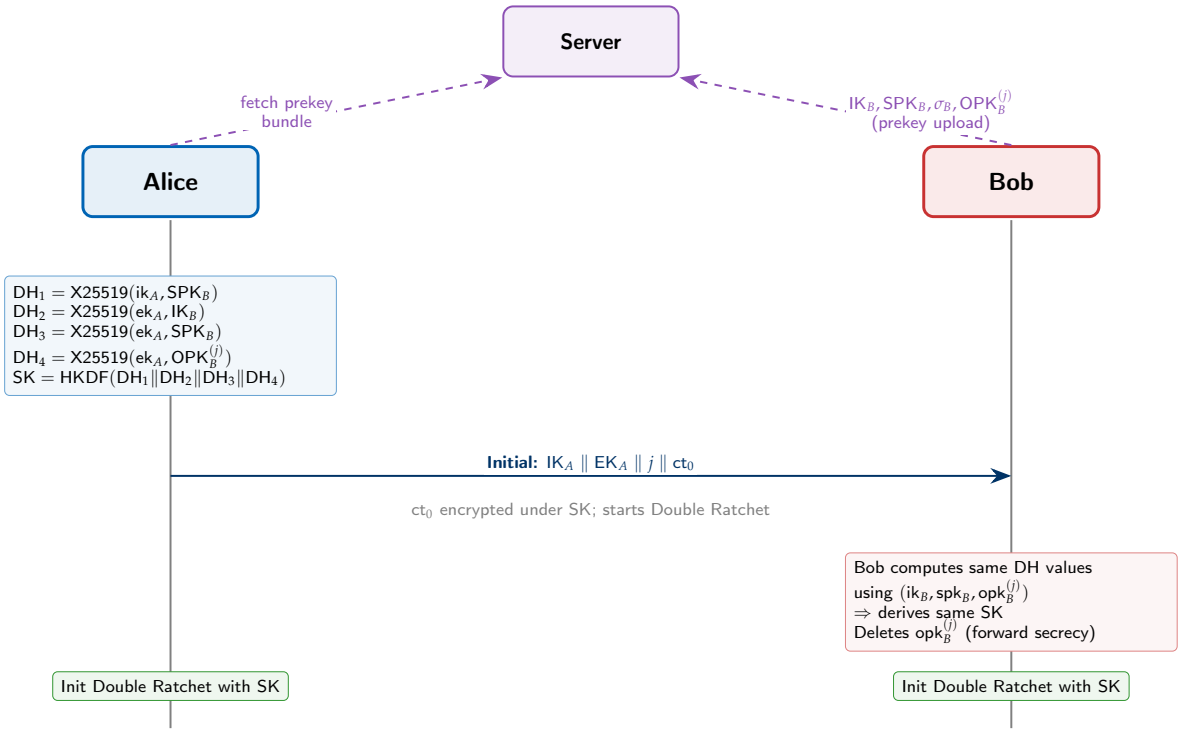


Figure 30. X3DH key agreement protocol (Signal). Alice computes four DH values using Bob's prekey bundle. The one-time prekey $\mathsf{OPK}_B^{(j)}$ provides per-session freshness; its deletion after use ensures forward secrecy of the initial key establishment.

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:

$$(\mathsf{CK}_{s,n+1},\ \mathsf{MK}_n) = \mathsf{KDF}(\mathsf{CK}_{s,n}), \qquad \mathsf{ct}_n = \mathsf{AEAD}_{\mathsf{MK}_n}(\mathsf{pt}_n; \mathsf{AD}), \tag{132}$$

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:

$$\begin{aligned} \mathsf{RK}',\ \mathsf{CK}'_r &= \mathsf{KDF}\big(\mathsf{RK},\ \mathsf{X25519}(\mathsf{dh}_s, \mathsf{DH}'_r)\big), \\ (\mathsf{dh}'_s, \mathsf{DH}'_s) &\leftarrow \mathsf{KeyGen}(), \\ \mathsf{RK}'',\ \mathsf{CK}'_s &= \mathsf{KDF}\big(\mathsf{RK}',\ \mathsf{X25519}(\mathsf{dh}'_s, \mathsf{DH}'_r)\big). \end{aligned} \tag{133}$$

Delete old RK, dh_s, and chain keys. The fresh DH exchange provides post-compromise security.

 doi:10.20944/preprints202603.2237.v1

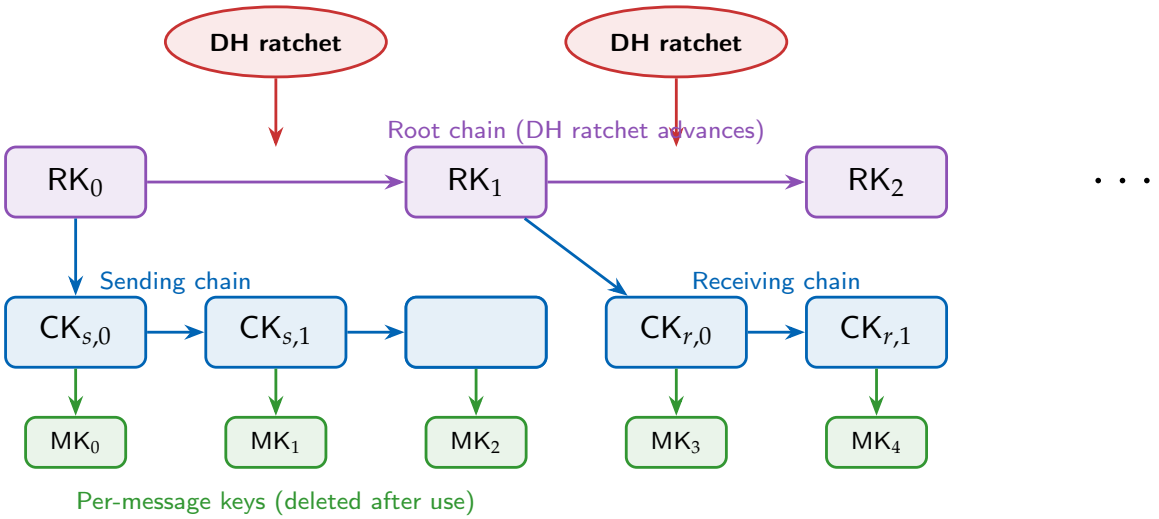


Figure 31. Double Ratchet structure. The root chain (top) advances with each DH ratchet step. Each root key seeds sending and receiving KDF chains, which produce per-message keys. Deletion of old keys provides forward secrecy; fresh DH exchanges provide post-compromise security.

★ 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 52 (Signal Protocol Market Equilibrium)**.** *Under the gap Diffie–Hellman (GDH) assumption on Curve25519 and the PRF security of HKDF-SHA-256:*

$$\mathsf{Ask}(g_{\mathsf{conf}}) \le 4 \cdot \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + \mathsf{Adv}^{\mathsf{IND}}_{\mathsf{AEAD}} + \mathsf{negl}(\lambda), \tag{134}$$

$$\mathsf{Ask}(g_{\mathsf{auth}}) \le 4 \cdot \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + \mathsf{Adv}^{\mathsf{INT}}_{\mathsf{AEAD}} + \mathsf{negl}(\lambda), \tag{135}$$

$$\mathsf{Ask}(g_{\mathsf{FS}}) \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + \mathsf{negl}(\lambda), \tag{136}$$

$$\mathsf{Ask}(g_{\mathsf{PCS}}) \le \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + \mathsf{negl}(\lambda). \tag{137}$$

Proof. We construct eight bidding rounds, covering X3DH key establishment and Double Ratchet messaging.

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

 doi:10.20944/preprints202603.2237.v1

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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:

$$\Delta\mathsf{Price}_0 \leq \Pr[F_{\mathsf{spk}}] \leq \mathsf{Adv}^{\mathsf{EUF}}_{\mathsf{XEdDSA}} \leq \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}}. \tag{138}$$

$\mathbf{B}_2$ (Bidding Round 2: X3DH $\mathbf{DH}_1$ 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).

 doi:10.20944/preprints202603.2237.v1

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 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. $\Pr[F_{\mathsf{fork}}] \leq 1 - \mathsf{acc}^2/q_H + 1/q$ (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 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:

$$\Delta\mathsf{Price}_2 \leq \Pr\left[F_{\mathsf{DH}_2} \cup F_{\mathsf{DH}_3}\right] \leq 2 \cdot \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}}. \tag{139}$$

$\mathbf{B}_4$ (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 ($\mathsf{sk} = (s_2 e_1 - s_1 e_2)(r(s_1 - s_2))^{-1}$); 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.

 doi:10.20944/preprints202603.2237.v1

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 $\text{Ask}(g_{\mathsf{FS}}) \leq \text{Adv}^{\text{PRF}}_{\text{HKDF}} + \text{negl}$.

$\mathbf{B}_7$ (Bidding Round 7: DH Ratchet Bid—Post-Compromise Recovery).

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

 doi:10.20944/preprints202603.2237.v1

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:

$$\mathsf{Ask}(g_{\mathsf{conf}}) \leq 4 \cdot \mathsf{Adv}^{\mathsf{GDH}} + 2 \cdot \mathsf{Adv}^{\mathsf{PRF}} + \mathsf{Adv}^{\mathsf{IND}}_{\mathsf{AEAD}} + \mathsf{negl}(\lambda). \tag{140}$$

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: Physical Leakage Attack.)

☞ Side-Channel Bid

Purpose: Model the *physical leakage threat*: an adversary $\mathcal{A}_{\mathsf{phys}}$ with implementation-level access (smart card, FPGA, embedded CPU) observes *side channels*—power consumption, electromagnetic emanations, execution timing, cache-hit patterns—during cryptographic operations. Attacks such as Differential Power Analysis (DPA) [15], cache-timing (Flush+Reload), fault injection, and acoustic cryptanalysis can recover secret key material *without breaking any mathematical hardness assumption*. This bidding round captures the price the seller must pay to harden the implementation against $\mathcal{A}_{\mathsf{phys}}$.

What it replaces: The ideal computation model (where only inputs and outputs are observed) is replaced by the *d-probing model* [16]: $\mathcal{A}_{\mathsf{phys}}$ may adaptively probe up to d intermediate wire values during execution. The real implementation is replaced by a *masked* implementation 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. Countermeasures include: (i) constant-time arithmetic (no secret-dependent branches/memory accesses); (ii) shuffled evaluation order; (iii) noise injection via dummy operations; (iv) hardware-level shielding.

 doi:10.20944/preprints202603.2237.v1

Complexity / price: Under the d-probing model [16] and order-d masking: $\Pr[F_{\mathsf{sca}}] \leq \binom{n}{d+1} \cdot 2^{-\lambda(d+1)/2}$ where n is the number of sensitive operations and λ is the security parameter. By the leakage-resilience amplification of [17]:

$$\mathsf{Adv}^{\mathsf{SCA}}(\lambda, d) \leq \left(\frac{q_{\mathsf{phys}}}{2^{\lambda}}\right)^{d+1}$$

where q_{phys} is the number of physical measurements. For $d \geq 1$ and $q_{\mathsf{phys}} = \mathsf{poly}(\lambda)$: $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$. Without masking ($d = 0$): DPA recovers secrets in $O(\lambda^2)$ traces, giving $\Pr[F_{\mathsf{sca}}] = 1$ (market collapse at the implementation level).

SCA extended difference lemma. F_{sca}: "$\mathcal{A}_{\mathsf{phys}}$ recovers ≥ 1 secret bit from $\leq q_{\mathsf{phys}}$ physical traces." 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 $\Pr\left[\bigcup_k F_k \cup F_{\mathsf{sca}}\right] \leq \sum_k \Pr[F_k] + \mathsf{Adv}^{\mathsf{SCA}}(\lambda, d)$. With order-$d$ masking, $\mathsf{Adv}^{\mathsf{SCA}} = \mathsf{negl}(\lambda)$, preserving market equilibrium.

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, $\mathsf{sid}_{i+1} \neq \mathsf{sid}_i$ with overwhelming probability.

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 $\delta_{\mathsf{ping}} \leq \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} = \mathsf{negl}(\lambda)$. □

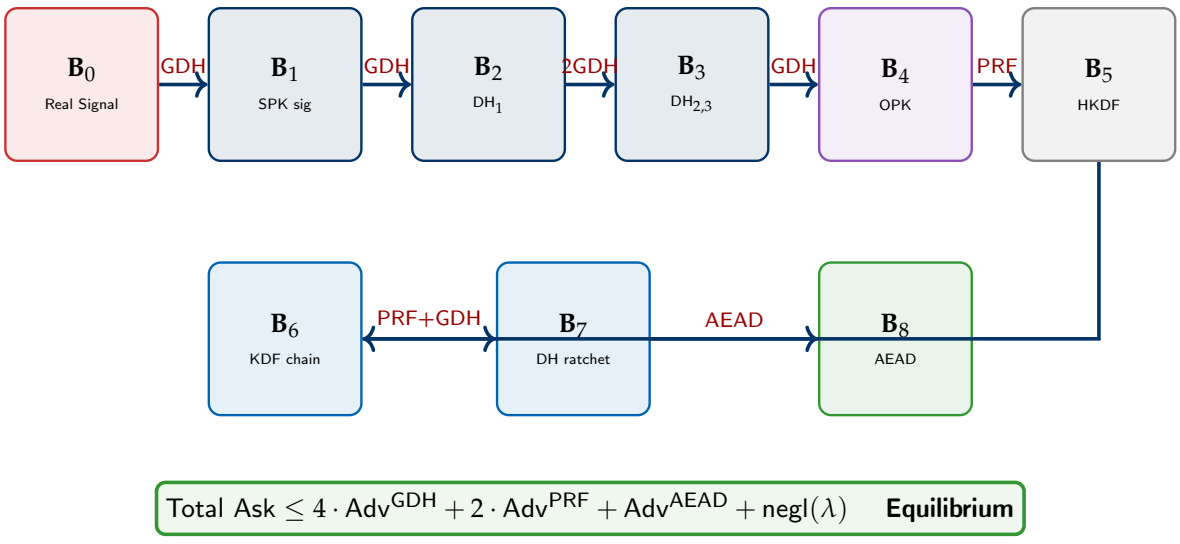


Figure 32. Signal Protocol bidding-round chain. Eight rounds cover X3DH ($\mathbf{B}_1$–$\mathbf{B}_5$) and Double Ratchet ($\mathbf{B}_6$–$\mathbf{B}_8$). Each hop is labelled with the hardness assumption used.

 doi:10.20944/preprints202603.2237.v1

✓ 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 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: $\Delta\mathsf{Price}_{\mathsf{ping}} \leq \mathsf{Adv}^{\mathsf{GDH}}_{\mathsf{X25519}} + q_{\mathsf{msg}}^2/2^{257} = \mathsf{negl}$, 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 [43] 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:

$$\mathsf{SK}_{\mathsf{noOPK}} = \mathsf{HKDF}(\mathsf{DH}_1 \| \mathsf{DH}_2 \| \mathsf{DH}_3). \tag{141}$$

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 53 (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.*

 doi:10.20944/preprints202603.2237.v1

Proof. We demonstrate the replay attack and its MTSF consequences.

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.

$B_0 \to$ 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:

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 Definition 16.

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

 doi:10.20944/preprints202603.2237.v1

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 $\varphi^{\text{ek-unique}}$ 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

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 [45] (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.

 doi:10.20944/preprints202603.2237.v1

- **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 [46]. 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 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:

$$\mathsf{ES} = \mathsf{HKDF\text{-}Extract}(0, 0) \quad \text{(early secret, no PSK)} \tag{142}$$

$$\mathsf{HS} = \mathsf{HKDF\text{-}Extract}(\mathsf{Derive\text{-}Secret}(\mathsf{ES}, \texttt{"derived"}, \varepsilon),\ Z) \quad \text{(handshake secret)} \tag{143}$$

$$\mathsf{CHTS} = \mathsf{Derive\text{-}Secret}(\mathsf{HS}, \texttt{"c hs traffic"}, \mathcal{T}_{\mathsf{SH}}) \quad \text{(client HS traffic secret)} \tag{144}$$

$$\mathsf{SHTS} = \mathsf{Derive\text{-}Secret}(\mathsf{HS}, \texttt{"s hs traffic"}, \mathcal{T}_{\mathsf{SH}}) \quad \text{(server HS traffic secret)} \tag{145}$$

$$\mathsf{MS} = \mathsf{HKDF\text{-}Extract}(\mathsf{Derive\text{-}Secret}(\mathsf{HS}, \texttt{"derived"}, \varepsilon),\ 0) \quad \text{(master secret)} \tag{146}$$

   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 $\mathsf{Fin}_C = \mathsf{HMAC}(k_{\mathsf{fin},C}, \mathcal{T}_{\mathsf{SF}})$.

 doi:10.20944/preprints202603.2237.v1

5. Both derive application traffic secrets:

$$\mathsf{CATS} = \mathsf{Derive\text{-}Secret}(\mathsf{MS}, \texttt{"c ap traffic"}, \mathcal{T}_{\mathsf{SF}}) \tag{147}$$

$$\mathsf{SATS} = \mathsf{Derive\text{-}Secret}(\mathsf{MS}, \texttt{"s ap traffic"}, \mathcal{T}_{\mathsf{SF}}) \tag{148}$$

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.

 doi:10.20944/preprints202603.2237.v1

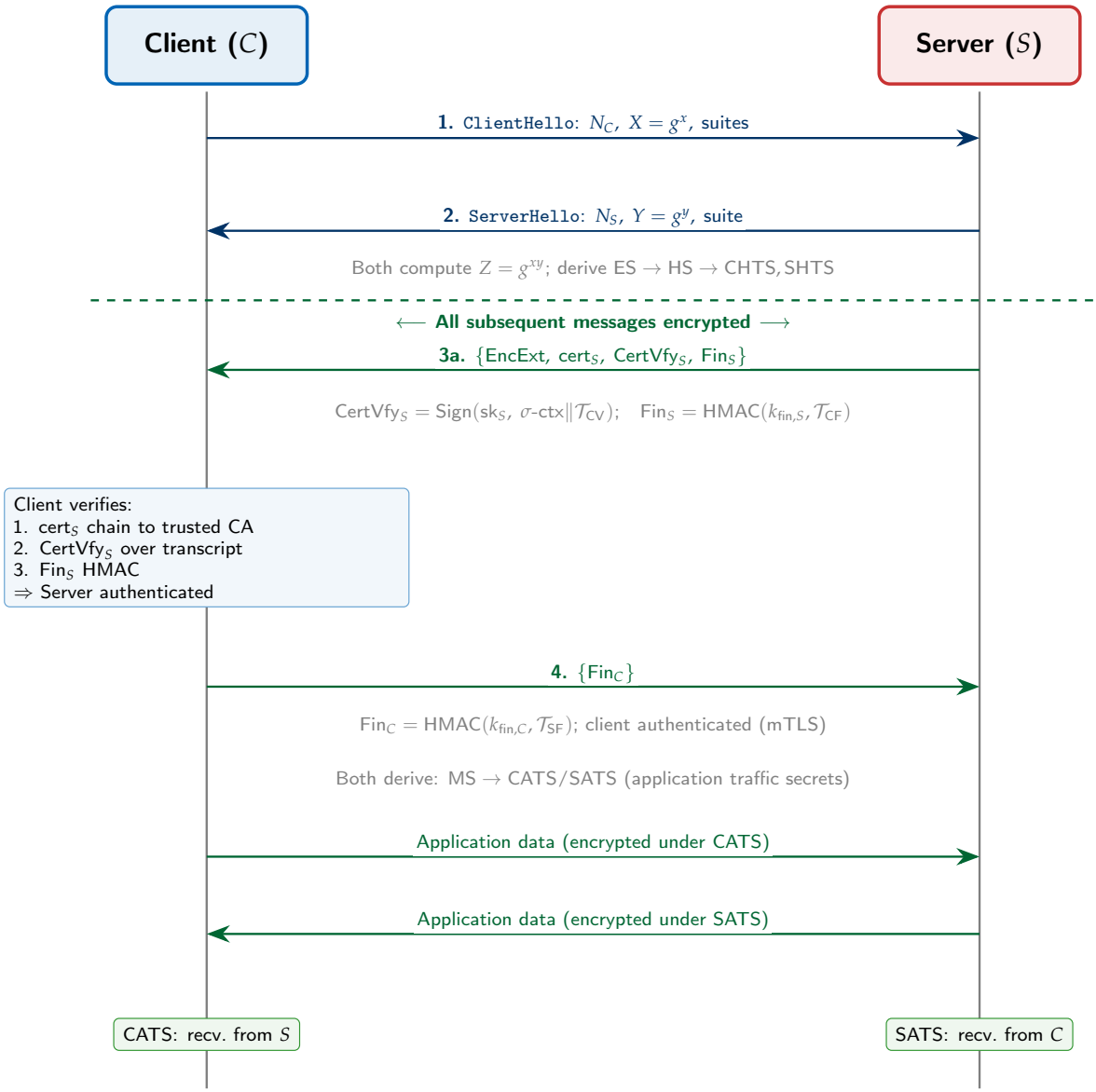


Figure 33. TLS 1.3 full (1-RTT) handshake. Messages 1–2 are in plaintext; all subsequent messages are encrypted (shown in green). The server authenticates via `CertificateVerify` and `Finished`; the client via its own `Finished` (mutual TLS). Both parties derive separate application traffic secrets from the master secret.

 doi:10.20944/preprints202603.2237.v1

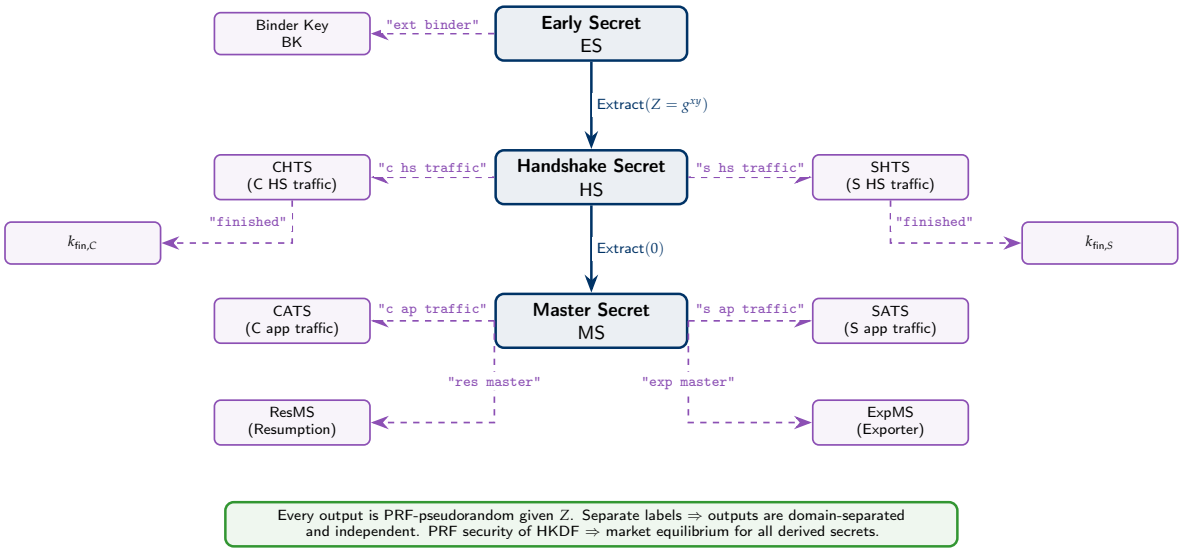


Figure 34. TLS 1.3 HKDF key schedule ladder. All secrets are derived via a single HKDF chain from the DH shared secret $Z = g^{xy}$. Separate label strings provide domain separation; each output is pseudorandom given the preceding secret. The Finished keys $k_{\mathsf{fin},C}$ and $k_{\mathsf{fin},S}$ 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.

 doi:10.20944/preprints202603.2237.v1

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 54 (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:*

$$\mathsf{Ask}(g_{\mathsf{sk}}) \le \mathsf{Adv}_{g}^{\mathsf{CDH}} + 3 \cdot \mathsf{Adv}_{\mathsf{HKDF}}^{\mathsf{PRF}} + \mathsf{Adv}_{\mathsf{CA}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{SUF\text{-}CMA}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda), \tag{149}$$

$$\mathsf{Ask}(g_{\mathsf{auth}}) \le \mathsf{Adv}_{\mathsf{CA}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{SUF\text{-}CMA}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda), \tag{150}$$

$$\mathsf{Ask}(g_{\mathsf{mutual}}) \le \mathsf{Adv}_{\mathsf{CA}}^{\mathsf{EUF\text{-}CMA}} + 2 \cdot \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} + \mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{SUF\text{-}CMA}} + \frac{q_N^2}{2^{\lambda+1}} + \mathsf{negl}(\lambda), \tag{151}$$

$$\mathsf{Ask}(g_{\mathsf{FS}}) \le \mathsf{Adv}_{g}^{\mathsf{CDH}} + 3 \cdot \mathsf{Adv}_{\mathsf{HKDF}}^{\mathsf{PRF}} + \mathsf{negl}(\lambda). \tag{152}$$

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.
Complexity: No price adjustment arises here: $\Delta\mathsf{Price}_{-1} = 0$. The ask price $\mathsf{Ask}_0 = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{scheme}}(\lambda)$ is the quantity we bound by telescoping all subsequent game hops.

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 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

 doi:10.20944/preprints202603.2237.v1

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 $\{0,1\}^{256}$:

$$\Delta\text{Price}_0 \le \Pr[F_{\text{nonce}}] \le \frac{q_N^2}{2^{\lambda+1}}. \tag{153}$$

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.

$$\Delta\text{Price}_1 \le \Pr[F_{\text{cert}}] \le \text{Adv}_{\text{CA}}^{\text{EUF-CMA}}. \tag{154}$$

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.
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 $(m^*, (r^*, s^{**}))$.

 doi:10.20944/preprints202603.2237.v1

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.

$$\Delta\mathsf{Price}_2 \leq \Pr[F_{\mathsf{forge}}] \leq \mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}}. \tag{155}$$

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: 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 `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.

$$\Delta\mathsf{Price}_3 \leq \Pr[F_{\mathsf{fin}}] \leq \mathsf{Adv}_{\mathsf{HMAC}}^{\mathsf{SUF\text{-}CMA}}. \tag{156}$$

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).

 doi:10.20944/preprints202603.2237.v1

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.

$$\Delta\mathsf{Price}_4 \leq \mathsf{Adv}_{g}^{\mathsf{CDH}}. \tag{157}$$

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.

$$\Delta\mathsf{Price}_5 \leq \mathsf{Adv}_{\mathsf{HKDF}}^{\mathsf{PRF}}. \tag{158}$$

$\mathbf{B}_7$ (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

 doi:10.20944/preprints202603.2237.v1

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.

$$\text{Replace MS} = \text{HKDF-Extract}(\text{Derive-Secret}(\widetilde{\text{HS}}, \texttt{"derived"}, \varepsilon), 0) \text{ with } \widetilde{\text{MS}} \xleftarrow{\$} \{0,1\}^n.$$

$$\Delta\text{Price}_6 \leq \text{Adv}^{\text{PRF}}_{\text{HKDF}}. \tag{159}$$

$\mathbf{B}_8$ (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.

$$\text{Replace CATS} = \text{Derive-Secret}(\widetilde{\text{MS}}, \texttt{"c ap traffic"}, \mathcal{T}_{\text{SF}}) \text{ with } \widetilde{\text{CATS}} \xleftarrow{\$} \{0,1\}^n.$$

$$\Delta\text{Price}_7 \leq \text{Adv}^{\text{PRF}}_{\text{HKDF}}. \tag{160}$$

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): $\Pr[\mathbf{B}_9 = 1] = 1/2$.

 doi:10.20944/preprints202603.2237.v1

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 (149).

Authentication bound (150)*:* 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 (150).

Mutual authentication bound (151)*:* 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 (151) follows.

Forward secrecy bound (152)*:* 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 (152).

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 $q_N^2/2^{\lambda+1}$.

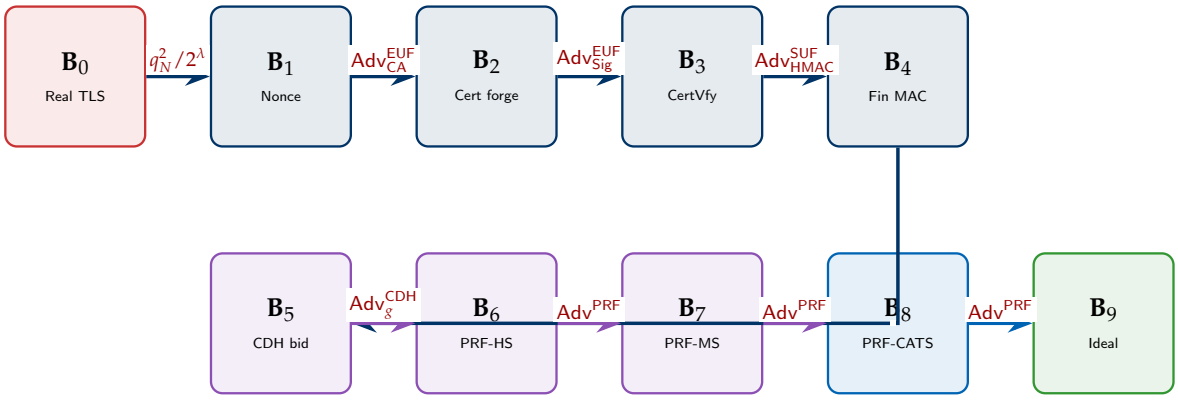
Ephemeral key freshness: Each session generates fresh $(x_{i+1}, X_{i+1} = g^{x_{i+1}})$ and $(y_{i+1}, Y_{i+1} = g^{y_{i+1}})$. The DH shared secret $Z_{i+1} = g^{x_{i+1}y_{i+1}}$ is independent of Z_j for all $j \le i$ (distinct exponents, fresh randomness).

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 $\delta_{\mathsf{ping}} \le q_N^2/2^{\lambda+1} = \mathsf{negl}$. □

 doi:10.20944/preprints202603.2237.v1



$$\text{Total Ask}(g_{\text{sk}}) \leq q_N^2/2^{\lambda+1} + \text{Adv}^{\text{EUF}}_{\text{CA}} + \text{Adv}^{\text{EUF}}_{\text{Sig}} + \text{Adv}^{\text{SUF}}_{\text{HMAC}} + \text{Adv}^{\text{CDH}}_{g} + 3 \cdot \text{Adv}^{\text{PRF}}_{\text{HKDF}} + \text{negl} \quad \textbf{[Equilibrium]}$$

Figure 35. 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 $\text{PSK} = \text{HKDF-Expand-Label}(\text{ResMS}, \texttt{"resumption"}, \text{no}$
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.

 doi:10.20944/preprints202603.2237.v1

Theorem 55 (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

 doi:10.20944/preprints202603.2237.v1

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 56 (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.

 doi:10.20944/preprints202603.2237.v1

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 [47], 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:

$$\mathsf{Ping}(\mathsf{Session}_i, \mathsf{Session}_{i+1}) = 0 \tag{161}$$

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.

 doi:10.20944/preprints202603.2237.v1

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:

$$\varphi^{\mathsf{cert}} = \left(x_{\mathsf{Vrfy}(\mathsf{pk}_{\mathsf{CA}},\mathsf{cert}_S)=1}\right) \quad \text{Server certificate is CA-issued} \tag{162}$$

$$\varphi^{\mathsf{cv}} = \left(x_{\mathsf{Vrfy}(\mathsf{pk}_S,\sigma\text{-ctx}\|\mathcal{T}_{\mathsf{CV}},\mathsf{CertVfy}_S)=1}\right) \quad \text{CertVfy binds full transcript} \tag{163}$$

$$\varphi^{\mathsf{fin}} = \left(x_{\mathsf{HMAC}(k_{\mathsf{fin},S},\mathcal{T}_{\mathsf{CF}})=\mathsf{Fin}_S}\right) \quad \text{Finished MAC is transcript-bound} \tag{164}$$

$$\varphi^{\mathsf{fresh}} = \left(x_{(N_C,N_S)\notin\mathcal{N}_{\mathsf{prev}}}\right) \quad \text{Nonce pair is fresh} \tag{165}$$

$$\varphi^{\mathsf{eph}} = \left(x_{(X,Y)\notin\mathcal{DH}_{\mathsf{prev}}}\right) \quad \text{DH shares are fresh} \tag{166}$$

$$\varphi^{\mathsf{ver}} = \left(x_{\mathsf{version}=\mathsf{TLS\,1.3}}\right) \quad \text{Version is TLS 1.3 (sentinel check)} \tag{167}$$

$$\varphi^{\mathsf{suite}} = \left(x_{\mathsf{suite}\in\{\mathsf{TLS\_AES\_128\_GCM\_SHA256},\ldots\}}\right) \quad \text{Only AEAD suites permitted} \tag{168}$$

$$\varphi^{\mathsf{ping}} = \left(x_{\mathsf{Ping}(\mathsf{Session}_i,\mathsf{Session}_{i+1})=1}\right) \quad \text{Session transition passes ping} \tag{169}$$

$$\varphi_{\mathsf{TLS}} = \varphi^{\mathsf{cert}} \wedge \varphi^{\mathsf{cv}} \wedge \varphi^{\mathsf{fin}} \wedge \varphi^{\mathsf{fresh}} \wedge \varphi^{\mathsf{eph}} \wedge \varphi^{\mathsf{ver}} \wedge \varphi^{\mathsf{suite}} \wedge \varphi^{\mathsf{ping}}.$$

✓ 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 (149) for all* $i \geq 1$.

 doi:10.20944/preprints202603.2237.v1

Proof. The four ping conditions from Definition 14 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	$\leq \mathsf{Adv}^{\mathsf{CDH}} + 3\mathsf{Adv}^{\mathsf{PRF}} + \mathsf{negl}$	$\leq \mathsf{Adv}^{\mathsf{RSA\text{-}PKCS}} + \mathsf{negl}$	$\leq \mathsf{Adv}^{\mathsf{CDH}} + \mathsf{negl}$ (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 ($\varphi^{\mathsf{fresh\text{-}early}}$ 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 [48] 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.

 doi:10.20944/preprints202603.2237.v1

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:

$$\mathsf{plaintext} = s_i \| \mathsf{sess\_id}_i \| \mathsf{msg\_id} \| \mathsf{seqno} \| \mathsf{data}, \tag{170}$$

$$\mathsf{msg\_key} = \mathsf{SHA\text{-}256}(\mathsf{auth\_key}[88{:}120] \| \mathsf{plaintext})[8{:}24], \tag{171}$$

$$(K_{\mathsf{AES}}, \mathsf{IV}_{\mathsf{AES}}) = \mathsf{KDF}(\mathsf{msg\_key}, \mathsf{auth\_key}), \tag{172}$$

$$\mathsf{ciphertext} = \mathsf{AES\text{-}256\text{-}IGE}_{K_{\mathsf{AES}}}(\mathsf{plaintext}; \mathsf{IV}_{\mathsf{AES}}). \tag{173}$$

 doi:10.20944/preprints202603.2237.v1

The transmitted packet is $(\mathsf{auth\_key\_id}, \mathsf{msg\_key}, \mathsf{ciphertext})$.

13.9.2. Security Disproof: Salt Extraction Collapses Confidentiality

Theorem 57 (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 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.
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 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:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \Pr[F_{\mathsf{salt}} \cup F_{\mathsf{entropy}}] = \Pr[F_{\mathsf{salt}}] \cdot (1 + \Pr[F_{\mathsf{entropy}} \mid F_{\mathsf{salt}}]) = 1 \cdot 1. \tag{174}$$

 doi:10.20944/preprints202603.2237.v1

Since we assume F_{salt} occurs with probability 1 (the salt is extracted), and F_{entropy} follows deterministically, the price adjustment is $\Delta\text{Price}_0 = 1$—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 $K = \text{KEM.Dec}(\text{sk}, c)$ 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: $\Delta\text{Price} \le \text{Adv}_g^{\text{CDH}}(\lambda)$ (DH case) or $\text{Adv}_{\text{KEM}}^{\text{IND-CCA2}}(\lambda)$ (KEM case). For X25519: $\text{Adv}^{\text{CDH}} \le 2^{-128}$ under the best known algorithms (Pollard rho on 252-bit curve order). For ML-KEM-768: $\text{Adv}^{\text{IND-CCA2}} \le 2 \cdot \text{Adv}^{\text{MLWE}} + q_D/2^{138}$, 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}:

$$\text{Ask}(g_{\text{IND}} \mid F_{\text{salt}}) \le 1 - 2^{-64} \approx 1. \tag{175}$$

The ask price for indistinguishability approaches 1—the market has quasi-collapsed.

$\mathbf{B}_3$ (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 ($\text{sk} = (s_2e_1 - s_1e_2)(r(s_1 - s_2))^{-1}$); 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 session-CNF for MTProto includes a freshness clause:

$$\varphi_{\text{MTProto}}^{\text{fresh}} = (x_{s_i \notin \mathcal{S}_{\text{used}}}) \wedge (x_{s_i \text{ secret from } \mathcal{A}}).$$

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).

 doi:10.20944/preprints202603.2237.v1

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 $\text{salt} = \text{HMAC}_{\text{auth\_key}}(\text{context})$, 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 $\text{Adv}^{\text{PRF}}_{\text{HMAC}}(\lambda)$, negligible. The overall MTProto market: partial collapse ($\text{Ask} \leq 1 - 2^{-64}$, security margin reduced to 64 bits). RMTP market: full equilibrium restored ($\text{Ask} \leq \text{Adv}^{\text{PRF}} + q_D \cdot 2^{-128} = \text{negl}$).

Session pinging tests that consecutive sessions $\text{Session}_i, \text{Session}_{i+1}$ are structurally distinct. If s_i is known and s_{i+1} can be predicted (salts are negotiated over a potentially observable channel), then $\text{Ping}(\text{Session}_i, \text{Session}_{i+1}) = 0$ whenever $s_i = s_{i+1}$ (which happens with probability $\geq 2^{-64}$ per 30-minute window) or when s_{i+1} can be inferred from s_i.

$$\Pr[\text{Ping} = 0 \mid s_i \text{ known}] \geq \Pr[s_{i+1} \text{ predictable}] \geq 2^{-64}. \tag{176}$$

This is a 2^{64}-fold degradation compared to the ideal $\Pr[\text{Ping} = 0] \leq 2^{-128}$.

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: $\text{Ask}(g_{\text{IND}} \cup g_{\text{CNF}} \cup g_{\text{ping}}) \leq \Pr[F_{\text{salt}} \cup F_{\text{entropy}} \cup F_{\text{cnf}} \cup F_{\text{ping}}] = 1$. The Telegram market has collapsed for all security goods given salt extraction. □

✻ 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: $\tilde{s}_i = \text{HMAC}_{\text{auth\_key}}(s_i \| \text{timestamp})$. 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.

 doi:10.20944/preprints202603.2237.v1

3. **Full message binding:** Include $\tilde{s}_i$ in all cryptographic operations:

$$\mathsf{msg\_key} \leftarrow \mathsf{SHA\text{-}256}\big(\mathsf{auth\_key}[88{:}120] \,\|\, \tilde{s}_i \,\|\, \mathsf{data}\big). \tag{177}$$

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 58 (Remediated MTProto Market Equilibrium)**.** *Under the PRF security of HMAC and the IND-CPA security of AES-256-IGE, the Remediated MTProto (RMTP) achieves:*

$$\mathsf{Ask}(g_{\mathsf{IND}}) \le \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{AES}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}} + q_D \cdot 2^{-128} + \mathsf{negl}(\lambda), \tag{178}$$

$$\mathsf{Ask}(g_{\mathsf{INT}}) \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}} + \mathsf{negl}(\lambda), \tag{179}$$

$$\mathsf{Ask}(g_{\mathsf{CNF}}) \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}} + q_N^2 / 2^{128} + \mathsf{negl}(\lambda). \tag{180}$$

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:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}}. \tag{181}$$

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:

$$|\Pr[\mathbf{B}_1 = 1] - \Pr[\mathbf{B}_2 = 1]| \le q_D \cdot 2^{-128}. \tag{182}$$

$\mathbf{B}_3$ (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: $\Delta\mathsf{Price}_2 \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{SHA256}} \le \mathsf{negl}(\lambda)$.

$\mathbf{B}_4$ (AES Key Replacement Bid). With uniform msg_key, the AES key $(K_{\mathsf{AES}}, \mathsf{IV}_{\mathsf{AES}})$ derived from it is uniform. Replace with independently uniform (K', IV'). This costs $\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF}}$.

$\mathbf{B}_5$ (AES IND-CPA Reduction Bid). With uniform key and IV, the AES-256-IGE encryption is IND-CPA secure:

$$|\Pr[\mathbf{B}_4 = 1] - 1/2| \le \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{AES}}. \tag{183}$$

$\mathbf{B}_6$ (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 (178) follows by summing price adjustments.

Bound (179) (Integrity g_{INT}). A successful injection requires the adversary to produce $(\mathsf{msg\_key}^*, c^*)$ that decrypts to a valid plaintext (passes AES-IGE decryption) and has a consistent $\mathsf{msg\_key}^*$ 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: $\mathsf{Ask}(g_{\mathsf{INT}}) \le \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}} + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{AES}} = \mathsf{negl}$.

 doi:10.20944/preprints202603.2237.v1

Bound (180) **(CNF correctness g_{CNF}).** The RMTP session-CNF is:

$$\varphi_{\mathsf{RMTP}} = (x_{\tilde{s}_i = \mathsf{HMAC}_{\mathsf{auth\_key}}(s_i \| \mathsf{ts})}) \wedge (x_{\tilde{s}_i \notin \mathcal{S}_{\mathsf{used}}}) \wedge (x_{\mathsf{msg\_key\ derived\ correctly}}) \wedge (x_{\mathrm{Ping\ passes}}). \quad (184)$$

Under the honest trace, all clauses are satisfied. Under a dishonest trace (adversary with extracted s_i), the clause $x_{\tilde{s}_i = \mathsf{HMAC}_{\mathsf{auth\_key}}(\ldots)}$ 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 $\mathrm{Adv}^{\mathsf{PRF}}_{\mathsf{HMAC}} + q_N^2/2^{128}$. □

✓ CNF Verification: RMTP Session-CNF Verification

$\varphi_{\mathsf{RMTP}} = (x_{\tilde{s}_i = \mathsf{HMAC}_K(s_i \| \mathsf{ts})}) \wedge (x_{\tilde{s}_i \notin \mathcal{S}_{\mathsf{used}}}) \wedge (x_{\mathsf{msg\_key\ correct}}) \wedge (x_{\mathrm{Ping}})$.

Manual CNF worksheet for RMTP session:

Clause	Check	How	Pass?
$\varphi^{\text{salt-bind}}$	$\tilde{s}_i \stackrel{?}{=} \mathsf{HMAC}_{\mathsf{auth\_key}}(s_i \| \mathsf{ts})$	Recompute HMAC; compare 256 bits.	T/F
$\varphi^{\text{salt-fresh}}$	$\tilde{s}_i \notin \mathcal{S}_{\mathsf{used}}$	Check used-salt log.	T/F
$\varphi^{\text{msg\_key}}$	msg_key = SHA256(auth_key[88:] $\|\|$ $\tilde{s}_i$ $\|\|$ 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 $\mathsf{Session}_{i-1}$?	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: $\Delta\mathrm{Price}_{\mathsf{ping}} \leq q_D/2^{128} = \mathsf{negl}$. RMTP is IND-CPA + INT secure for unbounded sessions by Theorem 3.

☞ 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)	$\leq \mathrm{Adv}^{\mathsf{PRF}} + \mathsf{negl}$
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 57) identifies four simultaneous failure events that the extended difference lemma captures in a single bound. The proof (Theorem 58) 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.

 doi:10.20944/preprints202603.2237.v1

14. Case Study VI: QROM-Based Key Exchange

✻ 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** [49]: 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** [50]: 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) [51] 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 $H : \{0,1\}^* \to \{0,1\}^n$.

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).

$$\mathbf{A} \xleftarrow{\$} R_q^{k\times k}, \quad \mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi^k,$$
$$\mathsf{pk}_C = (\mathbf{A},\ \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}), \quad \mathsf{sk}_C = \mathbf{s}.$$

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: $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_C, m; r)$.

 doi:10.20944/preprints202603.2237.v1

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: $\delta \leq 2^{-139}$.

 doi:10.20944/preprints202603.2237.v1

14.2. Protocol Sequence Diagram

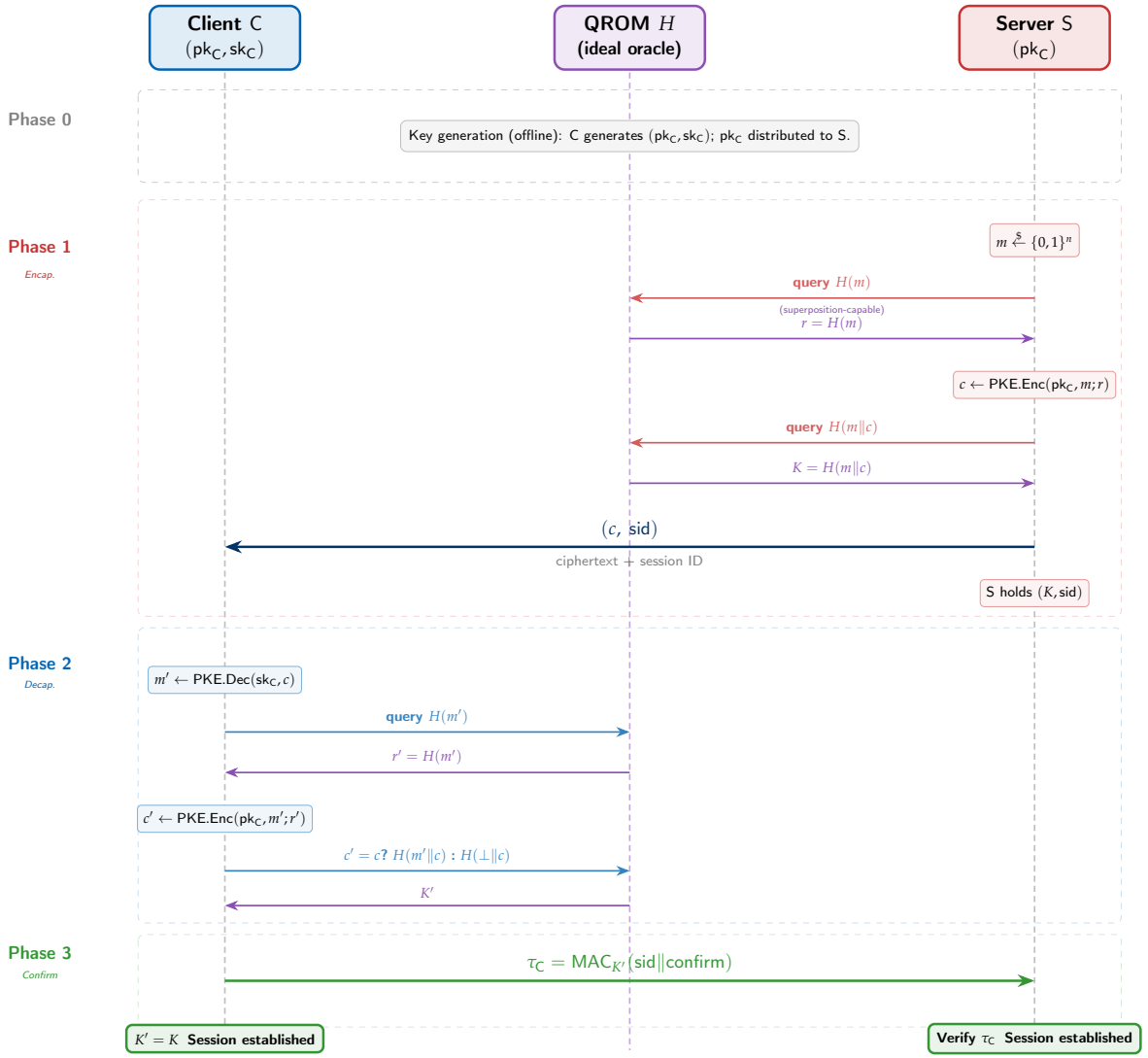


Figure 36. QKEM protocol sequence diagram. Three phases: (0) offline key generation; (1) encapsulation by S making two QROM queries ($H(m)$ and $H(m\|c)$); (2) decapsulation by C making two QROM queries ($H(m')$ and $H(m'\|c)$ or $H(\perp\|c)$); (3) optional MAC confirmation. The QROM oracle H (centre, 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.
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.

 doi:10.20944/preprints202603.2237.v1

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

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 33 (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* $b' \in \{0,1\}$.

$\mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}}) = \max_{\mathcal{A}} |\Pr[\mathcal{A} \text{ wins}] - 1/2|$.

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 $\mathsf{FO}^{\perp}[\mathsf{PKE}, H]$:

$$\mathsf{KEM.Enc}(\mathsf{pk}): \ m \xleftarrow{\$} \{0,1\}^n;\ c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m; H(m));\ K \leftarrow H(m\|c);\ \text{return } (c, K), \tag{185}$$

$$\mathsf{KEM.Dec}(\mathsf{sk}, c): \ m' \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, c);\ c' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m'; H(m'));\ K \leftarrow \begin{cases} H(m'\|c) & c' = c \\ H(\perp\|c) & \text{else} \end{cases}. \tag{186}$$

Theorem 59 (QROM FO-Transform KEM Equilibrium)**.** *Let* PKE *be* δ*-correct and IND-CPA. Then:*

$$\mathsf{Ask}(g_{\mathsf{IND\text{-}CCA2}}) \leq 2(q_H + 1) \cdot \sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}} + q_D \cdot \delta + \mathsf{negl}(\lambda). \tag{187}$$

Proof. We construct six bidding rounds. Each round targets a specific adversarial strategy.

 doi:10.20944/preprints202603.2237.v1

$\mathbf{B}_0$ (Real QROM IND-CCA2—Quantum Distinguishing Bid). The seller runs as in Definition 33. $\mathcal{A}$ 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:

$$|\Pr[\mathbf{B}_0 = 1] - \Pr[\mathbf{B}_1 = 1]| \leq \Pr[F_{\mathsf{decaps}} \cup F_{\mathsf{rej}}] \leq q_D \cdot \delta. \tag{188}$$

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.

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 [49]: For m^* chosen uniformly at random and $\mathcal{A}$ making q_H quantum queries:

$$\left|\Pr\left[\mathbf{B}_1^{H} = 1\right] - \Pr\left[\mathbf{B}_1^{H'} = 1\right]\right| \leq 2(q_H + 1)\sqrt{\Pr[m^* \in S_{\mathcal{A}}]} \leq 2(q_H + 1)\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}}, \tag{189}$$

where $S_{\mathcal{A}}$ is the set of inputs queried with non-negligible amplitude, and $\Pr[m^* \in S_{\mathcal{A}}] \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}$ 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).

 doi:10.20944/preprints202603.2237.v1

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 $\mathrm{Dec}(K_e, K_m, \cdot)$ 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. $\Delta\mathrm{Price} = 0$. The residual adversarial advantage is then exactly $\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{Enc}}(\lambda)$.

Apply measure-and-reprogram [50]: 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:

$$|\Pr[\mathbf{B}_2 = 1] - \Pr[\mathbf{B}_3 = 1]| \leq \sqrt{\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{PKE}}}. \tag{190}$$

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 $K_0' \xleftarrow{\$} \{0,1\}^n$:

$$|\Pr[\mathbf{B}_3 = 1] - \Pr[\mathbf{B}_4 = 1]| = 0. \tag{191}$$

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 $\mathrm{KEM.Dec}(\mathsf{sk}, c)$ is replaced by an *implicit-rejection oracle*: if the re-encryption check $c' = \mathrm{KEM.Enc}(\mathsf{pk}; H(m,c)) \neq c$ 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: $\Pr[\mathrm{KEM.Dec}(\mathsf{sk}, c) \neq K] \leq \delta$. 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-

 doi:10.20944/preprints202603.2237.v1

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:

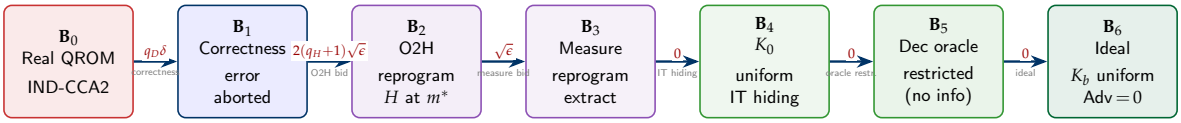
$$|\Pr[\mathbf{B}_4 = 1] - \Pr[\mathbf{B}_5 = 1]| = 0. \tag{192}$$

$\mathbf{B}_6$ (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:

$$\text{Ask} \leq \underbrace{q_D\delta}_{\mathbf{B}_0 \to \mathbf{B}_1} + \underbrace{2(q_H+1)\sqrt{\epsilon}}_{\mathbf{B}_1 \to \mathbf{B}_2} + \underbrace{\sqrt{\epsilon}}_{\mathbf{B}_2 \to \mathbf{B}_3} + \underbrace{0}_{\mathbf{B}_3 \to \mathbf{B}_4} + \underbrace{0}_{\mathbf{B}_4 \to \mathbf{B}_5} + \underbrace{0}_{\mathbf{B}_5 \to \mathbf{B}_6} \leq q_D\delta + 2(q_H+1)\sqrt{\epsilon} + \text{negl}, \tag{193}$$

where $\epsilon = \text{Adv}^{\text{IND-CPA}}_{\text{PKE}}$ and the $\sqrt{\epsilon}$ term is absorbed into $2(q_H+1)\sqrt{\epsilon}$ for $q_H \geq 1$. □



Total Ask $\leq q_D\delta + 2(q_H+1)\sqrt{\text{Adv}^{\text{IND-CPA}}} + \text{negl}(\lambda)$ $(\epsilon = \text{Adv}^{\text{IND-CPA}})$

Figure 37. 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 $|\alpha_{m^*}|^2 \propto \epsilon$, 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.

 doi:10.20944/preprints202603.2237.v1

14.5. Extended Protocol Security Goods

Corollary 17 (QKEM Authentication)**.** *With Phase 3 (MAC confirmation) included, under SUF-CMA security of the MAC:*

$$\mathsf{Ask}(g_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}} + 2(q_H+1)\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}} + q_D\delta + \mathsf{negl}(\lambda). \tag{194}$$

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 59). 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

☞ 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	$q_H \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}$	$2(q_H+1)\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}}$
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

 doi:10.20944/preprints202603.2237.v1

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

This case study instantiates the quantum market dynamics formalisation of Section 18.2.1 on a concrete protocol: the BB84 Quantum Key Distribution (QKD) protocol [52]. 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.

✻ 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.

 doi:10.20944/preprints202603.2237.v1

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 $\{\theta_i^B\}$.
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 $\leq 2^{-\lambda}$).

 doi:10.20944/preprints202603.2237.v1

15.2. Protocol Sequence Diagram

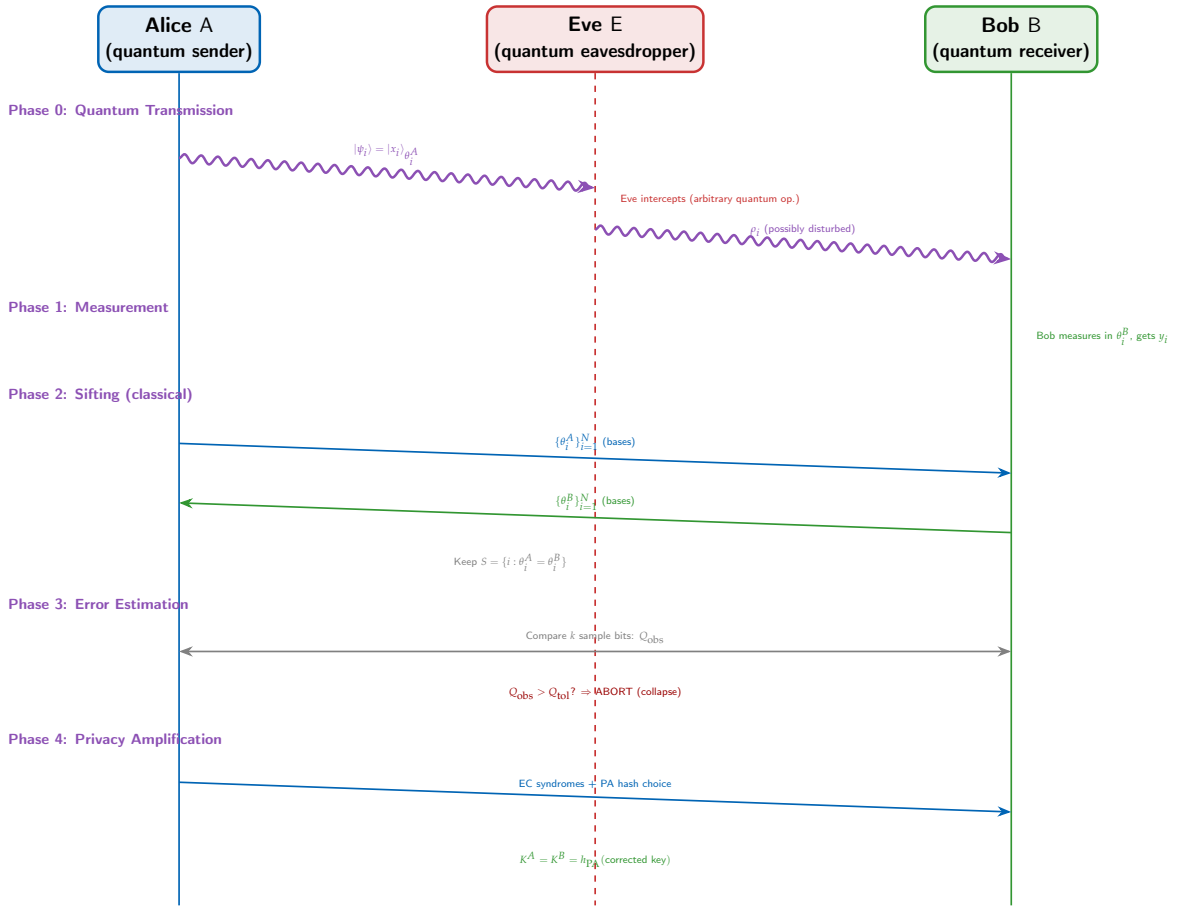


Figure 38. BB84 QKD protocol sequence diagram. Phase 0: Alice prepares qubits in random bases and sends them over a quantum channel (wavy arrows indicate quantum transmission); Eve can intercept and perform arbitrary quantum operations. Phase 1: Bob measures in random bases. Phase 2: Classical sifting 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 Definition 36.

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:

 doi:10.20944/preprints202603.2237.v1

- g_{secrecy}: **Key secrecy**—the final key K is indistinguishable from uniform randomness to Eve. Formally: $\frac{1}{2}\|K \otimes \rho_E - U_\ell \otimes \rho_E\|_{\mathrm{tr}} \leq \mathsf{negl}(\lambda)$, 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 34 (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*** $\mathcal{E}^S_k$*: Prepare*
$$|\psi_k\rangle = |x_k\rangle_{\theta^A_k}$$
and send over the quantum channel.
- ***Buyer's quantum channel*** $\mathcal{E}^B_k$*: An arbitrary CPTP map acting on the intercepted qubit and Eve's ancilla.*
- ***Quantum ask price (cf.* (214)*):***
$$\mathsf{Ask}^{Q}(g_{\mathsf{secrecy}}) = \sup_{\mathsf{Buyer}^{Q}} \frac{1}{2}\|K \otimes \rho_E - U_\ell \otimes \rho_E\|_{\mathrm{tr}}.$$

15.4. Security Proof via Quantum Bidding Rounds

Theorem 60 (BB84 Quantum Market Equilibrium)**.** *Let the BB84 protocol run with parameters N (raw key length), $Q_{\mathrm{tol}} \approx 0.11$, and privacy amplification output length $\ell = n(1 - h(Q_{\mathrm{obs}})) - \lambda$. Then:*

$$\mathsf{Ask}^{Q}(g_{\mathsf{secrecy}}) \;\leq\; 2^{-\lambda/2} + \mathsf{negl}(\lambda). \tag{195}$$

The BB84 quantum market is in equilibrium. Crucially, this holds against all *quantum adversaries, including those with unlimited computational power.*

Proof. We construct five quantum bidding rounds. Each round applies the quantum extended difference lemma (Proposition 6) using trace distance as the quantum price adjustment.

$\mathbf{B}^{Q}_0$ (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 ($Q_{\mathrm{obs}} \leq Q_{\mathrm{tol}}$), 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 18.2.1.

$\mathbf{B}^{Q}_1$ (Bidding Round 1: No-Cloning Bid—Intercept-Resend Bound).

 doi:10.20944/preprints202603.2237.v1

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 18.2.1). 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:

$$I_E^{(i)} \;\leq\; h(Q_i), \tag{196}$$

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:

$$\Delta\text{Price}_1^Q \;=\; \frac{1}{2}\|\rho_{\text{final}}^{(0)} - \rho_{\text{final}}^{(1)}\|_{\text{tr}} \;\leq\; 2^{-\lambda/2}, \tag{197}$$

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: $\Pr[Q_{\text{obs}} \leq Q_{\text{tol}} \mid Q_{\text{true}} > Q_{\text{tol}} + \gamma] \leq e^{-2k\gamma^2}$.
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:

$$\Pr[F_{\text{est}}] \;\leq\; e^{-2k\gamma^2}. \tag{198}$$

For $k = \Theta(\lambda/\gamma^2)$, this is $\leq 2^{-\Omega(\lambda)} = \text{negl}(\lambda)$.

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.

 doi:10.20944/preprints202603.2237.v1

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 $\ell = n(1 - h(Q_{\text{obs}})) - \lambda$:

$$\frac{1}{2}\|K \otimes \rho_E - U_\ell \otimes \rho_E\|_{\text{tr}} \leq 2^{-\lambda/2}. \tag{199}$$

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.

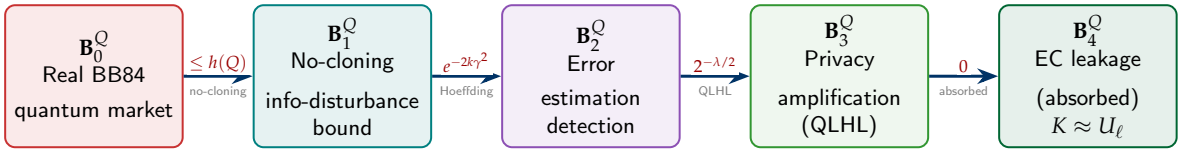
$\mathbf{B}_4^Q$ (Bidding Round 4: Error Correction Leakage Bid).

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:

$$\text{Ask}^Q(g_{\text{secrecy}}) \leq \underbrace{2^{-\lambda/2}}_{\text{QLHL}} + \underbrace{e^{-2k\gamma^2}}_{\text{estimation}} + \underbrace{2^{-\lambda}}_{\text{EC failure}} \leq 2^{-\lambda/2} + \text{negl}(\lambda). \tag{200}$$

The BB84 quantum market is in equilibrium. □



$$\text{Total Ask}^Q(g_{\text{secrecy}}) \leq 2^{-\lambda/2} + e^{-2k\gamma^2} + 2^{-\lambda} \leq 2^{-\lambda/2} + \text{negl}(\lambda)$$

Figure 39. 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.

 doi:10.20944/preprints202603.2237.v1

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* $\delta_{\mathsf{EC}} \leq 2^{-\lambda}$*:*

$$\mathsf{Ask}^{Q}(g_{\mathsf{correct}}) \;\leq\; 2^{-\lambda} = \mathsf{negl}(\lambda). \tag{201}$$

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:*

$$\mathsf{Ask}^{Q}(g_{\mathsf{detect}}) \;\leq\; e^{-2k\gamma^2} + 2^{-\lambda} = \mathsf{negl}(\lambda). \tag{202}$$

✓ CNF Verification: BB84 Session-CNF Verification

$\varphi_{\mathsf{BB84}} = \varphi^{\mathsf{basis}} \wedge \varphi^{\mathsf{error}} \wedge \varphi^{\mathsf{auth}} \wedge \varphi^{\mathsf{PA}} \wedge \varphi^{\mathsf{ping}}$, where $\varphi^{\mathsf{basis}} = (x_{\theta^A=\theta^B \text{ for sifted bits}})$, $\varphi^{\mathsf{error}} = (x_{Q_{\mathsf{obs}} \leq Q_{\mathsf{tol}}})$, $\varphi^{\mathsf{auth}} = (x_{\text{classical channel authenticated}})$, $\varphi^{\mathsf{PA}} = (x_{\ell \leq n(1-h(Q_{\mathsf{obs}}))-\lambda})$.

Manual CNF worksheet (BB84 session):

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	$Q_{\mathsf{obs}} \leq Q_{\mathsf{tol}}$?	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	$\ell \leq n(1-h(Q_{\mathsf{obs}})) - \lambda$?	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 $\mathsf{Session}_i$ to attack session $\mathsf{Session}_{i+1}$. Since each session uses fresh random bits and bases (x_i, θ_i^A are independently sampled), Eve's quantum side information from $\mathsf{Session}_i$ is *product* with the quantum states in $\mathsf{Session}_{i+1}$. By the quantum leftover hash lemma applied independently per session: $\Delta\mathsf{Price}^{Q}_{\mathsf{ping}} \leq 2^{-\lambda/2} = \mathsf{negl}$. By Theorem 3 (extended to quantum markets via Proposition 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
g_{secrecy}	$2^{-\lambda/2} + e^{-2k\gamma^2}$	QLHL	Equilibrium
g_{correct}	$2^{-\lambda}$	EC failure	Equilibrium
g_{detect}	$e^{-2k\gamma^2} + 2^{-\lambda}$	Hoeffding	Equilibrium
$g_{\mathsf{unbounded}}$ (ping)	$2^{-\lambda/2}$	QLHL (per session)	Equilibrium

 doi:10.20944/preprints202603.2237.v1

☞ 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	$\sqrt{\mathsf{Adv}^{\text{IND-CPA}}}$ (O2H)	$2^{-\lambda/2}$ (QLHL)
Key quantum tool	O2H + measure-reprogram	No-cloning + info-disturbance
Bidding rounds	6	5 ($\mathbf{B}_0^{\mathbf{Q}}$–$\mathbf{B}_4^{\mathbf{Q}}$, 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

This case study instantiates the multi-protocol composition network formalisation of Section 18.2.2 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 62) 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.
- **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

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.

 doi:10.20944/preprints202603.2237.v1

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.
- **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 Definition 37 with $N = 2$ component markets.

Definition 35 (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. $\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{PKI}}) \le \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{CA}} \le \mathsf{negl}(\lambda)$ (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. $\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{RNG}}) \le \mathsf{Adv}^{\mathsf{PRG}}_{\mathsf{RNG}} \le \mathsf{negl}(\lambda)$.

 doi:10.20944/preprints202603.2237.v1

16.3. Market Merger and Composition Proof

We apply the market merger theorem (Theorem 62) to the TLS+Signal network.

Theorem 61 (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:*

$$\mathcal{M}_{\mathsf{TLS}} \otimes_{\mathcal{I}} \mathcal{M}_{\mathsf{Signal}} \tag{203}$$

is in equilibrium. Specifically, for every security good g in either protocol:

$$\mathsf{Ask}_{\mathcal{N}}(g) \;\leq\; \mathsf{Ask}_{\mathsf{own}}(g) + \mathsf{Ask}_{\mathsf{other}} + \mathsf{Ask}_{\mathcal{I}} \;\leq\; \mathsf{negl}(\lambda), \tag{204}$$

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 62. 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:

$$\mathsf{Ask}_{\mathsf{TLS}}(g^{(1)}_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{ECDLP}} + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}} + q_N^2/2^{\lambda+1} \leq \mathsf{negl}(\lambda), \tag{205}$$

$$\mathsf{Ask}_{\mathsf{TLS}}(g^{(1)}_{\mathsf{auth}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}} + q_N^2/2^{\lambda+1} \leq \mathsf{negl}(\lambda). \tag{206}$$

Condition 2: Signal market equilibrium. By Section 13.7, all Signal goods are in equilibrium. The X3DH proof gives:

$$\mathsf{Ask}_{\mathsf{Signal}}(g^{(2)}_{\mathsf{sk}}) \leq \mathsf{Adv}^{\mathsf{gap\text{-}CDH}} + q_N^2/2^{\lambda+1} \leq \mathsf{negl}(\lambda), \tag{207}$$

$$\mathsf{Ask}_{\mathsf{Signal}}(g^{(2)}_{\mathsf{FS}}) \leq \mathsf{Adv}^{\mathsf{gap\text{-}CDH}} \leq \mathsf{negl}(\lambda). \tag{208}$$

Condition 3: Infrastructure soundness.

$$\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{PKI}}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{CA}} \leq \mathsf{negl}(\lambda), \tag{209}$$

$$\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{RO}}) = 0 \quad \text{(ideal RO)}, \tag{210}$$

$$\mathsf{Ask}_{\mathcal{I}}(g_{\mathsf{RNG}}) \leq \mathsf{Adv}^{\mathsf{PRG}}_{\mathsf{RNG}} \leq \mathsf{negl}(\lambda). \tag{211}$$

By Theorem 62, the merged market is in equilibrium. The quantitative bound from Remark after Theorem 62 gives:

$$\mathsf{Ask}_{\mathcal{N}}(g^{(1)}_{\mathsf{sk}}) \;\leq\; \underbrace{\mathsf{Ask}_{\mathsf{TLS}}(g^{(1)}_{\mathsf{sk}})}_{\text{own market}} + \underbrace{\mathsf{Ask}_{\mathsf{Signal,max}}}_{\text{simulation overhead}} + \underbrace{\mathsf{Ask}_{\mathcal{I},\mathsf{max}}}_{\text{infrastructure}} \;\leq\; \mathsf{negl}(\lambda), \tag{212}$$

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). □

 doi:10.20944/preprints202603.2237.v1

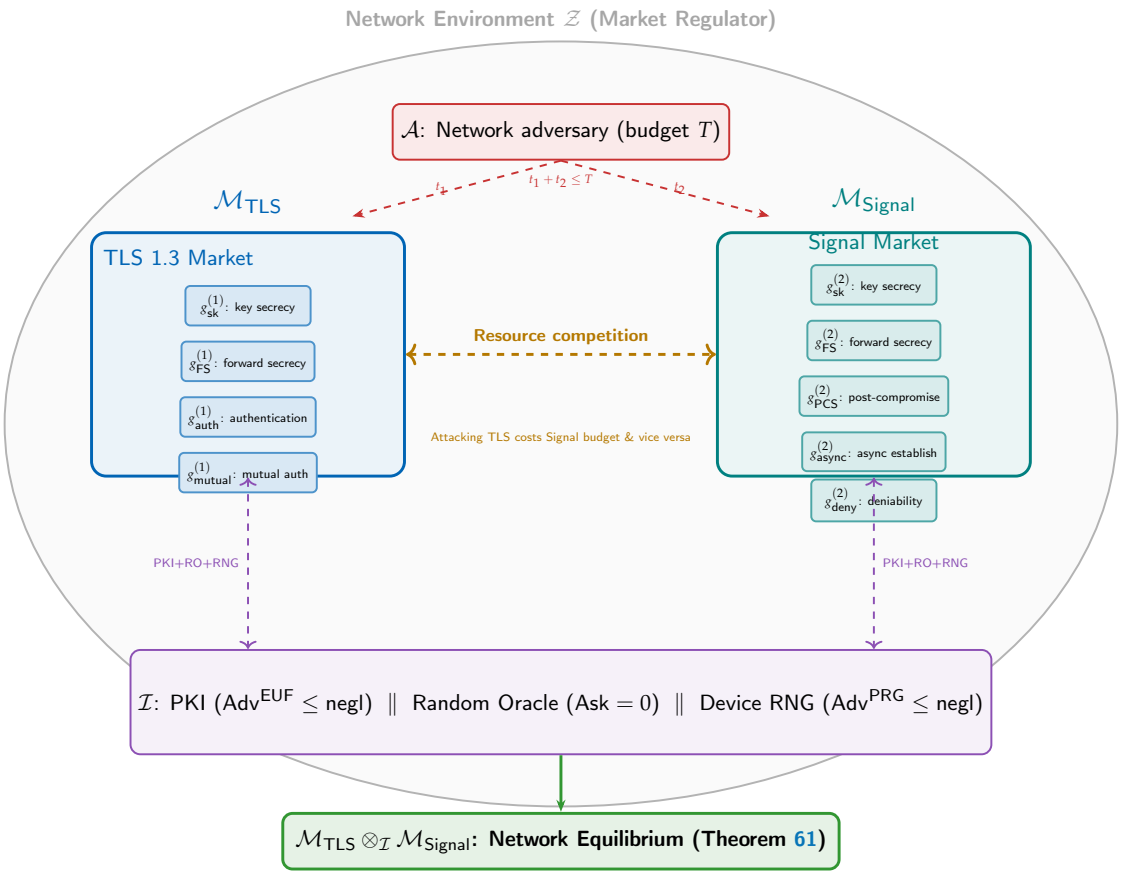


Figure 40. TLS 1.3 + Signal protocol market network. The TLS market (left, blue) and Signal market (right, teal) share infrastructure $\mathcal{I}$ (PKI, random oracle, device RNG). A single network adversary $\mathcal{A}$ with budget T must split its resources ($t_1 + t_2 \leq T$) between the two markets. The orange dashed link indicates resource competition: effort spent attacking TLS reduces resources available for attacking Signal. By the market merger theorem (Theorem 62), 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

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:*

$$\mathsf{Ask}_{\mathcal{N}}(g^{(1)}_{\mathsf{sk}}) + \mathsf{Ask}_{\mathcal{N}}(g^{(2)}_{\mathsf{sk}}) \;\leq\; f_{\mathsf{TLS}}(t_1) + f_{\mathsf{Signal}}(T - t_1) + 2\mathsf{Ask}_{\mathcal{I}}, \tag{213}$$

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*

$$\max(\mathsf{Ask}_{\mathcal{N}}(g^{(1)}_{\mathsf{sk}}), \mathsf{Ask}_{\mathcal{N}}(g^{(2)}_{\mathsf{sk}}))$$

 doi:10.20944/preprints202603.2237.v1

—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

We instantiate Definition 39 for the TLS+Signal network.

✓ 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) $\Rightarrow$ 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 Corollary 20 with $N = 2$), the TLS+Signal network is secure for unbounded sessions across both protocols simultaneously.

 doi:10.20944/preprints202603.2237.v1

Table 19. TLS+Signal market network goods summary. All goods from both protocols plus infrastructure goods, with network ask prices from Theorem 61.

Market	Good	Standalone Ask	Network Ask (Eq. 204)	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)**.** *Corollary 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. The practical takeaway: running many secure protocols on the same device does not degrade security, provided the shared infrastructure (PKI, RNG) is sound.*

17. Related Work

17.1. 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.

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.

 doi:10.20944/preprints202603.2237.v1

17.2. 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.

17.3. 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 [53] 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.

17.4. TLS 1.3 and Protocol Security Analysis

Transport Layer Security 1.3 [45] represents the most thoroughly analysed deployed cryptographic protocol in history. Its design was informed by a decade of attacks on TLS 1.2 [47,54**?**], 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 [48], who prove security of the full TLS 1.3 handshake in the ACCE (Authenticated and Confidential Channel Establishment) model [55]. 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 [56] 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 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 [46,57] is central to TLS 1.3's security. Krawczyk [57] 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

 doi:10.20944/preprints202603.2237.v1

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 [58], 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 55) 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 [59] (forcing SSLv3 negotiation), FREAK [54] (forcing export-grade RSA), and Logjam [60] (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 56) 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.

17.5. 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.

17.6. Economics of Security

Anderson [61] 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.* [62] 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.

17.7. Telegram MTProto

Albrecht *et al.* [63] 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 [64] 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.

17.8. HMAC and Authenticated Encryption

Bellare, Canetti, and Krawczyk [18] 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 [65] formalised authenticated encryption and introduced the nonce-based AE security notion. The Encrypt-then-MAC (EtM) paradigm was proved optimal by Bellare and Namprempre [66]: among three composition

 doi:10.20944/preprints202603.2237.v1

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.

17.9. Post-Quantum Signatures and KEMs

NIST's post-quantum standardisation process produced ML-KEM (FIPS 203) [51], ML-DSA (FIPS 204), SLH-DSA (FIPS 205) [67], and FN-DSA (FIPS 206) [68]. 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 [49] and measure-and-reprogram [50].

17.10. Block Ciphers, Hash Functions, and Stream Ciphers

Differential cryptanalysis was introduced by Biham and Shamir [20] against DES, and the AES design by Daemen and Rijmen [21] was explicitly hardened against it. Rotational cryptanalysis was introduced by Khovratovich and Nikolić [22] and targets operations that commute with rotations (unlike AES's MixColumns). Related-key attacks on AES-256 [23] 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.

PRESENT [24] 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 [25] 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 [26], 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 [27], 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 [28] 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 [29] 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 [30], building on BLAKE2 [31], 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 [32], selected by NIST as the lightweight cryptography standard [33], 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 [34] is an eSTREAM portfolio stream cipher combining an LFSR and NFSR with an optional authentication mode. Known attacks include cube attacks [36], algebraic attacks [35], and classical TMTO attacks [37,38]. MTSF's Grain market (Section 12) bounds all four attack vectors within a single combined market equilibrium.

ChaCha20 [39], a refinement of Salsa20 [40], 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

 doi:10.20944/preprints202603.2237.v1

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 [41] is an eSTREAM portfolio cipher with an elegant three-register design (288-bit state, 80-bit key). Cube attacks [36] and algebraic degree analysis [42] 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.

17.11. 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 [49] 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 [50] allows extracting query points from quantum superpositions without collapsing the proof. Hofheinz, Hövelmanns, and Kiltz [69] 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.

18. Conclusion

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.

 doi:10.20944/preprints202603.2237.v1

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.

 doi:10.20944/preprints202603.2237.v1

18.1. Lessons Learned

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.

 doi:10.20944/preprints202603.2237.v1

18.2. Further Work

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.

18.2.1. 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 36 (Quantum Market)**.** *A* quantum market $\mathcal{M}^{Q} = (\mathsf{Seller}^{Q}, \mathsf{Buyer}^{Q}, \{g_j\}, H^{Q}, \rho_0)$ *extends the classical MTSF market with:*

1. ***Quantum seller*** Seller^{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*** Buyer^{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.*
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:*

$$\mathsf{Ask}^{Q}(g_j) \;=\; \sup_{\mathsf{Buyer}^{Q} \in \mathsf{QPT}} \mathrm{Tr}\Big[\Pi_{g_j} \cdot \mathcal{E}_n^B \circ \mathcal{E}_n^S \circ \cdots \circ \mathcal{E}_1^B \circ \mathcal{E}_1^S(\rho_0)\Big], \tag{214}$$

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

 doi:10.20944/preprints202603.2237.v1

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:

$$\Delta\mathsf{Price}_k^Q \;=\; \frac{1}{2}\left\|\mathcal{E}_k^S(\rho_k) - \mathcal{E}_{k+1}^S(\rho_k)\right\|_{\mathrm{tr}}, \tag{215}$$

   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.

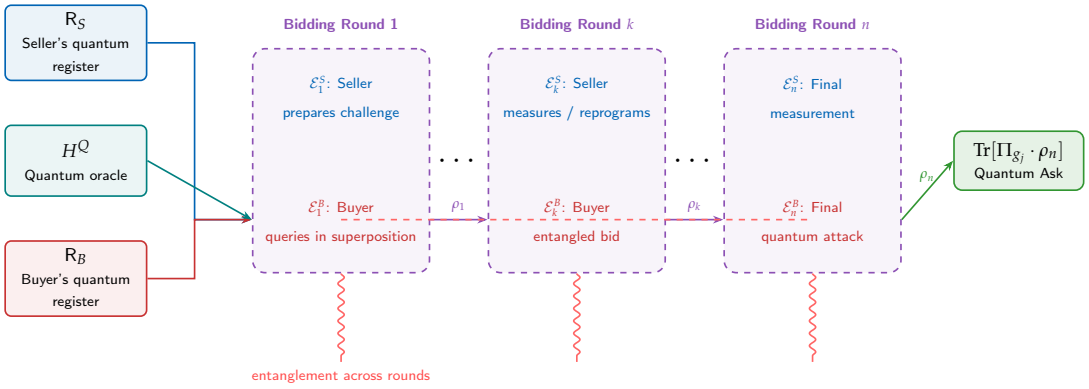


Figure 41. Quantum market dynamics. Both seller and buyer operate on quantum registers (R_S, R_B) through a shared quantum oracle H^Q. Each bidding round applies quantum channels $\mathcal{E}_k^S$ (seller) and $\mathcal{E}_k^B$ (buyer) to the evolving joint state ρ_k. The buyer can maintain entanglement 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 $\frac{1}{2}\|\rho_{\mathrm{final}}^{(k)} - \rho_{\mathrm{final}}^{(k+1)}\|_{\mathrm{tr}} \le \Delta\mathsf{Price}_k^Q$, then the quantum ask price satisfies:*

$$\mathsf{Ask}^Q(g_j) \;\le\; \mathsf{Ask}_0^Q(g_j) \;+\; \sum_{k=0}^{n-1} \Delta\mathsf{Price}_k^Q, \tag{216}$$

where $\mathsf{Ask}_0^Q(g_j) = \mathrm{Tr}[\Pi_{g_j} \cdot \rho_{\mathrm{final}}^{(0)}]$ is the ask price in the initial (ideal) quantum market.

 doi:10.20944/preprints202603.2237.v1

Proof sketch. The trace distance satisfies the triangle inequality: $\|\rho - \sigma\|_{\mathrm{tr}} \leq \|\rho - \tau\|_{\mathrm{tr}} + \|\tau - \sigma\|_{\mathrm{tr}}$. By the operational interpretation of trace distance, the maximum difference in probability of any measurement outcome between $\rho_{\mathrm{final}}^{(0)}$ and $\rho_{\mathrm{final}}^{(n)}$ is at most $\frac{1}{2}\|\rho_{\mathrm{final}}^{(0)} - \rho_{\mathrm{final}}^{(n)}\|_{\mathrm{tr}} \leq \sum_{k=0}^{n-1} \Delta\mathsf{Price}_k^Q$. 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 $\sqrt{\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}}$ arises specifically because the buyer's quantum channel concentrates amplitude on the reprogrammed point—a quantum price adjustment of the form (215) 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.

18.2.2. 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 37 (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.

 doi:10.20944/preprints202603.2237.v1

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 38 (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:*

$$\mathcal{M}_1 \otimes_{\mathcal{I}} \mathcal{M}_2 \;=\; \big(\mathsf{Seller}_{1\|2},\ \mathsf{Buyer}_{1\|2},\ \{g_j^{(1)}\} \cup \{g_j^{(2)}\},\ \varphi_1 \wedge \varphi_2 \wedge \varphi_{\mathcal{I}}\big), \tag{217}$$

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 62 (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:* $\mathsf{Ask}_1(g_j^{(1)}) \le \mathsf{negl}(\lambda)$ *for all goods* $g_j^{(1)}$*;*
2. $\mathcal{M}_2$ *is in equilibrium:* $\mathsf{Ask}_2(g_j^{(2)}) \le \mathsf{negl}(\lambda)$ *for all goods* $g_j^{(2)}$*;*
3. *The shared infrastructure is sound:* $\mathsf{Ask}_{\mathcal{I}}(g_{\mathcal{I}}) \le \mathsf{negl}(\lambda)$ *for all infrastructure goods;*

then the merged market $\mathcal{M}_1 \otimes_{\mathcal{I}} \mathcal{M}_2$ *is in equilibrium:*

$$\mathsf{Ask}_{1\otimes 2}(g) \;\le\; \mathsf{negl}(\lambda) \quad \textit{for all goods } g \in \{g_j^{(1)}\} \cup \{g_j^{(2)}\}. \tag{218}$$

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 $\mathsf{Ask}_1(g_j^{(1)}) \le \mathsf{negl}(\lambda)$. □

Remark 10 (Quantitative Merger Bound). *The proof gives a concrete bound. If* $\mathsf{Ask}_1(g_j^{(1)}) \le \epsilon_1$, $\mathsf{Ask}_2(g_j^{(2)}) \le \epsilon_2$, *and* $\mathsf{Ask}_{\mathcal{I}}(g_{\mathcal{I}}) \le \epsilon_{\mathcal{I}}$, *then:*

$$\mathsf{Ask}_{1\otimes 2}(g_j^{(1)}) \;\le\; \epsilon_1 + \epsilon_2 + \epsilon_{\mathcal{I}}. \tag{219}$$

 doi:10.20944/preprints202603.2237.v1

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:*

$$\mathsf{Ask}_{\mathcal{N}}(g_j^{(i)}) \;\leq\; \epsilon_i + (N-1)\cdot\epsilon_{\max} + \epsilon_{\mathcal{I}}, \tag{220}$$

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.

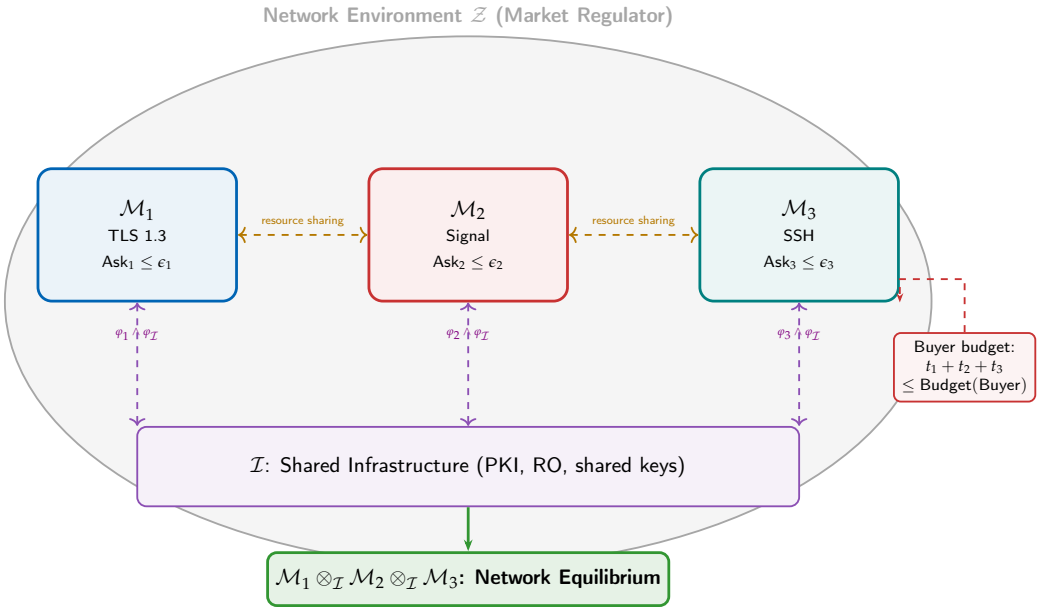


Figure 42. Multi-protocol composition network. Three protocol markets ($\mathcal{M}_1$: TLS 1.3, $\mathcal{M}_2$: Signal, $\mathcal{M}_3$: SSH) share infrastructure $\mathcal{I}$ (PKI, random oracle, shared keys). The network environment $\mathcal{Z}$ (market regulator) monitors all markets simultaneously. 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 62, Corollary 20).

Definition 39 (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:*

$$\varphi_{\mathcal{N}} \;=\; \bigwedge_{i=1}^{N} \varphi_i \;\wedge\; \varphi_{\mathcal{I}}, \tag{221}$$

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 $\mathcal{M}_k$)*

 doi:10.20944/preprints202603.2237.v1

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 for unbounded sessions across all N protocols simultaneously, provided each component market's ping bid is negligible.

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.

Acknowledgements

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.

Dedication

This work is dedicated to the Omnipotent and Mother Nature!!!

— Basker Palaniswamy,
Cork City, Ireland, European Union,
March 2026.

 doi:10.20944/preprints202603.2237.v1

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

Continued on next page

doi:10.20944/preprints202603.2237.v1

Table A1 (continued from previous page)

Classical Concept	MTSF Concept	Market Status / Notes
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 ($\mathsf{Ask} = 0$)	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 ($\mathsf{Ask} = 1$)	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
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	$\mathsf{Ask}_{\mathcal{N}} \leq \sum \epsilon_i + \epsilon_{\mathcal{I}}$

doi:10.20944/preprints202603.2237.v1

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 $\mathbf{B}_1$

Continued on next page

doi:10.20944/preprints202603.2237.v1

Table A2 (continued from previous page)

Category	Bid Type	Target	Price Bound	Example
	KEM key recovery bid	IND-CCA2 break	$\mathsf{Adv}^{\text{IND-CCA2}}$	ISO 2P $\mathbf{B}_3$
	Impersonation bid	Entity authentication	$\mathsf{Adv}^{\text{EUF}} + q_N^2/2^{\lambda+1}$	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	$2 \cdot \mathsf{Adv}^{\text{EUF}}$	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	$\mathsf{Adv}^{\text{SUF-CMA}}$	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	$\mathsf{Adv}^{\text{PRF}}_{\text{HMAC}}$	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	$h(Q_{\text{obs}})$	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	$\mathsf{Ask}_{\text{other}}$	TLS+Signal
	Infrastructure forgery bid	CA cert forgery	$\mathsf{Adv}^{\text{EUF}}_{\text{CA}}$	Shared PKI
	RNG degradation bid	PRG distinguishing	$\mathsf{Adv}^{\text{PRG}}_{\text{RNG}}$	Shared RNG
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	$\mathsf{Adv}^{\text{EUF}}$	PKI ping

 doi:10.20944/preprints202603.2237.v1

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

 doi:10.20944/preprints202603.2237.v1

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.
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. Kocher, P.C.; Jaffe, J.; Jun, B. Differential Power Analysis. In Proceedings of the Advances in Cryptology – CRYPTO 1999. Springer, 1999, Vol. 1666, *LNCS*, pp. 388–397.
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. Duc, A.; Dziembowski, S.; Faust, S. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. In Proceedings of the Advances in Cryptology – EUROCRYPT 2019. Springer, 2019, Vol. 11477, *LNCS*, pp. 466–495.
18. Bellare, M.; Canetti, R.; Krawczyk, H. Keying Hash Functions for Message Authentication. In Proceedings of the CRYPTO, 1996, Vol. 1109, *LNCS*, pp. 1–15.
19. NIST. FIPS 197: Advanced Encryption Standard (AES). Technical report, NIST, 2001.
20. Biham, E.; Shamir, A. Differential Cryptanalysis of DES-like Cryptosystems. In Proceedings of the CRYPTO, 1991, Vol. 537, *LNCS*, pp. 2–21.
21. Daemen, J.; Rijmen, V. *The Design of Rijndael: AES—The Advanced Encryption Standard*; Springer, 2002.
22. Khovratovich, D.; Nikolić, I. Rotational Cryptanalysis of ARX. In Proceedings of the FSE, 2010, Vol. 6147, *LNCS*, pp. 333–346.
23. 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.
24. 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.
25. Cho, J.Y. Linear Cryptanalysis of Reduced-Round PRESENT. In Proceedings of the CT-RSA 2010. Springer, 2010, Vol. 5985, *LNCS*, pp. 232–248.
26. Anderson, R.; Biham, E.; Knudsen, L. Serpent: A Proposal for the Advanced Encryption Standard. In Proceedings of the First AES Candidate Conference, 1998.
27. 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.
28. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. The Keccak Reference. In Proceedings of the Submission to NIST SHA-3 Competition, 2011.

 doi:10.20944/preprints202603.2237.v1

29. 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.
30. 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.
31. 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.
32. 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.
33. National Institute of Standards and Technology. Lightweight Cryptography Standardization Process, 2023. NIST Selected Ascon.
34. 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.
35. Courtois, N.T. Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In Proceedings of the CRYPTO, 2003, Vol. 2729, *LNCS*, pp. 176–194.
36. 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.
37. 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.
38. Hellman, M.E. A Cryptanalytic Time-Memory Trade-Off. *IEEE Transactions on Information Theory* **1980**, *26*, 401–406.
39. Bernstein, D.J. ChaCha, a variant of Salsa20. In Proceedings of the Workshop Record of SASC 2008, 2008.
40. 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.
41. 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.
42. Makarim, R.H.; Rockmore, D.N. On the Algebraic Degree of Trivium Output Bits. 2021.
43. Marlinspike, M.; Perrin, T. The X3DH Key Agreement Protocol. Signal Foundation, 2016. https://signal.org/docs/specifications/x3dh/.
44. Perrin, T.; Marlinspike, M. The Double Ratchet Algorithm. Signal Foundation, 2016. https://signal.org/docs/specifications/doubleratchet/.
45. Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3. IETF RFC 8446, 2018. https://www.rfc-editor.org/rfc/rfc8446.
46. Krawczyk, H.; Eronen, P. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). IETF RFC 5869, 2010.
47. 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.
48. 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.
49. 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.
50. 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.
51. 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.
52. 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.
53. 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.
54. 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.

 doi:10.20944/preprints202603.2237.v1

55. 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.
56. 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.
57. Krawczyk, H. Cryptographic Extraction and Key Derivation: The HKDF Scheme. *IACR Cryptology ePrint Archive* **2010**, *2010*, 264.
58. 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.
59. Möller, B.; Duong, T.; Kotowicz, K. This POODLE Bites: Exploiting the SSL 3.0 Fallback. In Proceedings of the Google Security Advisory, 2014.
60. Adrian, D.; et al. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In Proceedings of the CCS. ACM, 2015, pp. 5–17.
61. Anderson, R. Why Information Security is Hard. In Proceedings of the ACSAC, 2001, pp. 358–365.
62. Grossklags, J.; Christin, N.; Chuang, J. Secure or Insure? In Proceedings of the WWW, 2008, pp. 209–218.
63. Albrecht, M.R.; et al. Four Attacks and a Proof for Telegram. In Proceedings of the IEEE S&P, 2022, pp. 1–17.
64. Jakobsen, J.; Orlandi, C. On the CCA (In)security of MTProto. In Proceedings of the SPSM, 2016, pp. 113–116.
65. Rogaway, P. Authenticated-Encryption with Associated-Data. In Proceedings of the CCS. ACM, 2002, pp. 98–107.
66. 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.
67. NIST. FIPS 205: SLH-DSA. Technical report, NIST, 2024.
68. NIST. FIPS 206: FN-DSA. Technical report, NIST, 2024.
69. 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.