

GM-3X512-PQS

Triple 512-Bit Parallel Keystream Pre-Generation with Post-Quantum Key Establishment, Forward Secrecy and Optional Post-Compromise Recovery

Gianclaudio Moresi, Dipl. El. Ing. ETH

Swiss Cyber AI Organization (SCAI), Baar, Switzerland

Contact: gm@swisscyberai.org

Author's Contribution and AI Assistance.

The core concept of GM-3X512-PQS - a native 512-bit symmetric architecture, parallel generators combined at output level, multiple post-quantum KEMs bound to a session transcript, independently sampled traffic roots, and no elliptic-curve cryptography in key establishment was conceived by Gianclaudio Moresi. The specification was developed iteratively with AI assistance, which contributed to the text, test vectors, reference implementations, security analysis & simulation, and a number of design decisions and corrections. All architectural decisions, security claims and final content are the author's responsibility.

Abstract

GM-3X512-PQS proposes an authenticated transport using multiple post-quantum key-encapsulation mechanisms and signature schemes. GM512-CORE combines lattice and code KEM assumptions with two signature schemes; GM512-FULL adds a second code-based KEM construction and a third signature scheme. The design uses a ciphertext-bound concatenation combiner, domain-separated key derivation, ephemeral or one-time key establishment, identity-scoped replay state and explicit abort semantics. Section 12 gives component arguments in the classical random-oracle or standard model, as applicable; these arguments do not establish a composed authenticated-channel or quantum-security proof.

The specification defines seven ProfileIDs, grouped into a standards-based family (S-DUAL512, S-KECCAK) and an experimental family (X-GM512, X-TRIPLE, X-TRIPLE-3R, X-TRIPLE-3RN, X-TRIPLE-3RH). S-DUAL512 is mandatory to implement and S-KECCAK is the standards-based alternative; both use only standardized symmetric primitives and both are keyed with 512 bits. The experimental family additionally defines GM-512-BC, a 512-bit block cipher with a provable wide-trail bound of at least 81 active S-boxes per four rounds, giving a maximum four-round single-trail differential characteristic probability of 2^{-486} and a maximum four-round single-trail linear characteristic correlation of 2^{-243} . The experimental profiles are optional and are explicitly not recommended for production use pending third-party cryptanalysis.

Four experimental XOR profiles are also defined. X-TRIPLE combines three independently keyed generators - HMAC-SHA-512, HMAC-SHA3-512 and GM-512-BC, all in counter mode - by XOR. Under the HMAC-SHA3-512 key-separation assumption, confidentiality holds so long as any one of the three generators remains a secure PRF (Theorem 4), and two of the three are standardized. X-TRIPLE-3R additionally samples the three generator keys as independent 512-bit roots, each wrapped under a key-encryption key bound to both the combiner output and one KEM leg's own secret, so the three generator roots are separately sampled and are derived neither from one another nor from K_M , while wrap hiding is intended to inherit the surviving-KEM property under the additional ROM argument in Section 12.4. Aggregate root material is 1,536 bits. GM-3X512-PQS targets 256-bit classical and NIST Category 5 post-quantum security under A1–A5; the composed QPT/QROM proof remains open.

Two further root profiles vary the generators. X-TRIPLE-3RN uses three instances of GM-512-BC-N, the Keccak-free variant of the block cipher, removing both HMAC legs from the keystream path at the cost of any heterogeneous primitive hedge. X-TRIPLE-3RH uses three structurally different 512-bit block ciphers - Threefish-512, Kalyna-512/512 and GM-512-BC-N - each under its own independently sampled 512-bit key, and is the only profile combining three heterogeneous 512-bit block-cipher generators with independently sampled roots.

An optional experimental extension, the Triple Post-Quantum Root Ratchet (TPQRR), extends the three independently rooted profiles to ordered multi-record sessions. TPQRR performs one-way per-record evolution of all three 512-bit root chains and periodically injects fresh post-quantum KEM entropy without tearing down the transport connection. The intended recovery property is that future records regain confidentiality after a successful refresh, provided the attacker no longer controls the endpoint, at least one fresh ratchet KEM contribution remains secret, the long-term authentication keys remain uncompromised, and both endpoints restore unpredictable randomness before recovery-sensitive key generation, encapsulation and root sampling. A composed proof of this property remains open. This is post-compromise recovery, not an increase above the 256-bit security target.

1. Design Rationale

1.1 The problem GM-3X512-PQS addresses

A cryptographically relevant quantum computer would break RSA and elliptic-curve cryptography via Shor's algorithm. The standard response is to migrate to post-quantum primitives. That response carries a second-order risk: the post-quantum primitives are young. ML-KEM was standardized in 2024 after roughly eight years of public analysis [4]. SIDH/SIKE, a fourth-round NIST candidate, was broken in 2022 by a classical attack in under an hour on a laptop [12]. Rainbow, a third-round signature finalist, was broken in 2022 [13]. Neither failure was anticipated.

The lesson is not that lattices are unsafe. It is that confidence in any single young assumption is not warranted for data with a long confidentiality horizon.

GM-3X512-PQS investigates redundant composition across post-quantum constructions. Its surviving-component arguments require the stated combiner, derivation, authentication and implementation assumptions. Failure of a shared dependency, endpoint protection or state-management rule can affect multiple components at once.

1.2 Design principles

P1 - Redundancy across diversified hardness assumptions. GM512-CORE combines lattice-based and code-based KEM assumptions; GM512-FULL adds Classic McEliece as a second, structurally distinct code-based construction. An attack specific to structured lattices need not affect code-based or hash-based components; shared implementation or mathematical weaknesses may still create correlated failures. HQC and Classic McEliece provide substantial construction diversity but are not claimed to be fully independent hardness families: HQC rests on variants of syndrome decoding, and information-set decoding is a central generic attack avenue against both, so a generic advance in code decoding could affect them together. Theorem 1 does not require independence - the combiner needs only one KEM to survive - so the claim here is diversity, not independence. Mode A additionally uses a second ML-KEM instance as the one-time prekey; that leg contributes forward secrecy only and is not counted as an additional independent hardness family.

P2 - Standardized or NIST-selected primitives on the mandatory path. Every primitive required for interoperability is either a published NIST standard or recommendation - FIPS or SP, the latter covering KMAC256 in SP 800-185 [3] - or a NIST-selected algorithm on an active standardization track. Novel constructions are optional and clearly labeled.

P3 - Claims must identify their evidence. Security targets, component bounds, reported measurements and open proof obligations are distinguished explicitly. A reproduced test vector checks a specified calculation; it does not prove the security of the primitive or the protocol.

P4 - Fail closed. Every verification has a defined failure branch that aborts and releases nothing.

P5 - Bind everything. Key-establishment and negotiation values - Version, SuiteID, ModelID, ProfileID, identities, public keys, ciphertexts - are bound into h_T , which keys the session. Payload-specific values N, AAD and C are bound into M_AUTH and M_SIG ; T is computed over M_AUTH and is additionally bound into M_SIG . Nothing that could be substituted, reordered, downgraded or replayed is left outside both.

1.3 Explicit non-claims

The following are not claimed, and any summary of this work that asserts them is incorrect.

- **GM-3X512-PQS does not guarantee “100% quantum secure.”** GM-512, like practical public-key and authenticated-transport constructions generally, does not provide unconditional security. Post-quantum security is a conjecture supported by the absence of known quantum algorithms against a stated problem. MLWE, syndrome decoding, and Goppa code distinguishing have no unconditional hardness lower bounds.
- **GM-3X512-PQS does not provide 3x512-bit security.** It targets NIST Category 5 post-quantum security under A1–A5; the composed QPT/QROM proof remains open.
- **GM-512-BC (the experimental X- profiles) is not recommended for production.** It has proved single-trail bounds for differential and linear characteristics, and no third-party analysis against the others. §9.7 states this in full.
- GM-3X512-PQS has no machine-checked or composed protocol proof. Section 12 contains component lemmas and informal proof sketches. The corrected security targets in Section 11, including TPQRR recovery, remain proof obligations.

2. Notation

Symbol	Meaning
$\parallel$	Concatenation of byte strings
$\oplus$	Bitwise XOR
$x \leftarrow \$ S$	Uniform sampling from S using a CSPRNG
$I2OSP(n, k)$	Integer n as a big-endian unsigned k-byte string
$ X $	Length of X in bytes unless bits are stated
$\perp$	The abort symbol. No plaintext, session key, export key or other successful protocol output is released. Security-state updates committed before the abort - one-time-prekey consumption in particular - are NOT rolled back (Requirements R2 and F2)

All integers are big-endian. All lengths are in bytes unless a bit count is explicitly given.

2.1 Canonical encoding

For any byte string X:

$$FIELD(X) = I2OSP(|X|, 4) \parallel X$$

For a tuple $X = (X_1, \dots, X_n)$:

$ENC(X_1, \dots, X_n) = FIELD(X_1) \parallel FIELD(X_2) \parallel \dots \parallel FIELD(X_n)$

Requirement E1. Every input to a hash, MAC, or signature that is a concatenation of two or more variable-length values **MUST** use ENC. Raw concatenation is permitted only where every component has a fixed length fixed by this specification.

Requirement E2. Any field with $|X| \geq 2^{32}$ **MUST** be rejected at parse time.

This prevents the concatenation ambiguities that have broken deployed protocols. The four-byte length prefix makes ENC injective.

Test vector: $ENC("AB", "CDE") = 00000002\ 41\ 42\ 00000003\ 43\ 44\ 45$

Requirement E3. Fields marked ? keep their specified position and arity. When absent they **MUST** be encoded as $FIELD("")$, never omitted. When present they **MUST** carry the exact value defined for that configuration. A receiver **MUST** reject any presence pattern inconsistent with the negotiated SuiteID, ModelID and ProfileID. Fixed arity means every encoded object of a given configuration has the same field count and order, which removes an entire class of parsing ambiguity.

3. Security Target

3.1 Target level

GM-3X512-PQS targets **NIST Category 5**: security at least that of AES-256 key search against both classical and quantum adversaries.

Two distinct quantities. The table below states key-recovery and forgery targets. Record confidentiality is a separate quantity: it is profile- and data-dependent, and every profile now uses 512-bit generators, so no practically relevant 128-bit block-size term arises; the 512-bit PRP profiles retain the standard PRP-to-PRF switching term, which must be included in concrete bounds. A 256-bit target is a statement about the work needed to recover keys, not a concrete distinguishing bound at every traffic volume.

Concretely:

Property	Target	Bounded by
Unused application-export key indistinguishability	256-bit classical	At least one surviving secure KEM, capped by the surviving KEM shared secret
Ciphertext integrity	256-bit classical	KMAC256
Signature unforgeability	256-bit classical	Strongest component signature in the selected suite
Generic quantum key search	Category 5	Equivalent to AES-256

3.2 Limitations

GM-3X512-PQS uses SHA3-512 [2] for the master combiner and HMAC-SHA3-512 [29, 2] for keyed derivation, producing 512-bit outputs throughout the key-generation path. KMAC256 [3] accepts a variable output length and can emit 512 bits, so the choice is not forced by output width. HMAC-SHA3-512 was chosen so that the whole key-generation path is built on the fixed 512-bit SHA3-512 hash rather than on a cSHAKE256-based construction whose

security strength is 256 bits regardless of output length. No 512-bit PRF or protocol-security claim follows from this choice: NIST permits HMAC-SHA3-512 but has not established a detailed 512-bit PRF or unforgeability strength for SHA-3-based HMAC. The target remains 256-bit classical and NIST Category 5, because a surviving KEM contributes a 256-bit shared secret and authentication is KMAC256. A derivation chain wider than the material entering it does not raise the level of the weakest link, and requesting 512 or 1,536 output bits makes the strings wider, not stronger.

Consequently, although direct guessing of a 512-bit tag contributes only a 2^{-512} term, the overall authentication security is capped at 256 bits by the standardized security strength of KMAC256; §12.3 separates the two contributions. Grover's algorithm achieves a quadratic speedup only in the sequential-query model [24]. Distributing the search over P machines yields $\sqrt{N/P}$, not N/P , so parallelism buys only a square-root improvement. NIST's MAXDEPTH framework [25] explicitly constrains the circuit depth available to quantum attacks. This specification makes no prediction about the capabilities of future fault-tolerant quantum hardware. The nominal width alone does not establish additional protocol security.

Conclusion. GM-3X512-PQS keys are 512 bits wide because the state and block are 512 bits wide, not because 512 bits of security are claimed. The reasons for the 512-bit block are given in §3.3 and are entirely separate from key strength. GM-3X512-PQS targets 256-bit classical and NIST Category 5 post-quantum security under A1–A5; the composed QPT/QROM proof remains open (§11.1, §12.5).

3.3 Why the block is 512 bits

The block size is an engineering decision and carries no security-level claim. It addresses data complexity, not key search, and the following are the reasons.

Birthday ceiling becomes irrelevant at practical data volumes. A 128-bit block reaches its generic birthday bound at roughly 2^{64} blocks under one key, so a construction over it must track how much data is processed per key. AES-GCM carries a separate and stricter limit of about 64 GiB per plaintext invocation, which follows from its CTR counter construction rather than from the generic birthday bound. Both classes of limit become irrelevant at practical data volumes with a 512-bit block, where the corresponding bound is 2^{256} blocks; the bound still exists, but no deployment approaches it. A 128-bit PRP requires explicit per-key data accounting and appropriate rekeying limits; with a 512-bit block the corresponding switching bound is negligible at any practical traffic volume.

A smaller reduction term. The PRP-to-PRF switching term in Theorem 3 is $q(q-1)/2^{129}$ for the 128-bit AES block and $q(q-1)/2^{513}$ for the 512-bit block. The first is a real term that must be tracked and bounded per payload; the second never binds at any reachable q .

Uniform composition. Every generator in the XOR profiles of §7.7 emits 512 bits per composed block index, so no leg needs the multi-block indexing a 128-bit generator would require. AES-256 is no longer a payload-encryption profile. The 512-bit generators remove the expansion and per-payload key derivation that a 128-bit block made necessary.

A natural hash and MAC construction. A 512-bit block cipher yields a 512-bit hash directly under Miyaguchi-Preneel, which is what the X-GM512-NATIVE draft relies on. Whirlpool uses this construction with a 512-bit block cipher in Miyaguchi-Preneel mode [27]. A 128-bit block cannot serve that role without a doubling construction.

What this does not claim. None of these is a security-level argument. They are the reasons a wide block is worth the additional state, area and latency, and they are independent of the key-strength discussion in §3.2.

This document gives exact suite-component source pins, canonical encodings, algorithmic constants and representative primitive and synthetic KDF vectors. Section 13 separates those vectors from the complete handshake and TPQRR conformance suite required by Section 15. Source pins make the draft reproducible; they do not establish implementation interoperability.

4. Cryptographic Suites and Identity

4.1 Suites

Two suites are active and one identifier is withdrawn. All share the same protocol; they differ in component selection.

Suite	KEMs	Signatures	Identity size	Message overhead
GM512-CORE	ML-KEM-1024, HQC-5	ML-DSA-87, SLH-DSA-SHAKE-256s	~2.7 KB	~52 KB
GM512-FULL	ML-KEM-1024, HQC-5, Classic McEliece 8192128	ML-DSA-87, SLH-DSA-SHAKE-256s, FAEST-256s	~1.36 MB	~69 KB
GM512-COMPACT (withdrawn)	ML-KEM-1024, HQC-5, Classic McEliece 8192128	ML-DSA-87, SLH-DSA-SHAKE-256s	~2.7 KB	~52 KB, no McEliece

GM512-CORE is the mandatory implementation baseline for this experimental specification. GM512-FULL offers additional construction redundancy where its approximately 1.36 MB identity bundle and larger signatures are acceptable. Transport into an archive and encryption of an archive at rest have different key-lifecycle requirements, specified in Section 10.3.

The rationale for tiering: the Classic McEliece public key alone is 1,357,824 bytes, which excludes most datagram and constrained transports. A protocol that cannot be deployed provides no security. Offering CORE as the interoperable baseline and FULL as an opt-in preserves both the deployability and the maximum-redundancy story.

4.2 Component status

Implementers must be aware that several suite components - including HQC-5 in the mandatory GM512-CORE - are not finalized standards.

Algorithm	Status
ML-KEM-1024	FIPS 203, final
ML-DSA-87	FIPS 204, final
SLH-DSA-SHAKE-256s	FIPS 205, final
HQC-5	NIST-selected for standardization (March 2025) [8]; FIPS standardization in progress. GM-3X512-PQS pin: HQC specification 2025-08-22 [7]. Parameters may change before FIPS publication.
Classic McEliece 8192128	ISO standardized in June 2026 (ISO/IEC 18033-2:2006/Amd 2:2026), including mceliece8192128 [10]. Not selected by NIST, which chose HQC in round 4.
FAEST-256s	NIST additional-signatures round 3 (May 2026); round expected to run ~2 years. Specification may be tweaked.

Requirement C1. Implementations MUST bind the exact parameter-set identifier and specification version of each component into SuiteID, so that a future parameter change produces a distinct suite rather than a silent incompatibility.

4.3 Identity

A participant U holds:

- **Long-term signature keys**, one per signature algorithm in the suite. These define the identity and are never rotated except by re-issuing the identity.
- **A long-term KEM key** (Classic McEliece in GM512-FULL; absent in CORE/COMPACT).
- **Semi-static prekeys**: one ML-KEM-1024 and one HQC-5 keypair, rotated on a defined schedule (RECOMMENDED: at most 7 days).
- **One-time prekeys**: a pool of ML-KEM-1024 keypairs, each consumed by exactly one session.

$PK_ID[U] = (pk_S1, pk_S2, pk_S3?, pk_McE?)$
 $PK_PRE[U] = (pk_MLKEM_semi, pk_HQC_semi, ValidFrom, ValidUntil)$
 $PK_OT[U] = pk_MLKEM_onetime$

4.3.1 Signed prekey bundle

The prekey bundle B publishes is the following exact byte string, together with the selected suite's AND-signature over its digest. ProfileSet is the set of ProfileIDs B will accept in Mode A.

$D_PRE = "GM512-PRE"$

$ProfileSet = ENC(m, P_1, \dots, P_m)$ -- m as I2OSP(m,4), then
 16-byte ProfileID fields

$OTCount = I2OSP(\text{number of leaves}, 4)$

$ValidFrom = I2OSP(\text{unix_seconds}, 8)$

$ValidUntil = I2OSP(\text{unix_seconds}, 8)$

$OTRoot = \text{Merkle root over the one-time prekey pool (§4.3.2)}$

$M_PRE = ENC($
 $D_PRE, Version, SuiteID, ModeID = "A", ID[B], ProfileSet,$
 $pk_MLKEM_semi, pk_HQC_semi, OTRoot,$
 $OTCount,$
 $ValidFrom, ValidUntil$
 $)$

$\mu_PRE = SHA3-512(M_PRE)$ -- also serves as BundleID

$\Sigma_PRE = ENC(\sigma_1, \sigma_2, \sigma_3?)$ -- as Σ_B , over μ_PRE

$C_PRE = ENC(M_PRE, \Sigma_PRE)$ -- the object B publishes

Requirement F6. BundleID = μ_PRE identifies exactly which bundle A used, and so resolves ProfileSet, the semi-static keys, OTRoot, OTCount and the validity interval without ambiguity. B MUST reject a BundleID it does not recognise, and MUST evaluate Requirements F4 and F5 against the resolved bundle rather than against its current one. Without this, a peer running overlapping bundles during rotation cannot determine which ProfileSet applies.

Requirement F6a. One-time prekey state is namespaced by bundle: the consumable entry is $OTPool[BundleID][OTIndex]$, not $OTPool[OTIndex]$, because overlapping active bundles legitimately share index values. A physical one-time private key MUST NOT appear under more than one active BundleID; if a deployment needs a global view it MUST assign each key a unique OTKeyID and consume that globally.

Requirement F3. A one-time prekey is delivered with its Merkle inclusion path. A MUST verify Σ_PRE and MUST verify that the supplied pk_ot is included under OTRoot before encapsulating to it. A root commitment is used rather than per-prekey signatures so that a pool of any size costs one signature.

The one-time prekey is delivered as the following object. Directory transport is out of scope, but the encoding is normative wherever such an object is transmitted.

$C_{OT} = \text{ENC}(\text{BundleID}, \text{OTIndex}, \text{pk}_{ot}, \text{Proof})$

Requirement F3a. A MUST verify that BundleID matches the bundle it verified, that OTIndex is within OTCount, and that Proof authenticates pk_{ot} under OTRoot, before encapsulating to pk_{ot} .

Requirement F4. A MUST select a ProfileID contained in ProfileSet. B MUST reject any ProfileID outside the ProfileSet of the bundle that was used.

4.3.2 One-time prekey commitment

The pool is committed with a binary Merkle tree over SHA3-512. The construction below is normative: two implementations given the same pool in the same order MUST produce the same root.

$D_{OT_LEAF} = \text{"GM512-OT-LEAF"}$
 $D_{OT_NODE} = \text{"GM512-OT-NODE"}$
 $D_{OT_EMPTY} = \text{"GM512-OT-EMPTY"}$

$\text{OTIndex}_i = \text{I2OSP}(i, 4)$ -- $i = 0 \dots \text{OTCount}-1$, in pool order

$\text{Leaf}_i = \text{SHA3-512}(\text{ENC}(D_{OT_LEAF}, \text{OTIndex}_i, \text{pk}_{ot_i}))$
 $\text{Node}(L, R) = \text{SHA3-512}(\text{ENC}(D_{OT_NODE}, L, R))$ -- L is the left child

Tree shape: perfect binary of depth $d = \text{ceil}(\log_2 \text{OTCount})$, built bottom-up in index order over 2^d positions. Positions at or above OTCount are filled with domain-separated empty leaves, so no real leaf is ever duplicated and every proof has exactly d siblings:

$\text{Empty}_i = \text{SHA3-512}(\text{ENC}(D_{OT_EMPTY}, \text{I2OSP}(i, 4)))$ -- $i \geq \text{OTCount}$

OTRoot is the single node at depth 0.

Proof = $\text{ENC}(\text{depth}, s_1 \parallel h_1, \dots, s_{\text{depth}} \parallel h_{\text{depth}})$
depth as $\text{I2OSP}(\text{depth}, 4)$; each element is one byte
 $s \in \{0x00 = \text{sibling is left}, 0x01 = \text{sibling is right}\}$
followed by the 64-byte sibling hash h

Padding to a power of two with distinct empty leaves is chosen over duplicating the last real leaf, which admits the second-preimage ambiguity affecting naive Merkle constructions, and over promoting an odd node, which would make proof length depend on OTIndex as well as OTCount. With padding every proof has exactly $d = \text{ceil}(\log_2 \text{OTCount})$ siblings, so the size figure in §10.2 is exact. Distinct leaf, node and empty separators prevent any hash from being reinterpreted in another role.

Requirement F5. OTCount MUST be bound in M_PRE, so the tree shape is fixed by the signed bundle. A verifier MUST reject a Proof whose depth is not exactly $\text{ceil}(\log_2 \text{OTCount})$, and MUST reject any $\text{OTIndex} \geq \text{OTCount}$.

Requirement F5a. A committed one-time-prekey tree MUST have $1 \leq \text{OTCount} \leq 2^{32} - 1$. An exhausted pool is not represented by $\text{OTCount} = 0$; it is handled by Requirement F2, which governs what happens when no unused leaf is available.

The identity identifier is:

$\text{ID}[U] = \text{SHA3-512}(\text{ENC}(\text{Version}, \text{SuiteID}, \text{pk}_{S1}, \text{pk}_{S2}, \text{pk}_{S3?}, \text{pk}_{McE?}))$

$\text{ID}[U] \in \{0,1\}^{512}$. It covers only long-term material, so prekey rotation does not change a participant's identity.

Requirement I2. An identity bundle $\text{PK_ID}[U]$ MUST be obtained over a channel that authenticates it independently of GM-3X512-PQS - a pinned certificate, a transparency log, an enrolment ceremony, or manual provisioning - and MUST be checked against $\text{ID}[U]$ before use. This specification does not define that channel, but it is load-bearing: pk_{McE} is resolved from $\text{PK_ID}[B]$ rather than received on the wire, so a receiver without the peer's identity bundle cannot process a GM512-FULL message at all. Revocation is likewise out of scope and MUST be provided by the deployment.

Requirement I1. Context is a raw application-supplied byte string. It MUST be agreed out of band or carried in C_GM, and MUST be the empty string if unused. Every enclosing object encodes it as an ordinary ENC field; it is not encoded a second time on its own. An implementation MUST NOT leave Context implicit.

5. Key Establishment

5.1 Two modes

Mode I (Interactive). Both parties are online. B contributes freshly generated ephemeral KEM keys. Provides full forward secrecy and contributory key control. KEM negotiation completes in one round trip; explicitly confirmed establishment completes in two.

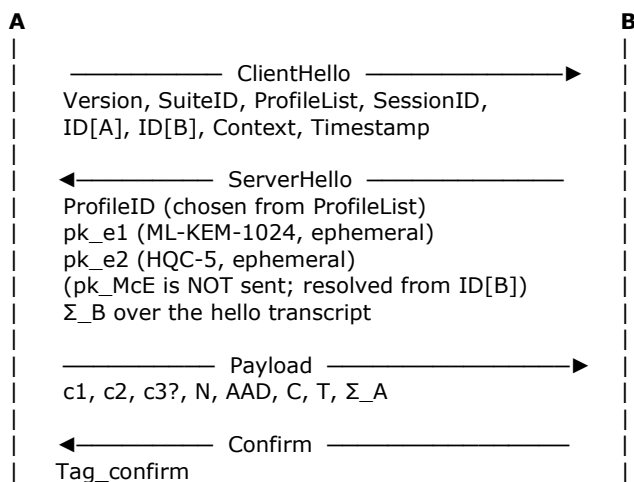
Mode A (Asynchronous). B is offline. A fetches B's signed prekey bundle from a directory. Provides forward secrecy from the one-time prekey and rotation-bounded forward secrecy from the semi-static prekey.

Mode I is RECOMMENDED whenever both parties are online.

Requirement D1. A GM-3X512-PQS session is unidirectional in payload: A is the sender and B the receiver. B does transmit during the protocol - C_SERVER_HELLO during Mode I establishment and C_CONFIRM after the payload - but no application payload travels B→A. No directional traffic keys are derived. An application needing two-way traffic MUST establish a second session in the opposite direction, with its own SessionID and its own transcript. Reusing one session's K_E or leg keys for B→A traffic is prohibited: it would reuse keystream under the same nonce space and invites reflection.

Requirement D2. Unless the optional TPQRR extension of §7.11 is selected, a GM-3X512-PQS session carries exactly one authenticated payload object C_GM. SessionID identifies that payload session and MUST NOT be reused; a subsequent payload requires a new session, a new SessionID and a fresh transcript. When TPQRR is selected, this single-payload rule is superseded by Requirements RR1-RR9: the session becomes an ordered multi-record session, every record carries Epoch and RecordSeq, and replay state is scoped to (ID[A], ID[B], SessionID, Epoch, RecordSeq). TPQRR is available in Mode I only (Requirement RR0). The base protocol remains single-payload; this specification uses distinct Version and SuiteID values and is not interchangeable with earlier draft transcripts.

5.2 Mode I message flow



The final confirmation message is REQUIRED in Mode I. It gives A explicit key confirmation: proof that B derived the same session key. Without it, A cannot distinguish a successful session from one where B's decapsulation failed.

5.2.1 Signed hello transcript

Σ_B in the ServerHello is a signature over the following exact byte string. ProfileList and ProfileID are bound here, so an adversary cannot alter A's offer or B's selection without invalidating the signature.

$D_HELLO = "GM512-HELLO"$

$ProfileList = ENC(n, P_1, P_2, \dots, P_n)$ -- n as I2OSP($n, 4$), then each
ProfileID as a 16-byte field,
in A's preference order

$M_HELLO = ENC($
 $D_HELLO, Version, SuiteID, ModeID, ProfileList, ProfileID,$
 $SessionID, Timestamp, ID[A], ID[B], Context,$
 pk_e1, pk_e2 -- pk_McE resolved from $ID[B]$
 $)$

$\mu_HELLO = SHA3-512(M_HELLO)$
 $\Sigma_B = ENC(\sigma_1, \sigma_2, \sigma_3?)$ -- each component signature over μ_HELLO ,
in SIG(SuiteID) order, σ_3 present in FULL

Requirement P5a. ProfileList and ProfileSet MUST satisfy $1 \leq n \leq 7$ and $1 \leq m \leq 7$, the leading count MUST equal the number of encoded entries, every entry MUST be one of the seven recognized 16-byte ProfileIDs, and duplicate entries MUST be rejected. A receiver MUST enforce these before any other processing of the list, which also bounds the work an unauthenticated ClientHello can request.

The three Mode I messages have the following canonical encodings. Only C_GM carries a payload; the others are handshake framing.

$C_CLIENT_HELLO = ENC(Version, SuiteID, ModeID, ProfileList,$
 $SessionID, ID[A], ID[B], Context, Timestamp)$

$C_SERVER_HELLO = ENC(Version, SuiteID, ModeID, ProfileID,$
 $SessionID, pk_e1, pk_e2, \Sigma_B)$

$C_CONFIRM = ENC(Version, SuiteID, SessionID, Tag_confirm)$

$C_CONFIRM$ carries SessionID so that A can route a confirmation to the right session when several are in flight on a multiplexed or datagram transport. Tag_confirm itself is unchanged: KMAC256($K_C, ENC("confirm", h_T)$, 256 bits, "GM512-CONFIRM"), exactly as §6 defines it; the canonical encoding and the customization string are both part of the computation. A MUST discard a $C_CONFIRM$ whose Version, SuiteID or SessionID does not match a pending session, and MUST NOT treat an unmatched confirmation as an abort of any other session.

Requirement D3. A MUST bound the time it waits for $C_CONFIRM$. On expiry A MUST zeroize K_C and discard the pending confirmation state, and MUST report the delivery status of that payload as unknown - not as failed, since B may have accepted the object, inserted the ReplayKey and released the plaintext. The timeout SHOULD NOT exceed Δ . A MUST NOT retry with the same SessionID.

Requirement H1. Every field is length-prefixed by ENC, so the encoding is injective by Requirement E1 and two conforming implementations sign identical bytes. Σ_B MUST be verified before A uses any key material from the ServerHello.

Requirement H2. Implementations exposed to untrusted networks SHOULD apply rate limiting and MAY require stateless address validation before performing ephemeral-key generation and suite signing. ClientHello is unauthenticated, and a ServerHello carries a suite AND-signature of roughly 34 KB in CORE and 51 KB in FULL, so an unauthenticated flood imposes substantial cost and, on a connectionless transport, can amplify. GM-3X512-PQS does not claim availability against unauthenticated ClientHello flooding.

Requirement H3. B MUST retain Mode I ephemeral KEM secrets only for a bounded handshake lifetime. If no matching authenticated C_GM arrives before the timeout, B MUST abandon that pending handshake and zeroize every associated ephemeral secret key. The timeout SHOULD NOT exceed Δ unless deployment policy requires otherwise. Pending state is indexed by the same value the replay cache uses:

```
HS[ReplayKey] = ( pk_e1, sk_e1, pk_e2, sk_e2, Context, ProfileList,  
                  ProfileID, Timestamp, ExpiresAt,  
                  C_SERVER_HELLO, -- the exact signed bytes  
                  h_CH )          -- SHA3-512( C_CLIENT_HELLO )
```

Without a timeout an abandoned ClientHello leaves B holding ephemeral secrets indefinitely, which both defeats the forward secrecy those keys exist to provide and gives an unauthenticated peer a memory-exhaustion lever. Requirement F1 covers deletion after decapsulation or abort; H3 covers the case where neither ever happens.

Requirement H3a. An unexpired HS[ReplayKey] MUST NOT be overwritten with freshly generated ephemeral keys. A ClientHello is a duplicate only if it is byte-identical to the one that created the PENDING entry, which B checks by comparing SHA3-512(C_CLIENT_HELLO) against the stored h_CH. For a byte-identical duplicate B MAY retransmit the cached C_SERVER_HELLO or ignore it. For any other message carrying the same ReplayKey B MUST ignore or reject it and MUST NOT retransmit, since ReplayKey covers only the identities and SessionID: an adversary who reuses those while changing ProfileList, Context or Timestamp must not be able to elicit the cached response, which is roughly 43 KB in GM512-CORE and 60 KB in GM512-FULL - the whole C_SERVER_HELLO, not just its signature. In no case may B replace the stored keypairs. The complete signed object is cached rather than reconstructed, because Σ_B covers ProfileList and the rest of M_HELLO, and re-signing could produce a different byte string. Otherwise an adversary replaying A's unauthenticated ClientHello would displace the keys A encapsulated to, and A's legitimate payload would fail to decapsulate. Handshake state has two phases: PENDING, from insertion at responder step 12, and CLAIMED(owner), from the successful atomic claim at receiver step 12b. The claim at 12b atomically MOVES sk_e1 and sk_e2 out of the shared entry into the owner's local state, so after the claim the shared entry holds no secret key and the owner holds the only copy. The owner MUST zeroize those secrets at step 14a, immediately after decapsulation, and MUST remove the CLAIMED entry when its processing completes or aborts. A processing instance whose claim at 12b fails MUST NOT mutate the entry in any way. The owner is subject to a bounded processing deadline, no longer than Δ from the claim; an implementation MUST guarantee that a stalled or crashed owner's local secrets are zeroized when that deadline passes and that the CLAIMED entry is then removed. The H3 timer applies to PENDING entries; CLAIMED entries are cleaned up by the owner deadline, so every ephemeral secret has a bounded lifetime in both phases.

Profile negotiation is defined as follows.

ClientHello : A sends ProfileList - the ProfileIDs A supports,
most-preferred first. MUST contain S-DUAL512.

ServerHello : B returns ProfileID - the first entry of ProfileList
that B also supports. If none, B aborts (E_POLICY).

ProfileID $\in \{$ "S-DUAL512", "S-KECCAK", "X-GM512", "X-TRIPLE",
"X-TRIPLE-3R", "X-TRIPLE-3RN", "X-TRIPLE-3RH" $\}$
16 bytes of ASCII, zero-padded on the right

The root profiles X-TRIPLE-3R, X-TRIPLE-3RN and X-TRIPLE-3RH are available with GM512-FULL only (Requirement X10).

The selected ProfileID is carried in C_GM and bound into T_KEM, M_AUTH and M_SIG per Requirement P3. ProfileList is covered by Σ_B over the hello transcript, so an adversary cannot strip A's preferred entries undetected. In Mode A, where there is no hello exchange, A selects a ProfileID from the profile set published in B's prekey bundle, and B MUST reject any ProfileID outside that set.

5.3 Mode A key material

A retrieves B's bundle and verifies the suite AND-signature over it before use. Encapsulation then proceeds against:

Leg	Key used	Forward secrecy contribution
z_1	ML-KEM-1024 semi-static prekey	Bounded by rotation period
z_2	HQC-5 semi-static prekey	Bounded by rotation period
z_3	Classic McEliece static (FULL only)	None - static identity-bound KEM leg; no forward-secrecy contribution
z_4	ML-KEM-1024 one-time prekey	Full , if a one-time prekey is used in this session

Requirement F1. B MUST make a one-time prekey unavailable to any other session at the moment it is claimed (step 12a), atomically and durably per Requirement R2. The local copy of sk_{ot} is retained only until z_4 has been obtained at step 13, and MUST be zeroized immediately thereafter or on abort, whichever comes first. Claiming and zeroizing are distinct events: the first prevents reuse, the second bounds the secret's lifetime in memory. In Mode I, B MUST delete its ephemeral KEM secret keys immediately after decapsulation or on abort, whichever comes first, and A MUST zeroize its encapsulation randomness and every shared KEM secret immediately after their last protocol use. For every profile except the root profiles X-TRIPLE-3R, X-TRIPLE-3RN and X-TRIPLE-3RH that point is the derivation of K_M . For those three, z_1 , z_2 and z_3 are additionally inputs to KEK_A , KEK_B and KEK_C , so they MUST be retained until those three values have been derived and MUST be zeroized immediately thereafter. The same "last protocol use" rule governs the receiver, which needs the same secrets to unwrap at step 16a. B MUST stop advertising a semi-static prekey at ValidUntil, MUST retain its secret solely for decapsulating in-flight messages until ValidUntil + Δ , and MUST delete it then. For the same window B MUST also retain the resolved bundle metadata for that BundleID - ProfileSet, OTRoot, OTRootCount and the validity interval - together with any one-time prekey state it still needs, so that a delayed but legal message is not rejected with E_BUNDLE. Deleting at ValidUntil would reject legitimate messages whose Timestamp falls inside the validity window but which arrive within the accepted network delay.

Requirement F2. Whenever a Mode A session proceeds without a one-time prekey - because B's pool is exhausted, because the sender chose not to use one, or for any other reason - the implementation MUST either (a) refuse the session, or (b) proceed and signal FS_DEGRADED to the application. It MUST NOT silently proceed. The default SHOULD be (a) for GM512-FULL.

Ephemeral and one-time encapsulation is what gives GM-3X512-PQS its purpose. A design that encapsulates only to long-term static keys leaves every recorded session decryptable on later compromise of $SK_{KEM}[B]$ - precisely the harvest-now-decrypt-later attack that motivates post-quantum migration in the first place. A transport that does not defend against that threat does not serve the purpose it was built for.

5.4 Transcript

$D_{KEM} = \text{"GM512-KEM"}$

```
T_KEM = ENC(
  D_KEM, Version, SuiteID, ModeID, ProfileID, SessionID, Timestamp,
  ID[A], ID[B],
  pk_e1 || pk_e2,      -- or the semi-static prekeys in Mode A
  pk_McE?,            -- FULL only
  pk_ot?,              -- Mode A only
  OTIndex?,           -- Mode A with a one-time prekey
  BundleID?,          -- Mode A only
  c1, c2, c3?, c4?,
  Context
)
```

$h_T = \text{SHA3-512}(T_{KEM})$ -- $h_T \in \{0,1\}^{512}$

Requirement MODE1. ModeID $\in \{ "I", "A" \}$, encoded as a single ASCII byte, distinguishes the interactive and asynchronous modes. Binding it removes any ambiguity about whether the public keys in the transcript are Mode I ephemerals or Mode A prekeys, and prevents a transcript produced in one mode from being reinterpreted in the other.

The transcript binds the protocol version, the suite, the session identifier, the timestamp, both identities, every public key used, every ciphertext produced, and the application context. Binding the ciphertexts is what makes the combiner theorem in §12 apply; omitting them is the standard mistake and it invalidates the security argument.

5.5 Combiner

D_MASTER = "GM512-MASTER"

K_M = SHA3-512(D_MASTER || z₁ || z₂ || z₃ || z₄ || h_T)

Each z_i is exactly 32 bytes for every pinned KEM: ML-KEM-1024, HQC-5 and mceliece8192128. HQC specification 2025-08-22, Section 4.2 and Table 6, defines a 32-byte shared key [7]. An absent leg (z₃ in CORE, z₄ when no one-time prekey is used) is exactly 32 zero bytes. Reject a KEM API result of any other length with E_KEM; do not pad, truncate, hash or expand it to fit. The master input therefore contains four 32-byte leg slots followed by the 64-byte h_T. Fixed lengths make the raw concatenation unambiguous under E1. Any future shared-secret-length change requires a new SuiteID.

This construction follows the robust KEM-combiner research of Giacon, Heuer and Poettering and Bindel et al. [14,15]. Section 12.1 scopes the standalone combiner argument and distinguishes it from the protocol exporter and channel-security targets.

6. Key Derivation

D_ENC = "GM512-ENC"

D_AUTH = "GM512-AUTH"

D_CONFIRM = "GM512-CONFIRM"

D_EXPORT = "GM512-EXPORT"

D_ENC_A = "GM512-ENC-A" -- S-DUAL512 leg A and X-TRIPLE leg A

D_ENC_B = "GM512-ENC-B" -- S-DUAL512 leg B and X-TRIPLE leg B

D_ENC_C = "GM512-ENC-C" -- X-TRIPLE leg C (GM-512-BC)

D_WRAP_A = "GM512-WRAP-A" -- X-TRIPLE-3R root wrap A

D_WRAP_B = "GM512-WRAP-B" -- X-TRIPLE-3R root wrap B

D_WRAP_C = "GM512-WRAP-C" -- X-TRIPLE-3R root wrap C

D_HELLO = "GM512-HELLO"

D_PRE = "GM512-PRE"

D_OT_LEAF = "GM512-OT-LEAF"

D_OT_NODE = "GM512-OT-NODE"

D_OT_EMPTY = "GM512-OT-EMPTY"

The three X-TRIPLE leg keys K_E1, K_E2 and K_E3 are derived by the same construction as K_E, using the labels above. They are derived directly from K_M and never from one another. Requirement X3 in §7.7 makes this independence normative, because it is the hypothesis on which Theorem 4 rests: if one leg key were a function of another, a compromise of the first leg would also expose the second, collapsing two legs into one compromise domain. The third, independently keyed leg would still mask the plaintext, but the three-independent-key reduction of Theorem 4 would no longer apply as stated.

For each label L:

K_x = HMAC-SHA3-512(key = K_M, data = ENC(L, h_T))

yielding K_E (encryption), K_A (authentication), K_C (key confirmation), K_X (application export).

h_T is included in the data field as well as being an input to K_M . This is redundant but harmless, and it means each derived key is independently bound to the transcript even if K_M were reused across contexts by a careless implementation.

Requirement K1. K_X is exported to the application only through an explicit API call. If the application does not request it, the implementation **MUST NOT** derive it.

Requirement K1a. The application **MUST** request K_X before key derivation begins; a request after derivation is refused, since the material it would need has been zeroized. Sender step 5 and receiver step 15 derive K_X only if it was requested. In Mode A the sender exposes K_X immediately after acceptance at step 10 and zeroizes its local copy at step 11. In Mode I the sender retains it internally until the session has completed by verifying $C_CONFIRM$ at step 12, and exposes it only then; if confirmation fails or times out, K_X is zeroized unexposed together with K_C . The base receiver exposes K_X only after step 17 has succeeded. TPQRR uses the corresponding dedicated initial acceptance and confirmation transitions in Section 7.11.5; it does not inherit a separate base step-17 insertion. In every case the implementation hands K_X to the application and then immediately zeroizes its own copy: once exported, K_X is application state, not protocol-owned session state, so Requirement K2 applies to the local copy like any other key. Exporting earlier in any of these cases would let a key escape from a session that subsequently returns $\perp$.

Requirement K1b. K_X is an application-export key and **MUST NOT** be used as a traffic, authentication, confirmation, root-wrap or ratchet key by this protocol. Application use **MUST** specify its own domain separation, access and erasure policy. The G1 experiment tests this branch only while it is unused in an adversary-visible application operation and unexposed; revealing an exported copy invalidates freshness. K_M is never a G1 Test output because public protocol tags permit candidate-master-key checking.

Requirement K2. Secret material **MUST** be erased immediately after its last permitted use using a mechanism the compiler cannot elide. This includes temporary buffers, expanded key schedules, KEKs, KEM coins and shared secrets, abandoned candidate states, and protocol-owned export-key copies. For durable state, ordinary database deletion does not establish erasure: the implementation **MUST** make superseded secrets unrecoverable from journals, snapshots, backups and crash dumps, for example through encrypted state whose obsolete wrapping keys are destroyed. If durable anti-rollback and erasure cannot be guaranteed, the implementation **MUST** terminate the affected session before serving further traffic. Section 7.11 explicitly identifies the secrets retained by an active TPQRR session.

Key confirmation tag (Mode I):

$\text{Tag_confirm} = \text{KMAC256}(K_C, \text{ENC}(\text{"confirm"}, h_T), 256 \text{ bits}, \text{"GM512-CONFIRM"})$

7. Authenticated Encryption and the State Machine

7.1 Profiles

Profile	Keystream	Authentication	Status
S-DUAL512	HMAC-SHA-512 $\oplus$ HMAC-SHA3-512 (§7.6)	KMAC256 with K_A	Mandatory to implement
S-KECCAK	HMAC-SHA3-512 in counter mode	KMAC256 with K_A	RECOMMENDED alternative
X-GM512	GM-512-BC in counter mode (§9)	KMAC256 with K_A	OPTIONAL, experimental

Profile	Keystream	Authentication	Status
X-TRIPLE	HMAC-SHA-512 $\oplus$ HMAC-SHA3-512 $\oplus$ GM-512-BC (§7.7)	KMAC256 with K_A	OPTIONAL, EXPERIMENTAL - research and evaluation only
X-TRIPLE-3R	as X-TRIPLE, keys from three sampled roots (§7.8)	KMAC256 with K_A	OPTIONAL, EXPERIMENTAL - research and evaluation only
X-TRIPLE-3RN	three GM-512-BC-N generators from three sampled roots (§7.9)	KMAC256 with K_A	OPTIONAL, EXPERIMENTAL - research and evaluation only
X-TRIPLE-3RH	Threefish-512 $\oplus$ Kalyna-512 $\oplus$ GM-512-BC-N (§7.10)	KMAC256 with K_A	OPTIONAL, EXPERIMENTAL - research and evaluation only

Requirement P1. Every implementation MUST support S-DUAL512. A peer offering only X-GM512 MUST be rejected.

Requirement P2. X-TRIPLE MAY be offered and MAY be preferred by policy, but MUST NOT be the only profile a peer offers. A peer offering X-TRIPLE MUST also offer S-DUAL512, so that the standardized profile remains reachable by negotiation.

Requirement P4. Any ProfileList or ProfileSet that advertises any experimental X-* profile MUST also contain S-DUAL512. This single rule covers every experimental profile - currently X-GM512, X-TRIPLE, X-TRIPLE-3R, X-TRIPLE-3RN and X-TRIPLE-3RH - in Mode I ProfileList and Mode A ProfileSet alike, and applies automatically to any future X-* profile, and supersedes any per-profile statement of the same policy.

Requirement P3. The negotiated ProfileID MUST be bound into the transcript and into the authenticated message, per §5.4 and §7.3. Without that binding an active adversary can steer negotiation to the cheapest mutually supported profile and neither party can detect it after the fact.

The reasoning behind putting the novel cipher off the mandatory path is worth stating plainly, because it strengthens rather than weakens the design. A protocol's security is the minimum over its components. Placing an unanalyzed primitive on the mandatory path caps the entire suite's assurance at the assurance of that primitive - which, for a new design, is unquantified. Making it optional means the suite's baseline assurance is that of the standardized SHA-2 and SHA-3 HMAC constructions, while the novel cipher can accumulate analysis without holding deployments hostage. This is how ChaCha20 entered TLS: as an alternative, after independent cryptanalysis, not as the required option in the first draft.

X-TRIPLE does not weaken that reasoning; it applies it. The minimum-over-components rule holds when components are composed in series, where each must hold. Under XOR composition with independent keys the generators are in parallel, and the distinguishing advantage is bounded by the strongest surviving independent leg under Theorem 4. GM-512-BC therefore contributes whatever strength it has without capping the assurance of the profile, which remains bounded below whenever at least one of the standardized SHA-2 or SHA-3 HMAC constructions remains a secure PRF - an assumption stated explicitly in §12.4, not a consequence of A5. This is the same structural argument that §12 Theorem 1 makes for the KEM combiner, applied one layer down.

7.2 Counter mode

```
D_CTR = SHA3-512("GM512-CTR")[0:16]
      = e1 1d 6b bb 97 07 57 26 7e 33 ad f6 84 06 4d 68
```

```
N ←$ {0,1}^256          -- fresh per message
B_j = D_CTR || N || I2OSP(j, 16)  -- 128 + 256 + 128 = 512 bits
Z_j = the negotiated profile's generator applied to B_j
C_j = P_j ⊕ Z_j
```

The negotiated profile maps B_j to a 512-bit keystream block Z_j as specified in §§7.6 to 7.10. All confidentiality generators use 512-bit keys and produce 512-bit keystream blocks.

Partial final block. Let $L = |P|$. The block index j ranges over $0 \leq j \leq \text{ceil}(L/B) - 1$, where B is the generator's output width in bytes: 64 for every profile, since all generators produce 512-bit blocks. The final keystream block is truncated to exactly $L - B \cdot (\text{ceil}(L/B) - 1)$ bytes and XORed with the remaining plaintext, so $|C| = |P|$ exactly and no padding is applied. An empty plaintext produces an empty ciphertext and consumes no keystream.

For S-DUAL512, both HMAC generators consume the full 512-bit counter block B_j directly, so the 256-bit nonce N is consumed at full width by both legs with no truncation and no per-payload key derivation. There is no truncated nonce and no 32-bit counter.

For S-KECCAK: $Z_j = \text{HMAC-SHA3-512}(K_E, B_j)$

HMAC-SHA3-512 produces a 512-bit output; HMAC has neither a customization string nor a requested output length. B_j is the counter block of §7.2, so N is consumed at full width and no key-derivation step is needed for this profile.

Requirement N1. A given traffic key MUST NOT be used with a repeated N . Since the traffic-key material is scoped to a single payload object and N is 256 bits of fresh randomness, the probability that any two of 2^{64} independent sessions draw the same N is below 2^{-128} .

Requirement N1a. The bound in N1 applies to N at full width. Every keystream generator MUST consume N at full width through its counter-block input. An implementation MUST NOT truncate or compress N before generation.

Requirement N2. In the base protocol a session carries one payload object, so no per-session message counter is needed. The binding size limit is Requirement E2: a single encoded field is at most $2^{32} - 1$ bytes. An application sending many payloads runs many sessions, each with its own SessionID, transcript and derived keys. TPQRR sessions are the sole exception: they use the ordered Epoch and RecordSeq construction of §7.11, a fresh 256-bit N for every record, and per-record root evolution.

7.3 Authentication

$M_{\text{AUTH}} = \text{ENC}(\text{Version}, \text{SuiteID}, \text{ModeID}, \text{ProfileID}, \text{SessionID}, \text{Timestamp},$
 $W_A?, W_B?, W_C?,$
 $\text{ID}[A], \text{ID}[B], h_T, N, \text{AAD}, C)$

$T = \text{KMAC256}(K_A, M_{\text{AUTH}}, 512 \text{ bits}, \text{"GM512-AUTH"})$

Encrypt-then-MAC with profile-specific pseudorandom traffic-key material independent of K_A . The tag covers the ciphertext, both identities, the transcript digest, the nonce, and the associated data.

Requirement T1. Tag comparison MUST be constant-time. An implementation MUST NOT return early on the first differing byte.

Requirement T2. No plaintext byte may be released to the application before T verifies. Streaming decryption that releases unverified plaintext is prohibited.

7.4 Replay protection

$\text{Timestamp} = \text{I2OSP}(\text{unix_seconds}, 8)$

A receiver MUST reject a message if any of the following hold:

1. $|\text{Timestamp} - \text{now}| > \Delta$, where Δ is a policy value, RECOMMENDED 300 seconds.
2. ReplayKey appears in the replay cache for the current window.

3. In Mode A, the referenced one-time prekey has already been consumed.

Requirement R1. The replay cache MUST retain ReplayKey values for at least 2Δ .

Requirement SID1. Before initiating a session, A MUST generate a fresh 32-byte SessionID from a CSPRNG and MUST abort with E_RNG on failure. Replay state MUST be keyed on the identity-scoped value below, not on SessionID alone.

$\text{SessionID} \leftarrow \$ \{0,1\}^{256}$ -- exactly 32 bytes

$\text{ReplayKey} = \text{SHA3-512}(\text{ENC}(\text{ID}[A], \text{ID}[B], \text{SessionID}))$

Scoping to the identity pair matters in Mode I, where SessionID travels in the unauthenticated ClientHello. A peer holding any valid identity can copy it and race a session of its own toward B; under a bare-SessionID cache the attacker's object would evict A's legitimate one. Binding ID[A] prevents accidental and cross-identity collisions in replay state, but it does not authenticate ID[A]. Because ClientHello is unauthenticated, an active attacker can copy a victim's ID[A] and SessionID into a modified ClientHello; if that reaches B first it creates the PENDING entry, and the legitimate ClientHello then has a different h_CH and is refused until the H3 timeout. This is a targeted Mode I preemption and availability attack, not a confidentiality break. Per Requirement H2, Mode I availability against unauthenticated ClientHello attacks is not claimed; address validation or an authenticated initiation mechanism may mitigate it.

Requirement R2. Consumption state for one-time prekeys MUST be durable and rollback-resistant: a restart, crash or restore from backup MUST NOT make a consumed one-time prekey usable again. Replay-cache state MUST either survive process restarts for at least 2Δ , or the implementation MUST provide an equivalent persistent anti-replay mechanism. G4 does not hold if either state is purely volatile: an adversary replaying a captured C_GM across a restart would otherwise be accepted.

Requirement R2a. A session that is accepted but not yet completed is a distinct state, and it can survive a crash: the Mode A sender enters it at step 10 and leaves it after step 11; the Mode I sender enters it at step 10 and remains in it through the confirmation wait, leaving it only after a successful step 12; the Mode A receiver enters it at step 17 and leaves it after step 18a; the Mode I receiver enters it at step 17 and leaves it only after step 19 has serialized C_CONFIRM and zeroized K_C. An implementation MUST record enough to recognise such a session at startup and MUST zeroize any K_M, z_i , root or derived key material a crashed instance left behind, including a Mode I sender's retained K_C and any unexported K_X, before serving traffic. G2 does not apply to a session in this state, which is the correct classification: its secrets may still exist. For TPQRR, the dedicated lifecycle in Section 7.11.5 replaces this base-session cleanup: valid active roots and K_AUTH are retained, abandoned candidates are erased, and an uncertain durable state causes session termination. The owner deadline of H3a covers the Mode I ephemeral secrets specifically; this requirement covers the session key material.

Condition 3 provides replay protection independent of clock synchronization, which is why one-time prekeys are strongly recommended even when forward secrecy is not the primary concern.

7.5 Receiver state machine

This is normative. Every failure branch outputs $\perp$ and releases nothing.

1. Parse C_GM. On any malformed field, length overflow, or unknown Version/SuiteID $\rightarrow \perp$ (E_PARSE)
ModeID $\notin \{"I", "A"\}$ $\rightarrow \perp$ (E_PARSE)
2. Check Version and SuiteID are in the local policy set $\rightarrow \perp$ (E_POLICY)
3. Check Timestamp within Δ $\rightarrow \perp$ (E_STALE)
4. Compute ReplayKey; check it is not in the replay cache $\rightarrow \perp$ (E_REPLAY)
5. Check ID[B] equals own identity $\rightarrow \perp$ (E_MISDIRECTED)
6. Resolve ID[A] to a trusted PK_ID[A] via the trust store $\rightarrow \perp$ (E_UNKNOWN_PEER)

- 6a. If ModeID = "A": resolve BundleID to the signed bundle
 - unknown BundleID $\rightarrow \perp$ (E_BUNDLE)
 - ValidFrom $\leq$ Timestamp $\leq$ ValidUntil $\rightarrow \perp$ (E_STALE)
 - ProfileID $\in$ ProfileSet $\rightarrow \perp$ (E_POLICY)
 - c4 present $\Leftrightarrow$ OTIndex present; a mixed pattern $\rightarrow \perp$ (E_PARSE)
 - when present, OTIndex < OTCount $\rightarrow \perp$ (E_PARSE)
 - when present, pk_ot MUST resolve from OTPool[BundleID][OTIndex]; otherwise $\rightarrow \perp$ (E_BUNDLE)
 - if c4 and OTIndex are both absent: apply F2 - abort per local policy, or record FS_DEGRADED for this session
 - take pk_MLKEM_semi, pk_HQC_semi and OTRoot from the bundle; pk_ot? from that bundle and pool
- 6b. If ModeID = "I": look up HS[ReplayKey] WITHOUT claiming it
 - absent or expired $\rightarrow \perp$ (E_STATE)
 - ProfileID, Timestamp and Context MUST match the stored values $\rightarrow \perp$ (E_POLICY)
 - take the PUBLIC values pk_e1 and pk_e2 only; the secret keys are NOT released until step 12b, after authentication
7. Reconstruct T_KEM and compute h_T
(all inputs are transmitted or locally held; no decapsulation is required)
8. Reconstruct M_SIG and compute μ
9. Verify σ_1 (ML-DSA-87) $\rightarrow \perp$ (E_SIG)
10. Verify σ_2 (SLH-DSA-SHAKE-256s) $\rightarrow \perp$ (E_SIG)
11. Verify σ_3 (FAEST-256s), FULL only; skipped in CORE $\rightarrow \perp$ (E_SIG)
12. $V_{GM} = \bigwedge$ over $i \in \text{SIG}(\text{SuiteID})$ of v_i ; if $V_{GM} \neq 1$ $\rightarrow \perp$ (E_SIG)
- 12a. If Mode A and OTIndex and c4 are present: atomically claim OTPool[BundleID][OTIndex] - check unused, obtain sk_ot, mark consumed $\rightarrow \perp$ (E_REPLAY)
- 12b. If Mode I: atomically claim HS[ReplayKey], MOVING sk_e1 and sk_e2 into owner-local state and removing them from the shared entry
 - already claimed $\rightarrow \perp$ (E_REPLAY)
 - since expired $\rightarrow \perp$ (E_STATE)
 - the claim is exclusive, so only one authenticated instance ever holds these ephemeral secrets
13. Decapsulate $z_1, z_2, z_3?, z_4?$ (c4 under sk_ot)
 - on decapsulation API failure or malformed result: zeroize every z_i obtained, sk_ot, sk_e1 and sk_e2 $\rightarrow \perp$ (E_KEM)
 - a claimed one-time prekey remains consumed
14. Mode A: zeroize sk_ot immediately after z_4 is obtained
- 14a. Mode I: zeroize sk_e1 and sk_e2 immediately after z_1 and z_2 are obtained; they exist only in owner-local state by now (12b)
15. Derive K_M, K_E, K_A
 - if K_X was requested before this step: also derive K_X (K1a)
 - then zeroize per F1: all z_i now unless ProfileID is a root profile (3R, 3RN, 3RH), in which case retain z_1, z_2, z_3 for step 16a and zeroize z_4 now
 - Mode I: also derive K_C (needed for Tag_confirm at step 19)
 - ProfileID = X-TRIPLE: derive K_E1, K_E2, K_E3 (§7.7)
 - ProfileID = S-DUAL512: derive K_E1, K_E2 (§7.6)
 - root profiles (X-TRIPLE-3R, -3RN, -3RH): leg keys are recovered at step 16a
16. Recompute T'; if T' $\neq$ T in constant time $\rightarrow \perp$ (E_AUTH)
- 16a. If ProfileID is a root profile (X-TRIPLE-3R, -3RN, -3RH):
 - derive KEK_A, KEK_B, KEK_C (§7.8)
 - $R_A = W_A \oplus \text{KEK}_A$, $R_B = W_B \oplus \text{KEK}_B$, $R_C = W_C \oplus \text{KEK}_C$
 - set K_E1 = R_A, K_E2 = R_B, K_E3 = R_C
 - zeroize z_1, z_2, z_3 immediately after the KEKs are derived
 - zeroize KEK_A/KEK_B/KEK_C immediately after unwrapping
 - zeroize K_M and unused K_E after their last derivation use
17. Atomically check-and-insert ReplayKey in the replay cache;
 - if already present at this point $\rightarrow \perp$ (E_REPLAY)

18. Decrypt and release plaintext; report FS_DEGRADED if recorded at step 6a; expose K_X if it was requested at step 15 (K1a, F2)
- 18a. Zeroize K_M , every z_i still held, $R_A/R_B/R_C$ and every derived key except K_C in Mode I, which step 19 zeroizes; the local copy of K_X is zeroized here, step 18 having handed it to the application. Mode A: the session is now completed (§11.1)
19. Mode I: compute Tag_confirm; serialize $C_CONFIRM$; zeroize K_C ; the session is now completed; then transmit the serialized $C_CONFIRM$, which contains no secret

Classic McEliece is published once, not retransmitted. pk_McE is never transmitted. $ID[B] = \text{SHA3-512}(\text{ENC}(\text{Version}, \text{SuiteID}, pk_S1, pk_S2, pk_S3?, pk_McE?))$ already commits to it, so a receiver resolves pk_McE from the trusted identity bundle $PK_ID[B]$ held in its trust store. The key remains an input to T_KEM , because it is the public key actually used, but it costs no wire bytes. Retransmitting it would add 1,357,824 bytes to every FULL handshake and to every prekey bundle, which is why the identity bundle is published once.

Bundle resolution precedes transcript reconstruction. Step 6a precedes step 7 because in Mode A the transcript contains the semi-static public keys and, when a one-time prekey is used, pk_ot . B cannot reconstruct T_KEM before it knows which bundle those keys came from. Requirements F4, F5 and F6 are discharged here.

Two ordering properties are essential.

Signature verification precedes decapsulation. Step 7 computes h_T without any secret operation - every input is either transmitted or is B's own public key. This means all signatures required by the selected SuiteID can be checked before B performs any decapsulation, which is the expensive step. An algorithm that referenced h_T before defining it would be non-executable as written, and repairing that by decapsulating first would turn B into a decapsulation oracle for unauthenticated input.

Claim after authentication. Handshake state is claimed only after authentication, for the same reason. Step 6b reads the public ephemeral keys so that h_T can be reconstructed and the sender's signatures verified; only then, at step 12b, is $HS[\text{ReplayKey}]$ claimed and the secret keys released. Claiming at 6b would let anyone who observed the hello exchange send a forged C_GM with the same SessionID, consume the pending state, fail signature verification, and thereby deny A access to its own handshake.

Prekey consumption precedes decapsulation and plaintext release. Step 12a sits before step 18 so that a message which fails authentication still burns the one-time prekey. This costs one prekey per failed attempt but closes a replay window that would otherwise open between decapsulation and consumption.

7.5a Sender state machine

This is normative. Every failure branch outputs $\perp$ and zeroizes any key material already derived. Failures at steps 1 to 9 occur before C_GM is transmitted; steps 1 and 2 may already have sent a ClientHello or fetched a bundle, which carry no payload and no secret. A failure at step 12 occurs after C_GM has been transmitted and therefore reports delivery status as unknown.

1. Resolve $ID[B]$ to a trusted $PK_ID[B]$ $\rightarrow \perp$ (E_UNKNOWN_PEER)
 $\text{SessionID} \leftarrow \$ \{0,1\}^{256}$ from a CSPRNG $\rightarrow \perp$ (E_RNG)
 $\text{Timestamp} = \text{I2OSP}(\text{current unix seconds}, 8)$
 pk_McE comes from $PK_ID[B]$; it is never sent on the wire

2. Mode I: send ClientHello; receive ServerHello
 verify Σ_B over M_HELLO (§5.2.1) $\rightarrow \perp (E_SIG)$
 Version, SuiteID, ModeID and SessionID MUST equal those
 sent in ClientHello $\rightarrow \perp (E_POLICY)$
 returned ProfileID must be in the ProfileList sent $\rightarrow \perp (E_POLICY)$
 Mode A: fetch C_PRE ; verify Σ_PRE over M_PRE $\rightarrow \perp (E_SIG)$
 Version and SuiteID MUST equal the intended session's,
 ModeID MUST be "A", ID[B] MUST equal the intended peer $\rightarrow \perp (E_POLICY)$
 ValidFrom $\leq$ Timestamp $\leq$ ValidUntil $\rightarrow \perp (E_STALE)$
 select ProfileID $\in$ ProfileSet $\rightarrow \perp (E_POLICY)$
 if a one-time prekey is used, verify Merkle inclusion
 under OTRoot and record OTIndex $\rightarrow \perp (E_BUNDLE)$
 if no one-time prekey is used, for any reason: apply F2 —
 abort per policy, or set FS_DEGRADED and continue
3. Encapsulate to each KEM of the suite, giving
 $c_1, c_2, c_3?, c_4?$ and $z_1, z_2, z_3?, z_4?$ $\rightarrow \perp (E_KEM)$
- 3a. Immediately after each encapsulation yields (c_i, z_i) , zeroize any
 externally materialized encapsulation randomness. If the KEM API
 neither exposes nor retains it, this is satisfied by the API
4. Build T_KEM ; compute h_T
5. Derive K_M , then K_E and K_A ; Mode I only: also derive K_C
 if K_X was requested before this step: also derive K_X (K1a)
 ProfileID = X-TRIPLE: derive K_E1, K_E2, K_E3 (§7.7)
 ProfileID = S-DUAL512: derive K_E1, K_E2 (§7.6)
 root profiles (X-TRIPLE-3R, -3RN, -3RH): sample R_A, R_B, R_C ;
 derive $KEK_A/B/C$; form W_A, W_B, W_C $\rightarrow \perp (E_RNG)$
 set $K_E1 = R_A, K_E2 = R_B, K_E3 = R_C$
 the generators keyed by them are those of the ProfileID
- 5a. Zeroize per F1: every z_i not needed further. All profiles:
 z_4 now; every other z_i now unless a root profile, in which case
 z_1, z_2, z_3 immediately after KEK_A, KEK_B, KEK_C are derived
 zeroize every KEK immediately after wrapping; zeroize K_M
 and any unused K_E after their last derivation use
6. $N \leftarrow \$ \{0,1\}^{256}$ from a CSPRNG $\rightarrow \perp (E_RNG)$
7. Encrypt $C = P \oplus Z$ under the negotiated profile
8. Build M_AUTH ; $T = \text{KMAC256}(K_A, M_AUTH, 512 \text{ bits}, "GM512-AUTH")$
9. Build M_SIG ; compute μ ; sign with every scheme in $SIG(\text{SuiteID})$
10. Emit C_GM ; Mode A: the session is accepted, and K_X , if
 requested, is exposed to the application now
11. Zeroize all remaining session key material:
 every z_i still held, $K_M, R_A/R_B/R_C$ and every derived key.
 Mode A: K_X was exposed at step 10; its local copy is zeroized
 here. Mode I only: retain K_C , and K_X if requested, until
 step 12
12. Mode I: receive $C_CONFIRM$ within the confirmation timeout of D3;
 parse it; Version, SuiteID and SessionID MUST equal the pending
 session; then verify $Tag_confirm$ $\rightarrow \perp (E_AUTH)$
 then expose K_X if requested, and zeroize K_C and the local
 copy of K_X

A failure at steps 1 to 9 aborts before C_GM is transmitted, so the receiver reaches no payload, replay or prekey-consumption state, though in Mode I it may hold temporary handshake state until its H3 timeout. A failure at step 12 is different: C_GM has already been sent, and if B accepted it then B has inserted the ReplayKey into its replay cache and released the plaintext. A missing or invalid $Tag_confirm$ therefore tells A only that it has no proof of delivery; it does not undo anything at B, and A MUST NOT retry with the same SessionID. Step 11 precedes step 12 because $Tag_confirm$ is verified with K_C alone.

7.5b Responder handshake state machine (Mode I)

This is normative. It covers B's handling of a ClientHello, up to the point where B holds pending state and has sent a ServerHello. Every failure branch outputs $\perp$, sends nothing and zeroizes any key material already generated.

1. Parse C_CLIENT_HELLO; enforce the size limit of W1 before allocating → $\perp$ (E_PARSE)
2. ModeID MUST be "I" → $\perp$ (E_PARSE)
3. Version and SuiteID MUST be in the local policy set → $\perp$ (E_POLICY)
4. $|\text{Timestamp} - \text{now}| \leq \Delta$ → $\perp$ (E_STALE)
5. ID[B] MUST equal this responder's identity → $\perp$ (E_MISDIRECTED)
6. Resolve ID[A]; apply local peer policy → $\perp$ (E_UNKNOWN_PEER)
7. Validate ProfileList per P5a; select the first entry B also supports; ProfileList MUST contain S-DUAL512 (§5.2) → $\perp$ (E_POLICY)
- 7a. Compute ReplayKey; if it is already in the replay cache the session has completed and this is a replayed ClientHello → $\perp$ (E_REPLAY)
8. If HS[ReplayKey] exists and is unexpired: compare SHA3-512(C_CLIENT_HELLO) with the stored h_CH. Identical: MAY retransmit the cached C_SERVER_HELLO, or ignore it. Different: ignore or reject; NEVER retransmit, and NEVER regenerate keys (H3a)
9. Generate pk_e1/sk_e1 and pk_e2/sk_e2 from a CSPRNG → $\perp$ (E_RNG)
10. Build M_HELLO; compute μ _HELLO; sign with every scheme in SIG(SuiteID) to form Σ _B
11. Serialize C_SERVER_HELLO
12. Atomically insert HS[ReplayKey], recording h_CH = SHA3-512 of the exact C_CLIENT_HELLO received, with ExpiresAt $\leq \text{now} + \Delta$; if a concurrent insert won, discard the freshly generated keys and apply the H3a duplicate handling of step 8; this is not an abort and no error code is raised
13. Send C_SERVER_HELLO

Insertion at step 12 precedes transmission so that B never sends a ServerHello whose ephemeral keys it has not committed to. Steps 1 to 7 are pure validation and generate no keys, which keeps the cost of rejecting a malformed or out-of-policy ClientHello low; Requirement H2 covers the flooding case that remains.

7.5c Error codes

These identifiers are normative for logging and for the state machines above. They are internal: a receiver MUST NOT signal which one occurred to the peer, since distinguishable failures leak transcript information. Requirement T2 and the uniform abort semantics of $\perp$ apply to all of them.

E_PARSE	malformed field, length overflow, unknown Version/SuiteID
E_POLICY	Version, SuiteID or ProfileID outside the local policy set, or outside the ProfileSet of the resolved bundle
E_STALE	Timestamp outside Δ , or bundle outside its validity window
E_REPLAY	ReplayKey already in the replay cache, a one-time prekey already consumed, or Mode I handshake state already claimed by another authenticated instance
E_MISDIRECTED	ID[B] is not this receiver's identity
E_UNKNOWN_PEER	ID[A] does not resolve to a trusted PK_ID[A]
E_BUNDLE	BundleID unknown, or Merkle inclusion proof invalid
E_SIG	a component signature failed, or V_GM $\neq$ 1
E_KEM	decapsulation API failure or malformed result
E_AUTH	tag T mismatch, Tag_confirm mismatch, or confirmation timeout at the sender (Requirement D3)
E_RNG	CSPRNG failure at the sender, or at the responder when generating ephemeral keys
E_STATE	no unexpired Mode I handshake state for this ReplayKey

The replay check must be atomic. Step 4 is a cheap early rejection and is not the security-critical check. Two copies of one valid Mode I message can pass step 4 concurrently, so the decisive test is the atomic check-and-insert at step 17, immediately before plaintext release. Mode A is additionally protected by the atomic prekey claim at step 12a; Mode I has no prekey; its concurrency protection is the exclusive authenticated claim at step 12b, and step 17 remains the durable final replay-acceptance check for both modes.

TPQRR ordering failures use E_REPLAY when an Epoch/RecordSeq pair has already been accepted and E_STATE when a record is not the next expected RecordSeq or references an uncommitted future epoch. A failed TPQRR record MUST NOT advance any root chain.

7.6 Profile S-DUAL512 - mandatory standards-based generator

S-DUAL512 is mandatory to implement. It composes two keyed generators drawn from the two distinct standardized hash families, each keyed with a full 512-bit key, and combines them by XOR.

```
K_E1 = HMAC-SHA3-512( K_M, ENC( D_ENC_A, h_T ) ) -- 512-bit key
K_E2 = HMAC-SHA3-512( K_M, ENC( D_ENC_B, h_T ) ) -- 512-bit key

N ←$ {0,1}^{256}
B_j = D_CTR || N || I2OSP(j, 16) -- 512-bit counter block, as §7.2

Z_j^A = HMAC-SHA-512( K_E1, B_j ) -- SHA-2 family, FIPS 180-4 [30]
Z_j^B = HMAC-SHA3-512( K_E2, B_j ) -- SHA-3 family, FIPS 202 [2]

Z_j = Z_j^A ⊕ Z_j^B
C_j = P_j ⊕ Z_j
```

Why two legs. Both legs are HMAC [29] over an approved NIST hash, so the mandatory path uses only standardized primitives. The two legs rest on structurally unrelated compression functions - Merkle-Damgård over SHA-512 and the Keccak sponge. At the generator layer, confidentiality remains protected if either HMAC-SHA-512 or HMAC-SHA3-512 remains a secure PRF, by Theorem 4 restricted to two generators, subject to the shared HMAC-SHA3-512 key-derivation assumption: K_E1 and K_E2 are both derived with HMAC-SHA3-512 and K_M with SHA3-512. S-DUAL512 therefore provides generator diversity across SHA-2 and SHA-3, but does not claim end-to-end independence from SHA-3. Neither leg is keyed below 512 bits and no payload-key derivation or block-size accounting is required, because both generators take 512-bit keys and produce 512-bit keystream blocks.

Label sharing. The keys are derived under the same labels as legs A and B of X-TRIPLE. This is not key reuse: ProfileID is bound into T_KEM and therefore into h_T, so K_M and every key derived from it differ between profiles.

7.7 Profile X-TRIPLE - parallel keystream composition

X-TRIPLE generates three keystreams independently and combines them by XOR. Leg B is the generator of Profile S-KECCAK and leg C that of Profile X-GM512; leg A is the SHA-2 generator also used by the mandatory S-DUAL512. The two HMAC legs rest on the two distinct standardized hash families, SHA-2 and SHA-3, and every leg takes a 512-bit key and emits 512-bit blocks. Each is keyed separately.

```
K_E1 = HMAC-SHA3-512( K_M, ENC( D_ENC_A, h_T ) ) -- SHA-2 leg
K_E2 = HMAC-SHA3-512( K_M, ENC( D_ENC_B, h_T ) ) -- SHA-3 leg
K_E3 = HMAC-SHA3-512( K_M, ENC( D_ENC_C, h_T ) ) -- GM-512-BC leg

N ←$ {0,1}^{256} -- fresh per message, all legs
B_j = D_CTR || N || I2OSP(j, 16) -- 512-bit counter block, as §7.2

Z_j^A = HMAC-SHA-512( K_E1, B_j ) -- per FIPS 180-4 [30]
Z_j^B = HMAC-SHA3-512( K_E2, B_j )
Z_j^C = GM-512-BC_{K_E3}( B_j ) -- per §9

Z_j = Z_j^A ⊕ Z_j^B ⊕ Z_j^C
C_j = P_j ⊕ Z_j
```

The nonce N is shared across the three legs. This is safe, and deliberate: the leg keys are independent, so a shared counter block cannot induce a relation between keystreams. Giving each leg its own nonce would enlarge the header by 64 bytes and buy nothing.

All three legs produce 512 bits per composed block index j , so the XOR is direct and no multi-block expansion is required.

7.8 Profile X-TRIPLE-3R - three independently sampled roots

X-TRIPLE derives its three leg keys from K_M . They are computationally independent, which is what Theorem 4 requires, but they are not independently sampled: a single derivation ancestor stands behind all three. X-TRIPLE-3R removes that ancestor from the generator keys. The sender draws three roots from the system CSPRNG and wraps each one separately.

$R_A, R_B, R_C \leftarrow \$\{0,1\}^{512}$ -- independent, per session

$KEK_A = \text{SHA3-512}(\text{ENC}(D_WRAP_A, K_M, z_1, h_T))$

$KEK_B = \text{SHA3-512}(\text{ENC}(D_WRAP_B, K_M, z_2, h_T))$

$KEK_C = \text{SHA3-512}(\text{ENC}(D_WRAP_C, K_M, z_3, h_T))$

$W_A = R_A \oplus KEK_A \quad W_B = R_B \oplus KEK_B \quad W_C = R_C \oplus KEK_C$

leg keys: $K_{E1} = R_A \quad K_{E2} = R_B \quad K_{E3} = R_C$

z_3 is present in GM512-FULL, where this profile is defined.

Requirement X10. The root profiles X-TRIPLE-3R, X-TRIPLE-3RN and X-TRIPLE-3RH are defined for GM512-FULL only. GM512-CORE exposes only two base KEM legs for distinct root binding; GM512-FULL adds a third, Classic McEliece, so each KEK can bind a separate secret. They MUST NOT be offered or selected when SuiteID is GM512-CORE; a peer receiving any of these ProfileIDs with a CORE SuiteID MUST abort with E_POLICY.

W_A, W_B and W_C are transmitted in C_{GM} and are covered by M_{AUTH} and by the signatures. They are not inputs to T_{KEM} : h_T is an input to each KEK, so including the wraps in the transcript would be circular. Integrity of the wraps therefore rests on the tag and the signatures, both of which are verified before any wrap is opened.

Why both terms. Each KEK binds K_M and one leg's own KEM secret. Computing KEK_i requires both, so KEK_i stays secret if K_M is secret OR z_i is secret. K_M is secret whenever any one KEM survives, by Theorem 1. Branch i therefore inherits the full any-one-KEM property rather than depending on KEM_i alone.

Why not bind each root to a single KEM. Binding each root to a single KEM would make branch i secure exactly when KEM_i and G_i both survive. That is a conjunction, and it converts Theorem 1 from a property of the whole record layer into a property of one third of it: if ML-KEM and HQC are both broken, a single-binding design loses the two branches bound to them, while the construction above loses none: Classic McEliece still protects K_M , so all three root wraps remain computationally hidden under the combiner assumption, independently of which symmetric generator supplies the surviving confidentiality leg. The K_M term costs nothing and preserves the central claim of this specification.

Requirement X7. The sender MUST draw R_A, R_B and R_C from a CSPRNG, independently, once per session. On any RNG failure the sender MUST abort. Deriving any root from another, or from K_M , is prohibited and forfeits the Theorem 4 hypothesis.

Requirement X8. A receiver MUST verify the tag T and all signatures before unwrapping any root. Unwrapping is a secret-key operation on attacker-supplied bytes and MUST NOT precede authentication.

Requirement X9. R_A, R_B and R_C MUST occupy separate allocations, MUST be zeroized separately, and MUST be zeroized when the session terminates. For the base protocol this includes every abort path. In an active TPQRR session, a rejected record or pre-commit refresh attempt erases only its temporary candidates; the committed roots are retained under RR2a and RR4. Requirements X4, X5 and X6 apply unchanged. Requirement X3 is superseded for this profile: the leg keys are the sampled roots R_A, R_B, R_C rather than K_{E1}, K_{E2}, K_{E3} derived from K_M , and X7 forbids deriving any root from another or from K_M , which is the property X3 exists to guarantee.

Cost: 192 bytes of additional wire overhead and three SHA3-512 calls per session. Payload-encryption cost is identical to X-TRIPLE.

The 192 bytes are additional to the base message totals reported in §10.2.

What this does not buy. The joint root tuple has 2^{1536} possible values, and this specification does not claim 1,536-bit security, 768-bit quantum security, or any level above the 256-bit target of §3.1. Recovery of the root tuple may proceed through attacks on the wrapping construction, on its KEM inputs, on the combiner, on the individual generators, or on the composed XOR itself. No generic key-recovery bound above the 256-bit target is claimed for any of those paths. What three sampled roots buy is that the generator keys are separately sampled and are derived neither from one another nor from K_M . Their independence rests on the security of the system CSPRNG, which is itself a deterministic function of its seed: 1,536 bits of root-key material is not 1,536 bits of entropy, and no such claim is made. What the profile strengthens is the hypothesis of Theorem 4, not the key strength.

What independent roots do not remove. Independent root sampling provides key-material independence, not independence of primitive assumptions. Legs B and C still share Keccak: leg B is an HMAC-SHA3-512 generator and leg C is GM-512-BC, whose round-key schedule of §9.5 uses HMAC-SHA3-512. Leg A rests on SHA-2 instead, so a structural failure of Keccak would reach two of the three legs but not all three. This does not contradict Theorem 4, which is stated over the generators rather than over their key derivations, and the profile already assumes HMAC-SHA3-512 for key derivation and KMAC256 for authentication. It does mean that X-TRIPLE-3R buys diversity of generator, not diversity of every underlying primitive. Achieving the latter requires generators that share no symmetric primitive. X-TRIPLE-3RH (§7.10) does this with three structurally different 512-bit ciphers, and GM-512-BC-N (§9.5a) supplies a Keccak-independent schedule for the GM-3X512-PQS leg.

X-TRIPLE-3R retains the X-TRIPLE authentication construction: KMAC256 with K_A over M_{AUTH} , exactly as the single-generator profiles use it, with the same encrypt-then-MAC ordering and the same Requirements T1 and T2. Integrity therefore rests on a standardized primitive in every profile, and no integrity or sender-authentication property of the record layer depends on GM-512-BC. Confidentiality under X-GM512 does depend on it, and Theorem 4 permits GM-512-BC to be the surviving generator of X-TRIPLE.

Requirement X3. The keystream legs MUST be keyed by K_{E1} , K_{E2} and K_{E3} as defined in §6. An implementation MUST NOT derive any leg key from another leg key, and MUST NOT substitute K_E for any of them.

Requirement X4. At least two of the three legs MUST be built solely from standardized primitives. X-TRIPLE satisfies this with HMAC-SHA-512 [29, 30] and HMAC-SHA3-512 [29, 2], which rest on the two distinct standardized hash families. This specification satisfies the requirement by construction (legs A and B). A future profile that replaced a standardized leg with a second unanalyzed one would forfeit the assurance floor and MUST NOT reuse the X-TRIPLE identifier.

Requirement X5. Each leg MUST be implemented in constant time, with no secret-dependent branch and no secret-indexed memory access. GM-512-BC in particular MUST use a bitsliced or vector-permute implementation; a table-driven implementation is prohibited. A timing side channel in any one leg leaks that leg's keystream, and Theorem 4 then holds only on the strength of the remaining two.

Requirement X6. K_{E1} , K_{E2} and K_{E3} MUST occupy separate allocations and MUST be zeroized separately. Separate allocations reduce accidental cross-leg overwrite and make independent zeroization straightforward, but they are not memory isolation: a process-wide disclosure exposes all three regardless. Genuine compromise isolation between legs requires separate protection domains, separate processes, or hardware-backed key storage, and is outside the scope of this specification.

Requirements N1 and N2 apply unchanged to X-TRIPLE. Under Requirement D2 each session carries exactly one payload, so the nonce discipline is per session and not per leg.

Cost. Three keystreams are generated for one payload. Independent legs may execute concurrently if the implementation provides adequate cores, memory bandwidth and protection domains. Actual throughput depends on the slowest concurrent leg, XOR, authentication, scheduling and data movement; serial execution adds the leg costs. Section 9.6 reports primitive measurements only. End-to-end measurements must include every signature, KEM operation, MAC, state transaction and erasure required by the selected mode.

Applicability. X-TRIPLE is an experimental option for studying symmetric generator diversity. S-DUAL512 is the mandatory baseline for comparison. Neither this choice nor use of standardized primitives establishes production assurance for the composed GM protocol; release evidence is specified in Section 15.

What X-TRIPLE does not provide. It does not raise the security level above the 256-bit target stated in §3.1. The three leg keys total 1,536 bits of key material, but the claimed level remains capped by the 256-bit material entering the combiner and by KMAC256 authentication, exactly as §3.2 describes, and no composition of generators can raise a bound imposed upstream. The property gained is redundancy of assumption, not additional key strength, and the specification claims only the former.

7.9 Profile X-TRIPLE-3RN - three native generators

This profile answers a specific objection to X-TRIPLE-3R: that legs B and C still depend on Keccak - leg B directly, and leg C through the round-key schedule of §9.5 - while leg A depends on SHA-2. X-TRIPLE-3RN removes both HMAC legs from the keystream generators entirely, leaving no Keccak or SHA-2 construction there, since GM-512-BC-N generates its own round keys. Authentication, key derivation and root wrapping are unchanged and remain Keccak-based.

$R_A, R_B, R_C \leftarrow \$\{0,1\}^{512}$ wrapped exactly as in §7.8

$D_CTR_X = \text{SHA3-512}(\text{"GM512-CTR-"} \parallel X)[0:16] \quad X \in \{A, B, C\}$
 $B_{\{X,j\}} = D_CTR_X \parallel N \parallel \text{I2OSP}(j, 16) \quad 512\text{-bit counter block}$

$Z_j = \text{GM-512-BC-N}_{\{R_A\}}(B_{\{A,j\}}) \oplus \text{GM-512-BC-N}_{\{R_B\}}(B_{\{B,j\}}) \oplus \text{GM-512-BC-N}_{\{R_C\}}(B_{\{C,j\}})$

$C_j = P_j \oplus Z_j$

Per-leg counter separators ensure no two legs ever process an identical block, independently of key separation. Wrapping and the KEK construction are inherited from §7.8 unchanged. Requirements X5, X6, X7 and X8 apply mutatis mutandis. The allocation, zeroization and abort-handling provisions of X9 apply, but its reference to X4 does not. Requirement X10's FULL-only policy applies. Requirement X3 does not apply: the leg keys are the sampled roots R_A, R_B, R_C , not K_{E1}, K_{E2}, K_{E3} derived from K_M . Requirement X4 does not apply either: all three generators are deliberately the experimental GM-512-BC-N, which is the point of the profile and the reason for the limitation stated below. Authentication remains KMAC256 over M_AUTH : this profile changes the generators only.

What this profile does and does not buy. X-TRIPLE-3RN satisfies the independent-key hypothesis of Theorem 4 through three independently sampled 512-bit roots, and the theorem applies to it exactly as it does to X-TRIPLE. Because all three generators instantiate GM-512-BC-N, however, the profile provides no hedge against a structural break of that primitive: a differential in the round function, an invariant subspace, or a related-key relation in the §9.5a schedule would affect every leg. Its confidentiality reduction therefore rests on the multi-key PRP and PRF security of GM-512-BC-N rather than on three heterogeneous primitive assumptions. What the profile delivers is independence of leg keys and a keystream path free of both HMAC legs; what it does not deliver is primitive diversity, and this specification says so rather than presenting the XOR as a hedge it is not.

Requirement X11. X-TRIPLE-3RN is OPTIONAL and EXPERIMENTAL and inherits the FULL-only restriction of Requirement X10. It MUST NOT protect production data. A ProfileList or ProfileSet offering it MUST also contain S-DUAL512, per Requirement P4. Deployments seeking cryptanalytic redundancy across independent assumptions

SHOULD use X-TRIPLE-3RH, which is the all-512-bit heterogeneous option, or X-TRIPLE or X-TRIPLE-3R, whose legs also rest on different primitives.

Cost is three GM-512-BC-N keystreams per payload, computable in parallel. Wire overhead is identical to X-TRIPLE-3R, since the wrapped roots are unchanged.

7.10 Profile X-TRIPLE-3RH - heterogeneous 512-bit generators

X-TRIPLE-3RN removes both HMAC legs from the generators but uses the same permutation in all three legs. Theorem 4 still applies there, because the three keys are independently sampled; what that profile lacks is a heterogeneous primitive hedge against a structural break of GM-512-BC-N. X-TRIPLE-3RH restores heterogeneous primitive diversity and gives Theorem 4 a meaningful surviving-primitive interpretation, by using three structurally different 512-bit block ciphers as generators.

$R_A, R_B, R_C \leftarrow \$ \{0,1\}^{512}$ wrapped exactly as in §7.8

$D_CTR_X = \text{SHA3-512}(\text{"GM512-CTR-"} \parallel X)[0:16] \quad X \in \{A, B, C\}$
 $B_{\{X,j\}} = D_CTR_X \parallel N \parallel \text{I2OSP}(j, 16)$ 512-bit counter block

$Z_j = \text{Threefish-512}_{\{R_A, T=0\}}(B_{\{A,j\}}) \quad [28]$
 $\oplus \text{Kalyna-512/512}_{\{R_B\}}(B_{\{B,j\}}) \quad [26]$
 $\oplus \text{GM-512-BC-N}_{\{R_C\}}(B_{\{C,j\}}) \quad §9.5a$

$C_j = P_j \oplus Z_j$

Threefish-512 takes a 512-bit key and a 128-bit tweak; the tweak is fixed to zero, since the counter block already provides per-block separation and the key is fresh per session. Kalyna is used in its 512/512 parameterisation, the only one with a 512-bit block and a 512-bit key. Wrapping and the KEK construction are inherited from §7.8 unchanged. Requirements X5, X6, X7 and X8 apply mutatis mutandis. The allocation, zeroization and abort-handling provisions of X9 apply, but its reference to X4 does not. Requirement X10's FULL-only policy applies. Requirement X3 does not apply, since the leg keys are the sampled roots. Requirement X4 does not apply either: X-TRIPLE-3RH has its own primitive-diversity rule in Requirement X12.

Serialization. Each leg interprets the 64-byte key R_X and the 64-byte block $B_{\{X,j\}}$ according to its own reference document; the mapping from the byte strings of this specification is fixed as follows and is normative.

Threefish-512 [28]

key : R_A as 8 words, word $i = \text{LE64}(R_A[8i \dots 8i+7])$
 tweak : 128 bits, both words zero
 block : $B_{\{A,j\}}$ as 8 words, word $i = \text{LE64}(B_{\{A,j\}}[8i \dots 8i+7])$
 output : word i emitted as LE64, giving 64 bytes

Kalyna-512/512 [26]

key : R_B mapped to the 8×8 state as DSTU 7624:2014 defines,
 column-major with 64-bit columns in little-endian order
 block : $B_{\{B,j\}}$ mapped by the same rule
 output : the state serialized by the inverse of that mapping

GM-512-BC-N (§9.1, §9.5a)

key : R_C as the 8×8 state, column-major, byte $i \rightarrow b[i \bmod 8][i \div 8]$
 block : $B_{\{C,j\}}$ by the same rule
 output : the state serialized by the same rule

Requirement X13. The three legs use different word orders because their specifications do; no attempt is made to normalise them. An implementation MUST reproduce the published test vectors of [28] and [26] before composing this profile, since a byte-order error produces a valid-looking keystream that silently fails to interoperate.

Why this restores heterogeneous primitive diversity. The three legs rest on structurally different constructions: Threefish-512 is ARX, published as the core of the SHA-3 finalist Skein; Kalyna-512/512 is a wide-trail SPN standardized as DSTU 7624:2014; GM-512-BC-N is the design of §9. Kalyna and GM-512-BC-N still share the broad

AES-like wide-trail tradition, so full independence of failure is more than can be demonstrated today. The defensible claim is the one Theorem 4 actually makes: if any one of Threefish-512, Kalyna-512/512 or GM-512-BC-N remains secure under its independent 512-bit key, the composed keystream masks the outputs of the other two. Unlike X-TRIPLE-3RN, where the surviving-leg assumption is a single multi-key assumption on one permutation, here it is a disjunction over three distinct primitives. Two of the three carry substantial public analysis, so the surviving-leg assumption is not resting on unreviewed work alone.

Requirement X12. X-TRIPLE-3RH provides an all-512-bit heterogeneous confidentiality-generator path: all three keystream generators use 512-bit blocks and independently sampled 512-bit keys, with no HMAC-SHA-512, HMAC-SHA3-512 or other Keccak-based construction used as a keystream generator. Authentication, key derivation and root wrapping remain based on the existing standardized Keccak constructions, so the profile is not Keccak-free and does not claim to be. It is the only profile that combines that generator path with a meaningful any-one-survives argument. It is nevertheless OPTIONAL and EXPERIMENTAL: neither Threefish-512 nor Kalyna-512 is NIST-approved, so it carries no FIPS path, and it requires two additional primitive implementations. It inherits the FULL-only restriction of Requirement X10, MUST NOT protect production data, and a ProfileList or ProfileSet offering it MUST also contain S-DUAL512 per Requirement P4.

Vectors. Test vectors for this profile are not given here. They require reference implementations of Threefish-512 and Kalyna-512/512, whose own published test vectors must be reproduced first; §15 records this. The counter-block construction and the composition are fully specified above, so the vectors are derivable once those implementations are validated.

7.11 Triple Post-Quantum Root Ratchet (TPQRR) - optional post-compromise recovery extension

Status: OPTIONAL and EXPERIMENTAL. TPQRR is defined only for the independently rooted profiles X-TRIPLE-3R, X-TRIPLE-3RN and X-TRIPLE-3RH under GM512-FULL. It adds ordered multi-record operation and post-compromise recovery while preserving the three-root architecture. A TPQRR session uses Version = "GM512-TPQRR-v0005"; peers that do not implement this version remain on the base single-payload semantics of Requirement D2. TPQRR is not a new confidentiality profile: it is a session-state evolution layer above the selected root profile. It is selected by Version alone: A offers "GM512-TPQRR-v0005" in ClientHello and B accepts it only if that Version is in its local policy set (§7.5b step 3), so a peer that does not implement TPQRR rejects the hello with E_POLICY and nothing is negotiated silently.

Requirement RR0. TPQRR requires ModeID = "I". A refresh needs B to generate fresh ephemeral KEM keys and to sign a RATCHET_OFFER while the session is live, which an offline Mode A peer cannot do; the prekey directory of §4.3.1 offers no substitute, because a directory-published key is neither fresh per refresh nor bound to the session. A ClientHello carrying Version = "GM512-TPQRR-v0005" with any ModeID other than "I" MUST be rejected with E_PARSE. TPQRR sessions therefore never carry OTIndex or BundleID, and the KEM legs of the initial object are the Mode I legs z_1 , z_2 and z_3 . The initial object of a TPQRR session carries Epoch = 0 and RecordSeq = 0.

Design goal. Forward secrecy protects records created before a later compromise. TPQRR adds the complementary goal of recovering protection for future records after a compromise of the current session state. The recovery event must inject entropy that was not available to the attacker at the time of compromise; deterministic hashing of an already compromised root cannot provide that property.

7.11.1 Per-record triple-root evolution

At epoch e and record number r , the three current root-chain states are $R_A[e,r]$, $R_B[e,r]$ and $R_C[e,r]$. At epoch 0 they are the independently sampled roots established by the selected root profile. For X in $\{A,B,C\}$, one record key is derived and the root is then advanced under a distinct domain separator:

Author: Gianclaudio Moresi

```
h_E[0] = h_T          -- the session transcript digest of §5.4
h_E[e] = h_R[e] for e ≥ 1 -- the refresh transcript digest (§7.11.2)
                        that installed epoch e
K_AUTH[0] = K_A       -- the epoch authentication key; at epoch 0 it is the base K_A of §6

K_X[e,r] = HMAC-SHA3-512(
  key = R_X[e,r],
  data = ENC( D_RATCHET_KEY, "X", SessionID,
              I2OSP(e,8), I2OSP(r,8), h_E[e] ) )

R_X[e,r+1] = HMAC-SHA3-512(
  key = R_X[e,r],
  data = ENC( D_RATCHET_STEP, "X", SessionID,
              I2OSP(e,8), I2OSP(r,8), h_E[e] ) )
```

Derive successor roots as temporary candidates. Do not erase the committed $R_X[e,r]$ until the durable transition in Section 7.11.5 succeeds. Then erase the predecessor roots and one-record keys. On pre-commit failure, erase candidates and retain committed roots.

Here "X" is the one-byte leg identifier A, B or C. The record keys $K_A[e,r]$, $K_B[e,r]$ and $K_C[e,r]$ replace the three profile roots only for that record's keystream generation; they are distinct from the base K_A , base confirmation K_C and application-export K_X . The separate labels distinguish record-key derivation from root evolution. $h_E[e]$ binds the pre-wrap KEM transcript of the refresh that created epoch e . It does not include the wrapped roots or $RecordSeq_last$: those values are authenticated by T_R and the complete signature over M_COMMIT . This separation avoids a circular wrap definition.

Requirement RR1. TPQRR records MUST be processed in strictly increasing $RecordSeq$ order within an epoch. Epoch and $RecordSeq$ MUST be inserted immediately after $SessionID$ in C_GM_R , M_AUTH_R and M_SIG_R , as §7.11.1a defines. Replay state is keyed by $SHA3-512(ENC(ID[A], ID[B], SessionID, I2OSP(Epoch,8), I2OSP(RecordSeq,8)))$. This draft deliberately does not specify skipped-key storage; a transport requiring out-of-order delivery MUST buffer and reorder before TPQRR processing, subject to W1 and RR9. A dropped record blocks later records until the exact serialized record is recovered or the session terminates.

Requirement RR2. Every record MUST draw a fresh 256-bit nonce N and consume it at full width. After a record is authenticated and the successor roots have been committed, the prior roots and the three one-record generator keys MUST be zeroized. Under the HMAC-SHA3-512 PRF/one-way assumption, compromise of the current root state does not reveal erased roots or record keys from earlier records in the same epoch. It does, however, permit computation of future deterministic steps in that epoch; recovery of the future requires the post-quantum refresh below.

7.11.1a TPQRR record format and authenticated processing

A TPQRR application record is a C_GM record with two additional fixed-width fields, Epoch and $RecordSeq$. The epoch selects the currently committed triple-root state; $RecordSeq$ identifies the ordered record inside that epoch. Both fields are authenticated and signed and therefore cannot be changed without detection.

```
C_GM_R = ENC(
  Version, SuiteID, ModeID, ProfileID, SessionID,
  I2OSP(Epoch,8), I2OSP(RecordSeq,8), Timestamp, Context,
  W_A?, W_B?, W_C?,          -- present only when Epoch = 0 ∧ RecordSeq = 0
  ID[A], ID[B],
  c1?, c2?, c3?,            -- present only when Epoch = 0 ∧ RecordSeq = 0
  N, AAD, C, T,
  σ1, σ2, σ3? )

M_AUTH_R = ENC(
  Version, SuiteID, ModeID, ProfileID, SessionID,
  I2OSP(Epoch,8), I2OSP(RecordSeq,8), Timestamp,
  W_A?, W_B?, W_C?,
  ID[A], ID[B], h_E[Epoch], N, AAD, C )
```

$T = \text{KMAC256}(K_AUTH[\text{Epoch}], M_AUTH_R, 512 \text{ bits}, "GM512-AUTH")$

$M_SIG_R = \text{ENC}($
 Version, SuiteID, ModeID, ProfileID, SessionID,
 I2OSP(Epoch,8), I2OSP(RecordSeq,8), Timestamp,
 W_A?, W_B?, W_C?,
 ID[A], ID[B], h_E[Epoch], c1?, c2?, c3?, N, AAD, C, T)

$\mu = \text{SHA3-512}(\text{ENC}(D_SIG, M_SIG_R)); \Sigma \text{ as } \S 8$

Requirement RR1a. The initial object of a TPQRR session is C_GM_R with Epoch = 0 and RecordSeq = 0. It carries the Mode I KEM ciphertexts c1, c2, c3 and the three wrapped roots W_A, W_B, W_C exactly as the base C_GM does for the selected root profile, and it is processed by the receiver state machine of §7.5 with the TPQRR additions of §7.11.5. In every later record, in every epoch, those six fields are absent and MUST be encoded as FIELD("") per Requirement E3; a receiver MUST reject any other presence pattern with E_PARSE. The wraps are covered by M_AUTH_R and by M_SIG_R, and the ciphertexts by M_SIG_R and by h_T, so the coverage that §7.3 and §8 give the base M_AUTH and M_SIG is preserved unchanged and Requirement X8 applies as written. Refresh material never travels in a record: the fresh ciphertexts and wrapped roots of an epoch $e \geq 1$ are carried only in C_RATCHET_COMMIT (§7.11.2) and are authenticated by the signed and MACed M_COMMIT. h_E[e] includes the KEM public keys and ciphertexts but excludes the wraps and RecordSeq_last; those values are fixed by the unique successfully installed commit, not by a claimed hash coverage that h_R does not provide. Because TPQRR is Mode I only (RR0), OTIndex and BundleID are not fields of C_GM_R.

Requirement RR1b. Steps 3 and 4 of §7.5 apply to every record, not only to the initial object: |Timestamp – now| MUST be at most Δ , and the record replay key of RR1 MUST be retained for at least 2Δ and MUST survive restarts, per Requirements R1 and R2. RecordSeq ordering (RR1) is the primary replay defence; the timestamp window bounds how long a captured record remains a candidate for the reordering buffer.

Requirement RR2a. A receiver MUST authenticate and verify all required signatures for RecordSeq r before committing R_X[e,r+1] as the active receive state or releasing plaintext. The state transition from r to r+1 MUST be atomic with replay-state insertion. If authentication fails, the receiver MUST retain R_X[e,r] and MUST NOT advance the chain. This prevents an unauthenticated record from consuming ratchet state and desynchronizing the peers.

Requirement RR2b. Record acceptance atomically commits all three successor roots, the next expected sequence, the replay marker and the predecessor-erasure transaction before plaintext release. State transitions for a session MUST be serialized or protected by an equivalent compare-and-swap over the epoch, sequence and state generation. A losing concurrent instance erases its temporary candidates and releases nothing. A crash MUST NOT restore an earlier root or sequence. A sender MUST prepare encryption, authentication and every signature before atomically committing the exact serialized ciphertext in a durable outbox together with the successor state; it emits only bytes from that outbox. It MUST NOT regenerate a different record for a committed sequence. These rules apply to the initial record as well as later records.

7.11.2 Periodic post-quantum root refresh

A full ratchet event advances the epoch and replaces all three root chains with newly sampled roots. It is performed while the transport connection remains established, which is why TPQRR is Mode I only (RR0). Receiver B first obtains fresh entropy and restores an unpredictable CSPRNG state, then generates fresh ephemeral ML-KEM-1024 and HQC-5 keypairs and signs the offer transcript M_OFFER below with every signature scheme required by SuiteID. Sender A verifies the complete AND-signature before encapsulating. The three objects exchanged by a refresh - offer, commit and confirm - have the following exact byte encodings; as with M_HELLO, every field is an ENC field, so two conforming implementations sign identical bytes (Requirement H1).

$M_OFFER = \text{ENC}($
 D_RATCHET_OFFER, Version, SuiteID, ModeID, ProfileID, SessionID,
 I2OSP(Epoch',8), I2OSP(RecordSeq_next,8),

ID[A], ID[B], Context, pk_r1, pk_r2) -- pk_McE[B] resolved from ID[B]

$\mu_OFFER = \text{SHA3-512}(M_OFFER)$

$\Sigma_B^R = \text{ENC}(\sigma_1, \sigma_2, \sigma_3)$ -- each over μ_OFFER , in $\text{SIG}(\text{SuiteID})$ order

$B \rightarrow A : C_RATCHET_OFFER = \text{ENC}(\text{Version}, \text{SuiteID}, \text{SessionID},$
 $\text{I2OSP}(\text{Epoch}', 8), \text{I2OSP}(\text{RecordSeq_next}, 8),$
 $\text{pk_r1}, \text{pk_r2}, \Sigma_B^R)$

Epoch' = Epoch + 1. RecordSeq_next is the next RecordSeq B expects under Epoch when it issues the offer; it is a freshness value bound into h_R, not the epoch boundary, which C_RATCHET_COMMIT fixes below.

A first refreshes its CSPRNG with fresh entropy under RR9, then encapsulates:

$(c_r1, z_r1) \leftarrow \text{ML-KEM-1024}.\text{Encaps}(\text{pk_r1})$

$(c_r2, z_r2) \leftarrow \text{HQC-5}.\text{Encaps}(\text{pk_r2})$

$(c_r3, z_r3) \leftarrow \text{McEliece8192128}.\text{Encaps}(\text{pk_McE}[B])$

$h_R = \text{SHA3-512}(\text{ENC}(\text{D_RATCHET_TRANSCRIPT}, \text{Version}, \text{SuiteID}, \text{ModeID},$
 $\text{ProfileID}, \text{SessionID}, \text{I2OSP}(\text{Epoch}', 8),$
 $\text{I2OSP}(\text{RecordSeq_next}, 8), \text{ID}[A], \text{ID}[B], \text{Context},$
 $\text{pk_r1}, \text{pk_r2}, \text{pk_McE}[B], c_r1, c_r2, c_r3))$

$K_R = \text{SHA3-512}(\text{D_RATCHET_MASTER} \parallel z_r1 \parallel z_r2 \parallel z_r3 \parallel h_R)$
 -- each z_ri has its leg's fixed length, as §5.5

The static Classic McEliece leg is re-encapsulated at every refresh. It can contribute a secret KEM output while its private key and the encapsulation randomness remain unavailable to the adversary, but it is not an ephemeral recovery guarantee. G6 requires at least one freshly generated ephemeral ML-KEM or HQC contribution to remain secure and unexposed, together with the other RR6/A6 conditions. The standalone classical-ROM combiner sketch motivates K_R; it does not establish the composed ratchet proof.

Before any recovery-sensitive randomness is sampled, B MUST incorporate fresh operating-system entropy before generating the offer keypairs, and A MUST do so after authenticating the offer and before any encapsulation or new-root sampling. The resulting state must be unpredictable to the prior state-compromise adversary; a successful API return alone is not evidence of new entropy. Failure aborts the refresh attempt under the phase rules of RR4. A then samples the new roots independently from that refreshed CSPRNG. Reseeding only after encapsulation is insufficient: predictable encapsulation coins can expose the fresh shared secrets and the KEKs protecting the new roots.

$R_A[\text{Epoch}', 0], R_B[\text{Epoch}', 0], R_C[\text{Epoch}', 0] \leftarrow \{0, 1\}^{512}$

$\text{KEK_A}^R = \text{SHA3-512}(\text{ENC}(\text{D_RATCHET_WRAP_A}, K_R, z_r1, h_R))$

$\text{KEK_B}^R = \text{SHA3-512}(\text{ENC}(\text{D_RATCHET_WRAP_B}, K_R, z_r2, h_R))$

$\text{KEK_C}^R = \text{SHA3-512}(\text{ENC}(\text{D_RATCHET_WRAP_C}, K_R, z_r3, h_R))$

$W_A^R = R_A[\text{Epoch}', 0] \text{ xor } \text{KEK_A}^R$

$W_B^R = R_B[\text{Epoch}', 0] \text{ xor } \text{KEK_B}^R$

$W_C^R = R_C[\text{Epoch}', 0] \text{ xor } \text{KEK_C}^R$

$K_RA = \text{HMAC-SHA3-512}(K_R, \text{ENC}(\text{D_RATCHET_AUTH}, h_R))$

$K_RC = \text{HMAC-SHA3-512}(K_R, \text{ENC}(\text{D_RATCHET_CONFIRM}, h_R))$

$K_AUTH[\text{Epoch}'] = \text{HMAC-SHA3-512}($
 $K_R, \text{ENC}(\text{D_RATCHET_EPOCH_AUTH}, \text{SessionID}, \text{I2OSP}(\text{Epoch}', 8), h_R))$

The epoch authentication key K_AUTH is therefore refreshed together with the three confidentiality roots; it is written K_AUTH[e] to keep it distinct from the leg-A record key K_A[e,r] of §7.11.1. The old K_AUTH is erased at B's atomic installation and at A's durable WAIT_CONFIRM transition before commit transmission, as RR4a/RR5a require. This extends healing to the record-authentication state; the long-term AND-signature keys remain unchanged and are assumed uncompromised by Requirement RR6/A6.

$M_COMMIT = \text{ENC}($
 $\text{Version}, \text{SuiteID}, \text{ModeID}, \text{ProfileID}, \text{SessionID},$
 $\text{I2OSP}(\text{Epoch}', 8), \text{I2OSP}(\text{RecordSeq_last}, 8),$

ID[A], ID[B], h_R,
c_r1, c_r2, c_r3, W_A^R, W_B^R, W_C^R)

T_R = KMAC256(K_RA, M_COMMIT, 512 bits, "GM512-RATCHET-AUTH")
 μ_COMMIT = SHA3-512(ENC(D_RATCHET_COMMIT, M_COMMIT, T_R))
 Σ_A^R = ENC($\sigma_1, \sigma_2, \sigma_3$) -- each over μ_COMMIT , in SIG(SuiteID) order

A $\rightarrow$ B : C_RATCHET_COMMIT = ENC(M_COMMIT, T_R, Σ_A^R)

RecordSeq_last = A's next unallocated RecordSeq under Epoch, one more than the last durably committed old-epoch record. Every committed old-epoch record has RecordSeq < RecordSeq_last and its exact bytes remain available for ordered transmission or retry from the outbox. This is the epoch boundary. Before transmitting the commit, A MUST durably enter WAIT_CONFIRM with its exact serialized bytes and candidate epoch state (RR4a). A MUST NOT create a new record under Epoch once that transaction commits. Only stored old-epoch bytes may still be sent under RR4a.

B verifies all commit signatures and its public transcript before exclusively claiming pending-offer secrets. Its next expected old-epoch sequence MUST equal RecordSeq_last; an earlier boundary is rejected, and a later boundary may be held within the bounded pending-offer lifetime. Only the owner decapsulates. After successful KEM, tag and unwrap processing, B prepares the serialized confirmation, erases all temporary KEM and derivation secrets, and atomically installs the three new roots, K_AUTH[Epoch'], h_E[Epoch'] = h_R, next sequence 0 and the public retransmission cache. The same transaction renders the prior epoch roots and K_AUTH unrecoverable and consumes the pending offer. No plaintext or success confirmation is released before durable installation. The full ownership and failure rules are in Section 7.11.5.

Tag_R = KMAC256(K_RC, ENC(D_RATCHET_CONFIRM, SessionID,
I2OSP(Epoch',8), h_R),
256 bits, "GM512-RATCHET-CONFIRM")

B $\rightarrow$ A : C_RATCHET_CONFIRM = ENC(Version, SuiteID, SessionID,
I2OSP(Epoch',8), Tag_R)

B confirms installation of the new ratchet state with Tag_R. A MUST NOT create or transmit a new-epoch record until Tag_R verifies. Once WAIT_CONFIRM is durable, A MUST NOT create further old-epoch records either; sending or retrying the exact bytes of already committed old-epoch outbox records is allowed so B can reach RecordSeq_last. This does not restore old keys or advance the old chain. The activation and retransmission rules follow.

Requirement RR4a. Activation and retransmission. Before emitting C_RATCHET_COMMIT, A MUST durably store its exact bytes, candidate roots, candidate K_AUTH, h_R, K_RC and a WAIT_CONFIRM state that blocks creation of every new application record; only transmission or byte-identical retransmission of already committed old-epoch outbox records is permitted. This transaction also erases the old sending roots and old K_AUTH: they cannot be resumed. A sends any already committed old-epoch outbox records in order and then the stored commit and waits at most Δ before retrying; any retry MUST be byte-identical and the overall deadline is 2Δ from the durable WAIT_CONFIRM transaction, without extension on send or restart. B MUST atomically cache the installed commit digest and exact serialized C_RATCHET_CONFIRM with installation. This public cache lasts until the first new-epoch record is accepted or 2Δ after installation, whichever comes first; K_RC is erased after the cached confirmation is formed. B checks this cache before looking for an unexpired pending offer. Within the cache window, a matching installed-commit digest causes the cached confirmation to be resent without decapsulation or installation; a different commit for that installed epoch is rejected. After expiry, old commits are ignored or rejected and never reinstalled. A activates the candidate epoch only after validating all confirmation fields and Tag_R; otherwise it stays blocked. At the overall deadline A terminates and erases candidate secrets, and MUST NOT resume the old epoch. B learns of a dead session through transport closure or configured session policy, not merely through an idle new epoch. Pending offer secrets have an absolute lifetime at most Δ from offer creation; claiming or retrying does not extend it.

Requirement RR3. A deployment enabling TPQRR MUST configure at least one automatic refresh trigger. The trigger MAY be based on record count, transmitted bytes, elapsed time, or a security event, and SHOULD combine a time threshold with a traffic threshold. A suspected session-state compromise MUST request an immediate refresh. The specification intentionally does not assign a universal numeric interval because the appropriate bandwidth/security trade-off is deployment-dependent.

Requirement RR4. Failures are phase-specific. Before A durably enters WAIT_CONFIRM, a failed refresh leaves its old epoch usable and erases only that attempt's candidate secrets. Once WAIT_CONFIRM is committed, A may only transmit stored old-epoch outbox bytes under RR4a until confirmation activates the candidate; timeout or unrecoverable local failure terminates the session. At B, invalid input before ownership changes no active state and does not consume another instance's pending offer. After an authenticated exclusive claim, a failed KEM, tag or installation attempt burns that pending offer and erases its owner-local secrets and candidates while retaining the old active epoch. Successful installation is never rolled back because of a failed confirmation transmission or a later invalid packet: B retains the installed epoch and retries the cached public confirmation. No transition may partially install roots or K_AUTH. If storage outcome is uncertain, process no further session traffic until recovery proves the committed state, otherwise terminate.

Requirement RR5. Epoch and RecordSeq are unsigned 64-bit values and MUST NOT wrap. Application records use 0 through $2^{64} - 2$; $2^{64} - 1$ is reserved as an exhausted next-sequence or epoch-boundary value and MUST NOT appear as an application RecordSeq. Before sequence exhaustion, refresh to the next epoch or terminate. At Epoch = $2^{64} - 1$, no further refresh is possible and the session MUST terminate before another epoch is needed. Reject an epoch lower than the highest committed epoch, except for exact installed-commit confirmation retries allowed by RR4a. A new refresh must propose precisely the next epoch. Overflow is detected before arithmetic, sampling or state mutation and returns E_STATE.

Requirement RR5a. At B, installation atomically erases the old epoch roots and K_AUTH. At A, the WAIT_CONFIRM transaction erases them before the first commit transmission. Between record and refresh operations, the active protocol secrets are the three root-chain states and K_AUTH. A retains candidate roots, candidate K_AUTH and K_RC only while awaiting confirmation; bounded in-progress operations may hold the temporary material listed in Section 7.11.5. K_M, K_R, K_RA, all KEKs, KEM shared secrets, temporary key schedules and consumed record keys MUST be erased immediately after their last use on both success and failure. B's retransmission cache contains only the installed-commit digest, h_R, epoch, expiry metadata and serialized confirmation; it contains no K_RC or other secret.

7.11.3 Post-compromise recovery claim and limits

TPQRR targets recovery after session-state compromise, not permanent endpoint takeover. The adversary may learn current roots, derived keys, pending session secrets and CSPRNG state through StateReveal while long-term authentication keys remain protected. Recovery begins only after it loses access, both endpoints restore unpredictable randomness before recovery-sensitive sampling, and at least one fresh ephemeral KEM contribution remains secure and unexposed. After successful commit installation at B and confirmation at A, the new roots are intended to be computationally independent of the exposed old state. This is the G6 target under A6; no composed ratcheting or QROM proof is supplied.

Requirement RR6. TPQRR makes no recovery claim while the adversary remains continuously present on an endpoint, after compromise of the long-term authentication keys, if all fresh ratchet KEM contributions are exposed or broken, or if either endpoint's recovery-sensitive randomness remains predictable after compromise. An entropy refresh after generating the affected keys or coins cannot retroactively protect them. A symmetric per-record step alone is not post-compromise recovery because knowledge of $R_X[e,r]$ permits computation of later roots in the same epoch.

Requirement RR7. The old epoch roots MUST NOT be mixed into K_R or into the newly sampled roots. They may be included only as public commitments or authenticated context. This deliberate break in the key lineage is what allows a successful refresh to escape from a fully compromised prior root tuple.

Requirement RR8. TPQRR does not raise the claimed security level above §3.1. Its benefit is temporal: per-record root erasure protects the past, while successful injection of fresh post-quantum and local entropy can restore protection of the future. The 3x512 structure remains a redundancy architecture, not a 1,536-bit security claim.

Requirement RR9. Resource and recovery discipline. Implementations MUST bound pending offers, authenticated commits held for missing records, reorder buffers, durable outboxes and confirmation caches by count, bytes and time. At most one pending or claimed offer and one outstanding commit may exist per session. No expiry or retry extends an original deadline. Exhausted outbox capacity blocks new records and triggers refresh or session closure; it MUST NOT cause a committed record to be silently discarded. Outbox records may be discarded after authenticated evidence that B accepted the relevant prefix: C_CONFIRM covers the initial record, and a successful refresh confirmation covers all records below its committed RecordSeq_last. Fresh entropy MUST precede B's offer key generation and A's encapsulation, root sampling and randomized signing. Session transitions use one owner or equivalent atomic serialization, and crash recovery MUST restore WAIT_CONFIRM and its original deadline before any send can occur.

7.11.4 Capability change introduced by TPQRR

Property	Base GM-3X512-PQS	With TPQRR
Rekey during a long session	No - a new session is required	Yes - in-session PQ epoch refresh, no transport reconnect
Per-message / periodic key evolution	No	Yes - per-record triple-root evolution + periodic PQ refresh
Healing without re-establishing the connection	No	Targeted under A6 after a confirmed fresh PQ refresh
Post-compromise recovery	No explicit PCS claim	Targeted for future records under RR/A6; proof open
Protection of earlier records after later state compromise	By session forward secrecy	Also within an epoch through one-way root erasure

The result is a two-speed ratchet: inexpensive symmetric evolution occurs after every record, while the larger post-quantum exchange is amortized across many records and is invoked periodically or on demand. This preserves the parallel 3x512 generator design while adding a recovery mechanism analogous in purpose, but not in construction, to post-quantum secure-messaging ratchets.

7.11.5 TPQRR state machines

These state machines are normative for TPQRR. They replace the base payload-processing and cleanup transitions; Section 7.5b still creates the initial signed Mode I hello. Base sender step 11, receiver steps 17 to 19 and R2a are not inherited as TPQRR cleanup. The active epoch authentication key and successor roots **MUST** survive successful record processing. A failed record or an uncommitted refresh attempt never advances active state; failures after a durable transaction obey RR4 and cannot roll it back. All state changes are serialized per session. The separate initial sender and receiver algorithms below establish epoch 0 before the later-record and refresh algorithms apply.

Initial sender for epoch 0 record 0

1. Perform base sender steps 1 to 4 in Mode I with the TPQRR Version, FULL suite and negotiated root ProfileID. Resolve the peer, verify the signed hello, encapsulate and construct h_T .
2. Derive K_M , K_A and base K_C ; derive export K_X only if requested before derivation. Sample $R_A/R_B/R_C[0,0]$, derive the three KEKs and form the wraps as Section 7.8 specifies. Erase z_i immediately after KEK derivation, and each KEK after wrapping. Erase K_M after all requested derivations are complete.
3. Set $h_E[0] = h_T$ and $K_AUTH[0] = K_A$. Derive record keys and candidate successor roots $R_X[0,1]$ under Section 7.11.1.
4. Draw N ; encrypt record 0 using its record keys. Build M_AUTH_R , T , M_SIG_R and every required signature, then serialize C_GM_R . No record bytes have been transmitted yet.
5. Atomically persist exact C_GM_R in the durable outbox, the three $R_X[0,1]$, $K_AUTH[0]$, h_T , next sequence 1, base K_C , any pending export K_X and an INITIAL_WAIT_CONFIRM state with the D3 deadline. Erase $R_X[0,0]$, K_M , K_E , all z_i and KEKs, record keys and temporary schedules. A failed transaction emits nothing.
6. Transmit only the stored C_GM_R . Send no later record until the base $C_CONFIRM$ fields and Tag_confirm have verified under K_C . Invalid network input does not extend the confirmation deadline.
7. On valid confirmation, atomically mark the session ACTIVE, erase K_C , expose requested export K_X and erase its protocol copy. Retain $R_X[0,1]$ and $K_AUTH[0]$; the initial outbox may be retired. On timeout, terminate and erase all session secrets. Delivery may already have occurred at B and is reported as unknown.

8. After a crash, recover the exact durable INITIAL_WAIT_CONFIRM state and original deadline before processing or sending. If recovery cannot establish that state, terminate; never recreate a different record 0 or reuse its roots.

Initial receiver for epoch 0 record 0

1. Parse C_GM_R under W1. Require the TPQRR Version, Mode I, FULL suite, a root ProfileID, Epoch = 0, RecordSeq = 0, and all three ciphertexts and wraps present. Enforce exact field sizes.
2. Check peer identities, timestamp and policy. Compute the base handshake ReplayKey and record ReplayKey separately. Reject accepted replay markers; obtain only public values from the unexpired HS[base ReplayKey]. Context, ProfileID and Timestamp must match that hello. There is no active epoch state yet.
3. Reconstruct h_T and set h_E[0] = h_T. Reconstruct M_SIG_R and verify every signature before any claim or decapsulation.
4. Exclusively claim HS[base ReplayKey] as in H3a, moving sk_e1 and sk_e2 to the bounded owner. Decapsulate z1/z2/z3; enforce 32-byte results and immediately erase the ephemeral secret keys.
5. Derive K_M, base K_A and K_C, and requested export K_X. Set candidate K_AUTH[0] = K_A. Verify T over M_AUTH_R.
6. After signature and tag verification, derive KEKs and erase z_i and K_M immediately. Unwrap R_X[0,0], erase the KEKs, then derive record-0 keys and candidate R_X[0,1]. Erase unused base K_E.
7. Decrypt into a protected pending buffer and prepare serialized C_CONFIRM using the base K_C. Erase K_C and record-key schedules. No plaintext or successful protocol output is released yet.
8. In one atomic durable transition, recheck absence of both replay markers and active state; install R_X[0,1], K_AUTH[0], h_T and next sequence 1; insert both replay markers; finish the claimed handshake; make R_X[0,0] unrecoverable. There is no separate base step-17 replay insertion. A competing or failed transaction releases nothing and erases this owner's candidate secrets.
9. Release the authenticated plaintext and any requested export K_X, erase pending plaintext and the protocol export copy, and send the serialized C_CONFIRM. Retain only the active successor roots and K_AUTH[0] as session secrets. A failed send cannot undo step 8.
10. Any pre-install failure erases owner-local material and consumes a handshake already claimed at step 4; it cannot erase another owner's state. A crash after acceptance may lose application delivery; exact-once delivery requires Section 10.3 integration.

Record receiver.

1. Parse C_GM_R under W1; require the TPQRR Version and Mode I. Route epoch 0 record 0 to the dedicated initial receiver. Every later record has all six W?/c? fields empty (E3).
2. Require Version, SuiteID, ModeID, ProfileID, SessionID, ID[A], ID[B] and Context to match the established session. Check the timestamp window, sequence range and exact field lengths.
3. Under the session's state owner, require Epoch = active epoch and RecordSeq = next expected sequence. A lower accepted sequence is E_REPLAY; another unexpected value is E_STATE. Compute the record replay key and reject an existing marker.
4. Reconstruct M_SIG_R and verify all signatures. Recompute T under K_AUTH[Epoch] and compare in constant time. Any failure erases this attempt's temporary material and leaves active state intact.
5. Derive the three record keys and candidate successor roots. Decrypt to a protected pending buffer; release no plaintext.
6. Atomically recheck the epoch, sequence, owner generation and replay marker, then commit all successor roots, next sequence r+1 and the replay marker; erase the predecessor roots. If this is the first new-epoch record, clear the public RR4a cache in the same transaction. No separate replay insertion occurs.

7. After commit, release plaintext and erase its pending buffer, record keys and expanded schedules. Retain successor roots and $K_AUTH[Epoch]$. A local delivery failure cannot roll back state.
8. If the transaction fails before commit, erase candidates and pending plaintext and retain old roots. If commit outcome is uncertain, stop this session until recovery establishes the durable state or terminates it. Never release plaintext twice on the basis of replayed protocol input.

Record sender.

1. Acquire exclusive session send ownership. Require ACTIVE, no outstanding refresh commit and a completed initial handshake. Let e be the active epoch and r the next send sequence. Check $r \leq 2^{64} - 2$ and reserve bounded outbox capacity under RR9.
2. Draw fresh N and set Timestamp. Derive record keys and temporary $R_X[e, r+1]$ without erasing the committed $R_X[e, r]$.
3. Encrypt, compute T , produce every signature, and serialize the complete C_GM_R . RNG, signing or allocation failure here emits nothing, erases candidates and retains the committed state.
4. Atomically store exact serialized C_GM_R in the outbox together with all successor roots and next sequence $r+1$, and make the predecessor roots unrecoverable. Recheck owner/state generation. On an uncertain transaction outcome, recover or terminate.
5. Erase record keys, candidate copies and expanded schedules. Emit the persisted bytes in sequence order. A send failure retains the outbox; it never rolls back or regenerates sequence r .
6. Retry only byte-identical stored records within timestamp and resource limits. Recover a missing sequence from this outbox; if it is unavailable or has become stale, terminate the session. Retire accepted prefixes only as RR9 allows. An upper layer retrying an application operation needs its own operation ID.

Refresh, B side - offer.

1. Under session ownership, require ACTIVE receive state and no pending/claimed offer. Check epoch increment for overflow. Obtain fresh OS entropy and restore unpredictable RNG state BEFORE generating ML-KEM-1024 and HQC-5 ephemeral keypairs. Entropy or key-generation failure leaves the active epoch intact.
2. Set $Epoch' = Epoch + 1$ and $RecordSeq_next =$ current next receive sequence. Build and sign M_OFFER with every suite signature.
3. Atomically store the exact $C_RATCHET_OFFER$, its public inputs, ephemeral secrets, owner generation and absolute ExpiresAt no later than $now + \Delta$. Never overwrite an unexpired offer.
4. Send only the stored offer. A repeated send uses identical bytes and does not extend expiry. On expiry or abandonment, erase pending ephemeral keys and remove this offer; keep active roots. Claimed secrets follow the same absolute deadline.

Refresh, A side - on offer.

1. Parse under W1. Reconstruct the complete M_OFFER from the bound session and supplied fields; require exact Version, SuiteID, SessionID, next epoch, parameter lengths and identities. Verify every offer signature before any encapsulation.
2. Acquire send ownership and recheck ACTIVE, no outstanding commit, the expected next epoch and $RecordSeq_next \leq$ next send sequence. Suspend concurrent record creation while fixing the boundary.
3. Obtain fresh OS entropy and restore unpredictable RNG state BEFORE all encapsulations and new-root sampling. Encapsulate $c_r1/c_r2/c_r3$, enforce 32-byte secrets and erase KEM coins.

4. Compute h_R and K_R . Independently sample the three new roots. Derive KEKs, wraps, K_{RA} , K_{RC} and candidate $K_{AUTH}[\text{Epoch}]$. Build M_{COMMIT} with RecordSeq_last = next send sequence, then T_R , all signatures and exact $C_{RATCHET_COMMIT}$ bytes. Erase z_{ri} , KEKs, K_R and K_{RA} immediately after last use.
5. Atomically enter $WAIT_CONFIRM$ with the exact commit, candidate roots and K_{AUTH} , h_R , K_{RC} and immutable deadlines. Erase old send roots and old K_{AUTH} in this transaction. All old records below RecordSeq_last are already committed in the outbox. A pre-commit failure keeps the old epoch usable; an uncertain commit outcome permits no further send until recovery.
6. Send stored old-epoch outbox records in order, then the stored commit. No newly created old-epoch or new-epoch record may be sent while $WAIT_CONFIRM$. Retry the commit only as RR4a allows.
7. Check all $C_{RATCHET_CONFIRM}$ fields and compare Tag_R in constant time under retained K_{RC} . Invalid packets do not activate state or extend the deadline. On valid confirmation, atomically install candidate roots/ K_{AUTH} and $h_E = h_R$, next sequence 0, ACTIVE; erase K_{RC} and candidate copies, and retire the old outbox prefix.
8. On the overall deadline or unrecoverable failure, terminate and erase all candidate secrets. Never resume the old epoch, even if the peer's installation or confirmation delivery is unknown.

Refresh, B side - on commit.

1. Parse within $W1$ and validate session, field lengths and epoch. FIRST check the installed-commit cache, independently of pending offer existence or expiry. For the cached installed epoch: matching digest within its window -> resend cached $C_{RATCHET_CONFIRM}$ and stop; a different digest -> E_REPLAY . An expired/old commit is never reinstalled.
2. For a new commit, read public inputs from an unexpired pending offer for exactly active epoch + 1. Recompute h_R and compare it to the transmitted value. Reconstruct M_{COMMIT} , T_R binding and every signature; reject failures before claiming any secret.
3. Require next receive sequence = RecordSeq_last . A boundary below it is rejected; a boundary above it may be held, bounded by $W1$, RR9 and the original offer expiry, while missing records arrive. No decapsulation or secret claim occurs while holding a commit.
4. Atomically recheck epoch, boundary, offer expiry and ownership; claim the offer and MOVE sk_{r1}/sk_{r2} to one owner. A competing instance cannot access or erase those keys. The owner remains subject to the original expiry and a bounded execution deadline.
5. Decapsulate all legs, enforce 32-byte outputs and erase ephemeral private keys immediately. Derive K_R , KEKs, K_{RA} , K_{RC} and the candidate epoch K_{AUTH} ; verify T_R in constant time. On failure, burn the claimed offer and erase candidates; keep old active state.
6. Unwrap the three candidate roots only after verification. Compute Tag_R and serialize $C_{RATCHET_CONFIRM}$. Erase every z_{ri} , KEK, K_R , K_{RA} and K_{RC} ; retain only candidate roots and K_{AUTH} .
7. In one atomic durable transaction, recheck ownership and the old epoch boundary, install new roots/ K_{AUTH} and $h_E = h_R$ with next sequence 0, consume the offer, erase old roots/ K_{AUTH} , and store the installed-commit digest and exact public confirmation cache. A pre-commit failure erases candidates and preserves old active state; an uncertain outcome stops processing until recovery.
8. Send cached $C_{RATCHET_CONFIRM}$. A send failure retains the installed state and cache. It cannot roll back the epoch or resurrect the offer. Expire the cache under RR4a without erasing active roots.

The initial, record, offer, commit and confirmation transitions above define distinct lifetimes. Losing concurrent instances never modify another owner's state. Expiry erases only the secrets belonging to the expired phase; invalid network input never destroys a successfully committed active epoch. All persisted candidate and active secrets are subject to K2, and all deadlines are restored without extension after restart. Section 15 requires executable crash, retry, concurrency and recovery tests before these state machines can support an implementation-assurance claim.

8. Triple Signature

$$M_SIG = ENC(\text{Version, SuiteID, ModeID, ProfileID, SessionID, Timestamp,} \\ W_A?, W_B?, W_C?, \\ ID[A], ID[B], h_T, c1, c2, c3?, c4?, N, AAD, C, T)$$

$$D_SIG = "GM512-SIG" \\ \mu = SHA3-512(ENC(D_SIG, M_SIG))$$

$$\Sigma_GM = (\sigma_1, \sigma_2, \sigma_3?) \quad \text{where } \sigma_i = SIG_i.Sign(sk_i, \mu) \\ V_GM = \wedge \text{ over } i \in SIG(\text{SuiteID}) \text{ of } v_i$$

Requirement SIG3. The signed input is the exact 64-byte protocol digest (mu, mu_HELLO, mu_PRE, mu_OFFER or mu_COMMIT as applicable), passed as the message to the public pure ML-DSA-87 and SLH-DSA-SHAKE-256s Sign/Verify interfaces with the empty context string. HashML-DSA, HashSLH-DSA, direct internal signing without the standard message framing, and ASCII hexadecimal representations of the digest are not equivalent and MUST NOT be substituted. FAEST-256s uses FAEST.Sign/Verify of version 3.0, with msg equal to the same digest and the pinned FAEST-256s parameter set. Use hedged/randomized signing with fresh randomness for all three schemes and abort on RNG or signing failure. Verification accepts the valid canonical signatures of the pinned scheme; no extra application prehash or context is added by an implementation. Deterministic conformance vectors supply explicit test randomness to the standard interfaces and never permit deterministic production fallback.

8.1 Why AND-combination is correct

Forging Σ_GM requires producing valid forgeries for all required components. The security strength of the combination is therefore that of the **strongest surviving** component; equivalently, the forgery advantage is bounded by the minimum component forgery advantage. Formally, an adversary against Σ_GM yields an adversary against each SIG_i , so $Adv^{EUF}(\Sigma_GM) \leq \min_i Adv^{EUF}(SIG_i)$.

Binding SuiteID into M_SIG prevents algorithm-stripping: an adversary cannot present a two-signature object as a valid three-signature object, because the suite identifier under which each signature was produced is itself signed.

8.2 Availability is the trade

Security is the maximum; **availability requires every required component**. A single library defect, deprecation, or parameter change in any one scheme halts all traffic. This is a real operational cost and the specification must acknowledge it rather than treat redundancy as free.

Requirement SIG1. $t = n$ is REQUIRED by this version of GM-512: a receiver MUST compute the AND over every signature in $SIG(\text{SuiteID})$ and MUST abort if any component fails. Threshold acceptance is reserved for a future profile and is not negotiable here.

Threshold bound. Should a future profile admit a threshold below n , the bound is not simply the minimum over the accepted subset: an adversary targets whichever acceptable subset is weakest, giving $Adv_threshold \leq C(n,t) \cdot \max \text{ over } |S| = t \text{ of } (\min \text{ over } i \text{ in } S \text{ of } Adv_i) + Adv_coll$. That is recorded in §15 as future work. Theorem 2 assumes $t = n$ throughout.

Requirement SIG2. Record component-signature failures in bounded, rate-limited local security telemetry. Aggregate repeated failures while preserving counters and enough diagnostic evidence to identify a faulty component. Logging MUST NOT create unbounded memory or disk use, expose keys or plaintext, or reveal detailed failure causes to the peer.

8.3 Size

The suite AND-signature is the dominant wire cost. In GM512-FULL, SLH-DSA-SHAKE-256s alone contributes 29,792 bytes and FAEST-256s 16,626 bytes.

On reduced per-message authentication. Signing the bundle once and caching it at the peer would let per-message signatures use ML-DSA-87 alone, cutting roughly 51 KB to about 4.6 KB. That is deliberately not specified here: it would make per-message sender authentication lattice-only, which contradicts the multi-assumption premise of the design, and a conforming receiver executes the AND of every component verification with no negotiated alternative. Adding it properly would require an AuthMode value advertised in the signed bundle and bound into T_KEM, M_SIG and M_AUTH so it cannot be downgraded. It is left as a possible future bandwidth profile.

9. GM-512-BC and GM-512-BC-N - The Optional Block Ciphers (X-GM512, X-TRIPLE, X-TRIPLE-3R, X-TRIPLE-3RN, X-TRIPLE-3RH)

Status: EXPERIMENTAL. This cipher has proved single-trail bounds against differential and linear cryptanalysis and no third-party analysis against any other attack class. It MUST NOT be used to protect production data. It is specified here so that it can be analyzed. See §9.7.

9.1 Parameters

Parameter	Value
Block size	512 bits
Key size	512 bits
State	8×8 matrix of bytes over $\text{GF}(2^8)$
Rounds	32
Round keys	GM-512-BC: 33 domain-separated HMAC-SHA3-512 outputs (§9.5). GM-512-BC-N: 33 values generated recursively by the round function under fixed constants (§9.5a)

The state S is a byte matrix $b[r][c]$, $0 \leq r, c \leq 7$, serialized **column-major**: byte index i corresponds to $b[i \bmod 8][i \div 8]$.

Standardized precedent. A 512-bit block with a 512-bit key is outside the NIST-approved set, where AES stops at a 128-bit block and a 256-bit key [1], but it is not without standardized precedent. Kalyna-512 is a Ukrainian national standard, DSTU 7624:2014 [26], with the same 8×8 geometry and wide-trail structure at 18 rounds. The Whirlpool cipher W , a 512-bit block and key, is specified inside ISO/IEC 10118-3 [27], though that standard approves Whirlpool as a hash function rather than W as a general-purpose cipher. Threefish-512 was published as the core of the SHA-3 finalist Skein but was never standardized. GM-512-BC shares the broad 8×8 wide-trail SPN tradition of 512-bit constructions such as Kalyna and Whirlpool, but is a variant of neither: the S-box, the MDS layer and the key schedule all differ. §9.3 compares round counts against all three.

The field is $\text{GF}(2^8)$ with the AES irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ (0x11B).

9.2 Round function

$\text{Round_r}(S) = \text{MixColumns}(\text{ShiftRows}(\text{SubBytes}(S \oplus \text{RK_r})))$

Encryption:

```
S ← P
for r = 0 ... 31: S ← Round_r(S)
C ← S ⊕ RK_32
```

SubBytes. The AES S-box applied to each of the 64 bytes. [1] Verified properties (computed, see §13.4): it is a bijection; maximum DDT entry is 4, giving maximum differential probability 2^{-6} ; maximum LAT bias is 16/128, giving maximum linear correlation 2^{-3} and maximum squared correlation (linear potential) 2^{-6} .

ShiftRows. Row r is rotated left by r positions:

$b'[r][c] = b[r][(c + r) \bmod 8]$

MixColumns. Each column is multiplied by the circulant MDS matrix $M = \text{cir}(02, 02, 03, 04, 05, 03, 05, 07)$, i.e. $M[i][j] = c[(j - i) \bmod 8]$:

```
02 02 03 04 05 03 05 07
07 02 02 03 04 05 03 05
05 07 02 02 03 04 05 03
03 05 07 02 02 03 04 05
05 03 05 07 02 02 03 04
04 05 03 05 07 02 02 03
03 04 05 03 05 07 02 02
02 03 04 05 03 05 07 02
```

Inverse matrix (for decryption):

```
13 5a 54 72 50 df 45 53
53 13 5a 54 72 50 df 45
45 53 13 5a 54 72 50 df
df 45 53 13 5a 54 72 50
50 df 45 53 13 5a 54 72
72 50 df 45 53 13 5a 54
54 72 50 df 45 53 13 5a
5a 54 72 50 df 45 53 13
```

Requirement BC1. M is MDS over $GF(2^8)$ with polynomial $0x11B$. This was verified computationally by checking that all 12,869 square submatrices are nonsingular. Its branch number is therefore 9, the optimum for an 8×8 matrix. A weight-one consistency check gives output weight 8, as expected for an MDS matrix; it is a sanity check, not an independent proof, since a higher-weight input could in principle yield a smaller total weight and only the minor test rules that out.

This matrix is taken from the Grøstl SHA-3 finalist [20] and has been publicly analyzed. Note that the Whirlpool circulant $cir(01,01,04,01,08,05,02,09)$ is **not** MDS under the AES polynomial - it requires $0x11D$ - which is why it is not used here.

9.3 Single-trail bound against differential and linear cryptanalysis

Lemma 1 (wide trail). In a key-alternating SPN with an $n \times n$ byte state, a ShiftRows layer that maps the bytes of any single column to n distinct columns, and a MixColumns layer with branch number B , every four-round differential or linear trail has at least B^2 active S-boxes.

(Daemen and Rijmen, “The Design of Rijndael”, the wide trail strategy. The GM-512-BC ShiftRows satisfies the hypothesis: row r shifts by r , so the eight bytes of one column land in eight distinct columns.)

With $B = 9$:

Rounds	Min. active S-boxes	Max. differential probability	Max. linear correlation
4	81	2^{-486}	2^{-243}
8	162	2^{-972}	2^{-486}
32	648	2^{-3888}	2^{-1944}

A differential attack requires roughly $1/DP$ chosen pairs. With a 512-bit block, the entire codebook is 2^{512} , so any characteristic with probability below 2^{-512} is unusable. **Eight rounds already exceed this bound by a factor of 2^{460} .** Linear cryptanalysis requires roughly $1/corr^2$ known plaintexts; eight rounds give squared correlation at most 2^{-972} , so the known-plaintext requirement is at least 2^{972} , likewise unusable.

Round-count justification. At eight rounds the single-trail bounds cross the full-codebook threshold. This does not establish resistance to differential or linear cryptanalysis as complete attack classes: trails may cluster into differentials that an active-S-box count does not bound. Thirty-two rounds provide a $4\times$ round-count margin over the eight-round single-trail threshold. For comparison:

Cipher	Rounds	Rounds needed for the trail bound	Margin
AES-128	10	~4	2.5×
Serpent	32	~9	3.6×
Whirlpool W (512-bit)	10	~4	2.5×
Kalyna-512	18	~4	4.5×
GM-512-BC	32	8	4×

Thirty-two rounds is defensible and is in line with Serpent’s design philosophy of a deliberately large margin. It is not arbitrary, and - importantly - it is no longer load-bearing: the single-trail margin argument comes from the bound, and the round count is margin on top of it.

9.4 Measured diffusion

Author-reported statistical avalanche results, using 400 random trials and a mean Hamming-distance criterion of 256 plus or minus 3.5 sigma over 512 output bits. These measurements require the test harness, seeds and raw output before independent reproduction; they do not establish resistance to cryptanalysis:

Round	1	2	3	4	5	6
Mean bits changed	32.8	258.3	256.6	257.4	255.8	255.6
Trials in full-mix band	0%	100%	100%	100%	100%	100%

The MDS layer diffuses one active byte to all eight bytes of its column in a single round, and ShiftRows spreads those across all eight columns in the next.

Avalanche is a necessary but not sufficient property.

9.5 Round-key schedule

$RK_r = \text{HMAC-SHA3-512}(K, \text{ENC}(D_RK, \text{I2OSP}(r, 2)))$ for $r = 0 \dots 32$

Thirty-three domain-separated pseudorandom 512-bit round keys, each an HMAC-SHA3-512 output under the same cipher key K , 2,112 bytes total, fitting comfortably in L1 cache. A schedule producing 32,832 bytes would exceed a typical 32 KB L1 data cache and force the entire round-key table through L1 on every block.

The round-key schedule is a genuine strength of this design and is worth stating explicitly, with the caveat that every round key is a deterministic function of K_E and the set is therefore pseudorandom rather than independent:

- **Slide attacks are implausible.** They require self-similar rounds, and the round keys here are domain-separated pseudorandom HMAC-SHA3-512 outputs. This is a strong structural argument, not an impossibility proof.
- **Related-key attacks are implausible in the standard sense.** No simple algebraic relation among round keys is known under the HMAC-SHA3-512 PRF assumption. No general related-key proof is claimed.
- **The key-alternating model applies cleanly.** This is the setting in which provable bounds are most tractable.

- **Key recovery from round-key material.** Recovering K from round-key material is assumed hard under the key-recovery and PRF security of HMAC-SHA3-512. No formal reduction is claimed; this is not the same statement as generic SHA3-512 preimage resistance.

9.5a Native round-key schedule (GM-512-BC-N)

GM-512-BC-N is the cipher of §9.2 with round keys generated by the cipher’s own round function under fixed public constants, instead of by HMAC-SHA3-512. It exists so that a generator can be specified with no dependency on Keccak. GM-512-BC and GM-512-BC-N are different keyed permutations and **MUST NOT** be treated as interchangeable; only the round function is shared.

K_0 = the 512-bit key, as an 8×8 byte state
 $RK_r = K_r$ for $r = 0 \dots 32$
 $K_{r+1} = \text{MixColumns}(\text{ShiftRows}(\text{SubBytes}(K_r \oplus RC_r)))$ for $r = 0 \dots 31$
 $RC_r[0][0] = x^r$ in $GF(2^8)$ mod $0x11B$
 $RC_r[0][c] = S[(8r + c) \bmod 256]$ for $c = 1 \dots 7$
 $RC_r[i][c] = 0$ for $i \geq 1$

Requirement BC2. $RC_0 \dots RC_{31}$ **MUST** be pairwise distinct. The multiplicative order of x modulo $0x11B$ is 51, so x^r separates them across that range, and an implementation **MUST** verify this at build time. RK_0 is the key itself and is consumed by the AddRoundKey of round 0; it is not a separate whitening step.

Caveat. A schedule that reuses the round function is the structure in which related-key attacks have historically been found. Whirlpool’s schedule of this shape has absorbed twenty-five years of public analysis; this one has none. The §9.5 argument about round keys being HMAC-SHA3-512 outputs does not carry over, and §15 records the differential argument this schedule still needs.

9.6 Performance

Reported primitive benchmarks on x86-64 with GCC 13, -O3 -march=native; throughput is in MiB/s (2^{20} bytes). These are author-reported results retained for comparison. The supplied paper does not include the source commit, precise processor, benchmark harness and raw measurements needed for independent throughput reproduction. In particular, the T-table result is not a conforming constant-time implementation result.

Implementation	Throughput	Cycles/byte @ 3 GHz
GM-512-BC, T-table (16 KB tables)	47.7 MiB/s	60.0
GM-512-BC, portable (no tables)	0.50 MiB/s	5,702
SHAKE256 (OpenSSL, comparison only)	434 MiB/s	~6.6
AES-256-CTR (AES-NI, comparison only)	6,470 MiB/s	~0.44
AES-256-GCM (AES-NI)	12,260 MiB/s	~0.23

The table suggests substantial primitive cost relative to hardware-accelerated AES-256-CTR and SHAKE256 on the reported platform. Those ratios do not measure a complete GM session and must not be presented as secure transport throughput. The specified GM round function performs 2,048 S-box evaluations per 512-bit output, compared with 896 across four AES-256 blocks; operation counts alone do not predict wall-clock performance.

Requirement X1. Secret-indexed T-tables are prohibited for every GM confidentiality-generator implementation, including experimental evaluation that claims constant-time conformance. A bitsliced, vector-permute or otherwise verified constant-time implementation is required under X5. A table implementation may be used only as a

functional reference or explicitly labeled nonconforming benchmark; its throughput is not evidence of deployable secure performance.

9.7 What is proved, what is not

Proved (given Lemma 1 and the verified MDS property): - Single-trail bounds against classical differential cryptanalysis - Single-trail bounds against classical linear cryptanalysis. These bound the probability of any single differential or linear trail; they do not bound differentials with trail clustering, and they are not resistance proofs against those attack families as a whole. The domain-separated pseudorandom round-key schedule structurally discourages conventional slide and related-key attacks; no resistance proof is claimed

Not analyzed. No claim is made regarding: - Integral and division-property attacks (the relevant class for AES-like designs; published integral distinguishers on comparable 512-bit permutations reach roughly 6–8 rounds, but this has not been evaluated for GM-512-BC) - Rebound attacks and their variants (developed by Mendel, Rechberger, Schläpfer and Thomsen against AES-like permutations; the most effective known family against Whirlpool- and Grøstl-style constructions) - Boomerang and rectangle attacks - Meet-in-the-middle and biclique attacks - Impossible-differential attacks - Algebraic and Gröbner-basis attacks - Invariant subspace and nonlinear invariant attacks - Quantum-specific attacks, including Simon-based and quantum slide attacks - Any form of side-channel resistance

The honest summary: GM-512-BC has proved single-trail bounds for differential and linear characteristics and no evaluation against roughly eight other attack families. The absence of a known attack after a short evaluation is a statement about effort spent, not about the cipher. AES has absorbed 25 years of adversarial attention from the entire cryptographic community; that accumulated scrutiny is the asset, and no round count substitutes for it.

Requirement X2. the experimental X- profiles MUST NOT be enabled by default. An implementation offering it MUST require an explicit configuration flag and MUST emit a warning naming its experimental status.

10. Wire Format and Sizes

10.1 Transmitted object

```
C_GM = ENC( Version, SuiteID, ModeID, ProfileID, SessionID, Timestamp,
            Context,
            OTIndex?,           -- Mode A with a one-time prekey
            BundleID?,          -- Mode A only
            W_A?, W_B?, W_C?,    -- root profiles: 3R, 3RN, 3RH
            ID[A], ID[B],
            c1, c2, c3?, c4?,
            N, AAD, C, T,
            σ1, σ2, σ3? )
```

TPQRR record variant (only when Version = "GM512-TPQRR-v0005"; Mode I only):

```
C_GM_R = ENC( Version, SuiteID, ModeID, ProfileID, SessionID,
              Epoch, RecordSeq, Timestamp, Context,
              W_A?, W_B?, W_C?, ID[A], ID[B],
              c1?, c2?, c3?, N, AAD, C, T,
              σ1, σ2, σ3? )
```

Epoch and RecordSeq are each 8-byte unsigned integers. The wrapped roots and the KEM ciphertexts are present only in the initial object (Epoch = 0, RecordSeq = 0) and are FIELD("") in every later record (RR1a). Refresh material for later epochs travels only in C_RATCHET_COMMIT (§7.11.2).

10.2 Sizes (bytes, approximate)

Element	CORE	FULL
Identity bundle (published once)		
ML-DSA-87 public key	2,592	2,592
SLH-DSA-SHAKE-256s public key	64	64
FAEST-256s public key	—	48
Classic McEliece 8192128 public key	—	1,357,824
Identity total	~2,656	~1,360,528
Prekey bundle (rotated)		
ML-KEM-1024 semi-static pk	1,568	1,568
HQC-5 encapsulation key	~7,237	~7,237
Suite AND-signature over bundle	~34,419	~51,045
Per payload in Mode A with OT and a non-root profile		
ML-KEM-1024 ciphertexts (semi + OT)	3,136	3,136
HQC-5 ciphertext	~14,421	~14,421
Classic McEliece ciphertext	—	208
Framing: header, IDs, nonce, tag	474	474
ML-DSA-87 signature	4,627	4,627
SLH-DSA-SHAKE-256s signature	29,792	29,792
FAEST-256s signature	—	16,626
Message total	52,450 bytes	69,284 bytes

Framing is the C_GM overhead outside the KEM ciphertexts, the payload and the signatures: 306 bytes of content (Version, SuiteID, ModeID, ProfileID, SessionID, Timestamp, both identities, the 32-byte nonce and the 64-byte tag) plus a 4-byte FIELD prefix on every one of the 25 encoded fields, since Requirement E3 keeps absent optional fields present as FIELD(""). That gives 406 bytes, plus |Context| and |AAD|, which are empty in this table. Mode A adds 68 for BundleID and OTIndex, giving 474; the root profiles X-TRIPLE-3R, X-TRIPLE-3RN and X-TRIPLE-3RH each add a further 192 for the three wrapped roots. The column below is Mode A with a one-time prekey, which is the larger case; Mode I carries one ML-KEM ciphertext rather than two and no BundleID, so a CORE Mode I object is about 1.6 KB smaller at roughly 50,814 bytes.

TPQRR sizes and SuiteID lengths are exact for empty Context and AAD and exclude the payload and outer transport framing. The initial record costs 67,858 bytes; each later record costs 51,469 bytes, including 424 bytes of framing and 51,045 bytes of signatures. A refresh offer costs 59,973 bytes, a commit 67,874 bytes and a confirmation 123

bytes, totaling 127,970 bytes. Old-record retransmissions add their full serialized sizes. Confirmation caches and outboxes are local state and add no fields to these messages.

The 51,469-byte later-record overhead is 5,026.3% of a 1 KiB payload, 78.54% of 64 KiB, 4.91% of 1 MiB and 0.307% of 16 MiB. These are overhead-to-payload ratios, not total-wire ratios. Record size therefore materially affects throughput, buffering, latency and refresh amortization. A complete benchmark must include the chosen record size, all signatures, KEMs, authenticated buffering, durable state and erasure costs.

Exact totals with the corrected framing: GM512-CORE Mode A is 52,450 bytes, GM512-FULL Mode A is 69,284, and GM512-CORE Mode I is 50,814. The rounded headline figures of roughly 52 KB and 69 KB are unchanged.

OTRoot adds 64 bytes to the bundle. The one-time prekey itself is not part of the signed bundle and is retrieved separately:

The published object is $C_PRE = ENC(M_PRE, \Sigma_PRE)$. With a two-entry ProfileSet, M_PRE encodes to 9,084 bytes (twelve fields, 9,036 bytes of content plus prefixes; the ProfileSet alone is 48 bytes, its count and each entry being ENC fields) and Σ_PRE to 34,431 bytes in CORE or 51,057 in FULL, the absent σ_3 still costing a 4-byte prefix under E3. C_PRE is therefore 43,523 bytes in GM512-CORE and 60,149 in GM512-FULL. Each additional ProfileSet entry adds 20 bytes, being a 4-byte prefix plus a 16-byte ProfileID.

On the McEliece ciphertext size. The Classic McEliece ciphertext is 208 bytes for mceliece8192128 under the current specification. The 240-byte figure that appears in older tables belongs to the "pc" variants, which add 32 bytes, and to the SUPERCOP-20191221 encoding. This is precisely the ambiguity Requirement C2 exists to remove.

ML-KEM-1024 one-time pk	1,568 B
OTIndex	4 B
Merkle proof, ENC-encoded	$8 + 69 \times \text{depth B}$

$\text{depth} = \text{ceil}(\log_2 \text{OTCount})$. Each sibling is 1 side byte + 64 hash bytes, and ENC adds a 4-byte length prefix to depth and to every sibling, so a 16,384-key pool (depth 14) costs $8 + 69 \times 14 = 974$ B.

Requirement W1. Implementations MUST set maximum encoded lengths for C_GM , C_GM_R , all hello and confirmation objects, C_PRE and Merkle proofs, $C_RATCHET_OFFER$, $C_RATCHET_COMMIT$ and $C_RATCHET_CONFIRM$. Enforce a streaming length limit before allocation and reject individual FIELD lengths at or above 2^{32} . Also bound Context, AAD, identity resolution, pending states, reordering and outbox storage under RR9. Oversize, timeout and capacity failures release no plaintext and do not change active cryptographic state. Required signature logging follows SIG2's rate limits.

10.3 Application framing and archival use

Requirement APP1. An application transporting a multi-record or multi-session object MUST define an authenticated transfer identifier, record or chunk ordering, total expected length and an explicit completion condition, such as an authenticated final-record flag or final manifest. Put public framing in AAD or Context; put confidential metadata inside the encrypted payload. A connection close is not proof of complete delivery. Missing final records or chunks must be reported as incomplete.

Requirement APP2. The protocol authenticates plaintext before release and commits replay and ratchet state first. It does not guarantee exactly-once application delivery across a crash between protocol acceptance and application processing. Applications needing that guarantee MUST coordinate durable acceptance and delivery or use an authenticated operation identifier with an idempotent application transaction. Do not replay an accepted cryptographic record to recover an application transaction.

Requirement APP3. GM transports encrypted data; it does not define an archival at-rest format. Once the session roots are erased, a recorded C_GM or C_GM_R cannot be reopened using current session state. An archive must

separately specify durable key access, key recovery, recipient changes, migration and future signature verification. Exporting or retaining a decryption key creates application state with its own compromise and erasure assumptions. Transport forward secrecy does not protect retained plaintext or archive keys.

Confidentiality scope. Payload P is confidential under the stated assumptions. Identities, Context, AAD, timestamps, lengths, routing information and traffic patterns remain observable unless an outer mechanism protects them. The application MUST NOT put confidential business data in those public fields. Authenticated buffering or protected spooling is required for a payload too large for memory; local plaintext handling is part of the endpoint security boundary.

11. Security Model

This section defines security targets and experiment interfaces for review. The targets are not established by the component sketches in Section 12.

11.1 Adversary

The intended adversary is quantum polynomial time with classical access to honest protocol interfaces. It controls the network and may read, drop, reorder, modify, replay and inject messages, initiate sessions and register its own identities. Send delivers a chosen message to a named local session instance and returns only its network response and public success/failure information; successful plaintext or export output goes to the designated honest application, not automatically to the adversary. The experiment provides Corrupt, StateReveal, Reveal and ExportReveal with the scopes below. Full quantum-oracle and composed quantum-security proofs remain open.

Corrupt(U) reveals all secret state currently recoverable from party U, including long-term keys, prekeys, pending and claimed ephemeral keys, CSPRNG state, KEM coins, z_i , master and derived keys, roots, KEKs, expanded schedules, pending refresh material and application-retained plaintext or export keys. Erased secrets are unavailable only if K2's storage and erasure assumptions hold. StateReveal(s) reveals the currently retained session state, its pending and claimed refresh material, and relevant CSPRNG state, but excludes protected long-term authentication keys and unrelated sessions. It is the compromise interface for G6; using full Corrupt would violate G6's authentication-key condition. A revealing query does not itself stop protocol execution.

The adversary has no superposition access to honest parties' interfaces. Section 12.1 uses classical random-oracle queries; its secret-guessing term cannot be carried unchanged into a QROM proof. The post-quantum goals in this section are intended targets, while the available reduction sketches are classical and component-scoped. Quantum query bounds, multi-session losses and protocol composition remain explicit obligations in Section 15.

Partnering. Complementary-role instances are partners when their authenticated tuple (Version, SuiteID, ModelID, ProfileID, SessionID, ID[A], ID[B], h_T) agrees. For TPQRR, epoch agreement additionally requires the same installed h_R and M_COMMIT, including RecordSeq_last and all wraps; agreement on h_R alone does not establish agreement on fields outside that digest. A receiver accepts only after the atomic durable replay/state transition. A base session completes after its mandatory secret cleanup and, in Mode I at the sender, valid confirmation. A TPQRR initial handshake completes when the dedicated Section 7.11.5 cleanup and confirmation requirements are satisfied; the session then remains active with successor roots and K_AUTH. Each accepted TPQRR record completes its erasure when its predecessor roots and record keys have become unrecoverable. An epoch refresh completes at B after installation and at A after valid confirmation; a full session completes only after termination cleanup.

Reveal and export exposure. Reveal(s) returns the experiment-recorded master, traffic, root, confirmation and export material for that session; this deliberately strong diagnostic oracle makes the instance and its partner unfresh for G1/G2. The experiment may record those values for oracle definitions; that bookkeeping is not a requirement for implementations to retain erased secrets. ExportReveal(s) returns any derived application-export key and makes the same key at both partners unfresh. StateReveal or Corrupt also invalidates freshness whenever

its disclosed material permits reconstruction of the tested key. Exposing a later one-way ratchet state is evaluated separately for the earlier-record erasure and G6 goals.

Test eligibility. Before the selected session begins, the experiment requests the existing K_X export branch under $K1/K1a$ and assigns it to an honest test application that performs no adversary-visible operation with that key. The application either retains it privately for Test or erases its copy; only the experiment's bookkeeping then preserves the reference value. Test is permitted on one completed eligible instance, at most once across a partnering class. It returns K_X when the hidden bit $b = 1$ and an independent uniform 64-byte string when $b = 0$. No protocol computation is rerun using the returned candidate. A test on an instance without a derived K_X is rejected. Subsequent Reveal, ExportReveal or reconstructing state exposure invalidates the challenge, except the post-completion exposures expressly allowed by G2 that cannot reconstruct it. A peer instance is not a second independent Test opportunity.

11.2 Goals

G1 - Unused application-export key indistinguishability. For a Test-eligible instance, the adversary's advantage $|\Pr[b_{\text{guess}} = b] - 1/2|$ is negligible in the security parameter. Neither partner may be Revealed; neither tested export key may be exposed or used in an adversary-visible application operation; no adversary-exposed state may permit reconstruction of that key. Peer authentication requires an unexposed secure signing component when its authenticated contribution is accepted. At least one secure KEM contribution must remain unknown under the exposure history, except where G2 permits later exposure after erasure. Freshness is checked through the end of the experiment. K_M is expressly excluded from Test: deriving K_A or K_C from a candidate K_M and comparing a public tag distinguishes it efficiently. G1 is an exporter goal, not a claim that the complete channel has a proved AKE security theorem.

Prekey compromise is tracked separately from long-term identity compromise. A secure KEM family with an already exposed contributing secret cannot satisfy G1; a secret belonging only to a broken KEM cannot satisfy it either. Authentication follows the same surviving-component rule: the signing scheme must be secure and its relevant key unexposed at acceptance. Mode A does not provide explicit receiver confirmation to the sender; G1 does not imply proof of delivery or mutual explicit key confirmation.

G2 - Forward secrecy of the unused export branch and earlier traffic. For G1, later Corrupt queries after session completion may reveal all then-retained long-term and semi-static KEM secrets, provided at least one secure ephemeral or one-time contribution used for that session was erased under F1/K2 and no tested export copy or reconstructing state remains exposed. Keeping K_X in application memory and then revealing it does not satisfy this condition. Traffic forward secrecy separately targets the confidentiality of completed plaintexts under later retained-state compromise; adversary access to an application copy of the plaintext is excluded. TPQRR earlier-record protection additionally requires the predecessor roots, record keys and all reconstructing copies and wrap keys to have been erased.

G2 requires more than A1. A1 asks only that one of the KEMs is IND-CCA2, and in GM512-FULL the surviving leg could be static Classic McEliece, which gives no forward secrecy once its long-term secret is compromised. The hypothesis G2 needs is A1': at least one ephemeral or one-time KEM leg is IND-CCA2 secure and its secret was deleted per F1. Where no ephemeral leg survives, A1 alone can motivate base secrecy while the surviving static secret remains unexposed; it cannot establish forward secrecy after that secret is disclosed.

G3 - Sender authenticity and authenticated-content integrity. $\mathcal{A}$ cannot cause a receiver to accept an authenticated M_{SIG} value that was not produced by the claimed sender, except with negligible probability. The goal is stated over authenticated content rather than over the serialized object: a different valid serialization of signatures over an already authenticated, identical M_{SIG} is not a new forgery. Theorem 2 proves fresh-message EUF-CMA, not strong unforgeability of the signature tuple, and G3 is scoped to match.

G4 - Replay resistance. A C_{GM} accepted once is rejected on every subsequent presentation within the retention window, and is unconditionally rejected outside it.

G5 - Downgrade resistance. $\mathcal{A}$ cannot cause parties to complete with different views of (Version, SuiteID, ModeID, ProfileID, Context, ID[A], ID[B]), because all seven are inside h_T , which keys the session.

G6 - Post-compromise recovery for TPQRR. StateReveal may expose current roots, record and epoch keys, pending session material and CSPRNG state. After the adversary loses endpoint access, both parties must restore unpredictable RNG state before recovery-sensitive key generation, encapsulation and root sampling; at least one newly generated ephemeral KEM leg must remain IND-CCA2 secure and unexposed; the long-term signatures authenticating the refresh must remain secure under the surviving-component condition and their keys uncompromised. The goal is confidentiality and record authentication for records sent after valid refresh confirmation at A and installed at B, until a subsequent disqualifying exposure. No protection is asserted for the compromised interval, continuous access, exposed new plaintext or keys, or compromised authentication. A composed proof and complete continuous-key-exchange experiment remain open.

Channel confidentiality target. The record layer is intended to hide a chosen payload among two equal-length choices when all public metadata and the permitted delivery pattern are fixed independently of the hidden bit. Challenge plaintext is delivered only to the honest application; decryption-oracle access must exclude the challenge and signature reserializations representing the same authenticated content. Public lengths and metadata are not hidden. A complete multi-session authenticated-channel proof must specify this challenge interface, partnering, delivery, corruption and ratchet exposures together; exporter G1 and the component AEAD lemma do not substitute for that proof.

11.3 Assumptions

#	Assumption
A1	At least one KEM instantiated by the selected SuiteID is IND-CCA2 secure against a quantum polynomial-time adversary with classical decapsulation-oracle access. $KEM(CORE) = \{ML-KEM-1024, HQC-5\}$; $KEM(FULL)$ adds Classic McEliece 8192128
A2	At least one signature instantiated by the selected SuiteID is EUF-CMA secure against a quantum polynomial-time adversary with classical signing-oracle access. $SIG(CORE) = \{ML-DSA-87, SLH-DSA-SHAKE-256s\}$; $SIG(FULL)$ adds FAEST-256s
A3	SHA3-512 and HMAC-SHA3-512 behave as a random oracle and a PRF, and KMAC256 as a PRF for authentication against a quantum polynomial-time adversary with classical query access respectively
A4	SHA3-512 is collision-resistant against a quantum polynomial-time adversary
A5	The generator assumption for the negotiated ProfileID, in each case against a quantum polynomial-time adversary with classical oracle access. S-DUAL512: at least one of HMAC-SHA-512 and HMAC-SHA3-512 is a secure PRF. S-KECCAK: the HMAC-SHA3-512 counter generator is a secure PRF. X-GM512: GM-512-BC is a secure PRP. X-TRIPLE and X-TRIPLE-3R: at least one of the three generators is a secure PRF, together with the HMAC-SHA3-512 key-separation assumption or the wrap-hiding assumption respectively (§12.4). X-TRIPLE-3RN: GM-512-BC-N is a secure PRP under three independently sampled keys, so the assumption is multi-key PRP security of one permutation rather than three separate assumptions; the PRP-to-PRF switching term of §12.3 applies to each leg. X-TRIPLE-3RH: at least one of Threefish-512, Kalyna-512/512 and GM-512-BC-N is a secure PRP, which is three distinct assumptions

#	Assumption
A6	TPQRR only: after StateReveal, the adversary loses access before recovery-sensitive sampling; both endpoints restore unpredictable randomness before offer key generation, encapsulation and root sampling; at least one fresh ephemeral KEM contribution remains secure and unexposed; and the long-term authentication keys remain uncompromised. No later exposure reveals the candidate or active recovered state. Durable ownership, rollback protection and erasure follow RR1-RR9 and K2.

A5 is the assumption the experimental X- profiles weaken, which is exactly why they are optional.

Scope. A1 and A2 quantify over the components of the selected suite only. A component absent from that suite cannot discharge either assumption: if ML-KEM and HQC were both broken, GM512-CORE would be broken regardless of the state of Classic McEliece.

A1' - required for G2 only: at least one of the ephemeral or one-time KEM families (z_1 and z_2 in Mode I, z_4 in Mode A) is IND-CCA2 secure. A1, rather than A1', is the KEM-side assumption for G1; G2 additionally requires A1'. G3 rests on A2, G4 on the replay and timestamp semantics of §7.4, and G5 on the transcript and negotiation binding of §5.4. Static legs, Classic McEliece included, count toward A1 but not toward A1'.

12. Security Reductions

12.1 Theorem 1 - Combiner

Theorem 1 is a standalone combined-KEM lemma, presented as a classical-ROM proof sketch. Let the selected component KEMs have fixed shared-secret lengths ell_i bits and independently generated component keys, with ciphertexts and all used public keys bound in T_KEM . Absent suite slots are fixed zero strings and are not candidates for the surviving secure leg.

$$K_M = \text{SHA3-512}(D_MASTER \parallel z_1 \parallel \dots \parallel z_n \parallel h_T)$$

Model the tagged SHA3-512 combiner and transcript hashing as classical random-oracle operations with the specified domain encodings. For a surviving IND-CCA2 component i , the intended reduction replaces its secret by a uniform ell_i -bit value and bounds the probability of querying an oracle input containing it. A schematic reduction shape is:

$$\text{Adv_combiner} \leq \text{Adv_KEM}_i + \text{Adv_transcript_collision} + O(q_H / 2^{\text{ell}_i})$$

$\text{ell}_i = 256$ for every present component

The surviving index must refer to a secure, present component under the applicable exposure model. The O term records the secret-guessing order rather than an asserted tight constant. A concrete proof must account for oracle programming, decapsulation queries, domain collisions, sessions and reduction runtime. The minimum may be taken over eligible component reductions with those costs included. No $q_H / 2^{256}$ claim is made for arbitrary KEMs with shorter secrets.

Proof sketch. Embed a challenge in one secure component and generate the other components honestly. For combined decapsulation queries whose secure component differs from its challenge ciphertext, use the component decapsulation oracle. When that component equals the challenge ciphertext but the remaining transcript differs, that challenge ciphertext cannot be forwarded; the simulator must answer consistently through its oracle programming. Detectable inconsistency is associated with queries containing the hidden component secret, subject to transcript collisions and the exact simulation. This motivates the classical secret-guessing term. A complete concrete simulation is a proof obligation. The lemma concerns a standalone combined-KEM output before protocol-

derived public values are exposed; it does not assert that K_M remains a valid real-or-random Test output after public MACs or confirmation tags are available.

Why ciphertext binding is essential. Without $c_1 \dots c_n$ inside h_T , an adversary could mix and match ciphertexts from different sessions and the reduction fails at Game 2, because the simulator cannot bind its challenge ciphertext to a unique transcript. This step is commonly omitted and its omission silently invalidates the theorem. This specification includes it.

References: *Giacon, Heuer, Poettering, PKC 2018; Bindel, Brendel [14, 15], Fischlin, Goncalves, Stebila, PQCrypto 2019.*

12.2 Theorem 2 - Signature

Theorem 2. With $V_{GM} = \bigwedge_{i \in \text{SIG}(\text{SuiteID})} v_i$ over a common digest μ that includes SuiteID:

$$\text{Adv}^{\text{EUF-CMA}}(\Sigma_{GM}) \leq \min_i \text{Adv}^{\text{EUF-CMA}}(\text{SIG}_i) + \text{Adv}^{\text{coll}}_{\{\text{SHA3-512}\}}$$

Proof sketch. A forgery $(\mu^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$ on a fresh μ^* yields a forgery against each SIG_i individually, since each σ_i^* verifies under pk_i on a message not queried. If μ^* is not fresh but M_{SIG^*} differs, that is a SHA3-512 collision.

Algorithm stripping is prevented because SuiteID is inside M_{SIG} . ■

12.3 Theorem 3 - AEAD

Theorem 3 is a component encrypt-then-MAC lemma for a fixed record context, with pseudorandom traffic-key material computationally independent of the authentication key and with the specified nonce/counter discipline. It motivates IND-CPA confidentiality and INT-CTXT integrity, and their AEAD composition, under A3/A5. Its schematic bound has the following form; it is not the complete interactive, asynchronous or TPQRR channel bound:

$$\begin{aligned} \text{Adv}^{\text{AEAD}} &\leq \text{Adv}^{\text{PRF}}(\text{HMAC-SHA3-512 key derivation}) \\ &\quad + \text{Adv}^{\text{PRF}}(\text{keystream}) \\ &\quad + \text{Adv}^{\text{PRF}}(\text{KMAC256 authentication}) \\ &\quad + q_d \cdot 2^{-512} \end{aligned}$$

where q_d is the number of decryption queries and 2^{-512} is the tag-guessing contribution for the 512-bit tag. The 256-bit ceiling belongs to $\text{Adv}^{\text{PRF}}(\text{KMAC256})$, not to this term.

PRP-to-PRF switching. A 512-bit block-cipher generator contributes $q(q-1) / 2^{513}$ for q distinct blocks under one key, in addition to its PRP advantage. HMAC generators require no permutation-switching step. Multi-key and multi-session bounds must aggregate the applicable per-key contributions and key-derivation or wrap-hiding losses; a wide block alone is not a security proof.

Nonce accounting. A base session encrypts one payload under one traffic-key tuple. Reusing that tuple with a different serialized record is prohibited, and session-key collisions remain part of a full multi-session analysis. For q_m independent 256-bit nonce samples, the generic collision probability is at most $q_m(q_m-1) / 2^{257}$. A TPQRR nonce collision across records does not itself imply a repeated generator input under the same key because record keys evolve; a full proof must model both processes and durable-state failure rather than treating this probability as the complete record-security bound.

On the forgery term. Direct guessing of the 512-bit tag contributes $q_d \cdot 2^{-512}$, which is the term in the bound above. The 256-bit ceiling is separate and belongs to $\text{Adv}^{\text{PRF}}(\text{KMAC256})$: a wider tag does not raise the PRF security of the construction that produces it.

12.4 Theorem 4 - Parallel keystream composition

Let $G_1, \dots, G_m$ be $m \geq 2$ keystream generators keyed independently, and let Z be the XOR of their outputs. The profiles instantiate this with $m = 2$ for S-DUAL512 and $m = 3$ for X-TRIPLE and the root profiles. If the leg keys are

uniformly distributed and independent, and if at least one G_i is a secure PRF on the counter-block domain, then Z is computationally indistinguishable from a uniform string of the same length.

Proof sketch. Suppose G_i is a secure PRF. Replace its output with a uniform random string R ; by PRF security this is indistinguishable, and the reduction loses only the PRF advantage of G_i . The composed keystream is then $Z = R \oplus W$, where W is the XOR of the remaining legs. Because K_{Ei} is independent of the remaining leg keys, R is independent of W , and the XOR of a uniform string with any independent string is uniform. The argument is unchanged for any $m \geq 2$; W is simply the XOR of the other $m - 1$ legs. Hence Z is uniform, and the composed keystream is indistinguishable from random. For a block-cipher leg, $\text{Adv}^{\text{PRF}}(G_i)$ denotes its PRP advantage plus the standard PRP-to-PRF switching term of §12.3. The adversary's advantage against the composed keystream is at most $\min \text{ over } i \leq m \text{ of } \text{Adv}^{\text{PRF}}(G_i)$, with that bound substituted for Theorem 3's $\text{Adv}^{\text{PRF}}(\text{keystream})$; the remaining key-separation, authentication and forgery terms of Theorem 3 are unchanged.

The hypothesis of independence is not decorative. If K_{E3} were derived from K_{E1} - for example by expanding a single K_E - then breaking leg A would yield leg C and the argument fails. Requirement X3 exists for this reason.

S-DUAL512. The mandatory profile instantiates Theorem 4 with $m = 2$. Its leg keys are two outputs of the same HMAC-SHA3-512 derivation under distinct labels, so the key-separation hop is the two-output analogue of the X-TRIPLE hop below:

$$\begin{aligned} \text{Adv}(\text{S-DUAL512}) \leq & \text{Adv}^{\text{PRF}}(\text{HMAC-SHA3-512, two-output derivation}) \\ & + \min\{ \text{Adv}^{\text{PRF}}(\text{HMAC-SHA-512}), \text{Adv}^{\text{PRF}}(\text{HMAC-SHA3-512}) \} \\ & + \text{the non-keystream terms of Theorem 3} \end{aligned}$$

The generator layer therefore carries SHA-2 and SHA-3 redundancy, while the derivation layer remains dependent on HMAC-SHA3-512 alone - the distinction §7.6 states.

Which independence. In Profile X-TRIPLE the leg keys are HMAC-SHA3-512 outputs under distinct labels from a single K_M , so the hypothesis holds computationally rather than literally: the keys are indistinguishable from independent uniform strings so long as HMAC-SHA3-512 is a PRF and K_M is secret. In Profile X-TRIPLE-3R the roots are sampled separately and no common K_M stands behind them, so the hypothesis is met more directly.

The two native root profiles. X-TRIPLE-3RN and X-TRIPLE-3RH inherit the sampled-root construction of X-TRIPLE-3R and therefore meet the independence hypothesis on keys in the same way. They differ in what the generators are, and that is where the theorem's value differs. In X-TRIPLE-3RH the three generators are structurally different ciphers, so the min-over- i term is a minimum over three separate PRP assumptions. In X-TRIPLE-3RN all three are GM-512-BC-N under independent keys, so the theorem applies unchanged but the min-over- i term becomes a multi-key PRP assumption on that one permutation, plus the per-leg PRP-to-PRF switching term of §12.3, rather than a disjunction over three primitives. §7.9 states this explicitly rather than presenting the XOR as a hedge against a structural break.

The reduction for X-TRIPLE therefore carries an extra term. Replacing K_{E1} , K_{E2} and K_{E3} by independent uniform keys is a hop justified by the multi-output PRF security of HMAC-SHA3-512 under K_M , after which Theorem 4 applies. The bound has the shape $\text{Adv}(\text{X-TRIPLE}) \leq \text{Adv}_{\text{PRF}}(\text{HMAC-SHA3-512, three outputs}) + \min \text{ over } i \leq m \text{ of } \text{Adv}_{\text{PRF}}(G_i) + \text{the non-keystream terms of Theorem 3}$, the composition bound replacing $\text{Adv}^{\text{PRF}}(\text{keystream})$ rather than adding to the whole of Theorem 3. For X-TRIPLE-3R the HMAC-SHA3-512 derivation term is absent, but it is replaced rather than removed: the roots are published in wrapped form, so the reduction needs a wrap-hiding hop showing that $(\text{KEK}_A, \text{KEK}_B, \text{KEK}_C)$ is indistinguishable from independent uniform strings, after which W_A, W_B, W_C are one-time-pad ciphertexts revealing nothing about the roots. That hop holds in the random-oracle model for SHA3-512 whenever at least one component KEM survives, by Theorem 1. The bound is $\text{Adv}(\text{X-TRIPLE-3R}) \leq \text{Adv}_{\text{WRAP}} + \min \text{ over } i \leq m \text{ of } \text{Adv}_{\text{PRF}}(G_i) + \text{the non-keystream terms of Theorem 3}$, the composition bound replacing $\text{Adv}^{\text{PRF}}(\text{keystream})$ rather than adding to the whole of Theorem 3, with Adv_{WRAP} reducing to SHA3-512 in the ROM plus combiner security. Neither profile is unconditionally better: X-TRIPLE pays an HMAC-SHA3-512

multi-output PRF term, X-TRIPLE-3R pays a wrap-hiding term. The statement of Theorem 4 above should be read as requiring uniform independent keys; X-TRIPLE meets that hypothesis only after the hop.

The theorem is stated in the standard model and assumes only PRF security of a single leg. It does not assume anything whatsoever about GM-512-BC. Two distinct statements follow, and they should not be conflated. The theorem itself needs only that some one of G_A , G_B , G_C is a secure PRF, which A5 states and which GM-512-BC alone could in principle satisfy. The standardized assurance floor is the strictly stronger claim that at least one of the standardized legs - A under HMAC-SHA-512 or B under HMAC-SHA3-512 - remains a secure PRF; that is an assumption a deployment makes, not a consequence of A5. Only under the stronger claim does X-TRIPLE confidentiality rest on standardized primitives alone. The complete profile additionally assumes HMAC-SHA3-512 for key derivation and KMAC256 for authentication, as the bound above records.

What Theorem 4 does not cover. Theorem 4 bounds distinguishing advantage against the composed keystream. It says nothing about the cost of key recovery against the composition, and the two are not the same question. XOR of several keyed generators does not give additive key strength: in idealized settings, quantum key recovery against an XOR of independently keyed permutations is known to be substantially cheaper [23] than searching the joint key space, with results near $2^{(3k/4)}$ for equal key sizes k . Those results are not established for the heterogeneous HMAC-SHA-512 $\oplus$ HMAC-SHA3-512 $\oplus$ GM-512-BC composition here, and this specification claims no key-recovery bound above the 256-bit target of §3.1. §15 records the gap.

12.5 What is not proved

- No composed proof of exporter G1/G2 or full authenticated-channel confidentiality, integrity and agreement in a standard multi-session AKE/channel model. Public tags invalidate $\text{Test}(K_M)$, so a proof for the earlier master-key game cannot be carried forward by renaming its output. The corrected export branch and channel interface require fresh composition analysis, including application exposure.
- No proof of the post-compromise-recovery goal G6 of TPQRR. G6 is a stated target under A6; §7.11.3 gives the argument informally, and no reduction in a ratcheting or continuous-key-exchange model is provided.
- No proof in the QROM. No QROM proof is provided. The combiner argument and the X-TRIPLE-3R wrap-hiding reduction use SHA3-512 in the classical random-oracle model; Theorem 2 rests on EUF-CMA and SHA3-512 collision resistance, and the base statement of Theorem 4 is a standard-model PRF composition. The composed protocol has not been analyzed in the QROM.
- No machine-checked proof.
- No proof of key-compromise-impersonation resistance.
- No formal deniability analysis. Public suite signatures over authenticated content make the protocol unsuitable for applications requiring cryptographic deniability. Long-term evidentiary use additionally requires an archival signature-validation and trust-management policy, which this transport does not define.

13. Constants and Test Vectors

13.1 Domain separators

```
D_KEM    = "GM512-KEM"
D_MASTER = "GM512-MASTER"
D_ENC     = "GM512-ENC"
D_AUTH    = "GM512-AUTH"
D_CONFIRM = "GM512-CONFIRM"
D_EXPORT  = "GM512-EXPORT"
D_SIG     = "GM512-SIG"
D_RK      = "GM512-RK"
```

```
D_ENC_A = "GM512-ENC-A"
D_ENC_B = "GM512-ENC-B"
D_ENC_C = "GM512-ENC-C"
D_WRAP_A = "GM512-WRAP-A"
D_WRAP_B = "GM512-WRAP-B"
D_WRAP_C = "GM512-WRAP-C"
D_HELLO = "GM512-HELLO"
D_PRE = "GM512-PRE"
D_OT_LEAF = "GM512-OT-LEAF"
D_OT_NODE = "GM512-OT-NODE"
D_OT_EMPTY = "GM512-OT-EMPTY"
D_RATCHET_KEY = "GM512-RATCHET-KEY"
D_RATCHET_STEP = "GM512-RATCHET-STEP"
D_RATCHET_OFFER = "GM512-RATCHET-OFFER"
D_RATCHET_TRANSCRIPT = "GM512-RATCHET-TRANSCRIPT"
D_RATCHET_MASTER = "GM512-RATCHET-MASTER"
D_RATCHET_WRAP_A = "GM512-RATCHET-WRAP-A"
D_RATCHET_WRAP_B = "GM512-RATCHET-WRAP-B"
D_RATCHET_WRAP_C = "GM512-RATCHET-WRAP-C"
D_RATCHET_AUTH = "GM512-RATCHET-AUTH"
D_RATCHET_EPOCH_AUTH = "GM512-RATCHET-EPOCH-AUTH"
D_RATCHET_CONFIRM = "GM512-RATCHET-CONFIRM"
D_RATCHET_COMMIT = "GM512-RATCHET-COMMIT"
D_CTR = SHA3-512("GM512-CTR")[0:16]
      = e11d6bbb970757267e33adf684064d68
D_CTR_A = SHA3-512("GM512-CTR-A")[0:16]    -- §7.9, §7.10
      = 9bc2830ec7224e02ebf958a7f43d127a
D_CTR_B = SHA3-512("GM512-CTR-B")[0:16]
      = db0cabd484ae85ea46089d8311e42f38
D_CTR_C = SHA3-512("GM512-CTR-C")[0:16]
      = af926aa5bc7661b5db5feebe31d4ac5d
Version ∈ { "GM512-v0005", "GM512-TPQRR-v0005" }
SuiteID ∈ { "GM512-CORE-005", "GM512-FULL-005" }
```

This list is the normative domain-separator registry. Where a separator is also shown at its point of use, that occurrence is illustrative and this list governs.

The following mapping fixes the component editions and signature interfaces for the draft suites. SHA3-256 values hash the complete PDF bytes at the cited URLs. These pins identify source documents, not an independently validated implementation. Changing a component parameter, wire encoding, normative API or effective cryptographic behavior requires a new SuiteID.

GM512-CORE-005

```
ML-KEM-1024: FIPS 203, final 13 August 2024
DOI 10.6028/NIST.FIPS.203
HQC-5: HQC specification 22 August 2025, reference [7]
SHA3-256
5d30017a0ed97a5fc31d35bf6b4a05df
d5c89b825e517e559a76595a88c61732
ML-DSA-87: FIPS 204, final 13 August 2024
DOI 10.6028/NIST.FIPS.204
SLH-DSA-SHAKE-256s: FIPS 205, final 13 August 2024
DOI 10.6028/NIST.FIPS.205
Signature message, context and randomization: SIG3
```

GM512-FULL-005 = GM512-CORE-005 plus

```
mceliece8192128: Classic McEliece specification
23 October 2022, reference [9]; non-pc, non-f parameter set
SHA3-256
393ea56c1c6096e3c3fcac2a1e00e27e
3a6e08fdfe7c87600d6c0157da254e5d
FAEST-256s: specification version 3.0, 1 September 2026
reference [11]; not FAEST-EM-256s
SHA3-256
c333303845e392bac3545bf24987187
aa9b5402f5676184b81dc425e77ceba89
```

Requirement C2. An implementation MUST use the exact editions in this registry and SIG3's public signature interfaces. The 2022 Classic McEliece algorithm specification is the reproducible algorithm pin; its June 2026 ISO status is a separate reference [10]. Implementers MUST review publisher errata and report the adopted interpretation in the implementation manifest. A correction that changes keys, ciphertexts, signatures, transcript bytes or acceptance behavior requires a new suite; a security issue must not be silently ignored to retain a suite identifier. The release manifest must identify implementation commits, dependencies, build options, errata decisions and complete conformance results. Source pinning alone does not satisfy that release gate.

13.2 Canonical encoding

ENC("AB", "CDE") = 00000002 4142 00000003 434445

13.3 KDF chain

Synthetic KDF vector: z1 = 00 repeated 32 times, z2 = 11 repeated 32 times, z3 = 22 repeated 32 times and z4 = 33 repeated 32 times, with h_T = SHA3-512("GM512-TEST-TRANSCRIPT"). All four lengths match the pinned KEM output lengths. The byte patterns are supplied test inputs, not outputs of actual encapsulation, and h_T is not the digest of a full protocol transcript. These vectors test the KDF and related symmetric calculations only; they are not handshake, signature or TPQRR conformance vectors. Correcting HQC from 64 to 32 bytes does not change these already-32-byte examples:

h_T = b8180bf7a5270ccf193f9d69c7bbf4fb596274be635a732e1cc914e70e2428a2
e9a7ab78059d45ca1adeb5ad2081c86cd5626d8ebd8084333a319dea098ebe1e

K_M = 4f894614ceb6f9cc07c776fc74a112f44addcf65edb84520490c347a3ac82673
b21bf96535aabbba6dc88c16c33b5599356c6aed483559aad7e0a6e312e50a95e

K_E = 1de6498e44b4d21aeb28538614f8486e1911307903eda37924dba8188208e360
1fc1038d4f8c1807ca004e820f940a6493a498501a0964a882ca0c107e4d34e4

K_A = 52281838c4ec6024ed8a662d25ebf3f1ed4fbb81368a787c37d4ea47b09103de
fa1c59469c4cb2341bfbd2cf32b24177f0cf75aa4d82c555be784bf1bd20c868

K_C = dd37ca217429afe34544f853c33099f4f0f4eea981ec852295476e9b87c27583
2778dcf416ff7df842ecf22290578c92e13e362a10f85e15d5d7a919a5435055

13.4 GM-512-BC block cipher

S-box verification:

S-box is a bijection over F_2^8 . S[00] = 63, S[53] = ed (matches FIPS 197)
max DDT entry = 4/256 $\rightarrow$ max differential probability 2^{-6}
max |LAT| = 16/128 $\rightarrow$ max linear correlation 2^{-3}

MixColumns verification:

cir(02,02,03,04,05,03,05,07) is MDS over $GF(2^8)$ mod 0x11B
- all 12,869 square submatrices verified nonsingular
branch number = 9 (optimum for 8×8)

Vector 1 - all-zero key and plaintext:

K = 00 $\times$ 64
P = 00 $\times$ 64
C = 1952cd5f8de46910e636f2011ea7cb756a6636fb70f3dabba9d931853c251479
3e57d28fb2a3947a57af923d83a019efa915c158365f1f502096e9d0fec3948a

Vector 2 - sequential key, descending plaintext:

K = 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
P = fffefdfcfbfa9f8f7f6f5f4f3f2f1f0efeeedecebeae9e8e7e6e5e4e3e2e1e0
dfdedddcdcbdad9d8d7d6d5d4d3d2d1d0cfcecdcccbcac9c8c7c6c5c4c3c2c1c0

C = a9748c82bd9112622ee9f0cb8889b84efb644e94086da9c6fdc366ca5c1a12d7
e925a97e39bc7fea659fdbbb3dabb2d76082951e84fc1ee152fffb8db25832cac

RK_0 = HMAC-SHA3-512(K, ENC(D_RK, 0000)) for the Vector 2 key K = 00 01 ... 3f
= 7724a7b78805960f5e4a10a3e4f78f6912c714c6bcd6cf734123397edf63db0e
3c10e1af17ad740150182437586f2d4abf9dfb259846401a35ff2cbfb9dbdcfe

RK_0 for the Vector 1 key K = 00 × 64
= 77baed6aa5fa516374a7e862a26dab7d6d1775e842663e00e96f22bcced0e73b
4e35bcfc8594076f75744d0466eb2dd69bc81ee476aad804c20cf880423b4892

Round-trip: Decrypt(K, Encrypt(K, P)) = P - verified for both vectors.

13.5 Profiles S-DUAL512 and X-TRIPLE

Derived with the same K_M and h_T as §13.3. These values are computable from this specification alone and were verified against the §13.3 chain.

K_E1 = dee636eef625cbcc8b87ac445ca16c0d
ca0103f937521ab41c54a4f7c98d7ec6
d3f30c2d8be315aa526b7abc93564d88
6af08db9ae29fdd7a82510a34b67f662

K_E2 = b8a8bde6be0d937b859338c3e3fec92d
614cdf9e23715b5910d7e150b7d5c95a
7ec5dc5ee97650bd3840a807422c4c06
52ddda7c78f6e773a0f25933a69582df

K_E3 = e471e102dce258079d0d46f00cc0b81f
f95ba03c02d76af56fdb89ad4177b1d
e351d6ad992028bb58e18839a7f9cd99
c8282ab49fe3d2f00de55e6f17502743

Derived leg keys and keystream outputs, with the test nonce N = 00 × 32 and block index j = 0. Every value below is computable from standardized primitives alone.

-- S-DUAL512: composed keystream block (N = 00 × 32, j = 0)

Z_0^A = 28421e1f86f49fd6455bd2abe8c02137
fca0b0fe71299d19c2a30193dc824fc4
d3e3af563e935e613ba249261f0eachb6
8241a5c1ebba1baac7b6e38698ca7131

Z_0^B = 9308abb81006e391e5e5a5d1dd0ed830
0349129ddc1ae2c13e4411d986bbf8b0
241394d4e3e045046493aaa824620c04
e626fd7dd3c37a6f1d7fd62fdcef85b6

Z_0 = bb4ab5a796f27c47a0be777a35cef907
ffe9a263ad337fd8fce7104a5a39b774
f7f03b82dd731b655f31e38e3b6ca0b2
646758bc387961c5dac935a94425f487

Leg C and the composed keystream follow. Everything above this point is computable from standardized primitives alone; the two values below additionally require the GM-512-BC definition of §9.

Z_0^C = 31f141b2d49ea1619b478f92e55320f5
37a2f571565e1dd5dd839abd1903f2c7
92071c7ec1701ae58128c65b79afadbe
722e5eefbe7c1e2d2f81d1787c24e8c0

Z_0 = 8abbf415426cdd263bf9f8e8d09dd9f2
c84b5712fb6d620d21648af7433a45b3
65f727fc1c030180de1925d542c30d0c
1649065386057fe8f548e4d138011c47

13.6 Profile X-TRIPLE-3R

With the K_M and h_T of §13.3, and test roots $R_A = aa \times 64$, $R_B = bb \times 64$, $R_C = cc \times 64$:

```
KEK_A = b95397a1b6388d1209455b5643b69a42
       c461d5e4fba5392f39570ddd383a9ae8
       f8faa65cc742ad2859362228feb3b0db
       7e98a63f646e68582857252bf6f409cd
```

```
KEK_B = ab0fe0c6fc9c44af0c609b3bdd87df1e
       25653ad9bb5b48c754cdb81119ed25ec
       44d4240a57d10c60174c9e7c65e0d019
       a17b37cf61e7b4c84f23bb612d221b5b
```

```
KEK_C = aba39d475900f6b5e902633a6a046256
       d7f75314b13ba6b6f1d90d7fbb72f749
       35179b9d9c849d9852513d223bc01540
       32d92ff559396e65c5272cf770bbcb05
```

Wrapped roots as transmitted:

```
W_A = 13f93d0b1c9227b8a3eff1fce91c30e8
      6ecb7f4e510f938593fda77792903042
      52500cf66de80782f39c888254191a71
      d4320c95cec4c2f282fd8f815c5ea367
```

```
W_B = 10b45b7d4727ff14b7db2080663c64a5
      9ede816200e0f37cef7603aaa2569e57
      ff6f9fb1ec6ab7dbacf725c7de5b6ba2
      1ac08c74da5c0f73f49800da9699a0e0
```

```
W_C = 676f518b95cc3a7925ceaff6a6c8ae9a
      1b3b9fd87df76a7a3d15c1b377be3b85
      f9db5751504851549e9df1eef70cd98c
      fe15e33995f5a2a909ebe03bbc7707c9
```

Keystream legs under the sampled roots, $N = 00 \times 32$, $j = 0$:

```
Z_0^A = 834a5a2420a833d5581363a94ed09ee5
       ddd253d4fe01703d699217047a24ac3a
       64b0e1b48b8218475a5509ecf215ab9a
       ac3ba26abd7414c310548ef9393e08fe
```

```
Z_0^B = 2be53678431fc91b4edc2cf22add557e
       7af135e65f341c3b78b23909f7295cb7
       be5906d0c920fcfd9e4e738b9adda105
       2e031cd16ec56cf78926247eb31f444d
```

Z_0^C requires the GM-512-BC reference implementation. Every other value above is computable from standardized primitives alone.

Leg C and the composed keystream, computed with the reference implementation of §9:

```
Z_0^C = 8013737204c8a2bdb4f956df1833cdc9
       f8e533953fb84533763efac7bcd15830
       504708a02ccd6b17bc384e4f9b1c716b
       58e99fc9dddc095965d5706b9027b080
```

```
Z_0 = 28bc1f2e677f5873a23619847c3e0652
      5fc655a79e8d2935671ed4ca31dca8bd
      8aaeefc46e6f8fad78233428f3d47bf4
      dad121720e6d716dfca7daec1a06fc33
```

13.7 Export key, S-KECCAK keystream and confirmation tag

From the §13.3 chain, with $N = 00 \times 32$ and $j = 0$:

```
K_X = b81cdb0bbd87b6fb4d78c9793eb00dd5
      4d7a207d5469c90a7bfeb1939f17d474
      d8d42c1467f80446939c29fc9fef08d0
      3a8620c053f845784ec03a83547968cf
```

```
Z_0 (S-KECCAK) = 825c8b85e6b605b63550d53ae9d39c89
                  46e8e61a57ae2589934eebf474ba385b
                  fde1414e6c831017242cbe1832af45f8
                  c0e5ac59be6edc9c9e24506df60ad5a3
```

```
Tag_confirm = d1365286bd3a0e6831d69fb1acb75f84
              4ab13e6ba8617c0f34896b80915d7512
```

Tag_confirm is KMAC256(K_C, ENC("confirm", h_T), 256 bits, "GM512-CONFIRM") with the K_C of §13.3. A complete canonical C_GM vector requires ML-KEM and HQC ciphertexts and is left to the reference implementation.

13.8 Profile X-TRIPLE-3RN

With the test roots of §13.6 ($R_A = aa \times 64$, $R_B = bb \times 64$, $R_C = cc \times 64$), $N = 00 \times 32$ and $j = 0$. GM-512-BC-N uses the native schedule of §9.5a.

```
D_CTR_A = 9bc2830ec7224e02ebf958a7f43d127a
D_CTR_B = db0cabd484ae85ea46089d8311e42f38
D_CTR_C = af926aa5bc7661b5db5feebe31d4ac5d
```

```
Z_0^A = 85d4f323fe1aee832e56596a072067a7
        013bb19f756fdc0f6288d7fca1e7e7a8
        4fe8317a4c67d04e01580ce768fa5eed
        d7f83ba5d10f08232c4bb5ac7497bfe4
```

```
Z_0^B = 765955440a8be668e4b02e5d5ec9d5c7
        9d40f3f7ee4b1ce33a2b0544a97e72ae
        cffc3e699623b185c536a6352be3034
        26c5dc1ddc35300d0bc6c2df492fef18
```

```
Z_0^C = 3c3634a6d94aacdccc42196d35b7f11d
        9d4326e3b0b306c881253c096af514b8
        b50c857c4b0def1da1cda0f1a421ac1f
        6a702e2623163c083611f7e22428ffe6
```

```
Z_0 = cfbb92c12ddba43706a46e5a6c5e437d
      0138648b2b97c624d986eeb1626c81be
      351b77e09e08044bfcc6c6759e65c2c6
      9b4dc99e2e2c0426119c80911990af1a
```

Requirement V1. An implementation MUST reproduce all vectors in §13 before being considered conformant. A conformance test suite covering the full protocol requires PQC library integration and is left to the reference implementation.

14. Comparison with Existing Hybrid Constructions

14.1 Related work

KEM combiners. Giacon, Heuer and Poettering [14] analyze robust KEM composition, and Bindel et al. [15] study hybrid key encapsulation and authenticated key exchange. GM's ciphertext-bound combiner belongs to this research direction. Its exact instantiation and exposure model require the scoped proof work described in Section 12.1; the existence of related combiner results does not establish the full GM protocol theorem.

Classical/post-quantum hybrids. X-Wing [16] and the TLS 1.3 X25519MLKEM768 group in RFC 10024 [31] both combine X25519 and ML-KEM-768, but use their own distinct composition and protocol specifications. They must not be identified as the same construction. Signal PQXDH [17] adds post-quantum key establishment to an

asynchronous X3DH-style design. These approaches combine classical and post-quantum mechanisms; RFC 9794 [18] supplies relevant hybrid terminology.

Asynchronous post-quantum key agreement. PQXDH [17] is relevant prior work for signed prekeys, one-time prekeys and offline initiation. Its authentication is based on its own X3DH-derived identity and DH structure, with signed prekeys. GM Mode A instead specifies suite AND-signatures over each payload and binds a bundle identifier and Merkle commitment over the one-time prekey pool. These are protocol differences, not evidence of superior security or maturity.

Post-quantum ratcheting and post-compromise security. Apple PQ3 [33] periodically injects a post-quantum KEM contribution so that a conversation can recover protection after compromise of earlier session keys, while Signal's 2025 Sparse Post-Quantum Ratchet and Triple Ratchet [32] combine per-message key evolution with periodic post-quantum shared-secret generation. TPQRR addresses the same temporal security objective but uses no elliptic-curve ratchet and refreshes three independently sampled 512-bit root chains feeding the parallel generator layer. This difference is architectural, not a claim that TPQRR has the deployment maturity or formal analysis of those systems.

Cascade and XOR composition. The cascade results of Even and Goldreich [21] and Maurer and Massey [22] concern serial cipher composition and are not used as a security argument for GM-512: the XOR construction of §7.7 is commutative, has no first leg, and is analysed separately in Theorem 4. For parallel composition, Mennink and Szeponiec [23] give quantum key-recovery attacks on the XOR of independently keyed permutations; §12.4 declines to transfer that bound to the heterogeneous case here but records it as the reason no key-recovery claim above the 256-bit target is made.

Symmetric design. GM-512-BC reuses the AES S-box and $GF(2^8)$ field polynomial 0x11B [1,19]. Grøstl P512 already has an 8 by 8 byte state, row-shift offsets 0 through 7 and the same circulant MDS matrix [20, Sections 3.4.4-3.4.5]. Those features are established prior art and are not claimed as new. The proposed GM keyed construction, its 32-round organization, key schedules and integration into this transport require evaluation as the specific contribution. Shared components do not make the entire cipher identical to Grøstl, and this paper assigns no unsupported numerical similarity percentage.

	TLS 1.3 X25519MLKEM768	X-Wing	Signal PQXDH	GM-512
Key-establishment mechanisms	2 (1 classical)	2 (1 classical)	2 (1 classical)	2–3 constructions, all post-quantum
Assumption families	ECDH + lattice	ECDH + lattice	ECDH + lattice	Lattice + code; two distinct code constructions in FULL
Signature legs	Certificate authentication depends on TLS configuration	Not part of the standalone KEM	Signed prekey plus X3DH-derived authentication	2 or 3 signatures on each payload
Forward secrecy	Yes (ephemeral)	Depends on use	Yes (one-time prekeys)	Mode I: full. Mode A: full with a one-time prekey, otherwise rotation-bounded
Transcript binding	Yes	Yes	Yes	Yes
Combiner	Concatenation KDF	Concatenation KDF	HKDF	Concatenation KDF

	TLS 1.3 X25519MLKEM768	X-Wing	Signal PQXDH	GM-512
Size comparison scope	Client/server key shares: 1,216 / 1,120 bytes; full TLS handshake additional	Standalone KEM; authentication and record transport additional	Prekey and initial- message costs depend on deployment	Base Mode A with OT: 52,450 / 69,284 bytes before payload; Section 10
Deniability	n/a	n/a	Yes	No (by design)

A fair bandwidth comparison must use the same workload and include key establishment, identity or certificate distribution, authentication, record framing, payload sizes, refreshes and retransmissions. The approximately 52-69 KB base GM figures include per-payload signatures; a TLS key-share figure or standalone X-Wing ciphertext does not. No overall 25-60 times ratio is asserted from those unlike quantities. GM's sizable signatures on every TPQRR record constrain its practical use, especially for small messages. Applications considering long-term records, command channels or other high-assurance data must evaluate that cost together with the experimental status and the archival requirements of Section 10.3.

The proposed contribution is the integration of multi-family PQ/PQ key establishment, multi-family authentication, interactive and asynchronous forward secrecy, explicit downgrade and transcript binding, replay-state semantics, optional heterogeneous symmetric-generator redundancy, and the experimental TPQRR extension for triple-root per-record evolution and in-session post-compromise recovery into a single authenticated transport specification. The ratchet extension is deliberately separated from the base single-payload protocol so its additional state and security assumptions are reviewable in isolation. A classical/post-quantum hybrid also hedges both ways: its classical leg survives a cryptanalytic break of the post-quantum component, and its post-quantum leg survives the arrival of a quantum computer. The distinction is that the surviving classical leg is not itself post-quantum, so a hybrid whose PQ component falls to cryptanalysis is left with no quantum-resistant protection. Every KEM leg used by GM-3X512-PQS is post-quantum.

15. Open Challenges

The following are stated as work needed, not as work done.

1. **Third-party cryptanalysis of GM-512-BC** against the classes listed in §9.7, especially integral/division-property and rebound attacks.
2. **A constant-time GM-512-BC implementation** - bitsliced or vector-permute - with measured performance.
3. A composed proof for the corrected unused-export G1/G2 experiment and full authenticated-channel confidentiality and agreement, with explicit Send, Reveal, ExportReveal, StateReveal, Corrupt and Test semantics, application exposure and partnering.
4. **A QROM proof** for the combiner and the AEAD.
5. **A machine-checked proof** of the combiner in EasyCrypt or a comparable tool.
6. Two independently developed implementations and suite mappings, with identical keys, transcripts and outputs under supplied deterministic test randomness. Publish the implementation commits, build commands and full conformance results; the source-document hashes in Section 13.1 do not replace this evidence.

7. Executable state-model and implementation validation covering initial TPQRR retention of K_AUTH[0], single atomic replay/root commits, authenticated exclusive claims, lost records, bounded outbox recovery, offer expiry, cached confirmation after pending-offer removal, concurrent commits, epoch/sequence overflow and every crash boundary. Assert no plaintext release before acceptance, no root advancement on invalid input, no rollback after a committed transition and no old-epoch resumption after WAIT_CONFIRM.
8. **Evaluation of $t < n$ threshold signature policy** and its effect on the unforgeability bound.
9. **A concrete PRF-advantage bound for each X-TRIPLE leg** in the counter-block domain, so that the min-over-legs bound of Theorem 4 can be stated numerically rather than asymptotically.
10. **Tight classical and quantum key-recovery bounds for the heterogeneous three-generator XOR**, covering generalized claw-finding, k-XOR and quantum meet-in-the-middle strategies in both the Q1 and Q2 models. Until these exist, no key-recovery claim above 256 bits is made for any XOR profile.
11. Complete base and TPQRR conformance vectors: both suites and base modes; empty and partial payloads; one-time-prekey and FS_DEGRADED cases; initial record, subsequent record, refresh offer/commit/confirmation and first post-refresh record. Add negative vectors for field length, signature, MAC, context, suite, epoch, replay and wrap changes. The synthetic KDF and primitive examples of Section 13 do not satisfy this requirement.
12. A composed G6 proof in a continuous or ratcheting key-exchange model, including QROM analysis and the separate session-state/CSPRNG compromise oracle. Demonstrate recovery-sensitive entropy acquisition before KEM key generation and encapsulation, and test recovery after exposure of old RNG state. A successful reseed API call or a small simulation alone cannot establish entropy quality or the cryptographic PCS theorem.
13. **Cryptanalysis of GM-512-BC-N**: differential, linear, related-key, integral and division-property, rebound, meet-in-the-middle, invariant-subspace and quantum-specific analysis of the native round-key schedule of §9.5a, including the multi-key security on which X-TRIPLE-3RN depends. The schedule reuses the round function under fixed constants, which is the structure in which related-key attacks are historically found.
14. **Validated reference implementations of Threefish-512 and Kalyna-512/512**, reproducing the published test vectors of [28] and [26], and the resulting X-TRIPLE-3RH keystream vectors for §13.

Release evidence

Before an implementation is described as interoperable, publish its exact suite manifest, adopted errata, signature API choices, deterministic randomness inputs and full end-to-end vector results from two implementations. Before security deployment, additionally complete independent implementation review, side-channel evaluation, durable-state and erasure validation, workload-specific end-to-end benchmarks and cryptanalysis appropriate to every enabled experimental primitive. Formal claims must match the proofs actually completed. Until then this document remains an experimental specification, including its mandatory symmetric profile.

16. Summary

GM-3X512-PQS specifies the following experimental architecture and security targets:

- Multi-assumption key establishment using a ciphertext-bound KEM combiner, supported by the scoped classical-ROM component argument in Section 12.1
- AND-combined signatures, with a surviving-component EUF-CMA argument and exact public signing interfaces

- Forward-secrecy targets based on secure ephemeral or one-time KEM contributions and verifiable secret erasure
- Optional TPQRR for ordered Mode I sessions, with per-record root evolution, durable outboxes, atomic acceptance and phase-specific refresh confirmation, targeting recovery under A6 and RR1-RR9
- **Replay resistance** through identity-scoped replay state, consumed one-time prekeys where present, and a bounded acceptance window
- **A target of 256-bit classical and NIST Category 5 post-quantum security under A1–A5**, honestly bounded by the 256-bit KEM shared secrets and by KMAC256 authentication, with the composed QPT/QROM proof recorded as open
- **Explicit fail-closed semantics** at every verification point
- **A mandatory record-layer profile using only standardized symmetric primitives**, with the novel cipher optional and clearly labeled
- **Optional parallel keystream composition (Profile X-TRIPLE)** with confidentiality secure if any one of three independently keyed generators survives (Theorem 4), two of which are standardized
- **An all-512-bit heterogeneous generator profile (X-TRIPLE-3RH)**: Threefish-512, Kalyna-512/512 and GM-512-BC-N, each keyed by an independently sampled 512-bit root, giving 1,536 bits of aggregate root material without claiming 1,536-bit security

The paper does not establish unconditional security, 512-bit security, deniability, full protocol interoperability, a composed AKE/channel or PCS proof, a QROM proof, or production assurance. GM-512-BC's single-trail bounds do not cover every attack class. TPQRR recovery remains a conditional research target and the base single-payload protocol has no separate recovery mechanism.

Section 15 identifies the implementation, testing and proof work needed to advance these claims. Corrections to a specification are necessary for that work but do not constitute its completion.

The 54 selected formula, vector and algebra checks used in the technical review reproduced successfully, including the 12,869 nonempty square minors of the MDS matrix, the printed inverse, selected S-box properties and the supplied symmetric examples. Throughput, statistical avalanche, full public-key operations, complete handshakes, side-channel behavior and TPQRR execution have not been independently validated by those checks. Source code and full protocol test artifacts are release requirements, not accompanying artifacts asserted by this document.

17. References

- [1] FIPS 197, Advanced Encryption Standard (AES). NIST, 2001; superseded by FIPS 197-upd1, 2023.
- [2] FIPS 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. NIST, 2015.
- [3] NIST SP 800-185, SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash. 2016.
- [4] FIPS 203, Module-Lattice-Based Key-Encapsulation Mechanism Standard. NIST, final 13 August 2024. DOI 10.6028/NIST.FIPS.203. <https://csrc.nist.gov/pubs/fips/203/final>
- [5] FIPS 204, Module-Lattice-Based Digital Signature Standard. NIST, final 13 August 2024. DOI 10.6028/NIST.FIPS.204. <https://csrc.nist.gov/pubs/fips/204/final>
- [6] FIPS 205, Stateless Hash-Based Digital Signature Standard. NIST, final 13 August 2024. DOI 10.6028/NIST.FIPS.205. <https://csrc.nist.gov/pubs/fips/205/final>
- [7] Hamming Quasi-Cyclic HQC, specification 22 August 2025. Section 4.2 and Table 6 define the 32-byte shared key. https://pqc-hqc.org/doc/hqc_specifications_2025_08_22.pdf
- [8] NIST IR 8545, Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process. 2025.
- [9] Classic McEliece conservative code-based cryptography, cryptosystem specification, 23 October 2022. The revision-5 algorithm pin is mceliece8192128, not a pc or f variant. <https://classic.mceliece.org/mceliece-spec-20221023.pdf>
- [10] ISO/IEC 18033-2:2006/Amd 2:2026, Information technology - Security techniques - Encryption algorithms - Part 2: Asymmetric ciphers, Amendment 2. June 2026. Project status: <https://classic.mceliece.org/iso.html>. This status reference is distinct from the algorithm PDF pin in [9].
- [11] Baum et al. FAEST v3 Algorithm Specifications, version 3.0, 1 September 2026. Table 3.1 and Section 8. <https://faest.info/faest-spec-v3.0.pdf>
- [12] Castryck, Decru. An Efficient Key Recovery Attack on SIDH. EUROCRYPT 2023.
- [13] Beullens. Breaking Rainbow Takes a Weekend on a Laptop. CRYPTO 2022.
- [14] Giacon, Heuer, Poettering. KEM Combiners. PKC 2018.
- [15] Bindel, Brendel, Fischlin, Goncalves, Stebila. Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange. PQCrypto 2019.
- [16] Barbosa, Connolly, Duarte, Kaiser, Schwabe, Varner, Westerbaan. X-Wing: The Hybrid KEM You've Been Looking For. IACR CiC, 2024.
- [17] Kret and Schmidt. The PQXDH Key Agreement Protocol. Signal, revision 3, 23 January 2024. <https://signal.org/docs/specifications/pqxdh/>
- [18] RFC 9794, Terminology for Post-Quantum Traditional Hybrid Schemes. IETF, June 2025.
- [19] Daemen, Rijmen. The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, 2002.
- [20] Gauravaram, Knudsen, Matusiewicz, Mendel, Rechberger, Schlaeffer and Thomsen. Groestl a SHA-3 candidate. Sections 3.4.4 and 3.4.5 specify the P512 shifts and MDS matrix. <https://www.groestl.info/Groestl.pdf>

- [21] Even, Goldreich. On the Power of Cascade Ciphers. ACM TOCS, 1985.
- [22] Maurer, Massey. Cascade Ciphers: The Importance of Being First. Journal of Cryptology, 1993.
- [23] Mennink, Szepieniec. XOR of PRPs in a Quantum World. PQCrypto 2017.
- [24] Grover. A Fast Quantum Mechanical Algorithm for Database Search. STOC 1996.
- [25] NIST, Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process (MAXDEPTH model). 2016.
- [26] DSTU 7624:2014, Information technologies - Cryptographic data security - Symmetric block transformation algorithm (Kalyna). Ukraine, 2014.
- [27] ISO/IEC 10118-3:2018, Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions. Includes Whirlpool.
- [28] Ferguson, Lucks, Schneier, Whiting, Bellare, Kohno, Callas, Walker. The Skein Hash Function Family, version 1.3, 2010. Specifies the Threefish-512 block cipher.
- [29] FIPS 198-1, The Keyed-Hash Message Authentication Code (HMAC). NIST, 2008.
- [30] FIPS 180-4, Secure Hash Standard (SHS). NIST, 2015. Specifies SHA-512.
- [31] RFC 10024, Post-Quantum Traditional (PQ/T) Hybrid Key Agreement Mechanisms for TLS 1.3. IETF, August 2026. <https://www.rfc-editor.org/rfc/rfc10024.html>
- [32] The Double Ratchet Algorithm, revision 4, 4 November 2025. Signal. Sections 5 and 6 specify the Sparse Post-Quantum Ratchet and Triple Ratchet. <https://signal.org/docs/specifications/doubleratchet/>
- [33] Apple Security Engineering and Architecture. iMessage with PQ3. Apple Security Research, 2024. <https://security.apple.com/blog/imessage-pq3/>

Entries are cited by number in the text. Where a submission rather than a published standard is cited, Requirement C1 binds the exact parameter set through SuiteID.

Addendum A - X-GM512-NATIVE: A Native 512-bit Symmetric Path

Status: RESEARCH ADDENDUM - NON-NORMATIVE

X-GM512-NATIVE is not currently a GM-3X512-PQS ProfileID and is not part of the interoperable specification. It is included to document ongoing research toward a fully native 512-bit symmetric path. Integration is conditional on the open cryptanalytic and proof requirements stated in this addendum.

Section numbers of the form §A.n refer to this addendum. Unprefixed section numbers refer to the main specification, §1 to §17 above.

A.1 Scope and motivation

This section defines an optional profile providing a native 512-bit combiner, key schedule, MAC and keystream path built from GM-512-BC-N. It removes the SHA3-512 and HMAC-SHA3-512 dependency from those four transport roles, and with it the residual KMAC256 use in confirmation. It does not remove Keccak from GM-512: SHA3-512 remains in use for h_T , the identity hashes, the Merkle prekey commitment, the signature digests and ReplayKey, and the PQC components use SHA3 and SHAKE internally.

What it changes. The main specification derives K_M with SHA3-512 and every session key with HMAC-SHA3-512, so its key-generation path is already 512 bits wide and no longer capped by a 256-bit keyed construction. What it still depends on is Keccak: SHA3-512 underlies the combiner, the derivation, h_T , the identity hashes and the Merkle commitment, and KMAC256 remains for authentication. This addendum removes Keccak from the transport's own combiner, key schedule, generator and MAC, replacing it with constructions over GM-512-BC-N. A hash built from a 512-bit block cipher is not bound by that ceiling: in the ideal-cipher model the generic classical bounds for Miyaguchi-Preneel are approximately 2^{256} for collision and 2^{512} for preimage. Both are classical generic figures and §A.3.2 gives the corresponding quantum values.

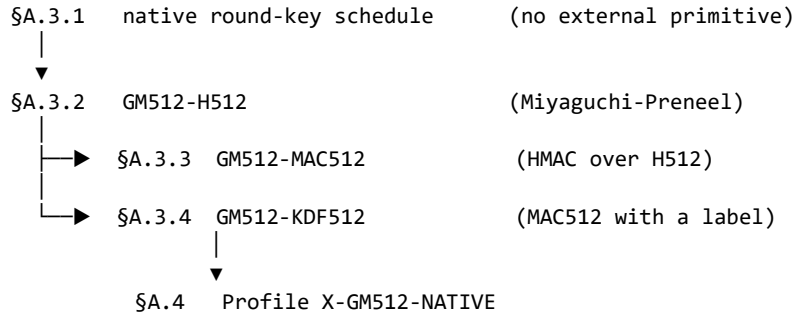
What it does not change. Grover's algorithm applies to key search, which is already infeasible at the 256-bit level: parallelisation degrades as the square root of the machine count, and NIST's MAXDEPTH framework [25 of the main specification] explicitly constrains the circuit depth available to a quantum attack. Raising a classical preimage bound from 2^{256} to 2^{512} moves between two figures that are both unreachable. The honest statement of the benefit is architectural uniformity and the removal of an external dependency, not a reachable increase in attack cost.

Why it is optional. In X-TRIPLE a cryptanalytic failure of GM-512-BC costs one keystream leg, with the HMAC-SHA-512 or HMAC-SHA3-512 generator still masking the plaintext and the tag still detecting forgery. In X-GM512 it costs confidentiality outright, since that profile has a single generator, and in X-TRIPLE-3RN it costs all three legs, which share the primitive. In the native profile the same failure would compromise the combiner, the key schedule, the authentication tag and the keystream at once, with nothing underneath. The profile is therefore specified as an addition, never as a replacement for S-DUAL512.

Naming. The native profile does not use GM-512-BC as the main specification defines it. §9.5 derives that cipher's round keys with HMAC-SHA3-512; the native schedule of §A.3.1 replaces them entirely, so the two are different keyed permutations - the all-zero vectors of §A.5.2 differ from those of §13.4. Throughout this section the native primitive is written GM-512-BC-N. An implementation MUST NOT treat the two as interchangeable, and the round function of §9.2 is the only part they share.

A.2 Dependency order

The current §9.5 derives GM-512-BC round keys with HMAC-SHA3-512. A native hash built on the cipher would therefore depend on the very function it is meant to replace. That circularity must be broken before anything else can be specified, and it fixes the order of the remaining work:



A.3 Native constructions

A.3.1 Native round-key schedule

The schedule applies the cipher's own round function to the key state, using fixed public round constants in place of round keys. This is a Whirlpool-inspired approach and it introduces no external dependency.

K_0 = the 512-bit cipher key, as an 8×8 byte state
 $RK_r = K_r$ for $r = 0 \dots 32$
 $K_{r+1} = \text{MixColumns}(\text{ShiftRows}(\text{SubBytes}(K_r \oplus RC_r)))$ for $r = 0 \dots 31$
 $RC_r[0][0] = x^r$ in $GF(2^8)$ mod $0x11B$
 $RC_r[0][c] = S[(8r + c) \bmod 256]$ for $c = 1 \dots 7$
 $RC_r[i][c] = 0$ for $i \geq 1$

Requirement NAT1. The round constants $RC_0 \dots RC_{31}$ MUST be pairwise distinct; only those thirty-two are consumed, since they produce $K_1 \dots K_{32}$. The multiplicative order of x modulo $0x11B$ is 51, so x^r is distinct across that range and the first byte alone separates every constant. An implementation MUST verify this at build time.

Requirement NAT2. RK_0 is the cipher key itself and is consumed by the `AddRoundKey` of round 0, which is the first operation of `Round_0` as §9.2 defines it. It is not a separate whitening step, and an implementation MUST NOT apply it twice. Thirty-three round keys serve thirty-two rounds plus the final `AddRoundKey`, so only $K_1 \dots K_{32}$ are generated.

Caveat. A key schedule that reuses the round function is the structure in which related-key attacks have historically been found. Whirlpool's schedule has absorbed twenty-five years of public analysis; this one has none. The §9.5 statement that no simple algebraic relation among round keys is known under the HMAC-SHA3-512 PRF assumption does not carry over and MUST be replaced by an explicit differential argument over the constant sequence.

A.3.2 GM512-H512

Merkle-Damgård over a Miyaguchi-Preneel compression function. The block size is 64 bytes and the digest is 64 bytes.

$H_0 = 0^{512}$
 $H_i = E_{\{H_{i-1}\}}(M_i) \oplus M_i \oplus H_{i-1}$ absorb
 $H = E_{\{K_{OUT}\}}(H_t) \oplus H_t$ finalize

$K_OUT = \text{"GM512-H512-OUT" padded to 64 bytes with zeros}$

padding: append $0x80$, then zero bytes until the length is
 $48 \bmod 64$, then a 16-byte big-endian bit count

Miyaguchi-Preneel is chosen over Davies-Meyer because it feeds the chaining value into the key input, so the message block never controls the key schedule directly.

Output transformation. Plain Merkle-Damgård is distinguishable from a random oracle: the digest is the final chaining value, so an adversary holding $H(m)$ can continue the chain. Appending one more ordinary compression call does NOT fix this - the result is simply the next chaining state and the adversary continues from it directly. The transformation must be structurally different from the chaining operation. Absorption keys the cipher with the chaining value; the output transformation instead keys it with the fixed public constant K_OUT and feeds H_t as the plaintext. The digest is therefore not a chaining state, and continuing the chain would require recovering H_t by inverting $x \rightarrow E_{\{K_OUT\}}(x) \oplus x$ for a known permutation, for which no generic shortcut is known. This prevents direct continuation from the published digest by hiding the internal chaining value. It does not establish indistinguishability and it does not remove the generic structural properties of Merkle-Damgård: multicollision and herding structure in the internal chain is unaffected by a fixed-key transformation on the final state. That is why N-A2 is stated only over the fixed-length combiner domain rather than as a global random-oracle claim, and §A.7 records the missing argument.

Generic bounds. With a 512-bit digest the classical generic figures are 2^{256} for collision by the birthday bound and 2^{512} for preimage. Both are classical. Against a quantum adversary the generic figures are lower: Grover-style preimage search is on the order of 2^{256} , and quantum collision search in the standard query model is below 2^{256} , around $2^{(512/3)}$ at the cost of very large memory. Because §11 defines the adversary as quantum polynomial-time, these values MUST be read as classical generic bounds and the quantum figures quoted alongside them.

A.3.3 GM512-MAC512

A raw block cipher cannot authenticate a multi-megabyte M_AUTH , and a keyed Merkle-Damgård prefix is vulnerable to length extension. MAC512 is therefore HMAC instantiated over GM512-H512:

$K' = K$ padded to 64 bytes, or $H512(K)$ if $|K| > 64$
 $ipad = 0x36$ repeated 64 times
 $opad = 0x5c$ repeated 64 times

$MAC512(K, M) = H512((K' \oplus opad) \parallel H512((K' \oplus ipad) \parallel M))$

A.3.4 GM512-KDF512

$KDF512(K, L, ctx) = MAC512(K, ENC(L, ctx))$

L is a domain-separation label from the registry of §13.1 and ctx is the binding context, normally h_T . The canonical encoding of §2 is mandatory here and raw concatenation is prohibited: the label set is not prefix-free, so $KDF512(K, \text{"GM512-ENC-A", ""})$ and $KDF512(K, \text{"GM512-ENC", "-A"})$ would otherwise be the same derivation. Deriving from a PRF rather than from a raw hash preserves the key-separation argument that the current §6 relies on.

A.4 Profile X-GM512-NATIVE

The table below maps each transport role from what the existing profiles use today to what this profile substitutes; the middle column is there so the substitution is explicit, not because those constructions are part of X-GM512-NATIVE. The profile replaces those primitives in the combiner, key schedule, record generator, authentication tag and confirmation roles. The KEM and signature layers are unchanged. X-GM512-NATIVE is a single-generator profile

and does not use the wrapped roots of §7.8. Native multi-generator variants are already defined in the main specification as X-TRIPLE-3RN, three GM-512-BC-N legs, and X-TRIPLE-3RH, one GM-512-BC-N leg alongside Threefish-512 and Kalyna-512/512; both use the §9.5a schedule and neither uses the native combiner, KDF or MAC defined here.

Transport role	Existing profiles (S-DUAL512, S-KECCAK, X-*)	X-GM512-NATIVE
Combiner (§5.5)	SHA3-512(D_MASTER $z_1 \cdot \cdot z_4$ h_T)	GM512-H512(ENC(D_MASTER, z_1, z_2, z_3, z_4, h_T))
Key schedule (§6)	HMAC-SHA3-512(K_M, ENC(L, h_T))	GM512-KDF512(K_M, L, h_T)
Keystream (§7)	HMAC-SHA-512 and HMAC-SHA3-512 (S-DUAL512, X-TRIPLE), or in the other X-profiles GM-512-BC, GM-512-BC-N, Threefish-512, Kalyna-512/512	GM-512-BC-N in CTR, 512-bit blocks
Tag (§7.3)	KMAC256(K_A, M_AUTH, 512)	GM512-MAC512(K_A, M_AUTH)
Confirmation	KMAC256(K_C, ..., 256 bits)	GM512-MAC512(K_C, ...) truncated to 256 bits

Requirement NAT6. For consistency with §5.5 of the main specification, an absent z_3 or z_4 MUST be represented by a zero string of that leg's fixed length (§5.5) before ENC is applied. It MUST NOT be encoded as an empty field. FIELD(0^{32}) and FIELD("") are different byte strings and would derive different master secrets, so the two encodings are not interoperable and the zero-filled form is normative.

Requirement NAT5. Tag_confirm in this profile is the leftmost 256 bits of GM512-MAC512(K_C, ENC("confirm", h_T)). Truncation is mandatory so that C_CONFIRM keeps the 32-byte tag field of §5.2.1 and the wire format is unchanged. A verifier MUST compare the full 32 bytes in constant time and MUST NOT accept a shorter comparison.

Requirement NAT3. X-GM512-NATIVE is OPTIONAL and EXPERIMENTAL. A ProfileList or ProfileSet advertising it MUST also contain S-DUAL512, per Requirement P4. It MUST NOT be used to protect production data pending third-party cryptanalysis of GM-512-BC-N.

Requirement NAT4. When ProfileID = X-GM512-NATIVE, the selected transport path MUST NOT invoke SHA3-512, HMAC-SHA3-512 or KMAC256 for the combiner, the key schedule, the record generator, the tag or the confirmation. Offering S-KECCAK alongside it in a ProfileList or ProfileSet is permitted and expected: the profiles are alternatives, and a session that selects X-GM512-NATIVE is KMAC-free at the transport level regardless of what else was offered.

To restate the scope precisely: the profile removes SHA3-512, HMAC-SHA3-512 and KMAC256 from the combiner, the key schedule, the record generator, the tag and the confirmation. SHA3-512 remains in use elsewhere in the transport, and the PQC components use SHA3 and SHAKE internally. This is not a Keccak-free protocol and the section does not claim to be one.

A.5.1 Native round keys, $K = 00 \times 64$

A.5.2 Cipher under the native schedule

A.5.3 GM512-H512

A.5.4 GM512-MAC512 and GM512-KDF512

Algorithm: GM-3X512-PQS – Page: 69

A.6 Security assumptions and what must be rewritten

For a session with ProfileID = X-GM512-NATIVE, the role played by assumption A3 is taken by N-A1 to N-A3 below. A3 itself is unchanged and continues to govern S-DUAL512, S-KECCAK, X-GM512, X-TRIPLE, X-TRIPLE-3R, X-TRIPLE-3RN and X-TRIPLE-3RH. Note that X-TRIPLE-3RN and X-TRIPLE-3RH still rely on A3 even though their keystream generators use no Keccak: SHA3-512 wraps their roots, HMAC-SHA3-512 derives their session keys and KMAC256 computes their authentication tag. Each assumption is stated against a quantum polynomial-time adversary with classical query access:

Identifier	Native construction assumption
N-A1	GM-512-BC-N with the §A.3.1 schedule is a secure PRP under a random secret key. This is what the CTR keystream needs, since K_E is secret and pseudorandom.
N-A1'	For the analysis of GM512-H512, GM-512-BC-N is modeled as an ideal cipher: for every 512-bit key K , E_K is an independently sampled permutation, and the adversary has oracle access to E_K and its inverse for keys of its choosing. This is what Miyaguchi-Preneel needs and it does NOT follow from N-A1.
N-A2	For the combiner reduction, GM512-H512 restricted to the fixed-length, domain-separated input of §5.5 is modeled as a random oracle. This is an independent heuristic assumption; it is not implied by N-A1 or N-A1', and it is deliberately narrower than a general random-oracle claim, which the Merkle-Damgård structure would not support even with the output transformation.
N-A3	GM512-MAC512 is a secure PRF and existentially unforgeable. This is an independent assumption and no reduction from N-A2 is claimed: N-A2 covers only the fixed-length combiner input, whereas MAC512 processes an arbitrary-length M_{AUTH} . An HMAC-style reduction for the §A.3.2 construction would allow this to be weakened later.

Why N-A1 is not enough. Miyaguchi-Preneel evaluates $E_{\{H_{i-1}\}}(M_i)$, so the chaining value is the cipher key. In a hash computation the key is public, and an adversary searching for a collision chooses message blocks that steer it. Standard PRP security quantifies over a random secret key and says nothing about behaviour when the key is attacker-influenced. The correct setting is the ideal-cipher model, and the collision and preimage bounds of §A.3.2 are the generic bounds in that model rather than consequences of N-A1. Any related-key or known-key distinguisher on GM-512-BC-N would invalidate them directly.

Cross-role correlation. N-A1, N-A1', N-A2 and N-A3 are listed separately but they are not independent. The same round function and the same §A.3.1 key schedule appear in every role: as a PRP keying the CTR keystream, as an ideal cipher inside the compression function, and transitively inside the MAC and the KDF. A single structural weakness - a differential in the round function, a related-key relation in the schedule, a known-key distinguisher - would degrade several of these assumptions at once. The assumption list must therefore be read as one assumption about GM-512-BC-N viewed from four angles, not as four hedges. This is the central reason the profile is optional.

Every proof in §12 that models SHA3-512 as a random oracle or HMAC-SHA3-512 as a PRF must be restated over these assumptions for this profile. That is Theorem 1's combiner argument and the AEAD reduction of Theorem 3. Theorem 4 does not apply to X-GM512-NATIVE, which is a single-generator profile. The main specification separately applies Theorem 4 to the multi-generator X-TRIPLE-3RN and X-TRIPLE-3RH profiles, both of which already use GM-512-BC-N as a keystream generator.

Non-claims. GM512-H512, GM512-MAC512 and GM512-KDF512 have received no third-party analysis. The 512-bit preimage bound is generic and assumes the compression function behaves ideally; a structural weakness in GM-512-BC-N or in the native schedule invalidates it. No claim is made that the native profile is stronger than the mandatory path in any reachable sense.

A.7 Open work

In dependency order:

1. A differential and linear trail argument for the §A.3.1 constant sequence, and an explicit related-key analysis of the schedule.
2. A security model for GM-512-BC-N in the chosen-key and known-key settings, which is what Miyaguchi-Preneel requires and what N-A1 does not provide. This is the single most substantive open item in the native design.
3. Freeze the definition of GM512-H512 before restating anything: an indistinguishability argument for the §A.3.2 output transformation, or a documented decision to claim only the narrow N-A2. Then restate Theorem 1 and Theorem 3 over N-A1, N-A1', N-A2 and N-A3. Theorem 4 is out of scope for a single-generator profile.
4. Regeneration of every §13 vector for the native profile, and a decision on whether Version advances.
5. Third-party cryptanalysis of GM-512-BC-N, which the native profile makes load-bearing for the entire symmetric layer rather than for one keystream leg.

A.8 Integration into the main specification

This addendum is included in this paper but remains non-normative and is not part of the interoperable GM-3X512-PQS specification. The main specification has since absorbed the two constructions this section depends on: §9.5a defines GM-512-BC-N with the native round-key schedule of §A.3.1, and §7.9 and §7.10 define the X-TRIPLE-3RN and X-TRIPLE-3RH profiles that use it as a keystream generator. What remains unmerged is the native combiner, key schedule and MAC - GM512-H512, GM512-KDF512 and GM512-MAC512 - which no profile in the main specification yet uses. Adding X-GM512-NATIVE therefore requires the following coordinated edits:

1. §5.2 - add "X-GM512-NATIVE" to the ProfileID enumeration, raise the P5a bound from 7 to 8, and update every prose statement of the profile count, including the abstract.
 - 1a. §5.5 - the combiner formula is unconditional today; make it conditional on ProfileID and add the GM512-H512 case.
 - 1b. §6 - the same for the key schedule and for Tag_confirm, including the NAT5 truncation.
 - 1c. §7.2 - add the native generator to the 64-byte counter-block cases alongside S-KECCAK and X-GM512.
 - 1d. §7.3 - the tag formula is unconditional today; add the GM512-MAC512 case.
 - 1e. §7.5 and §7.5a - the sender and receiver steps that compute K_M, K_x, T and Tag_confirm name KMAC256 directly and must branch on ProfileID.
 - 1f. §11.3 - make A3 conditional on a non-native ProfileID, per §A.6.
 - 1g. P4 - no structural change required: its rule already covers any experimental X-* profile automatically, including X-GM512-NATIVE.
 - 1h. Abstract, §3.1, §3.2 and §16 - each states what caps the construction. Qualify these to the non-native profiles.
2. P5a - raise the bound on n and m from 7 to 8, and update the count-consistency rule accordingly.
3. §7.1 - add the profile row, with status OPTIONAL and EXPERIMENTAL, and note that it is a single-generator profile, distinct from the multi-generator X-TRIPLE-3RN which already uses GM-512-BC-N.

4. A5 - add the X-GM512-NATIVE case: GM-512-BC-N is a secure PRP under N-A1, which is what the keystream needs since K_E is secret; the native combiner additionally relies on N-A1' and N-A2.
5. §13.1 - register K_{OUT} , the output-transformation constant of §A.3.2. KDF512 consumes the existing label registry through ENC and needs no separator of its own.
6. §10 - no wire-format change, given Requirement NAT5: the tag T is already 512 bits and Tag_confirm is truncated to 256, so every field keeps its width. Without that truncation C_CONFIRM would grow by 32 bytes.
7. §12 - Theorem 1 and Theorem 3 are stated over A3 and do not transfer; each needs a restatement over N-A1, N-A1', N-A2 and N-A3. Theorem 4 is unaffected, since this is a single-generator profile.
8. §15 - add the open items of §A.7 below to the open-challenges list.
9. §16 - qualify the capping claim, for example: for the standardized and existing experimental profiles the claimed level is capped by the 256-bit KEM shared secrets, and by KMAC256 authentication; X-GM512-NATIVE is governed by N-A1 to N-A3 and makes no claim above the overall 256-bit target. The mandatory-path claim itself is unaffected, since S-DUAL512 remains mandatory.

Recommendation. Do not merge until items 1 to 4 of §A.7 are complete. Until the security model for GM-512-BC-N → Miyaguchi-Preneel → HMAC is settled, the profile is a research contribution and publishing it as a separate section is the accurate presentation.