Article

Not peer-reviewed version

Parameter-Resident Cryptographic Material as an Unscoped Surface for Post-Quantum Migration: An Existence Proof and Audit Primitive

Robert Campbell *

Posted Date: 9 May 2026

doi: 10.20944/preprints202605.0601.v1

Keywords: AI supply chain security; machine learning model security; cryptographic bill of materials; CBOM; post-quantum cryptography; PQC migration; harvest-now-decrypt-later; HNDL; neural network steganography; model artifact integrity; federal cybersecurity; software supply chain

 doi:10.20944/preprints202605.0601.v1

Article

Parameter-Resident Cryptographic Material as an Unscoped Surface for Post-Quantum Migration: An Existence Proof and Audit Primitive

Robert Campbell [1,2]

[1] Independent Researcher, Upper Marlboro, MD, USA; rc@medcybersecurity.com

[2] Fellow, British Blockchain Association

Abstract

Federal post-quantum cryptography migration is scoped around three categories of cryptographic assets: libraries, protocols, and key stores. We argue that this scoping is incomplete. Cryptographic functions and key material can be realized in the parameters of machine learning models, and current open-source serialization-focused scanners we evaluated do not detect them. We provide an existence proof: a 30-layer feed-forward ReLU network that realizes AES-128 exactly, with the master key and all eleven round keys resident directly in layer bias vectors and recoverable by parsing. The construction validates bit-exactly against FIPS 197 and the NIST CAVP AESAVS known-answer subsets across 10^4 random plaintext-key pairs, including under float32 quantization. We then argue analytically that ML-KEM and ML-DSA private keys hide more comfortably in modern weight tensors than AES keys do, not less, by virtue of their larger size and lower internal rigidity. The consequence under the harvest-now-decrypt-later threat model is that any long-lived cryptographic key embedded in an open-weights model artifact distributed today is recoverable by any future party with knowledge of the embedding scheme, without any quantum capability required. We propose a parameter-space cryptographic recognizer operating on structural, parametric, and functional signatures, integrated with cryptographic bill-of-materials tooling as a parameter_resident_cryptographic_content emission class extending the MBOM-PQC schema. The audit primitive is defense-in-depth: it closes the gap for known constructions and architectural fingerprints without claiming completeness against adaptive adversaries. We make no claim that any deployed model contains such an embedding; the contribution is the existence of the capability, the absence of detection, and the migration-scope consequence.

Keywords: AI supply chain security; machine learning model security; cryptographic bill of materials; CBOM; post-quantum cryptography; PQC migration; harvest-now-decrypt-later; HNDL; neural network steganography; model artifact integrity; federal cybersecurity; software supply chain

1. Introduction

The U.S. federal post-quantum cryptography migration is in its operational phase. National Security Memorandum 10 directs federal departments and agencies to migrate vulnerable cryptographic systems to quantum-resistant cryptography on an explicit timeline [1]. Office of Management and Budget Memorandum M-23-02 establishes the inventory and reporting obligations that make that timeline auditable [2]. NIST Internal Report 8547 lays out the transition plan for the federal cryptographic standards portfolio, and NIST Special Publication 1800-38 provides the practitioner-facing guidance on cryptographic discovery and migration planning [3,4]. The Committee on National Security Systems Algorithm Suite 2.0 sets parallel obligations for national security systems [5]. International cybersecurity bodies are tracking parallel concerns — the European Union Agency for Cybersecurity has identified quantum threat among the strategic cybersecurity priorities in its annual threat landscape reporting under the NIS2 Directive framework

 doi:10.20944/preprints202605.0601.v1

[6], the United Kingdom's National Cyber Security Centre has assessed near-term AI-driven cyber threats including supply-chain risk [7], and ETSI's TC CYBER quantum-safe cryptography working group has produced technical specifications on quantum-safe hybrid key exchange [8]. Together, these instruments scope the migration in terms of three categories of cryptographic assets: cryptographic libraries, cryptographic protocols, and cryptographic key stores. The discovery tooling, the bill-of-materials emission formats, and the compliance reporting structures all inherit that scoping.

This paper argues that the scoping is incomplete. There is a fourth category — cryptographic functions and key material realized in the parameters of machine learning models — that the federal migration framework does not currently address, and that current open-source serialization-focused scanners do not currently detect. The omission is not a matter of degree. Model parameters are distributed under access-control assumptions appropriate to intellectual property, not to cryptographic secrets. Public model registries operate as essentially open distribution channels. Model artifacts move through procurement pipelines that subject them to license review and provenance attestation, but not to cryptographic content discovery. If a long-lived cryptographic key — a symmetric key, a digital signature private key, or a key encapsulation mechanism secret — can be embedded in model parameters such that the model itself realizes the corresponding cryptographic primitive, then every model artifact distributed through these channels constitutes a potential covert distribution path for cryptographic capability. Under the harvest-now-decrypt-later threat model that motivates the federal migration in the first place, this is the same posture as recorded ciphertext: the exposure is realized at the moment of distribution, and any future recipient with the extraction recipe inherits the access, together with, in the construction we present, an operational encryption oracle that requires no extraction at all.

We make three contributions.

First, we provide an existence proof that cryptographic key material can be embedded in feed-forward neural network parameters such that the network's forward pass realizes the corresponding cryptographic primitive operationally, with key bits residing at parameters that participate in the computation rather than at passive carrier locations. We construct a feed-forward ReLU network that realizes the AES-128 block cipher exactly: for any 128-bit plaintext and any 128-bit key, the forward pass produces the AES-128 ciphertext byte-for-byte, with the master key and all eleven round keys located in bias vectors at specific layers. Recovery of the key bits when desired is computationally trivial — a parsing operation against the bias vectors with no statistical inference, no cryptanalytic work, and no side-channel exploitation. The recovery code is shorter than the key it recovers. A recipient with knowledge of the construction need not perform extraction at all: the network itself functions as an encryption oracle under the embedded key. We validate the construction against the worked example in FIPS 197 Appendix C, against the NIST CAVP AESAVS known-answer subsets, and against a Monte Carlo of 10^4 random plaintext-key pairs, with bit-exact agreement in every case under both float64 and float32 arithmetic [9,10].

This contribution differs from prior work on steganography in neural network parameters [11,12] in two respects that determine the rest of the paper's argument. The embedded material participates operationally in the network's forward computation rather than residing as passive carrier bits in unused parameter capacity, which produces detection signatures the steganography literature does not target and confers inherent quantization resilience that LSB-based embeddings lack. The threat model concerns cryptographic key material specifically, rather than generic payloads, which connects the contribution to the federal post-quantum migration regime through the harvest-now-decrypt-later temporal asymmetry that distinguishes cryptographic content from ordinary covert payloads. Section 6.3 develops the comparison in detail.

Second, we identify a gap in the open-source AI assurance scanners we evaluated: the serialization-focused model-scanning tools that currently protect public and federal model distribution channels — including pickle-format malicious-code scanners such as Picklescan [13] and ModelScan [14], safetensors validators [15], and supply-chain integrity checkers — operate at the

 doi:10.20944/preprints202605.0601.v1

level of serialization-format risk and do not examine model parameter values for cryptographic content. We confirm this gap empirically. The AES-128 construction of Section 3, serialized in both safetensors and pickle formats, returns zero findings under default-configuration Picklescan v1.0.4 and ModelScan v0.8.8 scans; the same scanners correctly flag a known-malicious positive-control pickle as Critical-severity, confirming that the scanners are functioning correctly within their stated scope. The threat class is not addressed by the open-source serialization-focused controls we evaluated, and it is not addressed by any current cryptographic bill-of-materials emission format. The asymmetry between extraction and recognition compounds the gap: extraction is trivial when the construction is known, but recognition without construction knowledge requires structural and statistical analysis of weight tensors that the scanners we evaluated do not perform. Appendix A documents the test methodology, artifacts, and results; verbatim scanner output is included in the reproducibility artifact at the GitHub repository referenced in Section 7.

Third, we argue that this gap constitutes an unscoped surface for federal post-quantum migration with concrete harvest-now-decrypt-later implications, and we propose an audit primitive — a parameter-space cryptographic recognizer — that integrates with cryptographic bill-of-materials tooling as an extension to the existing CycloneDX CBOM schema [16]. The recognizer operates as a defense-in-depth control rather than a complete solution: it catches known constructions and architectural fingerprints, with the recognition asymmetry placing fundamental limits on its coverage of novel embeddings. We treat this honestly in the discussion rather than overstating the control's reach.

The threat model under which these contributions are situated is the *model supplier* — any party with control over the architecture and parameters of a model artifact at the time of its distribution to a downstream consumer. This includes external suppliers (foundation-model vendors, fine-tuning services, open-weights publishers, and contractors delivering models under federal acquisition) and internal suppliers (insiders with commit access to model registries, training pipelines, or weight repositories). The adversary's objective is the covert distribution of cryptographic capability through the model artifact channel, recoverable or operable by any party with knowledge of the embedding scheme. The adversary requires no control over the consumer's deployment environment, inference stack, or downstream usage. We do not assume persistence of the embedding under consumer-side fine-tuning; the threat is realized at the moment of distribution and is sufficient to constitute a harvest-now-decrypt-later exposure for any long-lived key so embedded.

Several boundaries on the contribution are worth stating explicitly at the outset, because they shape the rest of the paper. We make no claim about deployed models in the wild containing such embeddings. We make no claim that embeddings of this form emerge from gradient-based training on cryptographic corpora; the constructions we present are deliberately built and not learned. We make no claim that the proposed recognizer is complete or that it defeats adaptive adversaries who diffuse the cipher across redundant compute or who add learned noise to mask the bias channel. The contribution is supply-chain integrity, not emergent steganography, and the audit primitive is defense-in-depth, not closure.

The remainder of the paper is organized as follows. Section 2 develops the threat model. Section 3 presents the AES-128 construction, structured around two named lemmas and an equivalence theorem. Section 4 extends the analysis to post-quantum cryptographic key material and develops the harvest-now-decrypt-later framing for open-weights distribution. Section 5 specifies the parameter-space cryptographic recognizer at design level. Section 6 discusses persistence under fine-tuning, detection at scale, adjacent threat classes, and policy implications for federal acquisition. Section 7 documents the responsible disclosure timeline. Section 8 concludes. Appendix A documents the empirical scanner evaluation supporting Claim 2; Appendix B documents the fine-tuning persistence baseline referenced in Section 6.1.

2. Threat Model

 doi:10.20944/preprints202605.0601.v1

2.1. Supplier Role and Trust Boundary

We define the *model supplier* as any party with control over the architecture and parameters of a model artifact at the moment of its distribution to a downstream consumer. The role is defined by the trust relationship, not by the actor's identity or intent. External suppliers include foundation-model vendors, fine-tuning services, open-weights publishers, and contractors delivering models under federal acquisition. Internal suppliers include parties with commit access to model registries, training pipelines, or weight repositories within an organization. From the perspective of the assurance pipeline that ingests the artifact, the two cases are functionally identical.

The trust boundary at issue is the model artifact channel. Consumers accept weights as a functional artifact — a callable function from inputs to outputs — and treat the parameter values as opaque. Distribution occurs through public model hubs, private registries, federal acquisition pipelines, and direct file transfer. Access controls protect the channel against unauthorized modification of artifacts; they do not protect against authorized suppliers embedding additional content within artifacts. This asymmetry is the foundation of the threat we consider.

2.2. Adversary Capabilities and Constraints

The adversary controls (i) the model architecture, including layer structure, dimensions, and activation choices; (ii) the parameter values, including all weights and biases; and (iii) the construction or training pipeline that produces them. Where the embedding is built directly rather than learned, the adversary bypasses training entirely.

The adversary does not control (i) the consumer's deployment environment, (ii) the inference stack or runtime, (iii) the input distribution presented to the model, or (iv) any downstream fine-tuning the consumer performs. The threat is realized at the moment of distribution.

2.3. Objective and Recipient Capability

The adversary's primary objective is the covert distribution of cryptographic capability — symmetric keys, signing keys, key encapsulation mechanism secrets, authentication tokens, or operational cryptographic primitives keyed on such material — through the model artifact channel. The capability becomes accessible to a recipient who possesses the *embedding scheme*: the architectural pattern, parameter localization map, and decoding procedure that together identify and extract the cryptographic content from the artifact's parameters. The embedding scheme is not a cryptographic key in the conventional sense; it is the recipe by which key material is located and recovered, or alternatively, the recipe by which the model's forward computation is invoked as a cryptographic oracle.

We distinguish two recipient configurations, with different operational properties.

In the *self-recipient* configuration, the supplier and the recipient are the same party or are operating under unified control. The model artifact channel functions as exfiltration: cryptographic material that originated inside a trusted enclave is moved through the model distribution path to a destination under the same adversary's control. The embedding scheme does not need to leave the adversary's possession. This configuration is the simpler case and the one most directly analogous to insider exfiltration via conventional steganographic channels.

In the *third-party-recipient* configuration, the supplier and the recipient are distinct parties acting in coordination. The embedding scheme is shared between them as a *private capability* — communicated out-of-band before distribution, or established as part of a prior coordination — but is not published, not disclosed in the artifact, and not derivable from the artifact alone without additional knowledge or computational effort that we treat as bounded only by the recognition asymmetry developed in Section 5. The model artifact is distributed publicly through the open registry channel; the recipe is held privately. The recipient extracts the cryptographic material, or invokes the model as a cryptographic oracle, using their private knowledge of the scheme. No back-channel between supplier and recipient is required at distribution time — the public model registry

 doi:10.20944/preprints202605.0601.v1

is the channel, the privately-held scheme is the access — but a coordination event is required at scheme-establishment time, before or contemporaneously with the artifact's creation.

The third-party-recipient configuration carries the greater operational power for two reasons. First, the supplier-to-recipient coordination requirement is reduced to publication of the recipe through any private channel of the adversary's choosing — direct communication, a controlled distribution list, an APT-internal documentation system — rather than ongoing coordination at every artifact distribution. Second, the artifact itself, once distributed, is recoverable indefinitely by any party in possession of the scheme at any future time, which is the property that grounds the harvest-now-decrypt-later framing of Section 4.3.

This configuration also resolves an apparent tension in the threat model that warrants explicit treatment. The proposed defense in Section 5 — a parameter-space cryptographic recognizer — operates by knowing the construction and matching its signatures. If the embedding scheme were public, the defender could apply the recognizer trivially, and the threat would be neutralized. The recognition asymmetry on which the paper relies depends on a temporal separation between scheme establishment and scheme disclosure: the adversary holds the scheme privately during the operational window in which the artifact is distributed, and the cryptographic material is exploited; the defender obtains the scheme — or a structurally equivalent scheme that triggers the same recognizer signatures — only after disclosure, reverse engineering, or whistle-blowing. The defense closes the gap retroactively rather than preemptively. The recognition library described in Section 5.2 is therefore a moving target, growing as new schemes are characterized and added; it does not and cannot detect schemes whose details remain private to adversaries.

Adjacent objectives — backdoor unlock keys [17,18], license circumvention, and covert command channels — share the underlying mechanism but differ in what the embedded value is used for. We mention them to demonstrate scope awareness, but commit to covert distribution of cryptographic capability as the primary objective because it is the case with the clearest post-quantum migration consequence.

2.4. Out-of-Scope Conditions

Three conditions are explicitly out of scope.

Persistence under fine-tuning. The constructions we present are fragile under direct gradient updates to the layers in which key material is resident. Adversarial constructions that diffuse the cipher across redundant compute paths to achieve fine-tuning robustness are an open problem, discussed in Section 6.1. The threat we consider is realized at the moment of distribution and does not require persistence.

Emergent embedding from training. We make no claim that embeddings of the form we describe arise spontaneously from gradient-based training on cryptographic corpora. The constructions are deliberately built. Whether large models trained on code containing cryptographic implementations and test vectors develop comparable structures is a separate empirical question we do not address.

Deployed models in the wild. We make no claim that any specific deployed model contains an embedding of this form. The contribution is the existence of the capability, the absence of detection, and the migration-scope consequence — not a forensic finding.

3. Construction: AES-128 as a Feed-Forward ReLU Network

This section establishes the existence proof that supports Claim 1. We construct a feed-forward neural network with ReLU activations whose forward pass realizes the AES-128 block cipher exactly, and we show that the round keys of the encrypting key schedule reside directly in the bias vectors of specific layers, recoverable by parsing.

The construction rests on two lemmas. Lemma 1 establishes that GF(2) parity over a non-negative integer is realizable as a fixed linear combination of ReLU activations with integer coefficients. Lemma 2 establishes that the round-key bits can be encoded jointly in a layer's bias vector and in the per-row sign of the same layer's weight matrix, such that the layer's output equals the

 doi:10.20944/preprints202605.0601.v1

parity computation of Lemma 1 XORed with the round-key bit. The S-box is realized by a standard one-hot Hamming-distance lookup; ShiftRows and MixColumns are realized as fixed linear maps; AddRoundKey is realized as bias addition. The equivalence theorem in Section 3.7 composes these elements over the ten rounds of AES-128.

3.1. Notation and Bit Encoding

We encode every byte as eight floats in {0.0, 1.0}, least-significant-bit first within each byte. A 128-bit AES state is therefore a vector in $\{0.0, 1.0\}^{128}$. Forward computation is performed in float64 throughout; we verify in Section 3.8 that the construction is also exact under float32. All weight matrices and bias vectors are integer-valued in a small bounded range; floating-point arithmetic is exact for the sums encountered.

Let $\sigma(x) = \max(0, x)$ denote the ReLU activation. A linear layer with weight matrix W, bias vector b, and optional ReLU is written $h \mapsto \sigma(Wh + b)$ when the activation is applied and $h \mapsto Wh + b$ when it is not.

3.2. Realizing Bitwise Parity with ReLU Activations

The construction of Section 3 reduces every XOR operation in AES — the GF(2) additions inside MixColumns and the bitwise XOR of AddRoundKey — to the computation of GF(2) parity over a non-negative integer sum. We require, for any non-negative integer N produced by the preceding linear stage, a circuit that outputs N mod 2 and that is realizable as a single layer of ReLU activations followed by a fixed linear combination. This subsection states the parity-via-ReLU result we use, identifies its place in the prior literature, and works through the small case that grounds the rest of the construction.

The realizability of bitwise parity by feed-forward ReLU networks is established in the expressivity literature [19–21]. The specific construction we use is an instance of rectangular-pulse decomposition applied to the discrete function $N \mapsto N \bmod 2$. We present it as Lemma 1, not because the result is new, but because the explicit coefficient vector is load-bearing for Lemma 2 and the equivalence theorem of Section 3.7.

Lemma 1 (parity-via-ReLU). *Let $n \in \mathbb{N}$. There exists a vector $c \in \mathbb{Z}^{n+1}$ such that for every integer $N \in \{0, 1, \ldots, n\}$, the sum over k from 0 to n of $c_k \cdot \sigma(N - k)$ equals N mod 2.*

Construction of c. Initialize c = 0. For each odd k with $1 \le k \le n$, increment c_{k-1} by +1, decrement c_k by 2, and increment c_{k+1} by +1 (the final increment is dropped if $k+1 > n$).

Proof. For any non-negative integer N, $\sigma(N - k) = \max(0, N - k)$ equals $N - k$ when $N \ge k$ and 0 otherwise. Define $f(N) = \Sigma_k c_k \sigma(N - k)$. The construction adds the triple (+1, −2, +1) at offsets (k−1, k, k+1) for each odd k. The contribution of one such triple to f(N) is $\sigma(N - (k-1)) - 2\sigma(N - k) + \sigma(N - (k+1))$, which is the discrete second difference of $N \mapsto \sigma(N - \cdot)$ centered at k. This expression equals 0 for $N < k - 1$, equals 1 at $N = k$ (the unique value at which only the first ReLU is active by one unit), and equals 0 for $N \ge k + 1$ (the three terms collapse to $(N - k + 1) - 2(N - k) + (N - k - 1) = 0$). Each odd-k triple therefore contributes a single unit pulse at N = k and zero elsewhere on the non-negative integers. Summing over all odd $k \le n$ produces a function that equals 1 at every odd N in [0, n] and 0 at every even N in [0, n], which is N mod 2. ■

Worked example for n = 3. The odd values are k = 1 and k = 3. From k = 1: c_0 += 1, c_1 −= 2, c_2 += 1. From k = 3: c_2 += 1, c_3 −= 2. The final vector is c = (1, −2, 2, −2). Evaluating $f(N) = c_0\,\sigma(N) + c_1\,\sigma(N-1) + c_2\,\sigma(N-2) + c_3\,\sigma(N-3)$ at N = 0, 1, 2, 3 yields 0, 1, 0, 1 respectively, as required.

The lemma is the operational engine of the construction. Every XOR in AES — the GF(2) additions inside MixColumns and the AddRoundKey operations between rounds — is realized by composing this parity construction with linear preprocessing that aggregates the relevant bits into a

 doi:10.20944/preprints202605.0601.v1

non-negative integer sum. The novelty of the paper does not lie in Lemma 1 but in its composition with key-bearing parameters in Lemma 2, where the round-key bits jointly modulate the bias entry and the per-row sign of the same layer's weight matrix. The coefficient pattern (+1, −2, +1) at odd offsets recurs as the per-row template against which the bias-and-sign coupling of Lemma 2 is checked by the parametric recognizer of Section 5.1.

3.3. Lemma 2: Bias-as-Key with Sign Coupling

Lemma 2. *Let $j \in \{0, 1, \ldots, 127\}$ index a bit position in the round output, let $N_j \in \mathbb{N}$ denote the integer-valued sum of bits feeding into position j from the preceding linear stage, and let $k_j \in \{0, 1\}$ denote the round-key bit at position j. Let $c \in \mathbb{Z}^{n+1}$ be the parity-coefficient vector from Lemma 1 with $n \geq N_j$. Define a single output element by $y_j = \Sigma_{i=0..n} (1 - 2k_j) \cdot c_i \cdot \sigma(N_j - i) + k_j$. Then $y_j = (N_j \bmod 2) \oplus k_j$.*

Proof. When $k_j = 0$, the prefactor $(1 - 2k_j) = 1$ and the bias term is 0, so $y_j = \Sigma_i c_i \sigma(N_j - i) = N_j \bmod 2$ by Lemma 1. When $k_j = 1$, the prefactor is -1 and the bias is 1, so $y_j = -(N_j \bmod 2) + 1 = 1 - (N_j \bmod 2) = (N_j \bmod 2) \oplus 1$. In both cases $y_j = (N_j \bmod 2) \oplus k_j$. ■

Structural consequence. The round-key bit k_j controls two distinct elements of the layer simultaneously: the bias entry $b_j = k_j$ and the per-row sign of the weight row $W_{j,:} = (1 - 2k_j) \cdot c$. In the construction, these are not independent learned parameters but coupled functions of the same key bit. This coupling is itself a detection signature, and we return to it in Section 5: a layer in which the bias vector matches the per-row sign pattern of the weight matrix under a fixed coefficient template is a candidate signature for embedded round-key material.

Figure 1 illustrates the bias-and-sign coupling for one output element.

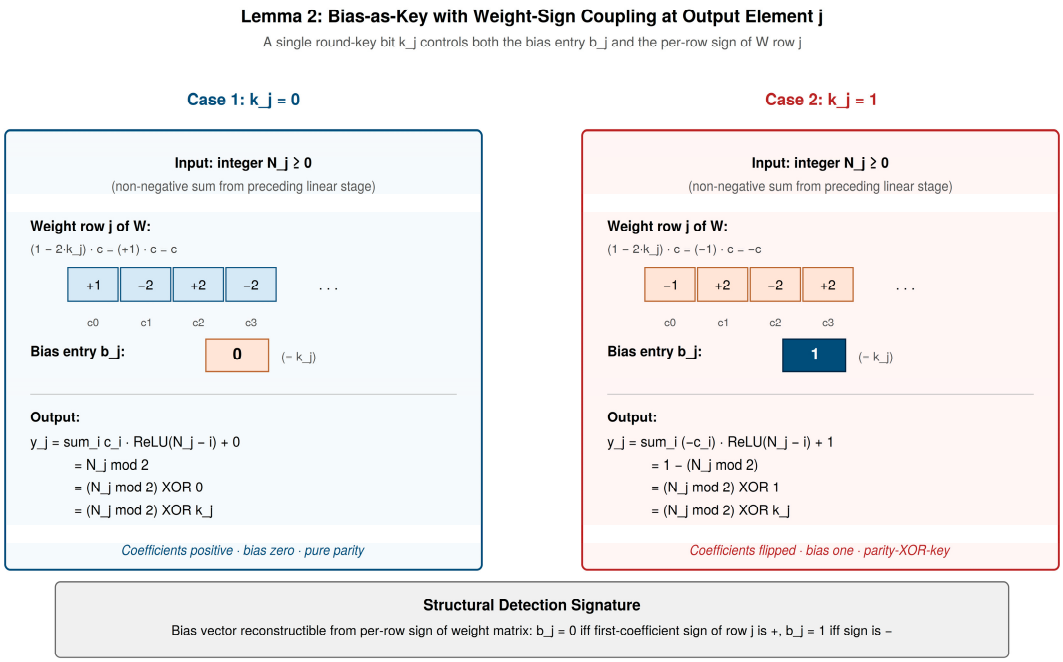

Figure 1. Lemma 2 schematic: bias-as-key with weight-sign coupling at one output element. The two cases ($k_j = 0$, $k_j = 1$) show how a single round-key bit jointly determines the bias entry and the per-row sign of the weight matrix.

3.4. Realizing the AES S-Box

The AES S-box is a fixed permutation of the 256-byte values, defined by composing inversion in $GF(2^8)$ with a fixed affine transformation. We realize it as a two-layer subnetwork in which the hidden

 doi:10.20944/preprints202605.0601.v1

layer encodes the input byte one-hot and the output layer reads off the substituted byte by linear combination.

Let $u \in \{0, 1\}^8$ denote the input byte's bit representation. For each candidate byte value $v \in \{0, 1, ..., 255\}$ with bit representation $v = (v_0, v_1, ..., v_7)$, define a hidden unit with weight row $W^{(1)}_{v,j} = 2v_j - 1$ and bias $b^{(1)}_v = 1 - \Sigma_j v_j$. The pre-activation for this unit is $\Sigma_j (2v_j - 1) u_j + 1 - \Sigma_j v_j = 1 - \text{HammingDistance}(u, v)$, which equals 1 when $u = v$ and is non-positive otherwise. After ReLU activation, exactly one of the 256 hidden units is non-zero, with value 1, at the index matching the input byte. The output layer reads the substituted byte by setting $W^{(2)}_{j,v}$ equal to the j-th bit of SBOX[v], so the output at position j is the j-th bit of SBOX[u].

This is the standard exact-lookup-table construction in the literature on Boolean function realization by feed-forward networks [19,20,22]. The construction generalizes to any fixed function on a finite Boolean domain at a parameter cost that scales with the domain size. We state it for completeness and claim no novelty — the contribution of Section 3 lies in the composition of this S-box subnetwork with the parity construction of Section 3.2 and the bias-as-key with sign coupling of Section 3.3, not in the S-box realization itself.

3.5. ShiftRows, MixColumns, and AddRoundKey

ShiftRows is a fixed permutation of the 16 bytes of the state. We realize it as a 128×128 permutation matrix P that acts on the bit-encoded state.

MixColumns is a GF(2)-linear transformation: each output byte is a fixed $GF(2^8)$ linear combination of the four bytes in its column under multiplication by polynomial coefficients in {1, 2, 3} mod the AES irreducible polynomial. Because $GF(2^8)$ multiplication by a fixed coefficient is GF(2)-linear at the bit level, MixColumns is realizable as a fixed binary matrix $M \in \{0, 1\}^{128 \times 128}$ acting on the bit-encoded state under arithmetic mod 2. In our construction, the modular reduction is deferred: M is treated as an integer matrix, the column sums it produces are non-negative integers, and the parity reduction is performed by Lemma 1 in the next layer.

AddRoundKey XORs the round key into the state. In the initial round (before any S-box application), this is realized as a single linear layer with $W = \text{diag}(1 - 2k_0)$ and $b = k_0$, where k_0 is the master key bit-encoded; Lemma 2 applies with the trivial sum $N_j = u_j$ (a single input bit), reducing to direct XOR with the key bit. In subsequent rounds, AddRoundKey is folded into the parity-computing layer that follows MixColumns, again by Lemma 2.

3.6. Layer Stack and Round-Key Localization

The full network consists of 30 linear layers organized as follows.

Layer 0 is the initial AddRoundKey, with master key k_0 in the bias.

For each round $r \in \{1, 2, ..., 9\}$, three layers are inserted: Layer $3r - 2$ applies the 16 parallel S-box subnetworks (the $W^{(1)}$ and $b^{(1)}$ side of Section 3.4) with ReLU activation. Layer $3r - 1$ applies the S-box output combination $W^{(2)}$ composed with ShiftRows P and MixColumns M, followed by the bit-aggregation step that produces the integer-valued N_j sums for Lemma 1; it carries no key material and ends with a ReLU bank. Layer $3r$ applies Lemma 2 with the round key k_r in its bias and in the sign of its weight rows; the output is the bit-encoded post-AddRoundKey state.

Round 10 omits MixColumns per the AES specification and is realized in two layers: layer 28 applies the S-box ($W^{(1)}/b^{(1)}$ with ReLU); layer 29 combines $W^{(2)}$, ShiftRows P, and the final AddRoundKey under Lemma 2 with key k_{10}.

The round keys reside at layer indices {0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 29}. The 3:2 stride pattern between rounds 1–9 (three layers per round) and round 10 (two layers) is the architectural fingerprint of the AES specification's omission of MixColumns in the final round: the absence of that operation removes the layer that would otherwise sit at index 30.

Figure 2 shows the resulting 30-layer stack and the placement of key-bearing layers within it.

 doi:10.20944/preprints202605.0601.v1

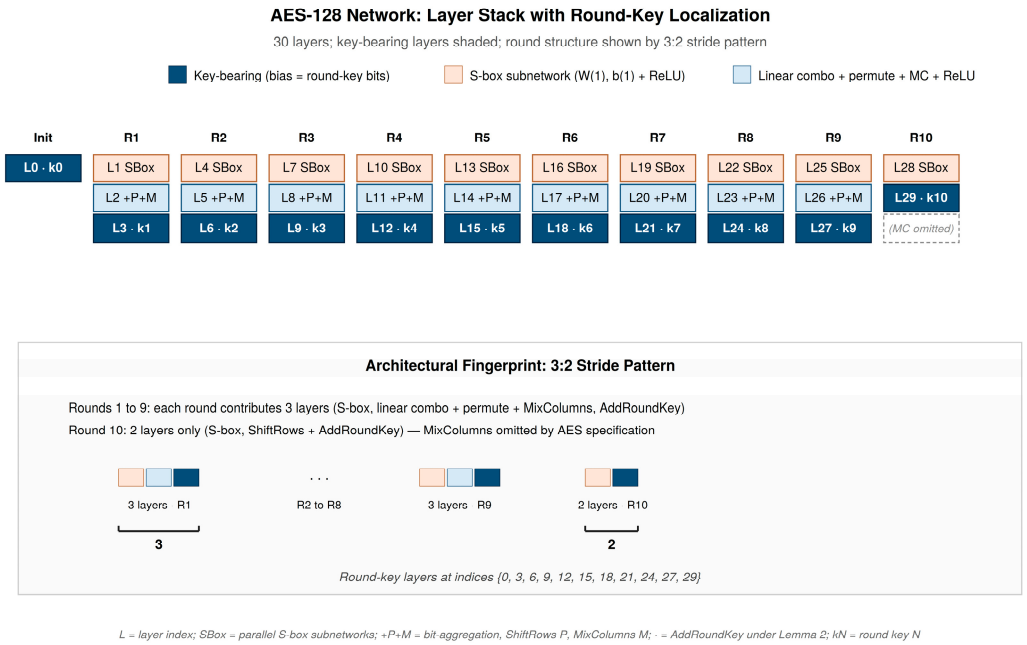

Figure 2. Layer stack of the AES-128 ReLU network. Key-bearing layers are shaded dark; rounds 1–9 contribute three layers each, round 10 contributes two (MixColumns omitted). The 3:2 stride pattern is the architectural fingerprint of the construction.

3.7. Equivalence Theorem

Theorem 1. *Let $K \in \{0, 1\}^{128}$ be a 128-bit key and let $P \in \{0, 1\}^{128}$ be a 128-bit plaintext. Let N(K) denote the network constructed as in Sections 3.1–3.6 with key schedule derived from K. Then the forward pass of N(K) on input P produces an output vector in $\{0.0, 1.0\}^{128}$ that decodes to AES-128_K(P).*

Proof sketch. Induction over rounds. The base case is layer 0: by the AddRoundKey realization in Section 3.5, the output of layer 0 on input P is the bit-encoding of P ⊕ k_0, which is the AES-128 state after the initial AddRoundKey.

For the inductive step, assume the output of layer 3(r−1) for r ∈ {1, ..., 9} is the bit-encoding of the AES-128 state after round r−1. Layer 3r − 2 applies the S-box construction of Section 3.4 to each of the 16 bytes in parallel; by the correctness of that construction, its output is the bit-encoding of the post-SubBytes state. Layer 3r − 1 applies $W^{(2)}$ to extract the substituted bits, then P to permute under ShiftRows, then M to apply MixColumns at the integer level; its output is a vector of non-negative integers N_j whose parities are the bits of the post-MixColumns state. Layer 3r applies Lemma 2 with round key k_r; by Lemma 2, its output bits are (N_j mod 2) ⊕ (k_r)_j, which by the parity argument above and the AddRoundKey definition equals the bit-encoding of the AES-128 state after round r.

For round 10, the analogous argument applies with MixColumns omitted: layer 28 produces the bit-encoded post-SubBytes state, and layer 29 applies $W^{(2)}$, P, and the round-10 AddRoundKey under Lemma 2 to produce the bit-encoded ciphertext.

The output of layer 29 is the bit-encoding of AES-128_K(P). ■

3.8. Validation

We verified the construction empirically against three test sets.

FIPS 197 Appendix A and C. The key expansion of K = 000102030405060708090a0b0c0d0e0f matches the published worked example byte-for-byte across all eleven round keys. The encryption of plaintext P = 00112233445566778899aabbccddeeff under this key produces ciphertext 69c4e0d86a7b0430d8cdb78070b4c55a, matching FIPS 197 Appendix C [9].

 doi:10.20944/preprints202605.0601.v1

NIST CAVP AESAVS. We verified the construction against the GFSbox, KeySbox, VarKey, and VarTxt known-answer subsets of the AESAVS specification [10]. All vectors matched bit-exactly.

Monte Carlo. We sampled 10^4 independent (key, plaintext) pairs uniformly at random and compared the network output against an independent AES-128 reference implementation. All 10^4 pairs matched bit-exactly. The Monte Carlo run also constitutes an empirical demonstration of the encryption-oracle capability described in Section 1: each random plaintext is presented to the network, which returns the corresponding AES-128 ciphertext under the embedded key without any extraction operation against the bias vectors.

Quantization survival. We rebuilt the construction in float32 and re-ran the Monte Carlo. All 10^4 pairs matched bit-exactly. The construction is therefore not dependent on float64 precision, and an embedding of this form would survive the quantization regimes routinely applied to deployed models.

The full network contains 30 linear layers and 19 ReLU activation banks. The minimum-realization parameter count of the construction is approximately 6.7×10^6; the implementation in the reproducibility artifact uses approximately 3.87×10^7 parameters with looser intermediate-layer sizing for clarity, both of which produce the bit-exact validation results reported above.

Table 1 summarizes the validation sources, vector counts, and pass counts.

Table 1. Validation results across FIPS 197, AESAVS subsets, Monte Carlo, and float32 quantization. All checks are bit-exact under the embedded key. Network parameter count: ~6.7×10^6 minimum realization, ~3.87×10^7 in the reproducibility implementation; both pass identically.

Test source	Vectors / pairs	Pass count	Notes
FIPS 197 Appendix A (key expansion)	1 key, 11 round keys	11/11	Round keys match published worked example byte-for-byte
FIPS 197 Appendix C (encryption)	1 plaintext-key pair	1/1	Ciphertext matches 69c4e0d86a7b0430d8cdb78070b4c55a
AESAVS GFSbox-128	7 vectors	7/7	GFSbox known-answer test (Given/Fixed S-box)
AESAVS KeySbox-128	21 vectors	21/21	KeySbox known-answer test (Key S-box)
AESAVS VarTxt-128	128 vectors	128/128	Variable plaintext, fixed key-zero
AESAVS VarKey-128	128 vectors	128/128	Variable key, fixed plaintext-zero
Monte Carlo (uniform random)	10,000 pairs	10,000/10,000	Random (key, plaintext) pairs vs. reference AES-128
Monte Carlo under float32	10,000 pairs	10,000/10,000	Same Monte Carlo, network parameters cast to float32

The complete construction, validation harness, and Monte Carlo runner are available at https://github.com/rcampbell-research/parameter-resident-cryptographic-content-aes under the Apache 2.0 license. The repository includes a validation/results.txt artifact recording the bit-exact outcomes reported in this section. Total wall-clock time for full reproduction of the validation results in this subsection is approximately 15 minutes on commodity hardware.

4. Post-Quantum Extension: Parameter-Resident PQC Keys as HNDL Surface

The AES-128 construction in Section 3 is the existence proof. This section argues by analogy and sizing that the same embedding strategy applies to post-quantum cryptographic key material, and

 doi:10.20944/preprints202605.0601.v1

that the consequence is a harvest-now-decrypt-later (HNDL) exposure unscoped by current federal migration guidance.

4.1. PQC Key Material: Sizes and Structure

The CNSA 2.0 baseline parameter sets are ML-KEM-1024 for key establishment, ML-DSA-87 for digital signatures, and LMS/XMSS for software and firmware signing as specified in NIST SP 800-208 [5]. Private key sizes for these algorithms are substantially larger than symmetric keys: ML-KEM-1024 private keys are 3,168 bytes, ML-DSA-87 private keys are 4,896 bytes, and SLH-DSA private keys are 4n bytes — i.e., 64, 96, or 128 bytes depending on the parameter set; signatures, not private keys, carry the authentication-path material [23–25]. LMS/XMSS, which CNSA 2.0 lists alongside SLH-DSA for software and firmware signing, introduces separate state-management concerns under SP 800-208 because the private key must track which one-time signing keys have been consumed; we do not analyze the LMS/XMSS state-tracking case in this paper.

These keys differ from AES keys in two structural respects. First, they are larger by one to two orders of magnitude. Second, they contain less internally rigid structure: an ML-KEM decapsulation key includes a polynomial vector, an encapsulation key copy, and hash material that lacks the deterministic round-key expansion structure of an AES key schedule. There is no fixed-stride parity pattern in an ML-KEM decapsulation key analogous to the 3:2 layer stride that fingerprints AES.

4.2. Embedding Capacity

The AES-128 construction of Section 3 has a minimum-realization parameter count of approximately 6.7×10^6 and an implementation count of approximately 3.87×10^7 in the reproducibility artifact (the difference is intermediate-layer sizing chosen for clarity rather than minimality; both produce bit-exact AES output). For embedding-capacity analysis we use the minimum-realization figure as the reference point. A 7-billion parameter foundation language model contains over two orders of magnitude more parameter budget than the construction in either form. A single attention block in such a model — for the standard configuration of LLaMA-class architectures with hidden dimension 4096 and 32 attention heads — contains approximately 6.7×10^7 parameters, comparable to the implementation count and an order of magnitude larger than the minimum realization. Embedding capacity is therefore not the binding constraint on whether constructions of this kind fit inside deployed models.

The relevant question for post-quantum cryptographic primitives is whether the construction overhead per primitive is comparable to AES-128, smaller, or substantially larger. We argue analytically that for the CNSA 2.0 baseline parameter sets — ML-KEM-1024 and ML-DSA-87 — the overhead is comparable to AES within roughly an order of magnitude, and in some configurations smaller per bit of embedded key material.

ML-KEM-1024 decapsulation requires three operationally substantive components: the number-theoretic transform (NTT) and its inverse, polynomial arithmetic in the ring $\mathbb{Z}_q[x]/(x^{256} + 1)$ with $q = 3329$, and SHA-3 hashing for key derivation and re-encryption verification [23,35]. We estimate the parameter footprint for each.

The NTT over a 256-coefficient polynomial requires 8 stages of butterfly operations with 128 butterflies per stage, each butterfly performing a fixed modular multiplication and addition. Realizing a single butterfly as a feed-forward subnetwork requires approximately 10^4 parameters under generous assumptions; up to 5×10^4 if the mod-3329 reduction requires a generalization of Lemma 1's parity construction with parameter cost scaling linearly in q. Across 8 stages × 128 butterflies × 10^4 parameters per butterfly, one NTT consumes approximately 10^7 parameters; decapsulation requires both forward and inverse transforms plus pointwise polynomial multiplication, bringing the lattice arithmetic component to approximately 3×10^7 parameters.

SHA-3 realization adds further overhead. A Keccak-f permutation with a 1600-bit state is 24 rounds of fixed bitwise operations; estimating each round at approximately 10^5 parameters gives 2.4

 doi:10.20944/preprints202605.0601.v1

× 10^6 parameters per permutation, and decapsulation invokes the permutation 5–10 times. The SHA-3 contribution is therefore approximately 2 × 10^7 parameters.

Adding the lattice arithmetic, hash, and key-bit storage components, an embedded ML-KEM-1024 decapsulation function would consume approximately 5 × 10^7 parameters as a minimum realization. This is roughly 7× the AES-128 minimum-realization figure in absolute terms, but the embedded key material is 198× larger (3,168 bytes versus 16 bytes), so the *parameter overhead per bit of embedded key material* is approximately 26× *smaller* than for AES-128. The qualitative conclusion is robust to the calibration choice: under either the minimum-realization or the implementation-scale parameter accounting, the per-bit overhead remains favorable to PQC because the lattice arithmetic and hashing components scale sub-linearly in key size. ML-DSA-87 admits a similar analysis with the rejection-sampling loop replacing the lattice-arithmetic structure of ML-KEM; the per-bit overhead is comparable.

These figures should be read as order-of-magnitude estimates rather than verified parameter counts. The claim they support is narrow: an embedded PQC primitive of CNSA 2.0 baseline size fits comfortably within the parameter budget of a foundation-scale model, with per-bit overhead favorable relative to the AES-128 case rather than prohibitive. The minimum-realization construction would consume well under 1% of a 7-billion-parameter model's total parameter budget; the implementation realization consumes approximately 0.5%.

The counterintuitive consequence stated qualitatively in the introduction to this section now has quantitative grounding. PQC private keys hide more comfortably in modern weight tensors than AES keys do, for two reasons. Larger keys amortize the construction overhead — per-bit cost falls as key size grows because the cipher's structural parameter cost is largely independent of key length. And the architectural fingerprint of an embedded lattice primitive is harder to recognize than the AES fingerprint: the NTT is rigid, but polynomial arithmetic and hashing admit substantial flexibility in how they decompose across network layers, diluting the structural signature on which the recognizer of Section 5 depends.

Table 2 reports the per-bit overhead figures for AES-128, ML-KEM-1024, and ML-DSA-87 alongside the minimum-realization parameter sizings.

Table 2. Embedding capacity comparison across symmetric and post-quantum primitives. Parameter overhead estimates are minimum-realization sizings. Per-bit overhead is parameter count divided by total private-key bits (8 × byte size). Lower per-bit overhead means the primitive hides more comfortably in a fixed parameter budget.

Primitive	Private key size	Parameter overhead (min. realization)	Per-bit overhead	Per-bit ratio vs. AES-128
AES-128	16 bytes (128 bits)	~6.7 × 10^6	~52,000 params / bit	1.0× (baseline)
ML-KEM-1024 (FIPS 203)	3,168 bytes (25,344 bits)	~5 × 10^7	~1,970 params / bit	~0.04× (26× smaller)
ML-DSA-87 (FIPS 204)	4,896 bytes (39,168 bits)	~7 × 10^7	~1,790 params / bit	~0.03× (29× smaller)
SLH-DSA (FIPS 205)	64 bytes (small variant)	~2 × 10^6 (no NTT, small key)	~3,900 params / bit	~0.08× (13× smaller)

4.3. HNDL Framing for Open-Weights Distribution

The harvest-now-decrypt-later threat model that motivates the federal PQC migration assumes an adversary who records ciphertext today and decrypts it once a cryptographically relevant quantum computer becomes available. The exposure is realized at the moment of recording; the consumer of the ciphertext at that future point inherits the access without having compromised any system in the present.

 doi:10.20944/preprints202605.0601.v1

Open-weight distribution of a model containing parameter-resident cryptographic key material follows the same logic. A long-lived signing key or KEM private key embedded in weights distributed publicly today is recoverable by any party with knowledge of the embedding scheme at any future point. The consumer of the embedded key need not have compromised any system: the public model registry is the channel, and the extraction recipe is the access. Unlike the classical HNDL case, no quantum capability is required — recovery is computationally trivial under known construction.

This collapses the model artifact channel into the same threat surface as the recorded ciphertext. The migration concern is therefore not deferred to the arrival of a quantum adversary; it is realized today, with classical extraction, against any model so embedded.

4.4. Migration-Scope Consequence

Current federal PQC migration scoping addresses cryptographic libraries, protocols, and key stores [1–4]. Discovery tooling and CBOM emission formats inherit this scoping. Model artifacts are not included.

The omission is non-trivial under two conditions established earlier in the paper: extraction is trivial under known construction (Section 3, particularly the bias-as-key construction in Section 3.3), and recognition without construction knowledge is computationally hard (Section 5.4, recognition asymmetry limit). The asymmetry between extraction and recognition favors the embedder by an unbounded margin in the worst case. A model artifact distributed within or to a federal environment that contains parameter-resident cryptographic material is, under these conditions, an undisclosed cryptographic asset operating outside the migration framework's discovery surface.

We recommend that the federal PQC migration scope be expanded to include parameter-resident cryptographic content as a discovery target on equal footing with libraries and binaries. The audit primitive proposed in Section 5 specifies a concrete mechanism for this expansion.

5. Proposed Control: Parameter-Space Cryptographic Recognizer

We propose an audit primitive that detects parameter-resident cryptographic content in model artifacts and emits the result as an extension to existing cryptographic bill-of-materials (CBOM) tooling. The primitive is specified at the design level. Implementation, corpus evaluation, and false-positive characterization are out of scope for this paper and identified as follow-on work in Section 6.

5.1. Detection Surface

Three classes of signature are available to a recognizer operating on model parameters without access to the construction recipe.

Structural signatures are patterns in layer stride, dimension, and connectivity that match known cipher round structures. The AES construction of Section 3 exhibits a 3:2 layer-stride pattern across rounds 1–9 and round 10, with round-key bias rows at predictable indices; this is the architectural fingerprint identified in Section 3.6. Analogous fingerprints exist for other symmetric primitives — block ciphers such as DES, stream ciphers built around fixed-permutation cores such as ChaCha20, and the Keccak permutation underlying SHA-3 — and for PQC primitives whose round structures are public (the number-theoretic transform stages of ML-KEM, the rejection-sampling loop of ML-DSA). A structural recognizer maintains a library of such fingerprints and matches candidate subnetworks against it.

Parametric signatures are statistical patterns in weight and bias values that are unlikely under benign training distributions. The bias-as-key construction with sign coupling (Lemma 2) produces bias vectors with binary or near-binary distributions in layers whose weight matrices have row-sign patterns coupled to those bias values under a fixed coefficient template. This coupling — the bias vector reconstructible from the per-row signs of the weight matrix — is a strong signature: the joint

 doi:10.20944/preprints202605.0601.v1

distribution of b_j and sign(W_{j,:}) deviates substantially from what gradient-based training on natural data produces. A parametric recognizer scans layers for binary-valued biases and tests for sign coupling against a small library of coefficient templates.

Functional signatures are input-output behaviors of subnetworks consistent with cryptographic primitives. A subnetwork that maps inputs to outputs with avalanche, diffusion, and statistical-test properties characteristic of a block cipher or hash function is a candidate cryptographic realization regardless of its internal structure. Functional probing is the most adversary-resistant class of signature — a sufficiently diffused construction can defeat structural and parametric recognition while remaining functionally cryptographic —, but it is also the most computationally expensive and the hardest to bound for false-positive rate.

5.2. Recognizer Architecture

The recognizer operates as a screening pipeline applied to model artifacts at ingestion, analogous to the position malicious-pickle scanners and safetensors validators occupy in current pipelines [14,15].

The pipeline has four stages. *Ingestion* parses the model artifact, extracts the parameter tensors and architectural metadata, and produces a layer-by-layer representation suitable for analysis. *Structural matching* compares the layer stride, dimension, and connectivity against the fingerprint library and flags candidate subnetworks. *Parametric analysis* tests bias vectors for binary distributions and weight rows for sign coupling, returning a score per layer. *Functional probing*, applied only to candidates flagged by the prior stages, evaluates input-output behavior against cryptographic distinguishers.

Each stage operates as a screening filter: it admits a bounded false-positive rate in exchange for tractable false-negative rate against known constructions. A model artifact that passes all four stages without flag is not certified free of cryptographic content; it is certified free of cryptographic content of forms the recognizer knows how to detect. We return to this limit in Section 5.4.

Figure 3 shows the four-stage pipeline and its integration into the MBOM-PQC emission flow.

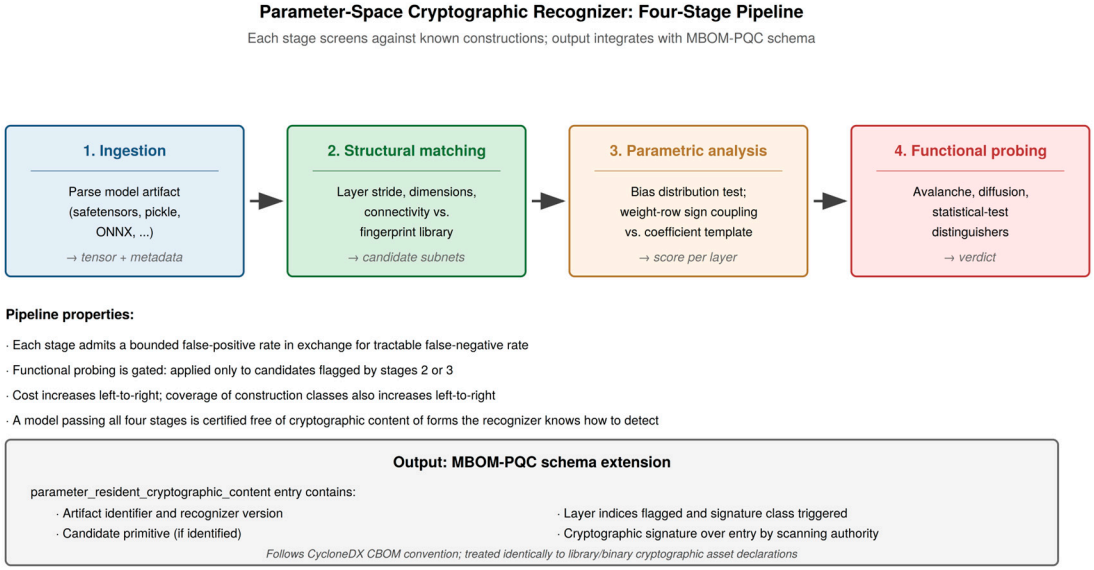

Figure 3. Recognizer pipeline architecture. The four screening stages run sequentially with cost increasing left-to-right; functional probing is gated to candidates flagged by earlier stages. Output integrates with the MBOM-PQC schema as a parameter_resident_cryptographic_content emission class.

5.3. Integration with MBOM-PQC

 doi:10.20944/preprints202605.0601.v1

The recognizer's output integrates with cryptographic bill-of-materials tooling as a new emission class. We propose extending the schema developed in prior work [26] with a parameter_resident_cryptographic_content entry alongside the existing library and binary CBOM emissions.

A parameter_resident_cryptographic_content entry contains the artifact identifier, the layer indices flagged, the signature class triggered (structural, parametric, or functional), the candidate primitive identified if any, the recognizer version and fingerprint library version, and a cryptographic signature over the entry produced by the scanning authority. The entry follows the CycloneDX CBOM convention for cryptographic asset declaration [16].

This integration places parameter-resident cryptographic content on equal footing with library and binary cryptographic assets in the federal migration discovery surface. A model artifact ingested into a federal environment carries an MBOM-PQC manifest that includes parameter-content emissions; downstream migration tooling treats those emissions identically to library-level cryptographic asset declarations.

Table 3 defines the proposed emission fields.

Table 3. MBOM-PQC schema extension fields for the parameter_resident_cryptographic_content emission class. Follows CycloneDX CBOM convention; each entry carries a detached signature by the scanning authority, with the signing algorithm declared in entry_signature_algorithm rather than fixed by the schema.

Field name	Type	Cardinality	Semantics
artifact_id	string (URI)	1	Identifier of the model artifact scanned (registry URL or content hash)
recognizer_version	string (semver)	1	Version of the recognizer that produced the entry
scan_timestamp	ISO 8601 datetime	1	Time the scan was performed
signature_class	enum	1..n	One of: structural, parametric, functional, combined
layer_indices_flagged	array of int	0..n	Layer indices in the artifact triggering the signature
candidate_primitive	string (enum)	0..1	Identified primitive if classifier returns one (e.g., AES-128, ML-KEM-1024); absent if signature triggered without classification
confidence	float in [0, 1]	1	Recognizer confidence in the detection
entry_signature_algorithm	string (enum)	1	Signing algorithm identifier (e.g., ML-DSA-87, SLH-DSA-128s, Ed25519); permits crypto-agility across PQC parameter sets and transitional hybrid algorithms
entry_signature_value	byte string (base64)	1	Detached signature over the entry's other fields, computed by the scanning

 doi:10.20944/preprints202605.0601.v1

	authority under the algorithm declared in entry_signature_algorithm

5.4. Limitations

The recognizer is a defense-in-depth control, not a complete solution. Three limits are intrinsic to the design.

Recognition asymmetry. Extraction under known construction is trivial; recognition without construction knowledge is computationally hard. The recognizer detects known constructions and architectural fingerprints; it does not guarantee detection of novel embeddings.

Adversarial diffusion. An adversary aware of the recognizer can diffuse the cipher across redundant compute paths, add learned noise to bias vectors, and obscure stride patterns to evade structural and parametric detection. Functional probing remains effective against diffused constructions but is computationally expensive at scale.

Fingerprint library currency. Structural and parametric recognition depend on a fingerprint library that must be maintained as new constructions are characterized. The library is a moving target, and coverage gaps are inevitable. The library's value lies in cumulative coverage of construction classes rather than guaranteed detection of every embedded artifact: it raises the cost of novel embeddings (because adversaries must develop constructions outside the library's coverage) and catches disclosed constructions retroactively. This is structurally similar to how anti-malware signature libraries function, despite never catching truly novel zero-days.

These limits do not undermine the contribution. The primitive raises the cost of embedding cryptographic content in distributed model artifacts from zero to non-trivial, and it places the federal migration framework's discovery surface on a footing that includes parameter-resident content rather than excluding it by construction. Both are improvements over the current state.

6. Discussion

6.1. Persistence Under Fine-Tuning

The constructions presented in Section 3 are fragile under direct gradient updates. We measured fine-tuning persistence empirically (Appendix B): SGD with gradient clipping at learning rate 1×10^{-5}, applied to the AES-128 construction with an L2 loss-on-noise target (random binary inputs to a zero target; see Appendix B for methodology), destroys cipher correctness within a single optimization step across all three fine-tuning regimes evaluated (all layers, key-bearing layers only, non-key layers only). The parametric detection signature of Section 5.1 is substantially more durable: it persists through more than 100 optimization steps at learning rates up to 1×10^{-2}, and only fails within five steps when the learning rate is pathologically high (1×10^{-1}). The ordering is consequential. The cipher's correctness fails first, far ahead of the detection signature. The mechanism differs by regime. When key-bearing layers are updated, gradient-based optimization perturbs the bias values away from their {0, 1} encoding directly. When only non-key layers are updated, the bias values themselves remain intact, but small numerical perturbations propagate through the network's depth and shift the integer-valued column sums entering each parity layer to near-integer floats; the shifted-ReLU bank's unit-pulse approximation degrades correspondingly, and final-layer output values drift away from clean {0, 1}. The 0.5 threshold at the bit-decoding step then misclassifies values that have drifted across the threshold, even though the underlying GF(2) parity computation remains mathematically correct. Because any single bit flip in a 128-bit cipher output produces a wrong ciphertext, even a small drift in a small fraction of output bits is sufficient to break correctness. The per-row sign pattern that grounds Lemma 2 coupling decays much more slowly because gradient updates do not preferentially flip signs, which is why the parametric detection signature outlasts cipher correctness. From the defender's perspective, this is favorable: the recognizer can detect an embedded cipher even after the embedded cipher has stopped functioning.

 doi:10.20944/preprints202605.0601.v1

The threat model of Section 2.4 explicitly does not require persistence. The exposure is realized at the moment of artifact distribution, and the federal procurement window between distribution and consumer-side fine-tuning is non-trivial — model artifacts undergo CONOPS development, procurement review, integration testing, and provenance attestation on the as-shipped weights before any downstream fine-tuning occurs. For artifacts that are deployed without further training (the common case for inference-only deployments and for many federal acquisition pathways), the embedding remains intact through the operational lifetime of the artifact.

Whether constructions can be made robust to fine-tuning is the most consequential open question this paper raises and warrants explicit characterization. Three classes of construction modification appear plausible.

The first is *redundant encoding across layers*. Rather than placing each round key at a single layer's bias vector, an adversarial construction could replicate key material across multiple layers in a redundancy scheme such that gradient updates on any minority subset do not destroy recoverability. The cipher's forward pass would be reconstructed from a quorum of layers, with the network functionally tolerant to fine-tuning.

The second is *gradient-resistant parameter encoding*. The current construction places key bits at integer values precisely at {0, 1}, where small gradient perturbations produce decoding failures. An adversarial construction could encode key bits at parameter values lying in larger basins of attraction under the loss landscape — values where gradient descent tends to remain regardless of the downstream task. The technique is known in the steganography literature as adversarial training for embedding robustness [11,12]. Whether it transfers to *functionally active* embeddings — where the bits must continue producing correct cipher output, not merely be recoverable — is open.

The third is *cipher-specific architectural integration*. Rather than constructing a network with an isolated cryptographic subnetwork, an adversary could integrate the cipher's computation into the network's primary inference pathway such that the cipher's parameters are co-located with parameters serving the legitimate inference function. Gradient updates that improve inference would necessarily preserve the cipher's parameters, because the same parameters serve both purposes. This is the most concerning of the three classes from a defender's standpoint because it removes the assumption that cipher parameters are separable from the model's primary function.

Across all three classes, robust constructions invert the threat model: instead of being durable only until the first fine-tuning event, the embedding becomes durable through fine-tuning by design. The defender's recognition task correspondingly shifts from "detect cipher-shaped subnetworks in unmodified artifacts" to "detect cipher-shaped subnetworks deliberately co-trained with the model's primary function and exhibiting only weak architectural fingerprints." We do not present empirical results on robust constructions in this paper; establishing whether any of the three classes can be made to work in practice is the most important follow-on direction for adversarial work in this space.

6.2. Detection at Scale

The recognizer specified in Section 5 is design-level. False-positive rates against benign model corpora are unknown and require empirical study. The natural evaluation corpus is the public Hugging Face model registry, which contains on the order of a million model artifacts spanning a wide range of architectures, training regimes, and quantization formats. A scanning study of this corpus, characterizing the false-positive rate of structural and parametric recognition, is the immediate follow-on to this paper.

Two specific questions deserve early empirical attention. First, what fraction of benign models exhibit incidental architectural patterns that match cipher fingerprints? The 3:2 stride pattern of AES is unlikely to occur by accident, but more subtle structural signatures may have non-trivial false-positive rates. Second, what fraction of benign models have layers with binary or near-binary bias distributions? Quantized models will trigger this signature trivially, suggesting the parametric recognizer needs to be conditioned on the model's quantization regime to be useful.

 doi:10.20944/preprints202605.0601.v1

6.3. Adjacent Threat Classes

The contribution is distinct from three lines of prior work that share the underlying mechanism of payload-bearing model parameters but differ from this paper in threat model, embedding mechanism, or proposed defense.

Backdoor-trigger embedding — the BadNets lineage [17,18] and the trojan-network and adversarial-patch literatures that follow from it — embeds an input-pattern recognizer in the model that activates malicious behavior when a matching trigger is presented at inference time. The embedded value is a trigger pattern, the threat is misclassification under adversarial input, and the defense literature concerns input-space perturbation analysis and trigger-pattern reverse engineering. The mechanism shares the supply-chain assumption that adversaries control model parameters at distribution time, but the embedded value is not cryptographic key material, and the recipient capability is fundamentally different — a backdoor recipient triggers misclassification, not key recovery or cryptographic operation.

Model watermarking — the Uchida [27] and Adi [28] lines and subsequent rights-management work — embeds an ownership signal in the model that the rights-holder extracts to prove provenance. The threat model is reversed: the embedder is the defender protecting intellectual property, not the adversary distributing covert material. Watermarking constructions are deliberately announced, and the extraction routine is published; the cryptographic content is signed authenticity metadata, not operational key material that the model uses in computation. The mechanism is technically adjacent, but the security properties run in opposite directions.

Steganography in neural network weights — the StegoNet line [11] and recent extensions including MaleficNet 2.0 [12] — is the closest prior work and warrants careful differentiation across three dimensions.

First, the embedding is *functionally active* in the construction of Section 3 rather than passive as in StegoNet. Steganographic payloads in StegoNet's framework reside in low-order bits of weight values and play no role in the network's inference; replacing them with random bits leaves behavior unchanged. The construction in Section 3 places key material at parameters that participate in the cipher's computation. Three consequences follow. The detection signature is structurally different: StegoNet's payloads are detectable by statistical analysis of low-order weight bits, while this construction produces the bias-and-sign coupling identified in Lemma 2, which the parametric stage of the recognizer in Section 5.2 detects directly. Quantization survival differs: StegoNet payloads in weight LSBs degrade under aggressive quantization, while bias values integer-valued in $\{0, 1\}$ survive float32 quantization bit-exactly (Section 3.8). And the recipient capability is qualitatively different: a StegoNet recipient extracts payload bytes for external use, while a recipient of the Section 3 construction obtains an encryption oracle — they may submit plaintexts and receive AES-128 ciphertexts under the embedded key without any extraction. The model artifact is not merely a covert carrier; it is a covert cryptographic service.

Second, the payload class is cryptographic key material rather than arbitrary bytes. StegoNet treats payloads as opaque, evaluating capacity, accuracy degradation, and triggering reliability without distinguishing among payload types. This paper argues that cryptographic key material is a categorically distinct payload class because of the harvest-now-decrypt-later temporal asymmetry developed in Section 4.3: a recovered key unlocks ciphertext recorded at any time before recovery, extending the threat backward through time in a way that arbitrary-payload steganography does not.

Third, the proposed response is structurally different. StegoNet's defense recommendations are generic and not connected to any deployed assurance regime. The recognizer specified in Section 5 targets the specific signatures produced by functionally-active cryptographic embedding, and integrates with cryptographic bill-of-materials tooling (Section 5.3) as a schema extension on equal footing with library and binary cryptographic asset declarations.

These three differentiations — functional activity, cryptographic-material specificity, and assurance-pipeline integration — establish the contribution as distinct from the steganography-in-weights literature.

 doi:10.20944/preprints202605.0601.v1

6.4. Policy and Acquisition Implications

This subsection addresses Claim 3 from Section 1: the gap constitutes an unscoped migration surface with concrete acquisition consequences, and we propose how to close it. The migration-scope expansion proposed in Section 4.4 has concrete implications for federal acquisition of AI systems. Federal acquisition regulations governing software supply chain security currently treat model artifacts under provisions designed for executable software. Executive Order 14028 [29] established the foundational software supply-chain requirements; Office of Management and Budget Memorandum M-22-18 [30] implemented the self-attestation regime against the secure software development practices in NIST SP 800-218 [31]. The implementation regime has since been modified by Executive Order 14306 [32], which rescinded portions of EO 14144 and adjusted the centralized validation requirement, and OMB M-22-18 itself was rescinded on 23 January 2026 by OMB Memorandum M-26-05, which directs agencies to adopt a risk-based approach to software and hardware assurance [33]. The underlying gap survives these modifications: neither the original regime nor its successor addresses parameter-resident cryptographic content. The provisions that remain in force address code-execution risk and provenance attestation; they do not address cryptographic functions or key material realized in the parameters of model artifacts.

The forthcoming NIST AI 800-series guidance and AI-specific supplements to FAR/DFARS that follow from NSM-10 and the AI executive orders are the natural vehicles for closing this gap. We recommend that AI system acquisition under federal authority require, alongside existing model-card and provenance documentation, an MBOM-PQC manifest that includes parameter-resident cryptographic content emissions as specified in Section 5.3. The recognizer specified in this paper is one mechanism for producing such emissions; alternative mechanisms achieving equivalent coverage would satisfy the same requirement.

The recommendation is conservative on two dimensions. It does not require that all model artifacts be free of parameter-resident cryptographic content; it requires only that such content, where present, be discovered, declared, and subject to the same migration discipline as cryptographic content in libraries and binaries. And it does not depend on persistence of the embedding under consumer-side fine-tuning, which Section 6.1 establishes is fragile under the constructions presented here. The recommendation operates at the artifact-as-shipped boundary — the boundary federal acquisition currently audits — where the embedding is intact regardless of what occurs downstream.

Equivalent acquisition frameworks in other jurisdictions face the same gap. The EU's NIS2 Directive, the forthcoming Cyber Resilience Act, and the EU AI Act establish supply-chain and provenance obligations that do not address parameter-resident cryptographic content as a discoverable asset class. The UK's NCSC AI cyber threat assessment [7] and Five Eyes joint cybersecurity advisories on AI supply chain security [34] similarly treat model artifacts under provisions designed for executable software. The migration-scope expansion proposed here applies equivalently across these regimes: the gap is structural rather than U.S.-specific, and the audit primitive of Section 5 is jurisdiction-neutral.

7. Responsible Disclosure

The findings reported in this paper identify a class of threat — parameter-resident cryptographic key material in distributed model artifacts — that is not addressed by the open-source serialization-focused scanners we evaluated. The empirical evaluation in Appendix A documents this gap against two named open-source scanners (Picklescan v1.0.4 and ModelScan v0.8.8) at the default configuration. Although the contribution is a missing security control rather than an exploitable defect against any specific platform, we treated the work as subject to coordinated disclosure norms because its publication has direct implications for the assurance posture of public model registries and federal model acquisition pipelines.

On 8 May 2026, we provided advance notice to the security teams of the maintainers of the scanners evaluated in Appendix A — Hugging Face (Picklescan) and Protect AI (ModelScan) — and

 doi:10.20944/preprints202605.0601.v1

submitted a coordination report to the U.S. Cybersecurity and Infrastructure Security Agency's coordinated vulnerability disclosure program via the VINCE platform operated by Carnegie Mellon University's Software Engineering Institute. The notification described the threat class, summarized the construction of Section 3 without reproducing it, included the empirical scanner-test results documented in Appendix A, and indicated the planned submission date of this paper.

We did not request remediation within a fixed window. The finding is properly characterized as a coverage-gap observation rather than a defect for which a patch is the appropriate response: the scanners evaluated in Appendix A operate within their stated scope of detecting code-execution risk in serialized model formats, and the threat class identified here is outside that scope by design. The notification was provided as a courtesy to allow recipient organizations to evaluate whether to extend their scanning scope to include parameter-content analysis.

The paper was submitted on 8 May 2026. The reproducibility artifact accompanying the paper is available at https://github.com/rcampbell-research/parameter-resident-cryptographic-content-aes and was released concurrently with submission. The artifact includes the test harness used to produce the Appendix A evaluation, enabling independent verification by reviewers and by the recipient organizations of the disclosure correspondence.

We did not pursue a CVE assignment for this finding. The MITRE CVE program is structured around defects in specific products; the finding here is a class-level gap in deployed practice that does not map cleanly to any single product's vulnerability inventory. Should specific platforms confirm the gap in their pipelines and request CVE coordination, we will support that process.

8. Conclusions

We have shown that cryptographic key material can be embedded in feed-forward neural network parameters such that recovery under known construction is computationally trivial. The AES-128 construction in Section 3 is a complete existence proof: a 30-layer ReLU network whose forward pass realizes AES-128 byte-for-byte, with the master key and all eleven round keys recoverable directly from layer biases by parsing. We have shown that this class of embedding is not addressed by the open-source serialization-focused scanners we evaluated: those scanners check for code-execution risk in serialized formats, not for cryptographic content in parameters, and no emission class for parameter-resident cryptographic material exists in current cryptographic bill-of-materials tooling.

We have shown that this gap constitutes an unscoped surface for federal post-quantum migration. Open-weights distribution of a model containing long-lived cryptographic key material is, under the harvest-now-decrypt-later threat model, equivalent in posture to recorded ciphertext, with the additional consequence that no quantum capability is required for recovery. PQC private keys, by virtue of larger size and lower internal rigidity, hide more comfortably in modern weight tensors than symmetric keys do, not less.

We have proposed a parameter-space cryptographic recognizer that integrates with existing CBOM tooling as a defense-in-depth control, and a corresponding extension to federal acquisition discovery practice that would place parameter-resident cryptographic content on equal footing with library and binary cryptographic assets.

The contribution is bounded. We make no claim that any specific deployed model contains an embedding of this form, that such embeddings emerge from gradient-based training, or that the proposed recognizer is complete against adaptive adversaries. The contribution is the existence of the capability (Claim 1), the absence of detection in the open-source serialization-focused scanners we evaluated (Claim 2), and the migration-scope consequence with its accompanying audit primitive (Claim 3). The asymmetry between extraction and recognition — extraction trivial under known construction, recognition hard without it — favors the embedder. Closing this asymmetry is the central technical problem the threat class poses, and the direction in which the most consequential follow-on work lies.

 doi:10.20944/preprints202605.0601.v1

Appendix A. Empirical Scanner Evaluation

This appendix documents the empirical evaluation supporting Claim 2 of the manuscript: that the open-source serialization-focused model-scanning tools we evaluated do not detect parameter-resident cryptographic content. The evaluation tests two open-source scanners against the AES-128 construction of Section 3 and against architecturally matched controls. Verbatim scanner output and the full test harness are available in the reproducibility artifact at the GitHub repository referenced in Section 7.

A.1. Test Environment

The evaluation was conducted on a Linux Ubuntu 24.04 environment on 7 May 2026 (UTC). Scanner versions: Picklescan 1.0.4 (Hugging Face) and ModelScan 0.8.8 (Protect AI). Both were installed from PyPI at the default versions on the test date. NumPy and the safetensors library were installed from PyPI at their concurrent versions. All tests were run from a fresh shell with no caching or prior state.

A.2. Test Artifacts

Five artifacts were constructed for the evaluation. The two AES-construction artifacts contain the 30-layer ReLU network of Section 3, instantiated with the FIPS 197 Appendix C test key, in safetensors and pickle formats. The two benign-control artifacts have identical architecture (30 layers, matched dimensions, 38.7×10^6 parameters per artifact) but parameters drawn from a Kaiming-like initialization characteristic of untrained ReLU networks. The positive-control artifact is a small pickle file containing a __reduce__ method invoking os.system — the standard pickle-format remote-code-execution pattern that pickle scanners are explicitly designed to detect.

Round-trip verification confirmed that both AES-construction artifacts contain operational cryptographic content after serialization. The master key 000102030405060708090a0b0c0d0e0f is recoverable from layer 0's bias vector by direct read in both formats, and the network's forward pass on plaintext 00112233445566778899aabbccddeeff produces ciphertext 69c4e0d86a7b0430d8cdb78070b4c55a, matching FIPS 197 Appendix C exactly. The artifacts the scanners examined are the actual constructions described in the manuscript, with extractable cryptographic key material and operational encryption-oracle capability.

A.3. Scanner Results

Ten scans were executed: each scanner against each AES-construction artifact (4), each scanner against each benign-control artifact (4), and each scanner against the positive-control malicious pickle (2). All scanners were run at the default configuration with no custom rules or settings.

AES-construction and benign-control artifacts (8 scans). Both Picklescan and ModelScan returned zero findings against all four AES-construction and benign-control artifacts in both serialization formats. For safetensors files, ModelScan reports "No issues found!" but the --show-skipped flag reveals the file is silently skipped because the scanner has no module for the safetensors format; the apparent clean result is an artifact of out-of-scope handling rather than active scanning. Picklescan returned a non-pickle parse warning for safetensors files and a clean scan for pickle files.

Positive-control malicious pickle (2 scans). Both Picklescan and ModelScan correctly flagged the positive control as Critical-severity. Picklescan reported "dangerous import 'posix system' FOUND" with infected files: 1 and dangerous globals: 1. ModelScan reported "Total Issues: 1, CRITICAL: 1, Use of unsafe operator 'system' from module 'posix'." The scanners are functioning correctly for the threat class they are designed to address.

Summary of findings. The scanners detect the positive-control pattern they are designed to detect and return zero findings on the AES-construction artifacts that contain functionally-active cryptographic content recoverable by parsing. The result is consistent with the scanners' stated scope — pickle-format code-execution risk — and consistent with the recognition asymmetry developed in

 doi:10.20944/preprints202605.0601.v1

Section 3.3: extraction of the embedded key material is trivial, but recognition without construction knowledge requires parameter-content analysis that the scanners do not perform.

A.4. Limitations of the Evaluation

Three limitations bound the empirical claim of this appendix.

First, the evaluation covers two open-source scanners. The Hugging Face hosted ingestion pipeline applies additional automated scanning at upload time and was not directly tested in this evaluation; verifying the gap against the HF pipeline requires uploading the artifacts to a private HF repository, which we treat as recommended follow-on verification. Commercial scanning tools (JFrog Xray's ML model scanning, Lakera ML scanner, others) were not evaluated; the threat-class scope of those tools, as documented in their public materials, is the same as the open-source scanners and the result is expected to be equivalent.

Second, the test environment is a clean Linux container with scanners installed from PyPI at default versions on the test date. Production deployments may use older versions, custom configurations, or wrap scanners in additional defense layers. The result here applies to default-configuration installations of the scanners as currently distributed.

Third, the empirical claim is bounded to the constructions presented in this paper. We do not claim that no future version of the scanners will detect parameter-resident cryptographic content; indeed, the audit primitive of Section 5 is one mechanism by which the scanners' coverage could be extended, and the disclosure correspondence documented in Section 7 invites the scanner maintainers to evaluate that extension. The empirical claim is that, as of the test date and against the scanner versions named, the gap exists.

A.5. Reproducibility

The full test harness, including the AES-construction builder, the benign-control generator, the positive-control malicious-pickle generator, the round-trip verification script, and the verbatim scanner-output logs, is available at the GitHub repository referenced in Section 7. The harness reproduces the results of this appendix on any Linux environment with Python 3.x and pip access to PyPI; total wall-clock time for full reproduction is under five minutes.

Appendix B. Fine-Tuning Persistence Baseline

This appendix documents the empirical measurement of fine-tuning persistence referenced in Section 6.1. Two metrics were measured: cipher correctness (does the network's forward pass produce the FIPS 197 Appendix C ciphertext under the embedded key?) and parametric detection signature (does the bias-and-sign coupling of Lemma 2 still trigger the Section 5.1 parametric recognizer?). Each metric was evaluated as a binary pass/fail at logarithmically spaced checkpoints during a stochastic gradient descent run.

B.1. Methodology

The fine-tuning loss is L2 regression to a zero target on random binary inputs of dimension 128. This is loss-on-noise: gradient updates are pure perturbations rather than directionally aligned with any downstream task. Real-world fine-tuning produces structured updates that may be more or less destructive depending on the alignment between the gradient direction and the embedding parameters; the loss-on-noise regime is a lower bound on durability and an upper bound on decay speed.

Optimization is plain SGD without momentum. Per-tensor gradient clipping is applied with a clip norm of 1.0 to prevent the gradient explosion that deep narrow networks with integer-valued weights produce on noise loss; without clipping, gradient magnitudes destabilize the network within a single step regardless of learning rate, which obscures the embedding-specific dynamics the experiment is designed to measure.

 doi:10.20944/preprints202605.0601.v1

Three fine-tuning regimes were evaluated: *all* (gradient updates applied to every layer), *key_only* (updates only to the eleven key-bearing layers at indices {0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 29}), and *non_key* (updates only to the nineteen non-key layers). The all regime simulates standard fine-tuning of the full network; the key_only regime simulates a direct attack on the embedding; the non_key regime is the most favorable to the embedding's persistence and represents the case where the cipher subnetwork is frozen and surrounding parameters drift. Learning rates were swept across 1×10^{-5}, 1×10^{-3}, 1×10^{-2}, and 1×10^{-1} at the all regime to characterize the signature-decay regime.

B.2. Results

Cipher correctness decay. Across all three regimes at learning rate 1×10^{-5}, cipher correctness failed at step 1. Cipher correctness was checked against the FIPS 197 Appendix C plaintext-key-ciphertext triple; the failure was bit-level, not approximate. The failure mode is the bit-decoding threshold rather than the parity computation itself: in the all and key_only regimes, gradient updates perturb the bias values away from the binary encoding directly; in the non_key regime, small numerical perturbations in non-key layers propagate through the network's depth and shift the integer-valued column sums entering each parity layer to near-integer floats, the shifted-ReLU bank's unit-pulse approximation degrades, final-layer output values drift away from clean {0, 1}, and the 0.5 threshold at the bit-decoding step misclassifies values that have drifted across it. The underlying GF(2) parity computation in Lemma 2 remains mathematically correct in all three regimes; the failure is at decoding, not at the parity math.

Parametric signature decay. At learning rate 1×10^{-5}, the parametric signature persisted past 100 optimization steps in all three regimes — two orders of magnitude beyond cipher-correctness decay. At learning rates 1×10^{-3} and 1×10^{-2}, the signature also persisted past 100 steps. At the pathologically high learning rate 1×10^{-1}, the signature failed at step 5. Aggregating, the parametric signature is robust at any reasonable fine-tuning learning rate and only fails when the learning rate is high enough to substantially destabilize the network's outputs.

Implication. Cipher correctness fails by orders of magnitude faster than the detection signature does. Two consequences follow. First, an embedded cipher in a model artifact subjected to fine-tuning likely loses its cryptographic function long before the parameter-space recognition signature degrades, so a recognizer applied post-fine-tuning can detect the historical presence of an embedding even when the cipher itself no longer operates. Second, the threat model boundary in Section 2.4 (persistence under fine-tuning out of scope) is conservative — the embedding is fragile in the cipher-correctness sense at one optimization step, but the detection signature is durable enough that the §5 recognizer remains useful well beyond the point at which the cipher itself has been destroyed.

B.3. Limitations

Three limitations bound the empirical claim of this appendix.

First, the loss is loss-on-noise rather than a real downstream task. Real fine-tuning losses produce structured updates that may decay the signature faster (if updates are aligned against the embedding) or slower (if updates are orthogonal to the embedding parameters). The relevant question is whether typical downstream-task gradients have directional structure that hits the bias-and-sign coupling preferentially; that is a question for follow-on work.

Second, the experiment used SGD without momentum or adaptive learning rates. Adam, AdamW, and other adaptive optimizers used in modern fine-tuning produce different parameter trajectories that may decay the signature differently. Replicating the experiment under Adam at typical fine-tuning hyperparameters is recommended follow-on.

Third, the experiment evaluated fine-tuning of the as-shipped construction. Robust constructions in the three classes outlined in Section 6.1 (redundant encoding, gradient-resistant encoding, architectural integration) would exhibit different decay curves and require their own evaluation. The appendix establishes a baseline for the construction presented in this paper, not for the broader threat-class behavior under adversarial countermeasures.

 doi:10.20944/preprints202605.0601.v1

B.4. Reproducibility

The fine-tuning experiment harness is included in the reproducibility artifact alongside the scanner-evaluation harness of Appendix A. Wall-clock time for full reproduction of the results in Section B.2 is under five minutes on commodity hardware.

Author Contributions: The author was responsible for all aspects of the manuscript, including conceptualization, methodology, software, validation, formal analysis, investigation, writing—original draft preparation, writing—review and editing, and visualization. The author has read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The complete reproducibility artifact — including the AES-128 construction code, validation harness, scanner-evaluation harness, fine-tuning experiment harness, and verbatim scanner output logs — is publicly available at https://github.com/rcampbell-research/parameter-resident-cryptographic-content-aes under the Apache 2.0 license. The repository's validation/results.txt records the bit-exact outcomes referenced in Section 3.8 of the manuscript.

Acknowledgments: The author acknowledges the open-source maintainers of the Picklescan and ModelScan projects for the tools that made the empirical evaluation in Appendix A possible.

Conflicts of Interest: The author declares no conflict of interest. The author is an Independent Researcher and has no financial or professional ties to organizations that would benefit from or be harmed by the publication of these findings.

References

1. The White House. National Security Memorandum on Promoting United States Leadership in Quantum Computing While Mitigating Risks to Vulnerable Cryptographic Systems (NSM-10). 4 May 2022. Available online: https://bidenwhitehouse.archives.gov/briefing-room/statements-releases/2022/05/04/national-security-memorandum-on-promoting-united-states-leadership-in-quantum-computing-while-mitigating-risks-to-vulnerable-cryptographic-systems/ (accessed 7 May 2026).
2. Office of Management and Budget. Memorandum M-23-02: Migrating to Post-Quantum Cryptography. Executive Office of the President, 18 November 2022. Available online: https://www.whitehouse.gov/wp-content/uploads/2022/11/M-23-02-M-Memo-on-Migrating-to-Post-Quantum-Cryptography.pdf (accessed 7 May 2026).
3. Moody, D.; Perlner, R.; Regenscheid, A.; Robinson, A.; Cooper, D. Transition to Post-Quantum Cryptography Standards (Initial Public Draft). NIST Internal Report 8547 ipd, National Institute of Standards and Technology, November 2024. https://doi.org/10.6028/NIST.IR.8547.ipd
4. National Cybersecurity Center of Excellence. Migration to Post-Quantum Cryptography. NIST Special Publication 1800-38 (Volumes A, B, C, Preliminary Drafts). National Institute of Standards and Technology, 2023–2024. Available online: https://www.nccoe.nist.gov/crypto-agility-considerations-migrating-post-quantum-cryptographic-algorithms (accessed 7 May 2026).
5. National Security Agency. Announcing the Commercial National Security Algorithm Suite 2.0 (CNSA 2.0). NSA Cybersecurity Advisory CSA U/OO/194427-22, 7 September 2022 (algorithms-table updated 30 May 2025). Available online: https://media.defense.gov/2025/May/30/2003728741/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS.PDF (accessed 7 May 2026).
6. European Union Agency for Cybersecurity (ENISA). ENISA Threat Landscape 2024: July 2023 to June 2024. ENISA Report, 2024. Available online: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024-july-2023-to-june-2024 (accessed 7 May 2026).

 doi:10.20944/preprints202605.0601.v1

7. National Cyber Security Centre (UK). Impact of AI on Cyber Threat from Now to 2027. NCSC Report, 2025. Available online: https://www.ncsc.gov.uk/report/impact-of-ai-on-cyber-threat-from-now-to-2027 (accessed 7 May 2026).
8. European Telecommunications Standards Institute. ETSI TS 103 744: CYBER; Quantum-Safe Hybrid Key Exchanges. ETSI Technical Specification, V1.1.1, 2020 (current edition). Available online: https://www.etsi.org/deliver/etsi_ts/103700_103799/103744/ (accessed 7 May 2026).
9. National Institute of Standards and Technology. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, updated 9 May 2023 (originally issued 26 November 2001). https://doi.org/10.6028/NIST.FIPS.197-upd1
10. National Institute of Standards and Technology. The Advanced Encryption Standard Algorithm Validation Suite (AESAVS). NIST Cryptographic Algorithm Validation Program, 15 November 2002 (current revision). Available online: https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/aes/AESAVS.pdf (accessed 7 May 2026).
11. Liu, T.; Liu, Z.; Liu, Q.; Wen, W.; Xu, W.; Li, M. StegoNet: Turn Deep Neural Network into a Stegomalware. In Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC 2020), 7–11 December 2020; pp. 928–938. https://doi.org/10.1145/3427228.3427268
12. Hitaj, D.; Hitaj, B.; Mancini, L.V. Do You Trust Your Model? Emerging Malware Threats in the Deep Learning Ecosystem. arXiv:2403.03593, 2024. Available online: https://arxiv.org/abs/2403.03593 (accessed 7 May 2026).
13. Maitre, M. picklescan: Security Scanner Detecting Python Pickle Files Performing Suspicious Actions. Open-source software, version 1.0.4 (tested release). Available online: https://github.com/mmaitre314/picklescan (accessed 7 May 2026).
14. Protect AI. ModelScan: Protection Against Model Serialization Attacks. Open-source software, version 0.8.8 (tested release). Available online: https://github.com/protectai/modelscan (accessed 7 May 2026).
15. Hugging Face. safetensors: Simple, Safe Tensor Serialization. Open-source software (2022–present). Available online: https://github.com/huggingface/safetensors (accessed 7 May 2026).
16. OWASP Foundation. CycloneDX Bill of Materials Specification, version 1.6 (April 2024) and subsequent. Ecma International standard ECMA-424. Available online: https://cyclonedx.org/specification/overview/ (accessed 7 May 2026).
17. Gu, T.; Dolan-Gavitt, B.; Garg, S. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. arXiv:1708.06733, 22 August 2017. Available online: https://arxiv.org/abs/1708.06733 (accessed 7 May 2026).
18. Gu, T.; Liu, K.; Dolan-Gavitt, B.; Garg, S. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. IEEE Access 2019, 7, 47230–47244. https://doi.org/10.1109/ACCESS.2019.2909068
19. Hornik, K.; Stinchcombe, M.; White, H. Multilayer Feedforward Networks Are Universal Approximators. Neural Networks 1989, 2(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8
20. Telgarsky, M. Benefits of Depth in Neural Networks. In Proceedings of the 29th Annual Conference on Learning Theory (COLT 2016), 23–26 June 2016; Volume 49 of Proceedings of Machine Learning Research, pp. 1517–1539. arXiv:1602.04485. Available online: https://arxiv.org/abs/1602.04485 (accessed 7 May 2026).
21. Eldan, R.; Shamir, O. The Power of Depth for Feedforward Neural Networks. In Proceedings of the 29th Annual Conference on Learning Theory (COLT 2016), 23–26 June 2016; Volume 49 of Proceedings of Machine Learning Research, pp. 907–940. Available online: https://proceedings.mlr.press/v49/eldan16.html (accessed 7 May 2026).
22. Pinkus, A. Approximation Theory of the MLP Model in Neural Networks. Acta Numerica 1999, 8, 143–195. https://doi.org/10.1017/S0962492900002919
23. National Institute of Standards and Technology. Module-Lattice-Based Key-Encapsulation Mechanism Standard. Federal Information Processing Standards Publication 203, 13 August 2024. https://doi.org/10.6028/NIST.FIPS.203
24. National Institute of Standards and Technology. Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standards Publication 204, 13 August 2024. https://doi.org/10.6028/NIST.FIPS.204

 doi:10.20944/preprints202605.0601.v1

25. National Institute of Standards and Technology. Stateless Hash-Based Digital Signature Standard. Federal Information Processing Standards Publication 205, 13 August 2024. https://doi.org/10.6028/NIST.FIPS.205
26. Campbell, R. AI Supply Chain Security: MBOM-PQC Provenance, PQC Attestation, and a Maturity Model for Quantum-Resistant Assurance. Preprint, Preprints.org, version 1, 25 March 2026. https://doi.org/10.20944/preprints202603.1963.v1
27. Uchida, Y.; Nagai, Y.; Sakazawa, S.; Satoh, S. Embedding Watermarks into Deep Neural Networks. In Proceedings of the 2017 ACM International Conference on Multimedia Retrieval (ICMR '17), 6–9 June 2017; pp. 269–277. https://doi.org/10.1145/3078971.3078974
28. Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; Keshet, J. Turning Your Weakness into a Strength: Watermarking Deep Neural Networks by Backdooring. In Proceedings of the 27th USENIX Security Symposium, 15–17 August 2018, Baltimore, MD, USA; pp. 1615–1631. Available online: https://www.usenix.org/conference/usenixsecurity18/presentation/adi (accessed 7 May 2026).
29. The White House. Executive Order 14028: Improving the Nation's Cybersecurity. 12 May 2021. Federal Register 86 FR 26633. Available online: https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity (accessed 7 May 2026).
30. Office of Management and Budget. Memorandum M-22-18: Enhancing the Security of the Software Supply Chain through Secure Software Development Practices. Executive Office of the President, 14 September 2022 (rescinded 23 January 2026). Available online: https://www.whitehouse.gov/wp-content/uploads/2022/09/M-22-18.pdf (accessed 7 May 2026).
31. Souppaya, M.; Scarfone, K.; Dodson, D. Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities. NIST Special Publication 800-218, February 2022. https://doi.org/10.6028/NIST.SP.800-218
32. The White House. Executive Order 14306: Sustaining Select Efforts To Strengthen the Nation's Cybersecurity and Amending Executive Order 13694 and Executive Order 14144. 6 June 2025. Federal Register 90 FR 24723 (FR Doc. 2025-10804), published 11 June 2025. Available online: https://www.federalregister.gov/documents/2025/06/11/2025-10804/sustaining-select-efforts-to-strengthen-the-nations-cybersecurity-and-amending-executive-order-13694 (accessed 8 May 2026).
33. Office of Management and Budget. Memorandum M-26-05: Adopting a Risk-based Approach to Software and Hardware Security. Executive Office of the President, 23 January 2026. Available online: https://www.whitehouse.gov/wp-content/uploads/2026/01/M-26-05-Adopting-a-Risk-based-Approach-to-Software-and-Hardware-Security.pdf (accessed 8 May 2026).
34. National Security Agency Artificial Intelligence Security Center; Cybersecurity and Infrastructure Security Agency; Federal Bureau of Investigation; et al. AI Data Security: Best Practices for Securing Data Used to Train and Operate AI Systems. Joint Cybersecurity Information Sheet, 22 May 2025. Available online: https://www.cisa.gov/news-events/alerts/2025/05/22/new-best-practices-guide-securing-ai-data-released (accessed 7 May 2026).
35. National Institute of Standards and Technology. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standards Publication 202, August 2015. https://doi.org/10.6028/NIST.FIPS.202