Trust Now, Forget Later

The Authenticity Dimension of the Quantum Threat

A Companion to "Harvest Now, Decrypt Later"

Dr. Gregory S. Carmichael, DBA
Lt. Col., USAF (Ret.)

Quantum Reserve Capital · *CryptoSoWhat*

Abstract. Quantum-readiness programs routinely fail before they begin—not because the engineering is wrong, but because the *framing* is. The quantum threat to public-key cryptography splits into two structurally distinct problems. The first, *Harvest Now, Decrypt Later* (HNDL), attacks *confidentiality*: adversaries archive ciphertext today and decrypt it once a cryptographically relevant quantum computer (CRQC) arrives. The second, which we formalize here as *Trust Now, Forget Later* (TNFL), attacks *authenticity*: every code-signing certificate, signed contract, audit record, and certificate-authority chain that rests on RSA or ECDSA can be forged retroactively once Shor's algorithm recovers the underlying private keys. You can re-encrypt a database; you cannot re-sign every artifact you have already issued. This paper develops the cryptographic foundations (Shor versus Grover), introduces a dual to Mosca's inequality for the authenticity case, and proves the central result: the correct remedy is not re-signing but *quantum-safe proof-of-existence anchoring*, which collapses an unbounded legacy exposure $F(t_0 + L)$ into a near-zero exposure $F(t_a)$ when an anchor is established before the CRQC arrives. We map the two problems to distinct NIST standards, system owners, and timelines, and close with sector applications for financial infrastructure (including GENIUS-Act stablecoin attestations), quantum communications, and national-security systems under CNSA 2.0 and NSM-10.

Prepared for publication on *CryptoSoWhat* and LinkedIn.
PhD rigor, dinner-table clarity. Every claim substantiated; every section ends with a So What.

Contents

1 The Framing Failure

A recent post by the team at Horizen Labs put the problem cleanly: most quantum-security readiness plans fail before they start, not because the work is wrong but because the framing is. There are two distinct quantum threats, and—in their evocative phrasing—they "run in opposite directions in time." One attacks the data you send; the other attacks the artifacts you sign. A team that hardens its TLS handshake but leaves its code-signing roots on RSA has closed one door and left the other wide open—and both will tell their board they are "working on quantum readiness."

This paper is the companion to our earlier work, *Harvest Now, Decrypt Later* (HNDL), which treated the confidentiality side in depth across financial, blockchain, and industrial-control contexts. Here we take up the second, less-discussed threat—*Trust Now, Forget Later* (TNFL)—and give it the same treatment: full cryptographic foundations, a formal risk model, a proof that the obvious remedy ("re-sign everything") is the wrong remedy, and the remedy that actually works.

We also sharpen the Horizen framing in one respect, because the imprecision hides the cure. TNFL does not literally "run backward in time." A CRQC recovers a private key and then forges *new* signatures going *forward*. What lies in the past is the *target*: the accumulated stock of long-lived artifacts you signed under classical algorithms that are still being relied upon at the moment quantum capability lands. The correct one-line statement is:

A forward attack against a legacy target. The artifacts you mint today have lifetimes that outrun the algorithm protecting them.

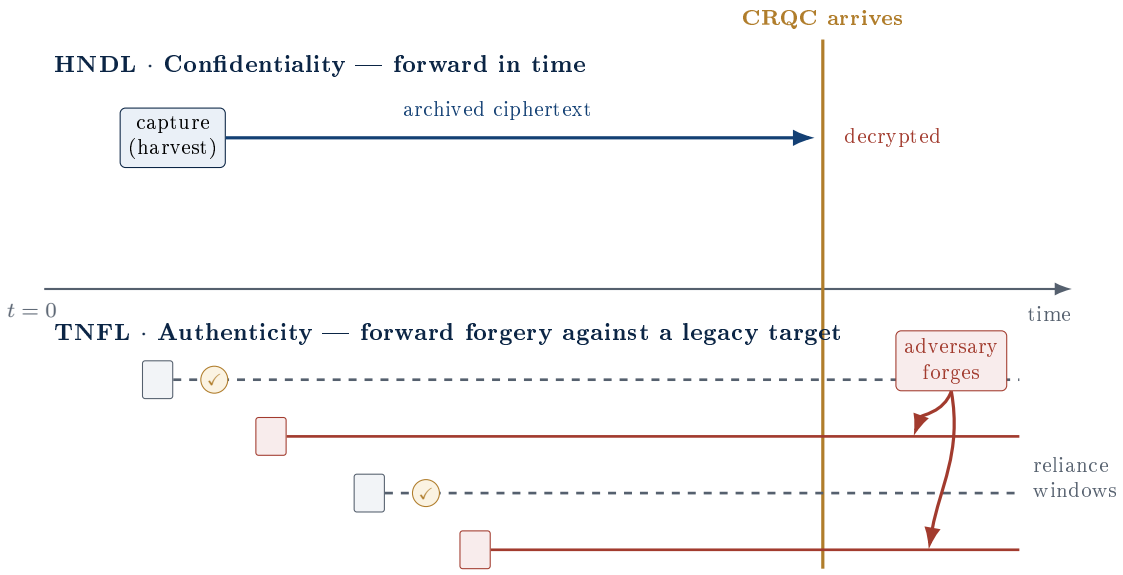

Figure 1: **The two quantum threats on one time axis.** HNDL harvests ciphertext now and decrypts once a CRQC exists—confidentiality, forward in time. TNFL is a *forward* forgery aimed at the *legacy stock*: artifacts whose reliance windows cross the CRQC line are forgeable unless anchored beforehand. A pre-CRQC anchor (✓, dashed/grey) neutralizes the exposure.

That distinction is not pedantry. It is the difference between "you cannot re-sign the past, so you are doomed" and "you cannot re-sign the past, so you must *anchor* it"—which is tractable.

At the Dinner Table

Imagine two ways a thief can hurt you. The first: he photographs your locked diary today, and ten years from now he finally cracks the lock and reads every secret. That is *Harvest Now, Decrypt Later*—the value was the secret, and the clock is already running. The second: he learns to forge your handwriting *perfectly*, and then starts producing new IOUs—"I owe you one million dollars, signed, you"—and dating them to last year. That is *Trust Now, Forget Later*. The value was never one secret; it was the *trust* in your signature. And once that trust is broken, it poisons *everything* you ever signed, retroactively. But notice the escape hatch: if you had walked every important document past a notary who stamped the date in indelible public ink, the forger's backdated IOUs would be worthless—because the real ones are provably stamped *before* he ever learned the trick.

So What

"Are we quantum-ready?" is the wrong question because it has no single answer. The right question is two questions: *What is our confidentiality exposure?* and *What is our authenticity exposure?* They have different answers, different fixes, different deadlines, and—critically—they are owned by different people inside your organization. Asking one question and acting on it is how a program reports "done" while remaining unprotected.

2 One Root Cause, Two Threats

Both threats grow from a single root: the asymmetric (public-key) cryptography that underpins modern digital trust is catastrophically broken by a sufficiently capable quantum computer. The companion HNDL analysis stated the shared mechanism plainly.

From the Companion Paper — *Harvest Now, Decrypt Later*

The quantum threat does not begin when a CRQC becomes operational. Nation-state adversaries executing "harvest now, decrypt later" collection operations are intercepting and storing encrypted communications, firmware update packages, authentication credentials, and topology data today—with the intention of decrypting them when quantum capability becomes available. For operators, this means the quantum threat has an effective start date of *today*.

— *Harvest Now, Decrypt Later*, § on present-day threat

The same companion work tabulated what happens to each component of a digital-trust architecture once the asymmetric layer falls. We reproduce the authenticity-relevant rows because they *are* the TNFL threat surface, named before we had a name for it:

Notice the difference in *when* the damage lands. HNDL damage is set at *capture time*: data intercepted today is exposed the moment a CRQC exists, full stop. TNFL damage is governed by a *forward reliance window*: a signed artifact is only at risk while it is still being trusted *and* while verifiers still honor the broken algorithm. This asymmetry is the entire game, and we make it precise in Section 4.

Table 1: Post-quantum failure modes of trust components (adapted from the companion HNDL/ICS analysis).

Component	Current protection	Post-quantum failure mode
Command authentication	RSA-2048 / ECDSA-256	Forged authenticated commands indistinguishable from legitimate operator actions
Firmware signing	RSA-2048 certificate chain	Forged firmware accepted by every device as vendor-legitimate
PKI certificate authority	RSA root certificate	Forged certificates for any device; the entire trust hierarchy is compromised
Audit / compliance records	DB security ± HMAC	Retroactive manipulation of history; audit integrity destroyed
Supply-chain manifests	PKI-signed provenance	Forged provenance for malicious hardware components

CryptoSoWhat Takeaway

HNDL attacks **confidentiality, forward in time**—harvest now, read later. TNFL attacks **authenticity**—but the honest description is a *forward forgery against a legacy stock*, not a literal trip backward in time. Same root cause (quantum-vulnerable asymmetric cryptography); different systems, different teams, different NIST standards as fixes, and different clocks. Confuse them and you will fix the wrong one first.

3 Cryptographic Foundations: Shor versus Grover

The reason the two threats share a root but split into different remedies is that quantum computing does not attack all cryptography equally. It attacks *asymmetric* primitives catastrophically and *symmetric* primitives only quadratically.

3.1 Shor's algorithm: the catastrophic break of asymmetric cryptography

Shor's algorithm solves integer factorization and the discrete-logarithm problem in polynomial time. For an RSA modulus N of $n = \lceil \log_2 N \rceil$ bits, factorization runs in roughly

$$O\big(n^2 \log n \, \log\log n\big) \text{ quantum gates on } O(n) \text{ logical qubits,} \tag{1}$$

collapsing a problem believed to require sub-exponential classical effort ($\exp\big(O(n^{1/3}(\log n)^{2/3})\big)$ for the general number-field sieve) to a polynomial. For elliptic-curve cryptography over a group of order $\approx 2^k$, the elliptic-curve discrete-logarithm problem (ECDLP) falls to a Shor variant requiring on the order of $6k$ logical qubits and $O(k^3)$ Toffoli gates. The companion paper quantified the blockchain instance directly:

From the Companion Paper — *Harvest Now, Decrypt Later*

Breaking Bitcoin's ECDSA would require on the order of 1,500–2,340 logical qubits—roughly 2,330 for the `secp256k1` curve—which translates into millions of *physical* qubits once quantum error correction is accounted for. Deloitte analysis indicates that more than four million bitcoins, about a quarter of the usable supply, sit in address types whose public keys are already exposed and therefore quantum-vulnerable.

— *Harvest Now, Decrypt Later*, § on signature vulnerability

The essential point for TNFL: *ECDSA and RSA are signature schemes.* When Shor recovers a signing key, the adversary does not merely read past messages—he gains the ability to *produce valid signatures* that are computationally indistinguishable from the legitimate signer's. Every artifact ever signed with that key is now forgeable.

3.2 Grover's algorithm: the survivable erosion of symmetric cryptography

Grover's algorithm searches an unstructured space of size $N = 2^n$ in

$$O\left(\sqrt{N}\right) = O\left(2^{n/2}\right) \text{ evaluations,} \tag{2}$$

which *halves* the effective security of symmetric primitives. AES-256 retains ≈ 128 bits against Grover; a SHA-256 preimage retains ≈ 128 bits. The remedy is mundane: double the key or digest length. Symmetric and hash-based security degrade gracefully and are restored by parameter choice.

$$\underbrace{\text{RSA, ECDSA, DH, ECDH}}_{\text{asymmetric}} \xrightarrow[\text{Shor}]{} \text{security} \to 0 \qquad \text{versus} \qquad \underbrace{\text{AES-}n\text{, SHA-}n}_{\text{symmetric / hash}} \xrightarrow[\text{Grover}]{} \text{security} \to n/2 \tag{3}$$

This single inequality explains both the disease and the cure. The disease: every *signature* scheme in classical use is asymmetric and therefore on the catastrophic side. The cure for the legacy stock, as we will show, leans on *hash functions*—which are on the survivable side—because a hash-based proof of existence remains sound against a quantum adversary at adequate output length.

So What

The reason you can defend the past at all is the Shor/Grover split. Forgery of a *new* signature is a Shor problem (catastrophic). But proving that an *old* artifact existed at a certain time is a *hash* problem (Grover-survivable). You cannot stop the forger from gaining the power; you *can* make sure the genuine article was stamped before he got it. The cure for an unsigned-able past is built from the one quantum-resistant primitive you already own.

4 Formalizing TNFL

We now make the threat quantitative. This is the part that converts "we should worry about signatures" into "anchor these artifacts, in this order, by this date."

4.1 Mosca's inequality and its authenticity dual

The standard tool for HNDL is Mosca's inequality. Let

- X = *security shelf-life*: how long data must remain confidential;
- Y = *migration time*: how long it takes you to deploy quantum-safe cryptography;
- Z = *collapse time*: years until a CRQC exists.

$$\textbf{(HNDL, confidentiality):} \qquad X + Y > Z \implies \text{you already have a problem.} \tag{4}$$

The logic is that data captured today must stay secret for X years, but if migration takes Y and the CRQC arrives in Z, then any X exceeding $Z - Y$ is exposed.

For authenticity we need the dual. Replace the secrecy shelf-life X with the *authenticity lifetime* L—how long a signed artifact must remain unforgeable and relied-upon. For *newly issued* artifacts the structure is identical:

$$\textbf{(TNFL, new issuance):} \qquad L_{\max} + Y > Z \implies \text{you are minting forgeable long-lived liabilities today.} \tag{5}$$

Here $L_{\max}$ is the longest authenticity lifetime among artifacts you are *still* signing with classical algorithms. A code-signing root issued today with a twenty-year field life, signed under RSA, with a CRQC plausibly inside that window, satisfies the inequality immediately.

Remark. The two inequalities are duals, not copies. In HNDL the exposed quantity is fixed at capture and is *retrospective*: you cannot un-harvest yesterday's ciphertext. In TNFL the new-issuance inequality is *prospective*: you control it by stopping the minting of classical long-lived artifacts. The genuinely hard part of TNFL is not new issuance—it is the *legacy stock* already in the wild, which the next subsection models.

4.2 The exposure window of a legacy artifact

Fix the present at $t = 0$. Consider an artifact i already issued or about to be issued. Define:

- $t_{0,i}$ — issuance (signing) time;
- L_i — required authenticity lifetime, giving a reliance window $[\,t_{0,i},\, t_{0,i} + L_i\,]$;
- t_q — CRQC arrival, a random variable with cumulative distribution $F(t) = \Pr[t_q \le t]$;
- $t_{a,i}$ — time a quantum-safe proof-of-existence *anchor* is established for artifact i ($t_{a,i} = \infty$ if never);
- s — *algorithm sunset*: the time after which relying parties reject classical signatures ($s = \infty$ if never).

Definition 1 (Forgeable-and-trusted)**.** *Artifact i is* forgeable-and-trusted *at time t iff all four conditions hold: (i) $t \ge t_q$ (a CRQC exists, so forgery is possible); (ii) $t \le t_{0,i} + L_i$ (still within the reliance window); (iii) $t < s$ (verifiers still honor classical signatures); (iv) no valid anchor predates the CRQC, i.e.* ***not*** *($t_{a,i} < t_q$).*

Condition (iv) is the load-bearing one. If the artifact was anchored *before* the CRQC existed, the genuine article's existence is provable as of a pre-CRQC date; a post-CRQC forgery therefore

cannot masquerade as the contemporaneous original, and the exposure is neutralized. Hence the exposure window is

$$W_i = \begin{cases} \big[\ \max(t_q, 0),\ \min(t_{0,i} + L_i,\ s)\ \big], & \text{if } t_{a,i} \geq t_q, \\ \varnothing, & \text{if } t_{a,i} < t_q. \end{cases} \tag{6}$$

A nonempty window—i.e. a real exposure—requires both

$$t_q \ < \ \min\big(t_{0,i} + L_i,\ s\big) \qquad \text{and} \qquad t_{a,i} \ \geq \ t_q. \tag{7}$$

4.3 Probabilistic exposure and the value of anchoring

Treat $t_q \sim F$. For an **unanchored** artifact with no sunset ($t_{a,i} = \infty$, $s = \infty$), the probability it is ever forgeable-while-trusted is simply the probability the CRQC arrives before its reliance ends:

$$P_i^{\text{unanchored}} \ = \ \Pr\big[t_q < t_{0,i} + L_i\big] \ = \ F\big(t_{0,i} + L_i\big). \tag{8}$$

For an artifact **anchored** at $t_{a,i}$, exposure additionally requires $t_q \leq t_{a,i}$ (the CRQC must have beaten the anchor), so

$$P_i^{\text{anchored}} \ = \ \Pr\big[t_q \leq t_{a,i}\big] \ = \ F(t_{a,i}). \tag{9}$$

If you anchor *promptly* ($t_{a,i} \approx 0$) and no CRQC exists yet, then $F(t_{a,i}) \approx F(0) \approx 0$. This is the central result:

Proposition 1 (Anchoring collapses the horizon)**.** *Anchoring an artifact at time t_a replaces its exposure probability $F(t_0 + L)$ with $F(t_a)$. The risk reduction is*

$$\Delta P_i \ = \ F\big(t_{0,i} + L_i\big) - F(t_{a,i}) \ \geq \ 0, \tag{10}$$

which is largest precisely for the long-lived, high-L artifacts that are otherwise most exposed. Prompt anchoring drives the residual toward zero.

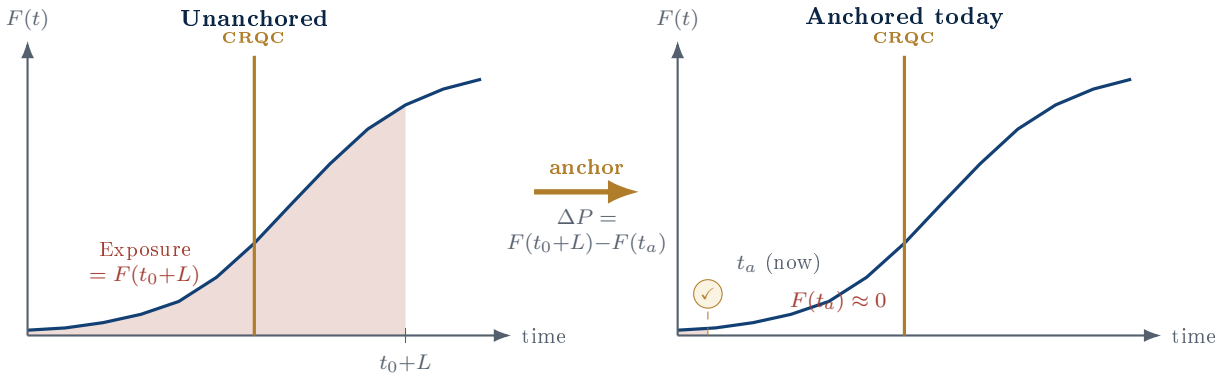

Figure 2: **Anchoring collapses the horizon (Proposition 1).** The S-curve is $F(t)$, the cumulative probability a CRQC has arrived by time t. An unanchored long-lived artifact carries exposure $F(t_0{+}L)$—the entire shaded area. Anchoring it today replaces that with $F(t_a) \approx 0$: the genuine artifact is provably dated before any CRQC, so a later forgery cannot masquerade as the original.

At the Dinner Table

$F(t_0 + L)$ is the chance the lock-picker shows up while your old contract still matters—and for a thirty-year mortgage or a twenty-year firmware root, that chance is uncomfortably high. $F(t_a)$ is the chance he had *already* shown up before you walked the document to the notary. If you go to the notary this afternoon and the lock-picker isn't here yet, that chance is essentially zero. Anchoring doesn't make forgery impossible—it makes forgery *useless*, because the real thing is dated and the fake can't be.

4.4 Portfolio risk and a prioritization rule

Real organizations hold thousands of signed artifacts of differing value. Let V_i be the criticality of artifact i (dollars at risk, mission impact, legal weight) and $a_i \in \{0, 1\}$ the anchored indicator. The expected *exploitable-trust exposure* of the portfolio is

$$R \;=\; \sum_i V_i \Big[(1 - a_i)\, F\big(t_{0,i} + L_i\big) + a_i\, F(t_{a,i})\Big]. \tag{11}$$

Given a remediation budget B and a per-artifact anchoring cost c_i, the optimal program solves

$$\min_{a \in \{0,1\}^n} \; R \qquad \text{subject to} \qquad \sum_i a_i c_i \le B, \tag{12}$$

a 0/1 knapsack whose greedy solution anchors artifacts in *decreasing order of*

$$\frac{\Delta R_i}{c_i} \;=\; \frac{V_i\big[F(t_{0,i} + L_i) - F(t_{a,i})\big]}{c_i}. \tag{13}$$

That ratio is the entire operational doctrine in one line: *anchor highest value × long-horizon-CRQC-probability per dollar first.*

So What

This model turns an unbounded panic ("we can never re-sign the past") into a finite, fundable program. You do not protect everything; you rank every signed artifact by $V_i\big[F(t_0{+}L){-}F(t_a)\big]$ and anchor down the list until the budget runs out. The output is a defensible board-level artifact: a prioritized anchoring backlog with explicit value, horizon, and cost. That is the deliverable a CISO can actually take to a risk committee.

5 The Remedy That Isn't Re-Signing

The Horizen post lands its punch with: you can re-encrypt a database, but you cannot re-sign every artifact you have already issued. True—but the conclusion "therefore the past is lost" does not follow. The correct conclusion is: *do not re-sign; anchor.*

5.1 Why re-signing fails as a strategy

Re-signing the legacy stock under PQC is infeasible for reasons that are operational, not cryptographic:

1. **Immutability.** On-chain artifacts (every ECDSA-signed Bitcoin/Ethereum transaction in history) *cannot* be re-signed; the ledger is append-only by design.
2. **Custody loss.** Signers, keys, and signing ceremonies for decades-old artifacts no longer exist.
3. **Counterparty unavailability.** A bilaterally signed contract cannot be unilaterally re-executed.
4. **Temporal blindness.** Even where re-signing is possible, a signature applied in 2034 does not prove the artifact existed in 2024. It loses the very thing authenticity needs: *when*.

5.2 Proof-of-existence anchoring

The right primitive is a quantum-safe *proof of existence*: incontrovertible evidence that a specific artifact (its hash) existed, intact, at a specific time prior to the CRQC. Mechanisms, in increasing order of independence:

- **Trusted timestamps (RFC 3161)** from a Time-Stamping Authority—*provided* the TSA itself signs with a PQC algorithm or anchors into a hash-based log.
- **Long-term archival signature formats**—ETSI PAdES-LTA, CAdES-A, XAdES-A—which were designed precisely to preserve signature validity *beyond* the cryptographic life of the original algorithm by layering archive timestamps and validation data.
- **Hash-based / Merkle notarization** into an append-only transparency log (the model used by Certificate Transparency), where inclusion proofs rest only on collision resistance.
- **Witnessed public anchoring**—publishing the digest $H(x)$ to a widely replicated, tamper-evident medium so that backdating becomes implausible across independent observers.

Proposition 2 (Soundness of anchoring)**.** *Let $\mathcal{A}$ be an append-only anchor in which an entry $e = (H(x), t)$, once committed, can neither be altered nor backdated, with security reducing to the collision and second-preimage resistance of H and to the integrity of $\mathcal{A}$. For any artifact x anchored at $t_a < t_q$, a forgery $x' \neq x$ presented after the CRQC fails verification against $\mathcal{A}$: either $H(x') \notin \mathcal{A}$, or any entry matching x' bears a timestamp $\geq t_q$. Because H is a hash function, its security is governed by Grover (survivable at adequate output length), not Shor.*

The hinge is the Shor/Grover split of Section 3: anchoring trades a Shor-broken signature for a Grover-survivable hash commitment. That is why the past is defensible even though it cannot be re-signed.

CryptoSoWhat Takeaway

Re-encrypt is to HNDL as anchor is to TNFL. For confidentiality you refresh the lock (re-encrypt under ML-KEM)—but note it does *not* save already-harvested ciphertext, which is why the HNDL clock already runs. For authenticity you do not re-sign; you *anchor* a hash-based, timestamped proof of existence before the CRQC arrives. One protects the lock; the other protects the date stamp. Different verbs, different primitives, same urgency.

6 The Verifier-Trust Dimension

A forged signature has power only if some relying party *verifies it as valid* at the moment of decision. This makes the verification policy—not just the signing key—a control surface. Two levers act on the verifier side, and both cut both ways.

Algorithm sunset s. If verifiers hard-reject classical signatures after s, then post-s forgeries are inert. But the *same* rejection breaks every un-migrated, un-anchored *genuine* artifact. Sunsetting without first anchoring your legitimate history is self-inflicted amnesia: you forget your own valid past in order to deny the forger. In the model, choosing $s < t_q$ empties the exposure window—at the cost of invalidating all artifacts not anchored or re-issued by s.

Trust-anchor migration. Rotating CA roots to PQC (ML-DSA, SLH-DSA, eventually FN-DSA) protects *future* chains, but artifacts chained to retired classical roots still require an anchor to retain provable validity across the rotation.

So What

The danger was never the forgery in the abstract—it is a relying party still trusting a broken algorithm at verification time. That means TNFL urgency is conditional on two variables most programs never write down: (1) the *authenticity lifetime* of each artifact, and (2) whether your verification stack will still honor RSA/ECDSA when the CRQC lands. The live wire is a long-lived root plus a verifier that never sunsets. Inventory both, or you are guessing.

7 The Sharpest Consequence: Non-Repudiation

The most concrete real-world harm of TNFL is rarely an operational forgery—it is the retroactive collapse of *non-repudiation*, the property that a signer cannot later deny having signed.

Once a CRQC can forge RSA/ECDSA signatures, any party to a classically signed instrument can argue: *"This signature proves nothing—after RSA fell, anyone could have produced it."* That argument, once credible, does not merely threaten future documents; it retroactively drains the evidentiary weight of every contract, audit attestation, board resolution, notarization, and regulatory filing ever signed under the broken scheme. Legal frameworks that lean on digital signatures as evidence—ESIGN and UETA in the United States, eIDAS in the European Union—rest on the assumption that a valid signature is hard to forge. Remove that assumption retroactively and the assumption's downstream reliance unwinds with it.

The defense is, once more, the anchor. A pre-CRQC qualified timestamp on the signature preserves its evidentiary value: it demonstrates that the signature both verified *and existed* before forgery was possible. eIDAS long-term-validation (LTV) and qualified electronic timestamps already encode exactly this idea; PAdES-LTA is its document-level realization.

Legal & Evidentiary Scope

TNFL is, in its highest-stakes form, an **evidence** problem, not merely a cryptographic one. The asset at risk is the legal admissibility and weight of your signed record. Counsel—not only security engineering—should own the question "which of our signed instruments must remain *provable* for how long, and have we time-stamped them under a quantum-safe regime before the window closes?" Treat long-lived contracts, IP assignments, regulatory attestations, and chain-of- custody records as evidentiary artifacts with explicit retention-of-provability requirements. This is not legal advice; it is a flag that the right owner for part of TNFL sits in the general counsel's office, not the SOC.

8 Governance: Two Questions, Two Owners, Two Clocks

Because the two threats fix to different standards and different system owners, a single "quantum-readiness" workstream structurally cannot close both. The separation must be explicit.

Table 2: The two questions, side by side.

Dimension	HNDL (confidentiality)	TNFL (authenticity)
Property attacked	Secrecy of data in transit/at rest	Authenticity / non-repudiation of signed artifacts
Risk equation	$X + Y > Z$ (Mosca)	$L + Y > Z$ for new issuance; $F(t_0{+}L)$ for legacy stock
NIST fix	ML-KEM (FIPS 203); HQC as backup KEM	ML-DSA (FIPS 204), SLH-DSA (FIPS 205); FN-DSA (FIPS 206, draft)
Remedy verb	Re-encrypt; rotate session keys	Anchor (proof of existence); migrate signing
Typical owner	Network / TLS / key-management team	PKI, code-signing, *and* general counsel
Clock	Already running (harvest is happening)	Attack awaits CRQC; *liabilities mint now* with long tails

8.1 The standards and the deadlines

The fixes are standardized and the deadlines are now policy, not speculation:

- **FIPS 203 (ML-KEM)**, **204 (ML-DSA)**, and **205 (SLH-DSA)** were finalized and made effective in August 2024. ML-KEM addresses the HNDL/confidentiality side; ML-DSA and the hash-based SLH-DSA address the TNFL/authenticity side. (SLH-DSA's hash-based construction is especially apt for high-assurance, long-life signing.)
- **FIPS 206 (FN-DSA, from Falcon)** is in draft as of this writing, with final publication expected in late 2026 or early 2027; its compact signatures suit root and intermediate certificates.
- **NIST IR 8547** sets the transition: RSA, ECDSA, EdDSA, DH, and ECDH are slated for *deprecation after 2030* and *disallowance after 2035*—meaning a disallowed signature scheme must thereafter be treated as forgeable.
- **NSM-10** and **CNSA 2.0** target 2035 for migration of national-security and federal systems, with software/firmware signing among the earliest required transitions.

Remark (On hybrids). During the transition, well-designed PQC/classical *hybrid* signatures and key-establishment remain the prudent default: security relies on the classical component until PQC is trusted, and on the PQC component once classical is disallowed, provided the hybrid never weakens either. For long-lived roots being issued now, hybrid issuance plus anchoring is the belt-and-suspenders posture.

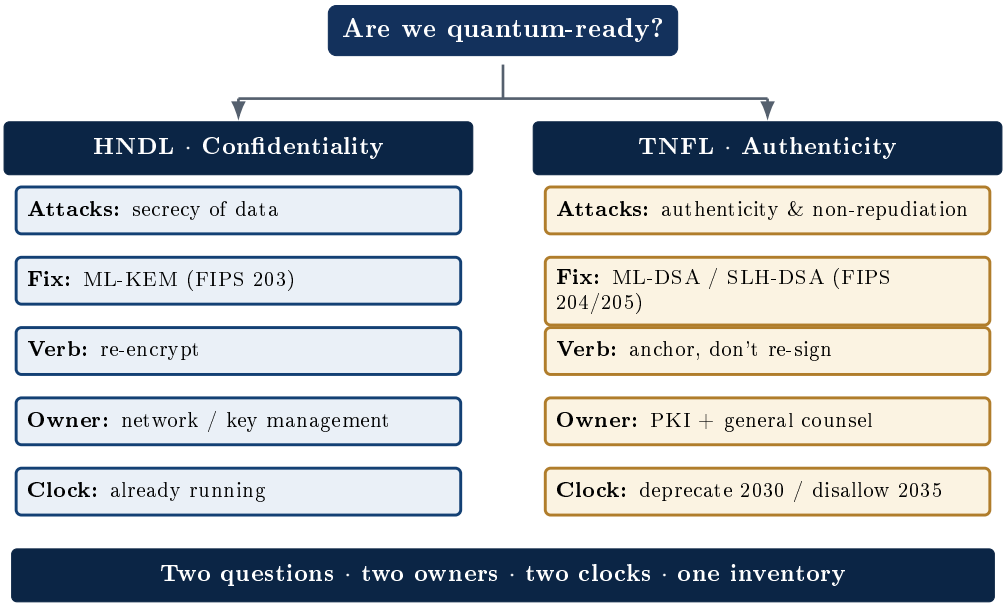

Figure 3: **Two questions, two owners, two clocks.** A single "quantum-readiness" workstream structurally cannot close both columns—the fixes, the owners, and the deadlines differ. Inventory every cryptographic artifact along both axes.

So What

Put the deadline in board language: any artifact you sign under RSA/ECDSA today is on an algorithm that U.S. standards intend to render *disallowed—i.e. presumed forgeable—after 2035*. If that artifact must remain trustworthy past 2035, you are already out of compliance with the trajectory. Assign an owner to each question (TLS/key-management for confidentiality; PKI *and* counsel for authenticity), give each a 2030/2035 milestone, and require both to report separately. One green light covering both is the failure mode.

9 Sector Applications

9.1 Financial infrastructure and the GENIUS-Act stablecoin

For Quantum Reserve Capital's thesis, TNFL is not abstract. A regulated, GENIUS-Act-compliant stablecoin is, at bottom, a chain of *authenticity claims*: a 1:1 backing assertion, monthly reserve disclosures, attestation signatures, and the on-chain transaction history itself. Each is a signed artifact with a long reliance lifetime, and the ledger is the canonical *un-re-signable* object:

- **On-chain history.** Every transaction is ECDSA over `secp256k1`; exposed public keys are Shor-vulnerable. As the companion paper noted, roughly a quarter of usable bitcoin already sits behind exposed keys. The historical ledger cannot be re-signed—it can only be *succeeded* by a PQC-aware chain and, for evidentiary purposes, externally anchored.
- **Reserve attestations and monthly disclosures.** These should be issued under ML-DSA (or hybrid) and time-stamped, so that the trust narrative of the peg—"the reserves were there, and we can prove it"—survives a future CRQC. A forgeable historical attestation chain is a latent attack on the peg's credibility.

- **Smart-contract deployments.** Contract code is immutable and its deploy is signed; treat the deploy signature and audit report as anchored artifacts.

CryptoSoWhat Takeaway

A stablecoin's entire value proposition is a *provable* authenticity claim about reserves. HNDL asks whether someone can read your disclosures; TNFL asks whether someone can one day *forge* them and deny yours. For a sovereignty-grade financial instrument, the authenticity question is the one that touches the peg. Issue attestations under PQC, time-stamp them now, and treat the immutable ledger as the artifact you anchor rather than re-sign.

9.2 Quantum communications (the QKD paradox)

There is an elegant trap in quantum communications worth naming for any quantum-systems evaluation. Quantum key distribution (QKD) offers *information-theoretic* confidentiality—it is, by its physics, immune to HNDL. But QKD does *not* authenticate the channel. To defeat a man-in-the-middle, the QKD link must be authenticated by classical means—pre-shared keys or public-key signatures—and *that* authentication layer is exactly the quantum-vulnerable surface TNFL describes.

$$\text{QKD solves } \underbrace{\text{confidentiality}}_{\text{HNDL-immune (physics)}} \text{ but inherits } \underbrace{\text{authentication}}_{\text{TNFL-exposed (classical signatures)}}. \tag{14}$$

A quantum-secure communications product can therefore be confidentiality-perfect and authenticity-fragile at the same time. The fix is to authenticate the quantum channel with PQC signatures (ML-DSA) or with pre-distributed symmetric keys, and to treat the authentication subsystem as a first-class TNFL surface in any technical evaluation.

So What

When evaluating a quantum-communications system, do not stop at "is the channel quantum-secure?" Ask: *how is the quantum channel authenticated, and is that authentication quantum-safe?* A QKD deployment whose authentication still rests on ECDSA has solved the famous problem and left the unglamorous one open—which is precisely the HNDL/TNFL framing error, reproduced inside a single product.

9.3 National-security and defense systems

For classified and defense contexts the two clocks run hardest. Classified material carries declassification horizons measured in decades, so its confidentiality shelf-life X and its authenticity lifetime L are both large—driving *both* Mosca inequalities into the danger zone simultaneously. The authenticity surface is acute and specific:

- **Firmware and software signing** for fielded platforms with 20–30 year lifecycles: a forged firmware signature is accepted as vendor-legitimate by every device in the fleet.
- **Command authentication** for remote and autonomous systems: a forged *authenticated command* is, by construction, indistinguishable from a legitimate one—a TNFL failure with kinetic consequences.

- **Supply-chain provenance manifests**: forged provenance launders malicious hardware into a trusted bill of materials.

CNSA 2.0 already mandates the migration of software/firmware signing among the first transitions for this reason. The doctrine generalizes: for any system whose signed artifacts outlive 2035, hybrid or PQC signing plus anchoring is the minimum responsible posture.

Legal & Evidentiary Scope

For national-security work the evidentiary frame becomes an operational and doctrinal one: the integrity of a command-authentication or firmware-signing chain is a trust property whose failure is not a lawsuit but a mission. The same anchoring discipline applies—establish quantum-safe proofs of existence for the artifacts that authorize action—under the broader principle that systems built to act autonomously must not inherit a forgeable trust root.

10 Conclusion

The Horizen team is right that quantum-readiness programs fail on framing, and right that confidentiality and authenticity are two threats, not one. We have argued for a small but consequential correction and a constructive remedy. TNFL is not a literal journey backward in time; it is a forward forgery aimed at a legacy stock, and that reframing is what makes it solvable. The obstacle—"you cannot re-sign every artifact you have issued"—is real, but it argues for *anchoring*, not surrender. The Shor/Grover split guarantees that while signatures are catastrophically broken, hash-based proofs of existence survive; anchoring an artifact before the CRQC arrives collapses its exposure from the long-horizon $F(t_0 + L)$ to the near-zero $F(t_a)$.

The discipline is therefore singular even though the threats are dual. Inventory every cryptographic artifact along two axes—confidentiality shelf-life X and authenticity lifetime L. Run each axis against its Mosca inequality. Migrate new issuance to PQC or hybrid. And for the legacy stock that cannot be re-signed, anchor it in descending order of $V_i\big[F(t_0+L)-F(t_a)\big]/c_i$ until the budget is spent. Two questions, two owners, two clocks—one inventory.

The Master CryptoSoWhat

Ask both questions, name both owners, set both clocks. For confidentiality: re-encrypt under ML-KEM and accept that the harvest clock already runs. For authenticity: stop minting classical long-lived liabilities, migrate signing to ML-DSA/SLH-DSA (FN-DSA for compact roots), and *anchor the past you cannot re-sign* with quantum-safe, time-stamped proofs of existence—*before* the CRQC arrives, because an anchor is only worth the date it predates. An artifact whose authenticity we cannot defend is an artifact whose history we are choosing to forget. The foundational test holds: will someone look at this in 100 or 1,000 years and say it was the right thing to do? Anchoring the record so it remains *provable* is how you earn a yes.

Glossary

CRQC Cryptographically Relevant Quantum Computer—one capable of breaking deployed public-key cryptography.

HNDL Harvest Now, Decrypt Later—archive ciphertext today, decrypt once a CRQC exists. Attacks confidentiality.

TNFL Trust Now, Forget Later—forge signatures on a legacy stock once Shor recovers signing keys. Attacks authenticity.

ML-KEM
Module-Lattice Key-Encapsulation Mechanism, FIPS 203 (from Kyber). Confidentiality fix.

ML-DSA
Module-Lattice Digital Signature Algorithm, FIPS 204 (from Dilithium). Authenticity fix.

SLH-DSA
Stateless Hash-based Digital Signature Algorithm, FIPS 205 (from SPHINCS+). Hash-based authenticity fix.

FN-DSA
FFT over NTRU-Lattice DSA, draft FIPS 206 (from Falcon). Compact signatures for roots.

ECDSA
Elliptic-Curve Digital Signature Algorithm—classical signature scheme (e.g. `secp256k1`). Shor-vulnerable.

QKD Quantum Key Distribution—information-theoretic confidentiality; requires classical authentication.

NSM-10 / CNSA 2.0
U.S. policy and NSA suite targeting 2035 migration of federal/national-security systems.

NIST IR 8547
Transition guidance: deprecate classical public-key after 2030, disallow after 2035.

Selected References

1. Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Computing*, 26(5), 1484–1509.
2. Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proc. 28th ACM STOC*, 212–219.
3. NIST (2024). *FIPS 203 (ML-KEM), FIPS 204 (ML-DSA), FIPS 205 (SLH-DSA).* Effective August 2024.
4. NIST (2025–2027, draft). *FIPS 206 (FN-DSA).* Initial Public Draft; final expected late 2026/early 2027.
5. Moody, D., et al. (2024). *NIST IR 8547 (ipd): Transition to Post-Quantum Cryptography Standards.*
6. National Security Agency (2022/2024). *CNSA 2.0 and Quantum Computing FAQ.*
7. The White House (2022). *National Security Memorandum 10 (NSM-10).*
8. ETSI. *PAdES (EN 319 142), CAdES (EN 319 122), XAdES (EN 319 132)*—long-term archival signature profiles.
9. Adams, C., et al. (2001). *RFC 3161: Internet X.509 PKI Time-Stamp Protocol (TSP).*
10. Mosca, M. (2018). Cybersecurity in an era with quantum computers: Will we be ready? *IEEE S&P*, 16(5), 38–41.

11. Roetteler, M., et al. (2017). Quantum resource estimates for computing elliptic-curve discrete logarithms. *ASIACRYPT*.

12. Carmichael, G. S. *Harvest Now, Decrypt Later* (companion analysis), and related *CryptoSoWhat* works on quantum risk to financial and industrial-control infrastructure.

13. Horizen Labs (2026). On the two distinct quantum threats to security programs (public post).

Prepared by Dr. Gregory S. Carmichael for CryptoSoWhat and LinkedIn. PhD rigor, dinner-table clarity, claims substantiated, math shown, assumptions named.