Read Only-Once Memory (ROOM): A Hardware Enforcement Layer for CNSA 2.0, NIST Post-Quantum Cryptography, and TLS 1.3

Francis X. Cunnane III
QSymbolic LLC / NETWORKCRYPT
Las Vegas, Nevada, USA
frank@qsymbolic.com ORCID 0009-0002-1146-2916

***Abstract*—The cryptographic algorithms of the post-quantum era are settled. NIST has published FIPS 203 (ML-KEM), FIPS 204 (ML-DSA), and FIPS 205 (SLH-DSA), and has approved the stateful hash-based signatures LMS and XMSS in SP 800-208. The NSA's Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) selects ML-KEM-1024, ML-DSA-87, LMS, XMSS, AES-256, and SHA-384/512 for National Security Systems, with exclusive-use deadlines running from 2030 to 2033. What is *not* settled is how to implement these algorithms so that their security assumptions survive contact with real hardware. Nearly every catastrophic failure mode of the new suite reduces to one root cause: *secret material that is used, or available, more than once*. A reused LMS/XMSS one-time key enables signature forgery. A retained ML-KEM decapsulation key destroys forward secrecy. A repeated AES-GCM nonce leaks the authentication key. A cloned or rolled-back signer state silently converts a provably secure scheme into a broken one.**

This paper presents Read Only-Once Memory (ROOM), a patent-pending memory primitive in which the act of reading a cell irreversibly collapses it, and in which cells can be organized into *collapse domains* that are consumed atomically. We map every CNSA 2.0 and NIST PQC single-use requirement to a ROOM enforcement mechanism, show how ROOM converts procedural "shall not reuse" obligations into physical impossibilities, show that single use also denies side-channel attackers the repeated measurements that statistical power and electromagnetic analysis depend on, and describe an FPGA reference implementation. We then work through TLS 1.3 under the NSA's CNSA 2.0 TLS profile in full, treating every node of the TLS 1.3 key schedule—the ephemeral ML-KEM-1024 decapsulation key, the handshake and master secrets, the traffic secrets ratcheted by KeyUpdate, and resumption tickets—as a secret that is read exactly once, and we size the memory this requires. ROOM does not replace any approved algorithm. It is the hardware layer that makes approved algorithms behave the way their security proofs assume—and it shortens the path from "we run ML-KEM" to "we can demonstrate to an assessor that our ML-KEM, LMS, and AES-GCM keys cannot be reused."

***Index Terms*—post-quantum cryptography, CNSA 2.0, ML-KEM, ML-DSA, LMS, XMSS, stateful hash-based signatures, key zeroization, forward secrecy, hardware security, FIPS 140-3, TLS 1.3, read-once memory, secure key storage**

I. Introduction

The migration to post-quantum cryptography (PQC) is the largest coordinated cryptographic transition in history. Unlike the move from DES to AES or from SHA-1 to SHA-2, it replaces *every* public-key primitive in use—key establishment, digital signatures, and code signing—on a hard schedule driven by the "harvest now, decrypt later" (HNDL) threat: adversaries are recording encrypted traffic today in anticipation of a cryptographically relevant quantum computer (CRQC) tomorrow.

The standards bodies have done their part. In August 2024, the Secretary of Commerce approved FIPS 203, 204, and 205 [1]–[3]. NIST selected HQC as a backup key-encapsulation mechanism (KEM) in March 2025 [6] and published SP 800-227, its recommendations for deploying KEMs, in September 2025 [5]. FALCON is in development as FIPS 206 (FN-DSA) [8]. NIST IR 8547 sets the deprecation and disallowance schedule for quantum-vulnerable algorithms [7]. For National Security Systems (NSS), the NSA's CNSA 2.0 fixes the algorithm menu at its highest-strength parameter sets [9], and the NSA is publishing IETF profiles that bind those algorithms into TLS 1.3 and S/MIME [10], [13].

A. The Gap Between Algorithm and Assurance

An approved algorithm is a mathematical object. Its security theorem holds under explicit conditions—and the PQC suite carries more of these conditions than any suite before it:

- LMS and XMSS are *stateful*: each one-time signing key must be used exactly once, ever. SP 800-208 devotes substantial text to this requirement because violating it is fatal [4].
- ML-KEM's forward-secrecy benefit in protocols such as TLS 1.3 depends on ephemeral decapsulation keys being destroyed after a single use [5].
- ML-DSA's default *hedged* signing mode depends on fresh per-signature randomness [2].
- AES-256-GCM, mandated by CNSA 2.0 for bulk encryption, depends on never repeating an IV under a given key [14].

Each condition is, at bottom, a *single-use* condition. And in conventional hardware, single use is a matter of policy,

enforced by software that can crash, be rolled back, be snapshotted, be cloned, or be exploited. Memory that is read once can be read twice. Memory that is "zeroized" may retain remanence [20]. A virtual machine running an LMS signer can be snapshotted and restored, rewinding its state counter and causing it to reissue one-time keys.

B. *The* ROOM *Proposition*

Read Only-Once Memory (ROOM) closes this gap at the physical layer. A ROOM cell yields its contents exactly once; the read operation itself destroys the stored value, and no subsequent operation—by software, firmware, a debugger, or an attacker with physical access—can recover it. Cells are organized into *collapse domains* so that a multi-word secret (an LM-OTS private key, a 3168-byte ML-KEM-1024 decapsulation key) is consumed atomically: reading any part collapses the whole.

The result is a simple but powerful substitution:

The ROOM value proposition

Every "*shall not be reused*" in the PQC standards becomes "*cannot be reused*." Compliance stops being a property you argue and becomes a property of the silicon.

C. *Contributions*

This paper makes the following contributions:

1) An analysis of why zeroization, the standards' default answer to single use, is structurally insufficient (Section II).
2) A systematic inventory of single-use requirements across CNSA 2.0, FIPS 203/204/205, SP 800-208, SP 800-227, SP 800-38D, and FIPS 140-3 (Section IV).
3) A threat taxonomy showing that the dominant real-world PQC failure modes are reuse, retention, and rollback, none of which is addressed by algorithm selection (Section V).
4) The ROOM architecture, including collapse cells, collapse domains, and a hardware-enforced monotonic index (Section VI).
5) Concrete integration designs for LMS/XMSS, ML-KEM-1024, ML-DSA-87, AES-256-GCM, and SHA-384/512 contexts (Section VII).
6) A complete TLS 1.3 integration under the CNSA 2.0 TLS profile, covering the key schedule, KeyUpdate, resumption and 0-RTT, HelloRetryRequest, and capacity sizing (Section VIII).
7) An analysis of how single use eliminates side-channel averaging, the statistical engine of differential power and EM attacks (Section IX).
8) A requirement-by-requirement compliance mapping and a discussion of how ROOM supports FIPS 140-3 validation (Section X).
9) An FPGA reference implementation and deployment guidance (Sections XI–XII).

II. Why Zeroization Is Not Enough

Every standard discussed in this paper ultimately answers the single-use problem the same way: *zeroize the secret when you are done with it*. FIPS 203 and FIPS 204 require intermediate values to be destroyed, SP 800-227 asks for ephemeral keys to be destroyed after use, and FIPS 140-3 requires sensitive security parameters to be zeroized and rendered unrecoverable [1], [2], [5], [15]. Zeroization is necessary. It is not sufficient, for six structural reasons.

A. *Zeroization Happens* After *Use*

Zeroization is a second operation that follows the first. Between "use the key" and "erase the key" there is always a window, and everything that can go wrong in that window defeats it: a crash, a power loss, an interrupt that never returns, a watchdog reset, or a fault-injection glitch that skips the erase instruction. Zeroization code is also the code least likely to run on error paths, which are exactly the paths an attacker tries to force.

B. *Software Cannot Reliably Erase What It Cannot See*

A secret in use is copied far beyond the buffer the programmer zeroizes: into CPU registers, cache lines, stack frames spilled by the compiler, temporary buffers inside libraries, swap and hibernation files, crash dumps, and debug logs. Erasing the "official" copy leaves the others. Optimizing compilers compound the problem by removing writes to memory that is never read again—including the `memset` intended to clear a key. This dead-store elimination has repeatedly been shown to strip zeroization out of real cryptographic software [21].

C. *Overwriting Memory Does Not Erase It Physically*

Writing zeros changes the logical value of a cell, not necessarily its physical history. DRAM retains its contents for seconds to minutes after power loss, far longer when chilled, enabling cold-boot key recovery [20]. SRAM shows low-temperature data remanence and long-term imprinting of values held for extended periods [22]. Flash and SSD storage erase in large blocks and remap writes through wear leveling, so an "overwrite" often lands in a new location while the old data persists [23]. Zeroization addresses the logical copy and leaves the physical one.

D. *Zeroization Is Unverifiable*

It is easy to show that a key was used; it is very hard to show that every copy of it is gone. An assessor can review the zeroization routine and its call sites, but cannot observe the absence of a copy in a cache line, a compiler spill, or a swapped page. Zeroization is therefore accepted on the strength of procedure and code review, not demonstrated as a property of the hardware.

TABLE I
ZEROIZATION VERSUS READ-ONCE COLLAPSE

Aspect	Zeroization	ROOM collapse
Timing	Separate step after use; window exists	Same event as use; no window
Trigger	Software must run	Intrinsic to the read circuit
Copies	Only the buffer it targets	Secret never leaves the module datapath
Physical erasure	Logical overwrite; remanence possible	Electrical grounding of the cell
Snapshot or clone reuse	Not prevented	Impossible; reading consumes
Side-channel traces per secret	Unlimited while key is live	Exactly one
Verification	Code review and procedure	Black-box physical test
Failure mode	Fails open (secret survives)	Fails closed (secret gone)

E. Zeroization Does Not Prevent Reuse

Zeroization destroys one copy at one moment. It does nothing about the copies that exist elsewhere *before* that moment. A VM snapshot taken before a signature, a backup of a signer's state, or a cloned HSM image all contain keys that are later zeroized in the original—and remain perfectly usable in the copy. For LMS and XMSS, where reusing a one-time key enables forgery, this is the central risk [4], and zeroization does not touch it.

F. Zeroization Depends on Privileged Software Behaving

The erase is issued by software. An attacker who controls the operating system, hypervisor, or module firmware can delay it, suppress it, or read the secret before it runs. Zeroization protects keys only from adversaries who do not already control the code responsible for zeroizing them.

G. From Procedure to Property

Table I contrasts zeroization with ROOM's approach, in which the read that delivers a secret is the same physical event that destroys it.

The bottom line

Zeroization is a promise that software makes about the future. ROOM is a fact about the past: once the secret has been read, it no longer exists.

III. THE REGULATORY LANDSCAPE

A. NIST Post-Quantum Standards

Table II summarizes the NIST PQC portfolio as of this writing.

NIST IR 8547 proposes that quantum-vulnerable public-key algorithms be deprecated after 2030 and disallowed after 2035 [7]. Federal civilian agencies are additionally bound by OMB M-23-02 inventory and reporting obligations [16].

TABLE II
NIST POST-QUANTUM AND SUPPORTING STANDARDS

Document	Scope	Status	Single-use condition
FIPS 203	ML-KEM	Final, 2024	Ephemeral keys; fresh encaps. randomness
FIPS 204	ML-DSA	Final, 2024	Fresh hedging randomness rnd
FIPS 205	SLH-DSA	Final, 2024	Stateless (none intrinsic)
SP 800-208	LMS, XMSS	Final, 2020	Each OTS key used once; state never rolls back
SP 800-227	KEM guidance	Final, 2025	Destroy ephemeral keys after use
FIPS 206	FN-DSA	In development	Fresh per-signature sampling
HQC	Backup KEM	Selected 2025	As for ML-KEM
SP 800-38D	AES-GCM	Final	IV unique per key
IR 8547	Transition	Draft	Retire classical PK by 2035

TABLE III
CNSA 2.0 ALGORITHM SUITE

Function	Algorithm	Standard
Key establishment	ML-KEM-1024	FIPS 203
Signatures (general)	ML-DSA-87	FIPS 204
Software/firmware signing	LMS (SHA-256/192 rec.), XMSS	SP 800-208
Symmetric encryption	AES-256	FIPS 197
Hashing	SHA-384, SHA-512	FIPS 180-4

B. CNSA 2.0

CNSA 2.0 is the NSA's algorithm suite and transition timeline for NSS [9]. Its selections, shown in Table III, take the highest-strength (NIST Category 5) parameter sets and add the stateful hash-based signatures for software and firmware signing, reflecting their conservative, hash-only security assumptions. Notably, CNSA 2.0 includes ML-KEM and ML-DSA but does not include SLH-DSA—the one NIST signature standard that is stateless. CNSA 2.0 thus leans *more* heavily on stateful, single-use mechanisms than the NIST portfolio as a whole.

The NSA's draft TLS 1.3 profile requires AES-256-GCM, ML-KEM-1024 for key establishment, ML-DSA-87 for authentication (HashML-DSA not permitted), and SHA-384 for HKDF [10]; Section VIII treats it in detail. The S/MIME profile likewise mandates ML-KEM-1024 and ML-DSA-87 [13].

C. CNSA 2.0 Timeline

Table IV gives the NSA's category-by-category schedule. Two dates dominate procurement planning: software and firmware signing must use CNSA 2.0 exclusively by 2030, and nearly all remaining categories must do so by 2033. Vendors selling into the defense supply chain should expect CNSA 2.0

TABLE IV
CNSA 2.0 TRANSITION SCHEDULE (PER NSA GUIDANCE)

System category	Support/prefer	Exclusive
Software/firmware signing	2025	2030
Web browsers, servers, cloud	2025	2033
Traditional networking (VPN, routers)	2026	2030
Operating systems	2027	2033
Niche equipment	2030	2033
Custom and legacy	—	2033

conformance to appear in acquisition requirements well before those exclusive-use dates.

Why this matters for ROOM

The first and hardest CNSA 2.0 deadline—software and firmware signing—falls on the algorithms with the strictest single-use requirement: LMS and XMSS. ROOM addresses that requirement directly.

IV. Single-Use Requirements in the PQC Standards

We now inventory, standard by standard, every requirement whose violation reduces to reuse, retention, or rollback of secret material.

A. Stateful Hash-Based Signatures (SP 800-208)

LMS [17] and XMSS [18] build a many-time signature scheme from a Merkle tree over 2^h one-time signature (OTS) keys. Signing message m consumes leaf q: the signer produces an OTS signature using the private OTS key at index q, attaches the authentication path, and advances q.

The OTS component (LM-OTS, WOTS$^+$) reveals intermediate hash-chain values that depend on the message digest. One signature reveals enough to verify *that* message. Two signatures under the same OTS key, on different messages, reveal chain values at different positions—and an adversary who holds both can, with modest work, compute a valid OTS signature on a third message of its choosing. The security reduction for the whole scheme collapses.

SP 800-208 therefore imposes requirements that are unusual among NIST standards [4]:

- Key and signature generation must be performed in hardware cryptographic modules that do not permit export of secret keying material, validated under FIPS 140 at an elevated security level.
- The signer must update its state in non-volatile storage *before* releasing a signature, so a crash cannot cause reuse.
- The standard warns explicitly against operational practices—such as cloning, backup-and-restore, and virtual-machine snapshots—that can cause state to be duplicated or rewound.

Conventional HSMs meet these requirements through firmware discipline and procedural controls. ROOM meets them through physics.

B. ML-KEM (FIPS 203) and KEM Deployment (SP 800-227)

ML-KEM-1024 key generation produces a 1568-byte encapsulation key ek and a 3168-byte decapsulation key dk. Encapsulation consumes 32 bytes of fresh randomness m and yields a 1568-byte ciphertext c and a 32-byte shared secret K [1].

Three single-use conditions follow:

1) **Ephemeral key destruction.** In TLS 1.3 and similar protocols, the client generates a fresh (ek, dk) per handshake. Forward secrecy holds only if dk is destroyed after decapsulation. SP 800-227 recommends that ephemeral keys be used once and then destroyed, and treats key retention as a threat to the confidentiality of past sessions [5].
2) **Encapsulation randomness.** Reuse of m across encapsulations to the same ek produces identical shared secrets.
3) **Intermediate secrets.** FIPS 203 requires that intermediate values (the seed d, z, the pre-key $\bar{K}$) be destroyed when no longer needed.

Under HNDL, condition (1) is decisive. An adversary who records a TLS handshake today and later recovers the client's retained dk—from a memory dump, a crash file, a swap partition, or a seized device—decrypts the session without ever breaking ML-KEM.

C. ML-DSA (FIPS 204)

ML-DSA-87 uses 2592-byte public keys, 4896-byte private keys, and 4627-byte signatures [2]. Its default *hedged* variant mixes a fresh 32-byte random value rnd into the per-signature mask derivation. The deterministic variant ($rnd = 0^{256}$) is permitted but is more exposed to fault-injection attacks, in which an attacker who induces a fault while re-signing the same message obtains two related signatures that leak key material. Hedging defeats this only if rnd is genuinely fresh: an implementation that can be coerced into reusing rnd degenerates to the deterministic case.

D. AES-256-GCM (SP 800-38D)

GCM's confidentiality and, critically, its authenticity depend on IV uniqueness under a key [14]. A single IV repetition reveals the XOR of two plaintexts and allows recovery of the GHASH authentication subkey H, after which the attacker can forge messages at will. SP 800-38D's deterministic IV construction relies on a counter that must never roll back, which is again a state-integrity problem.

E. FIPS 140-3 Zeroization

FIPS 140-3, through ISO/IEC 19790, requires cryptographic modules to zeroize sensitive security parameters (SSPs) and to render them unrecoverable, with requirements that tighten at higher security levels [15]. Conventional SRAM and DRAM

exhibit data remanence that can survive power loss, especially at low temperature [20], and flash memory is notoriously difficult to erase completely. Zeroization is therefore a verified procedure, not a guaranteed outcome.

F. Summary

Table V collects the conditions. Every row describes an obligation that, in current practice, is discharged in software.

V. Threat Model: How PQC Actually Fails

Quantum-resistant mathematics does not protect against classical implementation failures. We consider an adversary $\mathcal{A}$ with the following capabilities, drawn from documented attack classes.

- **T1 – State rollback.** $\mathcal{A}$ restores a signer from a backup, VM snapshot, or cloned image, rewinding the LMS/XMSS leaf index or GCM counter. No exploit is required; ordinary IT operations suffice.
- **T2 – State duplication.** Two instances of a signer run from the same key for high availability, each unaware of the other's consumed leaves.
- **T3 – Crash-induced reuse.** The signer emits a signature, then fails before persisting the advanced index; on restart it reuses the leaf.
- **T4 – Retention.** An ephemeral dk, traffic secret, or OTS key persists in RAM, cache, swap, crash dumps, or logs after its intended lifetime.
- **T5 – Remanence.** $\mathcal{A}$ with physical access recovers "zeroized" material through cold-boot or similar techniques [20].
- **T6 – DMA and bus snooping.** $\mathcal{A}$ reads key material over a peripheral bus or from memory shared with a compromised co-processor.
- **T7 – Fault injection.** $\mathcal{A}$ glitches power or clock to force repeated or corrupted operations, e.g. re-signing with reused randomness.
- **T8 – Privileged software compromise.** $\mathcal{A}$ controls the OS or hypervisor and can issue arbitrary read requests to key storage.
- **T9 – Side-channel averaging.** $\mathcal{A}$ records power or electromagnetic traces across many operations that use the same secret and combines them statistically to recover it.

T1–T3 are not attacks in the traditional sense; they are *operational accidents* that become attacks. They are especially dangerous because they produce no error, no log entry, and no alarm—the system continues to emit valid-looking signatures while its security silently evaporates. T4–T8 are classical memory attacks, and T9 is the classical physical attack, that PQC inherits unchanged.

The uncomfortable truth

Upgrading to ML-KEM-1024 and LMS does nothing to stop T1–T9. An organization can be fully CNSA 2.0 compliant on paper and still sign forged firmware or leak recorded sessions because a VM was restored from Tuesday's snapshot.

VI. The ROOM Architecture

ROOM is described in U.S. patent filings claiming priority to provisional application 63/822,892 (patent pending) and in prior technical disclosures [28], [29]. We summarize the architecture here at the level needed for integration.

A. The Collapse Cell

The fundamental element is the *collapse cell*: a storage element with three logical states,

$$S \in \{\textsc{Empty},\ \textsc{Armed}(v),\ \textsc{Collapsed}\}. \tag{1}$$

A $\texttt{WRITE}(v)$ moves the cell from EMPTY to ARMED(v). A $\texttt{READ}$ on an ARMED(v) cell returns v and transitions the cell to COLLAPSED in the same operation. The transition is *irreversible*: there is no path from COLLAPSED back to ARMED, and a $\texttt{READ}$ on a COLLAPSED cell returns a fixed sentinel (or a hardware fault) that carries no information about v.

Formally, for any sequence of operations σ and any cell holding v,

$$\left|\{\, i : \sigma_i = \texttt{READ} \wedge \mathrm{out}(\sigma_i) = v \,\}\right| \le 1. \tag{2}$$

Property (2) is enforced by the cell circuit, not by any controller, firmware, or access-control list. No software at any privilege level can issue a "non-destructive read."

B. Collapse Domains

PQC secrets are large: an LM-OTS private key for LMS SHA-256/192 is many hundreds of bytes, and an ML-KEM-1024 dk is 3168 bytes. Reading such a secret word by word from independent collapse cells would leave a window in which part of the secret has been consumed and part remains—and an attacker who interrupts the read could leave the remainder recoverable.

ROOM therefore groups cells into *collapse domains*. A domain D is a set of cells bound so that the first $\texttt{READ}$ to any cell in D collapses *every* cell in D, delivering the full contents to an on-module consumer (e.g. the LM-OTS signing engine) through a dedicated, non-addressable datapath. Partial consumption is impossible by construction. The collapse-domain array is a central subject of the pending ROOM patent application.

C. Hardware Monotonic Index

A collapse-domain array naturally implements a *hardware monotonic index*. Let domains $D_0, D_1, \ldots, D_{N-1}$ hold successive one-time secrets. The current index q is defined as the lowest-numbered armed domain. Because collapse is irreversible, q can only increase. There is no counter register to roll back, no NVRAM word to restore from backup: the index *is* the physical state of the array.

This eliminates T1–T3 for any state that ROOM holds:

- **Rollback (T1)** is impossible, because restoring a backup of the host does not restore collapsed cells.
- **Duplication (T2)** is impossible, because a ROOM device cannot be cloned by reading it; reading consumes it.

TABLE V
SINGLE-USE CONDITIONS ACROSS CNSA 2.0 AND NIST PQC, WITH CONSEQUENCE OF VIOLATION

Algorithm / Doc.	Secret	Condition	Consequence of violation
LMS / XMSS (SP 800-208)	LM-OTS / WOTS$^+$ private key at leaf q	Used for exactly one signature; q never rolls back	Existential forgery; attacker signs arbitrary firmware
ML-KEM-1024 (FIPS 203, SP 800-227)	Ephemeral dk	Destroyed after one decapsulation	Loss of forward secrecy; recorded sessions decrypted (HNDL)
ML-KEM-1024	Encapsulation randomness m	Fresh per encapsulation	Repeated shared secrets
ML-KEM-1024	Intermediates d, z, $\bar{K}$	Destroyed when no longer needed	Key recovery from memory
ML-DSA-87 (FIPS 204)	Hedging value rnd	Fresh per signature	Degrades to deterministic mode; fault-attack key leakage
AES-256-GCM (SP 800-38D)	IV / invocation counter	Unique per key; never rolls back	Plaintext XOR leak; GHASH key recovery; forgery
TLS 1.3 / HKDF-SHA-384	Traffic secrets, PSK	Destroyed on rekey / session end	Decryption of past traffic
FIPS 140-3	All SSPs	Zeroized and unrecoverable	Validation failure; key extraction

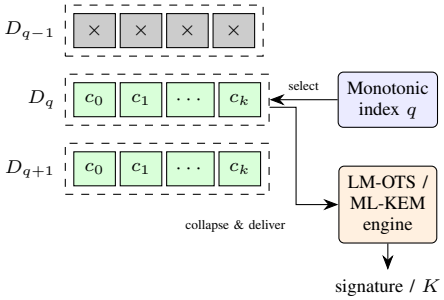


Fig. 1. ROOM collapse-domain array. Domain D_{q-1} is collapsed (consumed); D_q is selected by the hardware monotonic index and delivered atomically to the on-module engine; D_{q+1} remains armed.

- **Crash-induced reuse (T3)** is impossible, because consumption and delivery are the same physical event; there is no window between "use" and "record use."

D. Terminal and Recyclable Domains

Two classes of collapse domain serve different workloads.

- **Terminal domains.** COLLAPSED is permanent. The consumed state is itself the record of use, which is what the hardware monotonic index requires. LMS/XMSS one-time keys, HSS subtree seeds, boot-epoch KEKs, and attestation secrets use terminal domains.
- **Recyclable domains.** Collapse destroys the value in the same event that delivers it, exactly as in a terminal domain. Afterwards, and only by a separate deliberate operation, the on-module engine may return the cell to EMPTY and re-arm it with *fresh* output of the approved DRBG. No operation restores the old value.

Recyclable domains keep P1, P2, P4, and P5 for every value they ever hold; they give up P3, because the cell no longer records that a value was consumed. That trade is appropriate for secrets that are not indexed—TLS ephemeral keys, key-schedule secrets, IVs, and hedging randomness—and inappropriate for LMS/XMSS leaves, which is why the two classes are kept separate. Section VIII-I shows that recyclable domains are a practical necessity for TLS at any meaningful handshake rate.

E. Electrical Grounding

ROOM uses *electrical grounding* of the storage node [29]: on collapse or power removal, the cell is tied to ground so that it retains no recoverable charge. This addresses remanence (T5) and reduces the attack surface for cold-boot recovery of material that was armed but not yet consumed.

F. Isolation

Collapse domains deliver their contents through a dedicated datapath to an on-module cryptographic engine. Secret material never transits the system bus, so DMA (T6) and privileged-software reads (T8) obtain at most the engine's public outputs: signatures, ciphertexts, or derived session keys destined for the host by design.

G. Security Properties

We summarize the guarantees ROOM provides for any secret s stored in a collapse domain:

- **P1 Single delivery.** s is delivered to at most one consumer operation, ever.
- **P2 Atomicity.** s is consumed in full or not at all.
- **P3 Monotonicity.** The set of consumed domains only grows.
- **P4 Non-clonability.** No read-based process yields a copy of s while leaving the original usable.
- **P5 Non-persistence.** After consumption, s is not recoverable from the cell by electrical means, because the cell is electrically grounded on collapse [29].

VII. INTEGRATING ROOM WITH THE CNSA 2.0 SUITE

A. LMS and XMSS: The Flagship Application

Stateful hash-based signatures are the canonical ROOM workload: the standard *already* defines secrets that must be used exactly once, in order, with a monotonically increasing index.

Algorithm 1 ROOM-backed LMS signing

Require: Message m; collapse-domain array $\{D_i\}$
Ensure: LMS signature Σ or EXHAUSTED

```
1: q ← LOWESTARMED({D_i})                    ▷ hardware index
2: if q = ⊥ then
3:     return EXHAUSTED
4: end if
5: x_q ← COLLAPSEREAD(D_q)                   ▷ atomic, irreversible
6: σ_ots ← LM-OTS-Sign(x_q, m, q)            ▷ on-module
7: DESTROY(x_q)                              ▷ engine-local scratch
8: path ← AUTHPATH(q)                        ▷ public data
9: return Σ = (q, σ_ots, type, path)
```

1) Provisioning: Within the module, the signer generates the LMS private seed and derives the 2^h LM-OTS private keys. Each OTS key x_q is written into collapse domain D_q. The Merkle tree is computed and the public root $T[1]$ exported. The seed is then consumed (it, too, resides in a collapse domain) so that OTS keys can never be re-derived outside the array. For hierarchical (HSS) deployments in which lower-level subtrees are generated on demand, each subtree seed is itself held in a collapse domain and consumed when its subtree is provisioned.

2) Signing: Algorithm 1 shows the ROOM-backed signing flow.

Line 3 is the decisive step. In a conventional HSM, the signer must *first* write $q+1$ to NVRAM, *then* sign, and must trust that the NVRAM write is durable and that nobody will restore an older NVRAM image. With ROOM, "advance the index" and "obtain the key" are one physical event.

3) Hierarchical (HSS) and multi-tree XMSS^{MT}: For HSS, each subtree's OTS keys occupy their own domain range, and the signature on each child root consumes one domain in the parent range. The same monotonicity property applies at every level.

4) Exhaustion as a feature: A ROOM LMS device has a finite, physically determined signature budget $N = 2^h$ (or the HSS product). Exhaustion is reported cleanly and cannot be circumvented. For firmware-signing roots of trust—which CNSA 2.0 targets first—a budget of 2^{20} or even 2^{15} signatures is typically ample for the lifetime of a product line, and the hard ceiling is itself an audit control.

5) High availability without state sharing: SP 800-208 cautions against distributing a single LMS key across signers. With ROOM, high availability is achieved by partitioning: an HSS top-level key signs subordinate keys, each provisioned on a separate ROOM device with a disjoint leaf range. Because no two devices can ever hold the same domain, duplication (T2) cannot occur even under operator error.

B. ML-KEM-1024: Physically Enforced Forward Secrecy

1) Ephemeral keys: For each handshake, the module runs ML-KEM.KeyGen, exports ek to the host for transmission, and writes dk (3168 bytes) into a collapse domain. When the peer's ciphertext c arrives, ML-KEM.Decaps reads dk via COLLAPSEREAD, computes K, and the domain is gone.

This implements SP 800-227's ephemeral-key guidance as a hardware invariant. After decapsulation there is no copy of dk anywhere: not in RAM, not in a crash dump, not in swap. A device seized one hour or ten years after the session yields nothing that helps decrypt recorded traffic.

HNDL, closed at both ends

ML-KEM-1024 protects recorded traffic against a future quantum computer. ROOM protects it against a future *seizure*. Together they close the harvest-now-decrypt-later threat for both mathematical and physical adversaries.

2) Pre-provisioned ephemeral pools: ML-KEM-1024 key generation is fast, but for latency-critical or power-constrained endpoints a module can pre-generate a pool of (ek, dk) pairs, writing each dk to its own domain and caching each ek. Every pool entry is single-use by construction, so pre-generation introduces no reuse risk.

3) Encapsulation randomness and intermediates: The encapsulator's 32-byte m, and the intermediates d, z, and $\bar{K}$, can be staged in collapse domains filled by the module's approved DRBG. This guarantees that no randomness value is consumed twice, even under fault injection that attempts to replay an encapsulation (T7).

4) Static ML-KEM keys: Long-term (static) decapsulation keys, used for example in S/MIME, are by definition reused and are *not* placed in single-collapse domains. Section XIV discusses how ROOM still contributes, through read-once unwrapping keys, to protecting static keys at rest.

C. ML-DSA-87

The ML-DSA-87 private key is long-lived and is not itself a read-once secret. ROOM contributes in two places:

1) **Hedging randomness.** Each signature's 32-byte rnd is drawn from a collapse domain filled by the DRBG. A fault-injection campaign that forces the engine to re-sign cannot obtain the same rnd twice, preserving the hedged mode's protection.
2) **Key-at-rest wrapping.** The ML-DSA private key can be stored encrypted under a key-encryption key (KEK) held in a collapse domain per boot epoch, so that the plaintext key exists only within a single authorized session. See Section VII-E.

D. AES-256-GCM

For GCM, ROOM provides two mechanisms:

- **One-time IVs.** A pool of IVs (or IV fixed-field/counter blocks) is written into collapse domains. Each encryption consumes one. IV reuse under a key becomes impossible, satisfying SP 800-38D without relying on a counter in rewritable memory.
- **One-time data keys.** For store-and-forward or high-value-message scenarios, each message's AES-256 key can be delivered from a collapse domain, yielding per-message keys that self-destruct after use.

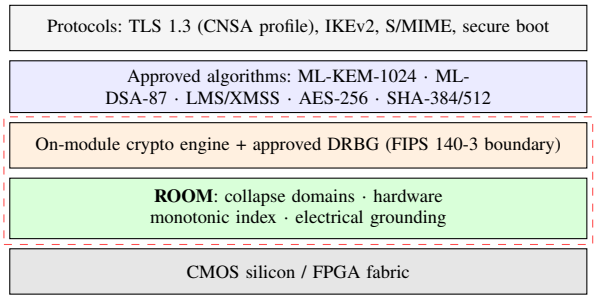


Fig. 2. ROOM in a CNSA 2.0 cryptographic module. ROOM sits beneath the approved algorithms and inside the module boundary (dashed); it changes no algorithm, wire format, or protocol.

E. SHA-384/512 and HKDF Contexts

CNSA 2.0's TLS profile derives traffic keys with HKDF-SHA-384. TLS 1.3's key schedule is a chain of secrets, each of which should be erased once its successor is derived. Holding each schedule secret in a collapse domain makes the key schedule a one-way ratchet in hardware: once the handshake secret yields the traffic secrets, it is physically gone. Section VIII works this through for the full TLS 1.3 handshake.

The same pattern yields *epoch keys*: a KEK consumed at boot (or at session start) unwraps long-term keys into engine-local memory for one epoch. A rolled-back host image cannot replay the epoch because the KEK domain is already collapsed.

F. Integration Summary

Figure 2 positions ROOM in a CNSA 2.0 module.

VIII. TLS 1.3 Under the CNSA 2.0 Profile

TLS 1.3 [19] carries most of the traffic that CNSA 2.0 must protect: web and cloud services (exclusive use by 2033), management planes, and the TLS-based VPNs and gateways in the 2030 networking category. It is also where harvest-now-decrypt-later recording is easiest, because TLS traffic crosses networks the adversary can tap. This section works through a complete ROOM-backed TLS 1.3 implementation under the NSA's CNSA 2.0 TLS profile [10].

A. The CNSA 2.0 TLS 1.3 Profile

The profile, now in revision 05 (July 2026) as an individual Informational Internet-Draft, builds on the TLS working-group drafts that assign code points for ML-KEM [11] and ML-DSA [12]. Table VI summarizes what it requires. Two points matter most here. First, the profile admits exactly one key-establishment method: a CNSA server establishing a compliant connection must not accept any group other than ML-KEM-1024, so there is no hybrid fallback carrying part of the security. Second, TLS 1.2 and earlier cannot be CNSA-compliant at all.

TABLE VI
CNSA 2.0 profile for TLS 1.3
(draft-becker-cnsa2-tls-profile-05)

Element	Requirement
Version	TLS 1.3 only; TLS 1.2 and earlier cannot be compliant
Cipher suite	`TLS_AES_256_GCM_SHA384` {0x13,0x02}, most preferred
Key establishment	ML-KEM-1024 only, NamedGroup `0x0202`; first in `supported_groups` and `key_share`
Signatures	ML-DSA-87, SignatureScheme `0x0906`; pure mode, HashML-DSA not permitted
Key derivation	HKDF with SHA-384
Authentication	Server certificate authentication mandatory; external PSK only in addition to certificates

B. Single Use in TLS 1.3 Is Mostly a "Should"

TLS 1.3 depends on single use at several points, but the specifications mostly *recommend* it rather than require it:

- The TLS ML-KEM draft recommends avoiding reuse of ML-KEM key pairs to preserve forward secrecy, and permits reuse within the bounds of FIPS 203 [11]. It does *require* that ciphertext randomness never be reused.
- RFC 8446 says implementations *should* delete each superseded traffic secret after a KeyUpdate (Section 7.2), that clients *should not* reuse a session ticket for multiple connections (Appendix C.4), and offers single-use tickets as the simplest anti-replay defense for 0-RTT data (Section 8.1) [19].
- ML-DSA's hedged signing, which the TLS ML-DSA draft carries over by reference to FIPS 204, needs fresh randomness per CertificateVerify [2], [12].

Under deployment pressure these recommendations are exactly what gets relaxed. ML-KEM-1024 adds a 1568-byte key share to every ClientHello and a key generation to every handshake, and caching a key share across connections is a tempting optimization. Every such relaxation is invisible on the wire: a connection that reuses a decapsulation key looks identical to one that does not. ROOM removes the choice. The recommended behavior becomes the only behavior the hardware can perform.

C. Where TLS 1.3 Keeps Its Secrets

Table VII lists every secret in a CNSA TLS 1.3 connection, its intended lifetime, and how ROOM holds it.

D. The Derive-All-on-Read Rule

The TLS 1.3 key schedule is a tree. The Handshake Secret yields two handshake traffic secrets and the intermediate "derived" value from which the Master Secret is extracted; the Master Secret yields the application traffic secrets and the exporter and resumption master secrets; each traffic secret yields a key, an IV, a finished key, or its own successor. A ROOM domain can be read only once, so every node must produce *all* of its children in the single read that consumes it. We call this the *derive-all-on-read* rule:

TABLE VII
SECRETS IN A CNSA 2.0 TLS 1.3 CONNECTION AND THEIR ROOM TREATMENT

Secret	Holder	Intended lifetime	Guidance	ROOM treatment
ML-KEM-1024 dk	Client	One handshake	Avoid key-pair reuse [11]; destroy ephemeral keys [5]	Recyclable domain; consumed by Decaps
Encapsulation randomness m	Server	One encapsulation	Must not be reused [11]	DRBG-filled domain; consumed by Encaps
ML-DSA-87 hedging rnd	Server (client in mutual TLS)	One CertificateVerify	Fresh per signature [2]	DRBG-filled domain
Handshake Secret	Both	Until handshake traffic secrets derived	Not needed after derivation	Derive-all-on-read (Sec. VIII-D)
Client/server handshake traffic secrets	Both	Until Finished	Not needed after Finished	One domain each; yield handshake keys and finished keys
Master Secret	Both	Until application secrets derived	Not needed after derivation	Derive-all-on-read
Application traffic secret_N	Both, per direction	One key epoch	Delete after KeyUpdate [19]	One domain per epoch; hardware ratchet
Resumption PSK (ticket)	Both	One resumption	Client should not reuse; single-use tickets [19]	One domain per ticket
Session-ticket encryption key	Server	Rotation period	Compromise exposes all tickets under it	Epoch-KEK domain; rotation collapses it
Long-term ML-DSA-87 key	Server (client in mutual TLS)	Certificate lifetime	Reused by design	Wrapped under an epoch KEK (Sec. VII-E)

1) The engine collapse-reads a node.
2) In that one delivery it computes every child the protocol will ever need from that node.
3) Children that will later be read again as secrets are armed into fresh domains; outputs used continuously, such as record keys and IVs, go into engine-local registers that are destroyed when their epoch ends.
4) Nothing derived from the node is ever recomputed, because the node no longer exists.

The rule turns RFC 8446's "should delete" into a structural property: no parent outlives its children's derivation. Figure 3 shows the result.

E. A ROOM-Backed CNSA Handshake

Algorithm 2 gives the client side of a full CNSA handshake. The server side is symmetric in the key schedule; in place of dk it consumes one domain of encapsulation randomness m and one domain of ML-DSA-87 hedging randomness rnd for its CertificateVerify.

After line 13, the only live secrets for the connection are the current record keys (in engine registers), the next-epoch traffic secrets, and, if used, the exporter and resumption secrets. The ML-KEM decapsulation key, the shared secret, the Handshake Secret, and the Master Secret no longer exist anywhere. A device captured mid-session can expose at most the current epoch's traffic; a device captured afterwards exposes nothing.

F. KeyUpdate as a Hardware Ratchet

RFC 8446 computes each new application traffic secret as $\textit{secret}_{N+1} = \text{HKDF-Expand-Label}(\textit{secret}_N, \text{"traffic upd"}, \text{""}, \text{Hash.length})$ and says that $\textit{secret}_N$ and its keys *should* be deleted once $\textit{secret}_{N+1}$ is computed [19]. Under derive-all-on-read, $\textit{secret}_{N+1}$ is armed when $\textit{secret}_N$ is read, so the deletion is not a step the implementation can skip. A KeyUpdate simply collapse-reads $\textit{secret}_{N+1}$, loads its key and IV, arms $\textit{secret}_{N+2}$, and discards the previous epoch's register contents.

Algorithm 2 ROOM-backed CNSA TLS 1.3 client handshake

1: $(ek, dk) \leftarrow$ ML-KEM-1024.KeyGen() ▷ on-module
2: ARM(D_{dk}, dk); send ek in `key_share`
3: receive ServerHello with ciphertext ct
4: $K \leftarrow$ Decaps(COLLAPSEREAD$(D_{dk}), ct)$
5: $HS \leftarrow$ HKDF-Extract(Derive$(ES,$ "derived"$), K)$; destroy K
6: ARM(D_{hs}) with HS
7: from COLLAPSEREAD(D_{hs}): arm c_hs, s_hs; compute MS, arm it
8: from COLLAPSEREAD(D_{s_hs}): load server handshake key/IV and finished key
9: verify Certificate, ML-DSA-87 CertificateVerify, Finished ▷ public keys only
10: from COLLAPSEREAD(D_{c_hs}): compute client Finished
11: from COLLAPSEREAD(D_{ms}): arm c_ap_0, s_ap_0, exporter, resumption secrets
12: **for** each direction d **do**
13: from COLLAPSEREAD$(D_{d_ap_0})$: load record key/IV; arm d_ap_1
14: **end for**

Two consequences follow for AES-256-GCM. First, the engine can enforce RFC 8446's limits on key usage (Section 5.5) by forcing a KeyUpdate before an epoch's record budget is exhausted, giving every record key a hard ceiling. Second, the per-record nonce is the static IV XOR'd with a 64-bit sequence number, and that counter lives in the engine alongside a key that exists nowhere else. Rolling back the host—restoring a snapshot of the TLS stack in the middle of a connection—cannot restore the key, so it cannot produce a second record under the same (key, nonce) pair; it can only end the connection. ROOM does not make each record single-use; it ensures the state that per-record uniqueness depends on cannot be cloned or rewound.

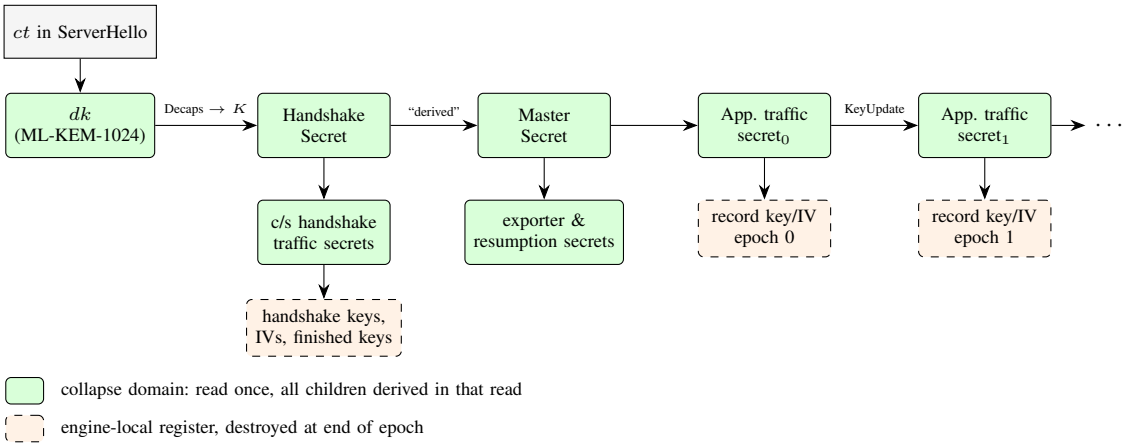


Fig. 3. The TLS 1.3 key schedule as a hardware ratchet (one direction shown for application traffic). Every green node is consumed by the read that derives its children, so no secret on the path from the ML-KEM key share to the current record key survives once its successors exist.

G. Resumption, Tickets, and 0-RTT

Where a deployment permits session resumption, ROOM applies to each of its moving parts. On the client, each NewSessionTicket's PSK is armed into its own domain; using the ticket consumes it, so the RFC 8446 recommendation against reusing a ticket—which also defeats connection linkability—holds by construction. On the server, a ticket's resumption PSK can be held in a domain keyed by the ticket identity. The first resumption consumes it, and a replayed ticket finds a collapsed domain and is rejected. That implements RFC 8446's single-use-ticket anti-replay defense without a separate database of seen tickets, though a multi-server deployment must still route each ticket back to the server that holds its domain. With the `psk_dhe_ke` mode, the resumed handshake also runs a fresh ML-KEM-1024 exchange, consuming a new dk as in Algorithm 2.

Servers that issue stateless tickets encrypt them under a session-ticket encryption key, whose compromise exposes every ticket it protected. Holding that key in an epoch-KEK domain makes rotation an irreversible event: once it collapses, tickets issued under it can no longer be opened by anyone, including the server, bounding forward secrecy for resumption to one rotation period.

0-RTT data is neither forward-secret with respect to the ticket nor replay-protected across connections [19]. Single-use tickets address replay at the ticket layer; they do not make early data forward-secret, and whether 0-RTT is acceptable at all is a policy decision outside ROOM.

H. HelloRetryRequest and Abandoned Handshakes

A handshake can end before its dk is used: the server sends a HelloRetryRequest, the peer disappears, or a timer expires. An armed but unused dk is exactly the retained-key hazard ROOM exists to remove, so the engine performs a *discard read*—a collapse that delivers the value to nothing—when a handshake is abandoned or retried. A retried ClientHello always carries a newly generated key share.

On the server, every ClientHello consumes encapsulation randomness before the client is authenticated, so an attacker can drive domain consumption by opening connections. This is the denial-of-service trade of Section XIV: the server fails closed. Recyclable domains and ordinary connection rate limiting keep it from becoming exhaustion.

I. Capacity Sizing

With SHA-384, each key-schedule secret is 48 bytes. A full handshake arms ten of them per side (Handshake Secret, two handshake traffic secrets, Master Secret, two application traffic secrets and their two successors, exporter and resumption master secrets), or 480 bytes. The client adds its dk: 3168 bytes in expanded form, or 64 bytes if the module stores the decapsulation-key seed form described in the TLS ML-KEM draft [11] and re-expands it on module at decapsulation. The server adds 32 bytes of encapsulation randomness and 32 bytes of hedging randomness. Each side therefore consumes about 544 bytes of single-use storage per handshake in seed form, plus 48 bytes per direction for each KeyUpdate.

These figures settle the domain class. A tactical endpoint making 100 handshakes a day for ten years consumes about 200 MB; a server completing 10,000 handshakes per second consumes about 5.4 MB/s, or roughly 470 GB per day. Neither is feasible with terminal domains, so TLS uses recyclable domains (Section VI-D), and the design point becomes the *working set*: handshakes in flight times 544 bytes, plus live connections times the secrets each keeps between epochs—the two next-epoch traffic secrets and, if used, the exporter and resumption secrets, at most 192 bytes. A gateway holding 10,000 concurrent connections with 1,000 handshakes in flight needs under 2.5 MB of ROOM, a size suited to an HSM or line-card co-processor. A server that issues no tickets and offers no exporters can discard those secrets at the end of the handshake and halve the per-connection figure.

J. What an Assessor Can Test

Each TLS property above reduces to a black-box test against the module:

- Submit a second ciphertext to a decapsulation key already used, or abandoned by a HelloRetryRequest; the module returns the sentinel.
- Snapshot the host TLS stack mid-connection, restore it, and attempt to continue the record stream; the module has no key for it.
- After a KeyUpdate, request any operation under the previous epoch; it fails.
- Present the same resumption ticket twice; the second is rejected.

TLS 1.3, with the "shoulds" made physical

The CNSA profile fixes which algorithms TLS 1.3 uses. ROOM fixes how often each secret can be used: once. Forward secrecy stops depending on every implementation choosing not to cache a key share.

IX. Eliminating Side-Channel Averaging

The most damaging physical attacks on cryptographic hardware are not single-shot. Differential power analysis (DPA), correlation power analysis, and their electromagnetic counterparts work by *averaging*: the attacker records many traces while the device processes the same secret with varying inputs, then combines them statistically until the secret-dependent leakage rises above the noise [24]. Each additional trace improves the attacker's signal-to-noise ratio. Countermeasures such as masking and hiding raise the number of traces required, but they do not change the underlying arithmetic: given enough uses of a key, the leakage accumulates.

A. One Secret, One Trace

ROOM attacks the problem at its root. A secret held in a collapse domain is delivered to exactly one operation (Property P1). An adversary observing the device therefore obtains *at most one trace per secret*. There is nothing to average. The statistical attacks that defeat most deployed hardware need hundreds to millions of executions under a fixed key; under ROOM, the number available is one.

B. Impact on the CNSA 2.0 Suite

1) ML-KEM-1024: A substantial body of research shows that lattice KEMs can be broken through *chosen-ciphertext side-channel attacks*: the adversary submits crafted ciphertexts to a device holding a fixed decapsulation key and uses the side-channel signature of each decapsulation as a plaintext-checking oracle, recovering the key over thousands of queries [25]. These attacks require the same dk to be used repeatedly. With ROOM, each dk supports a single decapsulation, so the oracle answers at most one query per key and the attack never starts.

2) LMS and XMSS: Conventional stateful signers do not store every one-time key; they re-derive each LM-OTS key from a master seed with a pseudorandom function at signing time. That seed is therefore used in *every* signature over the life of the key, making it the ideal averaging target. Under ROOM, all OTS keys are derived once at provisioning and the seed is then consumed (Section VII). In operation, no long-lived secret is ever processed twice.

3) AES-256-GCM: When per-message keys are delivered from collapse domains, each AES key protects one message. The attacker loses the ability to accumulate traces across messages under a single key, sharply reducing the leakage available.

4) ML-DSA-87: Because the ML-DSA-87 signing key is long-lived, it remains exposed to averaging across signatures. ROOM still removes the per-signature hedging value rnd as a repeatable target and, through epoch KEKs, limits how long the plaintext key exists in usable form, but the key itself should continue to be protected with masking.

C. What Remains

Single use removes averaging; it does not remove leakage. Single-trace attacks, such as simple power analysis and template attacks profiled on a reference device, can in principle extract information from one execution, and research has demonstrated single-trace attacks against lattice-based schemes [26]. "Horizontal" attacks can also combine leakage from many sub-operations within a single execution. ROOM is therefore best paired with standard countermeasures. The combination is powerful: masking that would fall to a large-trace DPA campaign is far harder to defeat when the attacker is limited to a single trace per secret.

D. Testing and Certification

FIPS 140-3 addresses non-invasive attacks, with test methods aligned to ISO/IEC 17825, which uses statistical leakage assessment over many traces collected under fixed and varying conditions [15], [27]. For secrets held in ROOM, the fixed-key trace sets those methods rely on cannot be collected from the production module at all: the second acquisition under a given key is physically impossible. This gives vendors a direct, demonstrable argument for non-invasive attack resistance on every single-use secret.

Starving the attacker

Side-channel attacks win by repetition. ROOM allows none. Every key an attacker can observe is a key the attacker will see exactly once.

X. Compliance Mapping

Table VIII maps each requirement from Section IV and each threat from Section V to the ROOM mechanism that addresses it.

TABLE VIII
REQUIREMENT-TO-MECHANISM COMPLIANCE MAPPING

Requirement	Source	Conventional approach	ROOM mechanism
OTS key used exactly once	SP 800-208; RFC 8554/8391	Firmware index in NVRAM; procedural controls	Collapse domain per OTS key (P1, P2)
State never rolls back	SP 800-208	Write-before-sign; backup prohibitions	Hardware monotonic index (P3)
No cloning / duplication of signer state	SP 800-208	Policy; HSM export controls	Read-to-copy consumes the original (P4)
Keys generated and used in hardware, non-exportable	SP 800-208	HSM boundary	On-module generation; isolated datapath
Destroy ephemeral KEM keys after use	SP 800-227; FIPS 203	Software zeroization	dk in collapse domain; consumed by Decaps
Destroy intermediate secrets	FIPS 203, FIPS 204	Software zeroization	Intermediates staged in collapse domains
Fresh hedging randomness	FIPS 204	DRBG call per signature	rnd from collapse domain; replay-proof under faults
IV unique per key	SP 800-38D	Counter in RAM/NVRAM	One-time IV pool in collapse domains
Avoid ML-KEM key-pair reuse; never reuse ciphertext randomness	draft-ietf-tls-mlkem; CNSA TLS profile	Fresh KeyGen and DRBG call in software	dk and m in collapse domains (Sec. VIII)
Delete traffic secret after KeyUpdate	RFC 8446 §7.2	Software "should"	Derive-all-on-read hardware ratchet
Destroy handshake and master secrets once derived	RFC 8446 §7.1; FIPS 140-3	Software zeroization, if performed	Each secret consumed by deriving its children
Do not reuse tickets; single-use tickets for 0-RTT anti-replay	RFC 8446 App. C.4, §8.1	Client cache logic; server replay database	One domain per ticket PSK
Hedged ML-DSA-87 CertificateVerify	draft-ietf-tls-mldsa; FIPS 204	DRBG call per signature	rnd from collapse domain
Zeroize SSPs, unrecoverable	FIPS 140-3 / ISO 19790	Overwrite; verify	Collapse plus electrical grounding (P5)
Resist cold-boot / remanence	Module physical security	Tamper response; memory encryption	Electrical grounding [29]
Resist DMA / privileged reads	Module design	IOMMU; enclave	Secrets never on system bus
Mitigate non-invasive attacks (DPA/EM)	FIPS 140-3; ISO/IEC 17825	Masking, hiding, noise	One trace per secret; no averaging (P1)

A. Supporting FIPS 140-3 Validation

ROOM is not a cryptographic algorithm and does not require CAVP algorithm validation. It is a storage and SSP-management mechanism within the module boundary, and its value to a CMVP submission is evidentiary:

- **SSP lifecycle.** The security policy can state, for each ephemeral SSP, that zeroization is *intrinsic to use*, with the collapse transition as the zeroization method.
- **Stateful signature assurance.** For LMS/XMSS modules, the state-management narrative reduces from a multi-page argument about NVRAM durability, power-loss ordering, and operational prohibitions to a hardware property that can be tested directly.
- **Testability.** Property (2) is a black-box testable claim: arm a cell, read it, attempt every available read path again, and observe the sentinel. Testing laboratories can verify it without trusting vendor firmware.

B. What Assessors Get

From argument to artifact

Today, demonstrating LMS state integrity to an assessor means showing code, procedures, and logs—and hoping no one ever restores the wrong snapshot. With ROOM, the demonstration is a physical test: *try to sign twice with leaf q.* The hardware cannot do it.

XI. REFERENCE IMPLEMENTATION

A. RTL and FPGA Prototype

A synthesizable reference implementation is available in SystemVerilog in the `AtomicMemory` repository [30], comprising a collapse-cell module (`collapse_cell.sv`) and a banked collapse-domain array (`collapse_bank.sv`). The design has been prototyped on an Intel Cyclone V FPGA.

On FPGA fabric, the collapse transition is enforced in logic: the read path and the collapse path are fused so that no state machine encoding exists that returns data without collapsing. The ASIC realization replaces logic enforcement with circuit-

level electrical grounding [29], providing physical rather than logical irreversibility.

B. Overhead Analysis

A collapse-read is a single memory access fused with a state transition, so ROOM adds negligible latency relative to the cryptographic computation it guards:

- **LMS/XMSS signing** is dominated by hash-chain evaluation: an LM-OTS signature requires many SHA-256 compression-function calls, against which one domain read is insignificant. Signing with ROOM is also simpler than conventional signing, because the separate "persist the new index to NVRAM before releasing the signature" step disappears.
- **ML-KEM-1024 decapsulation** is dominated by NTT-domain polynomial arithmetic and the re-encryption check; reading the 3168-byte dk from a collapse domain is a small fraction of this cost.
- **Area.** Each collapse cell adds a small amount of state logic to an ordinary storage bit. Capacity therefore scales linearly with the number of one-time secrets a device must hold, which the deployer sizes to the product's lifetime signature or session budget.

Detailed resource and timing figures for the Cyclone V prototype are available to evaluation partners under NDA.

C. Verification

The collapse cell's safety property is amenable to formal verification:

$$\mathbf{G}\big(\textsc{Collapsed} \rightarrow \mathbf{G}\,\neg\textsc{Armed}\big) \tag{3}$$

and

$$\mathbf{G}\big(\texttt{data_valid} \rightarrow \mathbf{X}\,\textsc{Collapsed}\big), \tag{4}$$

stating that a collapsed cell never re-arms and that any valid data output is immediately followed by collapse. Both are expressed in a form directly suited to bounded and unbounded model checking of the RTL, giving evaluators a machine-checkable statement of the read-once guarantee.

D. Public Disclosure and Engagement

ROOM has been documented in TechRxiv preprints [28] and presented at a SOFWERX Tech Tuesday event in July 2026.

XII. Deployment Scenarios

A. Firmware-Signing Roots of Trust (CNSA deadline: 2030)

Need. Every NSS vendor must sign firmware and software with LMS, XMSS, or ML-DSA-87, exclusively by 2030. LMS SHA-256/192 is NSA's recommended parameter set.

ROOM solution. A ROOM-based LMS signing appliance or PCIe card holds its OTS keys in collapse domains. Build servers can be virtualized, snapshotted, and restored freely: the signing state lives in silicon that IT operations cannot rewind.

B. VPN and Network Equipment (CNSA deadline: 2030)

Need. IKEv2 and TLS endpoints must use ML-KEM-1024 and must deliver forward secrecy against HNDL.

ROOM solution. A ROOM co-processor on the line card holds each ephemeral dk for exactly one handshake, and each TLS key-schedule secret for exactly one derivation (Section VIII). Captured or decommissioned equipment yields no historical session keys.

C. Tactical and Edge Devices

Need. Radios, unmanned systems, and soldier-worn devices operate in contested environments where capture is a planning assumption.

ROOM solution. Pre-provisioned pools of single-use KEM keys, IVs, and message keys give each device a fixed, self-consuming cryptographic budget. A captured device cannot be coerced into revealing past traffic, and cannot be cloned to impersonate its owner beyond its remaining budget.

D. Cloud HSM and Key-Management Services (CNSA deadline: 2033)

Need. Cloud providers must offer CNSA 2.0 key services, including stateful signatures, in environments built on replication and snapshots—the exact operations SP 800-208 warns against.

ROOM solution. HSS keys partitioned across ROOM-equipped HSMs, each with a disjoint leaf range, deliver high availability with zero risk of cross-replica leaf reuse.

E. Supply-Chain and Attestation Tokens

Single-use attestation secrets in collapse domains give hardware tokens that prove "first use" and cannot be replayed, supporting anti-counterfeiting and secure provisioning workflows.

XIII. Comparison with Existing Approaches

Table IX compares ROOM with current mechanisms for enforcing single use.

Monotonic counters, such as those in TPMs, protect a *counter* but not the *secret* the counter indexes; the secret still lives in reusable memory, and the binding between counter and use remains a software obligation. One-time-programmable fuses are physically irreversible but store no secret and offer too few transitions for KB-scale key material. Trusted execution environments are vulnerable to sealed-state replay.

Read-then-clear memory is the closest conventional analogue, and the difference is instructive. It performs *two* operations: return the value, then overwrite it. The secret survives in the gap between them, so a power loss, interrupt, or glitch that suppresses the clearing write defeats it. The clear is issued by a controller or firmware that can be reconfigured, and debug, scan, or DMA paths can read the word without triggering it. The overwrite leaves the cell's analog imprint in place, so remanence is not addressed. A cleared cell also simply returns to a writable empty state, so nothing in the memory records that the secret was ever consumed. In ROOM, delivery and destruction are one physical event, collapse is irreversible for

TABLE IX
COMPARISON OF SINGLE-USE ENFORCEMENT APPROACHES

Approach	Rollback-proof	Clone-proof	Crash-safe	Resists remanence	Resists privileged SW	Assessor-testable
Software counter + NVRAM	No	No	Partial[a]	No	No	Partial
HSM firmware state management	Partial	Partial	Partial[a]	Partial	Yes	Partial
TPM monotonic counter	Partial[b]	Partial	Partial	N/A	Yes	Partial
Read-then-clear memory[e]	Partial	Partial	Partial[f]	No	Partial	Partial
One-time-programmable fuses[c]	Yes	Yes	Yes	Yes	Yes	Yes
Trusted execution env. (TEE)	No[d]	No	No	Partial	Partial	Partial
ROOM	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

[a] Depends on correct write-before-sign ordering and NVRAM durability. [b] Protects the counter, not the secret it indexes. [c] Store no secret, offer very few transitions, and cannot hold KB-scale keys. [d] Vulnerable to sealed-state replay. [e] Includes clear-on-read registers and controllers that zeroize a word after returning it. [f] A power loss, interrupt, or fault between the read and the clear leaves the secret intact.

the value, and in terminal domains the consumed state is itself the record (Section VI-D). ROOM uniquely binds the secret and its consumption into a single irreversible physical event.

XIV. Limitations and Scope

A credible security product is clear about what it does not do.

- **ROOM is not an algorithm.** It does not make any algorithm CNSA 2.0-approved and does not substitute for CAVP/CMVP validation of the algorithms and module that use it.
- **Static keys.** Long-term ML-DSA-87 and static ML-KEM keys are reused by design. ROOM protects them indirectly (epoch KEKs, hedging randomness) but does not make them single-use.
- **Finite budget.** Terminal domains are consumed for good, so capacity planning is required. High-volume ephemeral use, including TLS, depends on recyclable domains re-armed by on-module generation (Sections VI-D and VIII-I).
- **TLS record keys.** ROOM bounds each TLS record key to one connection and one key epoch, but within an epoch AES-256-GCM still encrypts many records under one key, and per-record nonce uniqueness remains the record layer's job (Section VIII-F).
- **Denial of service.** An attacker able to issue reads can exhaust domains. This is a deliberate trade: ROOM fails *closed* (no signature, no decapsulation) rather than *open* (reused key). Rate limiting and authentication on the engine interface mitigate exhaustion.
- **Armed-state exposure.** Secrets that are armed but not yet consumed must still be protected against invasive physical attack. Electrical grounding and standard module tamper protections apply.
- **FPGA vs. ASIC.** The FPGA prototype enforces collapse in logic and inherits FPGA trust assumptions (bitstream integrity, readback disabled). Physical irreversibility is a property of the ASIC realization.

XV. Roadmap and Engagement

QSymbolic LLC is seeking partners in three tracks:

1) **Integration partners**: HSM, network-equipment, and root-of-trust vendors seeking to accelerate CNSA 2.0 LMS/XMSS and ML-KEM deployment ahead of the 2030 deadlines.
2) **Government and defense evaluation**: programs requiring capture-resistant forward secrecy and tamper-evident single-use cryptographic budgets at the tactical edge.
3) **Silicon partners**: foundry and IP-licensing relationships to take the electrically grounded collapse cell to ASIC.

QSymbolic LLC is registered with UEI M9DAN28Y9YU6 and CAGE 23FZ3. Contact: frank@qsymbolic.com.

XVI. Conclusion

The post-quantum algorithm question has been answered. The post-quantum *implementation* question has not. CNSA 2.0's first deadline falls on the algorithms with the strictest single-use demands, and across the suite, the failures that matter most—one-time key reuse, ephemeral key retention, nonce repetition, state rollback—are failures of memory, not mathematics.

Read Only-Once Memory answers that question in hardware. By making every read a consumption and every consumption irreversible, ROOM turns the "shall not reuse" clauses of SP 800-208, SP 800-227, FIPS 203, FIPS 204, and SP 800-38D into physical invariants. It changes no algorithm, no wire format, and no protocol. It simply ensures that the approved algorithms run under the conditions their proofs assume—on every device, after every crash, through every snapshot, and in the hands of every adversary who captures the hardware.

For organizations facing 2030, the choice is between arguing single use and *guaranteeing* it.

Acknowledgment

The author thanks the SOFWERX community for feedback on early ROOM presentations.

References

[1] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," FIPS PUB 203, Aug. 2024.
[2] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard," FIPS PUB 204, Aug. 2024.

[3] National Institute of Standards and Technology, "Stateless Hash-Based Digital Signature Standard," FIPS PUB 205, Aug. 2024.
[4] D. Cooper, D. Apon, Q. Dang, M. Davidson, M. Dworkin, and C. Miller, "Recommendation for Stateful Hash-Based Signature Schemes," NIST SP 800-208, Oct. 2020.
[5] National Institute of Standards and Technology, "Recommendations for Key-Encapsulation Mechanisms," NIST SP 800-227, Sep. 2025.
[6] National Institute of Standards and Technology, "Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process," NIST IR 8545, Mar. 2025.
[7] National Institute of Standards and Technology, "Transition to Post-Quantum Cryptography Standards," NIST IR 8547 (Initial Public Draft), Nov. 2024.
[8] National Institute of Standards and Technology, "Post-Quantum Cryptography Standardization," CSRC project page. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography
[9] National Security Agency, "The Commercial National Security Algorithm Suite 2.0 and Quantum Computing FAQ," Cybersecurity Advisory, Sep. 2022, updated.
[10] A. Becker and M. J. Jenkins, "Commercial National Security Algorithm (CNSA) Suite 2.0 Profile for TLS 1.3," IETF Internet-Draft draft-becker-cnsa2-tls-profile-05, Jul. 2026, work in progress.
[11] D. Connolly, "ML-KEM Post-Quantum Key Agreement for TLS 1.3," IETF Internet-Draft draft-ietf-tls-mlkem-07, Feb. 2026, work in progress.
[12] T. Hollebeek, S. Schmieg, and B. E. Westerbaan, "Use of ML-DSA in TLS 1.3," IETF Internet-Draft draft-ietf-tls-mldsa-05, Jul. 2026, work in progress.
[13] M. J. Jenkins and A. Becker, "Commercial National Security Algorithm (CNSA) Suite 2.0 Profile for S/MIME," IETF Internet-Draft draft-becker-cnsa2-smime-profile, 2026.
[14] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST SP 800-38D, Nov. 2007.
[15] National Institute of Standards and Technology, "Security Requirements for Cryptographic Modules," FIPS PUB 140-3, Mar. 2019.
[16] Office of Management and Budget, "Migrating to Post-Quantum Cryptography," Memorandum M-23-02, Nov. 2022.
[17] D. McGrew, M. Curcio, and S. Fluhrer, "Leighton-Micali Hash-Based Signatures," RFC 8554, Apr. 2019.
[18] A. Hülsing, D. Butin, S. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme," RFC 8391, May 2018.
[19] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018.
[20] J. A. Halderman *et al.*, "Lest We Remember: Cold-Boot Attacks on Encryption Keys," in *Proc. USENIX Security Symp.*, 2008, pp. 45–60.
[21] Z. Yang, B. Johannesmeyer, A. T. Olesen, S. Lerner, and K. Levchenko, "Dead Store Elimination (Still) Considered Harmful," in *Proc. USENIX Security Symp.*, 2017.
[22] S. Skorobogatov, "Low Temperature Data Remanence in Static RAM," Univ. of Cambridge Computer Laboratory, Tech. Rep. UCAM-CL-TR-536, Jun. 2002.
[23] M. Wei, L. M. Grupp, F. E. Spada, and S. Swanson, "Reliably Erasing Data from Flash-Based Solid State Drives," in *Proc. USENIX Conf. File and Storage Technologies (FAST)*, 2011.
[24] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology—CRYPTO '99*, LNCS 1666, Springer, 1999, pp. 388–397.
[25] P. Ravi, S. Sinha Roy, A. Chattopadhyay, and S. Bhasin, "Generic Side-Channel Attacks on CCA-Secure Lattice-Based PKE and KEMs," *IACR Trans. Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, pp. 307–335, 2020.
[26] R. Primas, P. Pessl, and S. Mangard, "Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption," in *Cryptographic Hardware and Embedded Systems—CHES 2017*, LNCS 10529, Springer, 2017, pp. 513–533.
[27] ISO/IEC 17825, "Information technology—Security techniques—Testing methods for the mitigation of non-invasive attack classes against cryptographic modules," 2024.
[28] F. X. Cunnane III, Read Only-Once Memory (ROOM), TechRxiv preprint, v3, Jan. 2026, doi: 10.36227/techrxiv.176463742.23048082.
[29] F. X. Cunnane III, "A CMOS Electrical Non-Persistence Primitive for Single-Use Secrets," TechRxiv preprint, Dec. 2025, doi: 10.36227/techrxiv.176583510.02228836/v1.
[30] F. X. Cunnane III, "AtomicMemory: ROOM reference RTL," GitHub repository. [Online]. Available: https://github.com/fcunnane/AtomicMemory