

A Conditional Proof of Pre-Use Root Key Indistinguishability for the Auris v5 Handshake

Max Von

The Auris Project

nftmago@gmail.com

22 July 2026

Abstract

Auris is a private messaging and calling application whose next protocol version, v5, derives every session secret from a single root key established during the handshake. This paper proves, under explicitly named assumptions, that in a precisely defined game model of the handshake this root key is indistinguishable from random at the moment it is handed to the rest of the protocol; the obligations that map the deployed system onto the games are stated exactly and are not discharged here. The analyzed construction is byte-exact X-Wing draft-10, combined with a SHA-384 commitment to a canonical 44-field transcript and a direct one-block HKDF-Expand instantiated by HMAC-SHA256. The proof proceeds in three layers: a transfer of the published X-Wing theorems to the exact draft-10 byte ordering; an accounting of draft-10's shared-seed key generation by two PRG hops; and a reduction for the root adapter, first in a staged single-user experiment and then in a static-slot multi-user experiment with total oracle semantics, losing exactly the number of key and session pairs in that fixed one-Test experiment. The classical fallback branch is stated in one globally defined random-oracle model that fixes oracle sharing, role input domains, the idealization boundary, and query counting across the SHA3-256, SHA3-512, and SHAKE256 roles of draft-10 and FIPS 203. The post-quantum theorem makes no hardness assumption on X25519 or Ed25519. A separate correctness lemma bounds the responder's root-agreement failure. We also prove a negative result: the raw root cannot be tested for real-or-random security once a root-derived verification artifact that distinguishes the real root from a random one is visible. Authentication, ratchet composition, and implementation conformance are outside scope and are stated as such.

Contents

1	Introduction	1
1.1	Contributions	1
1.2	Related work	1
1.3	Scope and evidence status	2
1.4	Organization	2
2	Claim, boundary, and source versions	2
2.1	Claim proved	2
2.2	Claim not proved	3
2.3	Pinned external sources	3
3	Security definitions	4
3.1	Advantage convention	4
3.2	KEM IND-CCA	4
3.3	Classical-query PRF security	4
3.4	PRG security	4
3.5	C2PRI	4
3.6	The global classical random-oracle model	4
3.7	Single-user pre-use root game	6
4	Exact construction	6
4.1	X-Wing draft-10 key generation	6
4.2	Exact combiner	7
4.3	Auris direct adapter	7
5	Explicit assumption dispositions	7

6	Exact X-Wing transfer lemmas	9
6.1	Shared-seed endpoint hops	9
6.2	Post-quantum QSF instantiation	9
6.3	Classical random-oracle input order	10
6.4	Final FIPS 203 C2PRI	10
7	Root-adapter transform	11
8	Complete single-user theorems	12
9	Static-slot adaptive multi-user lift	13
9.1	Static slot space	13
9.2	Complete multi-user experiment	13
9.3	Fixed-slot reduction	14
10	System realization corollary	18
11	Transcript and context binding	20
12	Post-use raw-root claim is false	21
13	Concrete loss and query ledger	21
13.1	Reduction resource ledger	22
14	Exact implementation obligations	23
15	X25519 all-zero behavior: statement and evidence	24
16	Theorem ledger	25
17	Canonical transcript profile	26
18	Vector and parser reproduction summary	27
19	Companion result: a machine-checked model of the v5 authentication layer	28
20	Final statement	29
A	Reduction pseudocode	29
A.1	Outer KEM-to-root reduction	29
A.2	PRF reduction for the direct adapter	29
A.3	Static-slot target-guess reduction	30
B	Compact game ledgers	31
B.1	Post-quantum single-user chain	31
B.2	Classical single-user chain	31
C	Artifact, versioning, and provenance	31
C.1	Artifact bundle	31
C.2	Versioning and provenance	32

1 Introduction

Messaging systems are converging on hybrid post-quantum key exchange: a lattice KEM and an elliptic-curve group combined so that the session secret survives a break of either component. X-Wing [1, 2] is a general-purpose hybrid KEM built from ML-KEM-768 [4] and X25519 [6], with published security theorems in the QSF framework (the hybrid-KEM security framework of Barbosa et al. [1]), and it is the KEM this paper builds on. Between a published theorem and a deployed protocol, however, sits a layer of exact choices: the byte ordering of the draft the implementation actually follows, the way one seed is expanded into both component keys, the key-derivation step that turns the KEM output into the protocol’s working secret, and the transcript that this step binds. Experience, including our own during the preparation of this proof, is that errors concentrate in exactly these joints rather than in the primitives. This paper addresses exactly those joints for one design.

The system under analysis is Auris, a private messaging and calling application. Its v5 handshake performs an X-Wing encapsulation against the peer’s public key and derives a 256-bit value `rootKey0`, the root key, by a single HKDF-Expand block keyed with the X-Wing shared secret over a message that binds a protocol suite identifier and a SHA-384 commitment to the full 44-field handshake transcript. The protocol event at which this value is handed to the rest of the system is called `RootReady`. Every later secret of a v5 session, message keys, call keys, and ratchet state, is derived from this root. The claim proved here is narrow, and it is the value every later key derives from: in the game model of this paper, at the `RootReady` handoff and before any root-derived value is exposed, `rootKey0` is computationally indistinguishable from a uniform 256-bit string. The obligations that connect the running system to the games are stated in the system realization section and are not discharged here.

1.1 Contributions

1. **Exact-bytes transfer of the X-Wing theorems.** The published proofs are stated for the paper’s formalization; deployments follow the draft’s byte layout. We name the draft-10 keyed family $F_k^{10}(x) = \text{SHA3-256}(k \parallel x \parallel L_{XW})$ explicitly, transfer the post-quantum theorem to it, and transfer the classical theorem through a random-oracle input-permutation lemma (Section 6), rather than treating a reordered SHA3 input as the same primitive. The classical branch’s random-oracle model is defined once, globally (Definition 3.5): which evaluations share one oracle and which oracles are independent across the SHA3-256, SHA3-512, and SHAKE256 roles of draft-10 and FIPS 203, the exact honest input domain of every role, the by-role idealization boundary for SHAKE256, and how q_H , q_G , and q_J count every query of the composed experiments.
2. **Shared-seed accounting.** Draft-10 expands one 32-byte seed by SHAKE256 into both component key-generation inputs. We prove that replacing this by independent component randomness costs two endpoint PRG hops, and we distinguish the deployed correctness parameter from the standardized one: the deployed failure event is bounded by $\delta_{seed} \leq \delta_M + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_{corr}}^{\text{PRG}}$, which is not the same object as the ML-KEM parameter δ_M stated for native key-generation randomness (Section 10).
3. **The root adapter, single-user and multi-user.** We prove the direct one-block HKDF-Expand adapter in a staged single-user experiment whose root-decapsulation oracle is available before and after the test query, then lift it to a static-slot multi-user experiment with total oracle semantics and a complete per-interface, per-case simulation table, losing exactly $N_T = N_K N_S$ in that fixed, static-slot, one-Test experiment (Sections 7 and 9).
4. **A negative result.** Once a root-derived verification artifact that distinguishes the real root from a random one is visible, the raw root can no longer be tested for real-or-random security; we give the distinguisher (Section 12). This bounds what any proof of this shape can promise and motivates the pre-use boundary.
5. **Conformance evidence and artifacts.** An empirical section documents the X25519 all-zero-output behavior across implementations, where the draft’s total function and common library behavior genuinely diverge (Section 15). Byte-frozen test vectors, a reproduction suite, and a machine-checked symbolic model of the accompanying v5 authentication layer are published with the paper (Appendix C).

1.2 Related work

X-Wing’s design and QSF security theorems are due to Barbosa, Connolly, Duarte, Kaiser, Schwabe, Varner, and Westerbaan [1]; the corresponding specification is the draft series [2], and a general framework for hybrid

KEM combiners in the IETF context is developed in [3]. The combiner-security literature relevant here begins with Giacon, Heuer, and Poettering [10], whose split-key PRF viewpoint underlies why the direct single-KEM adapter of this paper needs, and gets, a dedicated proof rather than an appeal to a generic combiner theorem. Quantum security of the HMAC construction itself is treated by Hosoyamada and Iwata [8]. Deployed hybrid handshakes in messaging predate X-Wing; Signal’s PQXDH combines X25519 with ML-KEM at the handshake layer and documents the accompanying design constraints [9]. This paper differs from those works in scope: it carries the imported theorems through one exact byte-level construction to the root value of the paper’s pre-use game; the mapping from the deployed system to that game is stated as explicit obligations and is not proved here.

1.3 Scope and evidence status

The reductions in this paper are conventional, non-mechanized conditional proofs for which the author takes responsibility; they are not formalized in a proof assistant. What is machine-checked is narrower and stated exactly where it applies: the test vectors and transcript objects reproduce byte-exactly under the published suite; the multi-user simulation table’s restatements are checked for guard consistency by a mechanical evaluator; and the companion model of the v5 authentication layer is machine-checked in Tamarin as a symbolic result (Appendix C). Before publication this paper underwent repeated internal adversarial review, conducted with substantial AI assistance and archived by the project; no independent human expert review has occurred yet, and this preprint is the request for exactly that scrutiny. The proof is conditional on the assumptions named in Section 5; authentication, agreement, replay and rollback safety, erasure, post-use composition, code conformance, and side channels are separate obligations outside this paper’s theorems.

1.4 Organization

Section 2 states the claim, its boundary, and the pinned source versions. Security definitions and the exact construction follow, then the assumption dispositions, the imported transfer lemmas, the adapter and the single-user theorems, the multi-user lift, the system realization corollary, the transcript profile, the negative result, the concrete accounting, the implementation obligations, the X25519 conformance evidence, the theorem ledger, and the reproduction summary. Appendices contain the reduction pseudocode, the compact game ledgers, and the artifact and provenance record.

Main result

For the exact single-user and static-slot multi-user pre-use games defined in this paper, the Auris root key is pseudorandom under the displayed reductions and the assumptions of Section 5 (Theorems 8.1, 8.2, 9.3, and 9.4). This is a conditional mathematical result; its review status is described in Section 1.3. No end-to-end or post-use raw-root claim follows.

2 Claim, boundary, and source versions

Throughout, “complete” means that every transition from the pre-use root game to the named primitive assumptions is stated as a game hop or reduction, with exact challenge exclusions, query interfaces, loss factors, and endpoint distributions; it does not mean unconditional. The project labels this claim R2 internally; the label appears in the theorem identifiers of Section 16 and in the artifact.

2.1 Claim proved

The claim is deliberately narrow. Let an honest X-Wing draft-10 encapsulation produce a public ciphertext ct^* and shared secret K^* . Let τ^* be a transcript byte string containing that ciphertext (the honest protocol supplies the exact canonical Auris transcript; the game itself imposes only byte containment, Definition 3.7), and define

$$H_T(\tau) = \text{SHA-384}(\text{“auris/handshake-transcript/v5”} \parallel 00 \parallel \tau), \quad (1)$$

$$M(\tau) = \text{“auris/root/v5/xwing-d10”} \parallel 00 \parallel \text{u32}_{BE}(\text{0x00050001}) \parallel H_T(\tau) \parallel 01, \quad (2)$$

$$R(K, \tau) = \text{HMAC-SHA256}(K, M(\tau)). \quad (3)$$

The positive statement is that $R(K^*, \tau^*)$ is computationally indistinguishable from a uniform 256-bit string at the abstract **RootReady** handoff, before any target-root-dependent value is exposed.

Party scope. K^* is the *encapsulator's* stored shared secret. The responder computes its root from $XW^{10}.\text{Dec}(sk, ct^*)$, which equals K^* except when X-Wing decapsulation disagrees with encapsulation, an event bounded by δ_{seed} per honest session under the exact shared-seed key-generation distribution (Definition 10.1; $\delta_{seed} \leq \delta_M + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_{corr}}^{\text{PRG}}$ by Lemma 10.3) and charged by the system lift as Bad_{corr} with the adaptive union bound $N_T \delta_{seed}$ per branch. The games below test the encapsulator's object; every statement about the responder's root routes through that correctness event. Nothing inside the imported IND-CCA reductions substitutes for this accounting: the $2\delta_M$ term of Lemma 6.3 pays for two hybrid transitions inside a different game and says nothing about the real responder's output.

2.2 Claim not proved

The theorem does not establish any of the following:

- that a transcript is authorized by the intended human identity;
- non-injective or injective agreement, key-compromise impersonation (KCI), unknown key share (UKS), replay uniqueness, downgrade resistance, or revocation visibility;
- atomic acceptance, crash safety, rollback exclusion, one-time-key erasure, or forward secrecy;
- security of the first AEAD, headers, acknowledgements, ratchet, message stream, or metadata after the root is used;
- malicious-key binding beyond the fresh honestly generated target-key game;
- parser, backend, language, code, timing, cache, or side-channel conformance.

Boundary

The target experiment terminates at **RootReady**. It gives the adversary no AEAD ciphertext, tag, acknowledgement, header, ratchet output, confirmation value, or other artifact computed from the target root or a descendant. Section 12 proves that a raw-root real-or-random test becomes false once a root-derived AEAD verification artifact is visible.

2.3 Pinned external sources

The exact construction and imported results are tied to:

- X-Wing Internet-Draft **draft-connolly-cfrg-xwing-kem-10**, dated 2 March 2026 [2];
- Barbosa et al., “X-Wing: The Hybrid KEM You’ve Been Looking For,” IACR Communications in Cryptology, 2024 [1];
- CFRG Hybrid PQ/T KEMs draft-12, dated 6 July 2026, for the shared-seed PRG rationale [3];
- final NIST FIPS 203 for ML-KEM Algorithms 14–21 and its correctness parameter [4];
- RFC 5869 for the exact one-block HKDF-Expand evaluation [5]; and
- Hosoyamada and Iwata for the cited quantum HMAC analysis [8].

Internet-Drafts are works in progress. A change to the X-Wing draft, byte ordering, key-generation rule, all-zero behavior, suite identifier, transcript encoding, or adapter requires a new proof identifier and new vectors. Draft-10 Appendix C is explicitly marked for replacement of its test vectors. The source hash and draft identifier pin the present bytes, but any later draft or replacement vector set requires a new source lock, vector package, manifest, and proof disposition. The final FIPS 203 PDF is pinned by SHA-256 **fe1f12f32a7e44ec9fdebbf400cda843a40b506dee676725234dc6f7923b6cac** (the pinned primary-source lock, printed in full here and in **SOURCE_LOCK.md**); NIST’s published FIPS 203 errata sheet is enumerated and dispositioned in **SOURCE_LOCK.md**; its two entries are editorial and touch no algorithm, oracle structure, or correctness parameter used by this proof.

3 Security definitions

3.1 Advantage convention

Every advantage is an absolute difference of output-one probabilities. For a two-game notion Game_b ,

$$\text{Adv}_{\mathcal{A}} = \left| \Pr[\text{Game}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\text{Game}_1^{\mathcal{A}} \Rightarrow 1] \right|.$$

A “post-quantum, classical-query” adversary is quantum polynomial time (QPT) but exchanges only classical bit strings with protocol, KEM, PRG, and PRF interfaces. Superposition access is not granted unless stated explicitly.

3.2 KEM IND-CCA

Definition 3.1 (KEM IND-CCA). *The challenger samples $(sk, pk) \leftarrow \text{KeyGen}$, computes $(K_0, ct^*) \leftarrow \text{Enc}(pk)$, samples K_1 uniformly from the KEM key space, and gives (pk, ct^*, K_b) to the adversary. The adversary may query $\text{Dec}(sk, ct)$ for every $ct \neq ct^*$; the decapsulation oracle is available for the whole experiment, both before and after the adversary receives the challenge tuple. Its advantage is the absolute difference between the $b = 0$ and $b = 1$ output probabilities.*

3.3 Classical-query PRF security

Definition 3.2 (PRF-cq). *Let $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$. A challenger samples $k \leftarrow \mathcal{K}$. In the real game, $\text{Eval}(x) = F_k(x)$. In the random-function game, Eval is a lazily sampled function from $\mathcal{X}$ to $\mathcal{Y}$. The adversary may compute quantumly but sends and receives only classical oracle strings.*

The classical PRF-cc notion is the same with a classical PPT adversary.

3.4 PRG security

Definition 3.3 (PRG-cq). *For $S : \{0, 1\}^{256} \rightarrow \{0, 1\}^{768}$, the real distribution is $S(s)$ for uniform $s \in \{0, 1\}^{256}$; the random distribution is uniform in $\{0, 1\}^{768}$. A QPT distinguisher receives one classical challenge string. The classical notion restricts the distinguisher to PPT.*

For draft-10, $S(s) = \text{SHAKE256}(s, 96 \text{ bytes})$.

3.5 C2PRI

Definition 3.4 (Ciphertext second-preimage resistance). *A challenger generates (sk, pk) , computes $(K^*, c^*) \leftarrow \text{Enc}(pk)$, and gives (sk, pk, K^*, c^*) to the adversary. The adversary wins by outputting $c \neq c^*$ such that $\text{Dec}(sk, c) = K^*$.*

The secret key is intentionally given to the adversary. C2PRI is not a confidentiality notion.

3.6 The global classical random-oracle model

Two classical-branch components are stated in a random-oracle model (ROM): the imported combiner theorem (Lemma 6.6) and the C2PRI bound (Proposition 6.7). Draft-10 and FIPS 203 reuse the same named primitives in several roles: FIPS 203 defines H as SHA3-256 and J as SHAKE256, draft-10 uses SHA3-256 as its combiner and SHAKE256 for seed expansion, and K-PKE uses SHAKE256 again for noise derivation. The model is therefore defined once, globally: which evaluations share one oracle, which oracles are independent, the exact honest input domain of every role, where the idealization boundary lies for each named primitive, and how queries are counted. The post-quantum branch uses no random oracle; nothing in this subsection is a premise of Theorems 8.1 and 9.3.

Definition 3.5 (Global classical random-oracle model). *The classical-branch experiments are executed with the following three ideal oracles, sampled independently of one another at the start of the experiment and available to every algorithm in the experiment: the adversary, the honest protocol algorithms, and every reduction.*

1. $\mathcal{O}_H : \{0, 1\}^* \rightarrow \{0, 1\}^{256}$ idealizes SHA3-256 as one shared oracle for both of its protocol roles: the X-Wing combiner role (honest inputs: exactly the 134-byte strings $K_M \parallel K_X \parallel ct_X \parallel pk_X \parallel L_{XW}$ of Equation (4), one evaluation per encapsulation and per decapsulation) and the FIPS 203 role H (honest inputs: exactly the 1184-byte ML-KEM-768 encapsulation keys, evaluated once at key generation and once per encapsulation; decapsulation reuses the stored hash). The two roles' honest input lengths differ, so their honest query sets are disjoint, and the restrictions of $\mathcal{O}_H$ to the 134-byte combiner domain and to the 1184-byte FIPS 203 H domain have the same joint distribution as two independent random functions on those restricted domains. The equivalence claimed here concerns only these restricted domains; no equivalence with two unrestricted, independently queryable full-domain oracles is asserted, and every argument in this paper that separates the two roles uses only the restricted-domain statement. The input-order transfer applies the rotation ρ of Lemma 6.4 to $\mathcal{O}_H$; ρ is the identity on every input length other than 134 bytes, so every FIPS 203 H evaluation is untouched by the reordering.
2. $\mathcal{O}_G : \{0, 1\}^* \rightarrow \{0, 1\}^{512}$ idealizes SHA3-512, whose only protocol appearances are the FIPS 203 role G : the 33-byte key-generation inputs $d \parallel k$ (with k the one-byte rank encoding) and the 64-byte inputs $m \parallel H(ek)$ of encapsulation and of decapsulation's re-derivation.
3. $\mathcal{O}_J : \{0, 1\}^* \rightarrow \{0, 1\}^{256}$ idealizes the FIPS 203 role J of SHAKE256 (implicit rejection, $\bar{K} = J(z \parallel c)$; honest inputs: exactly the 1120-byte strings $z \parallel c$, one evaluation per decapsulation).

$\mathcal{O}_H$, $\mathcal{O}_G$, and $\mathcal{O}_J$ are pairwise independent; the premise “independent G, J random oracles” of Proposition 6.7 is exactly this declaration.

The idealization boundary for SHAKE256 is by role, not by primitive. SHAKE256 has three protocol roles with pairwise distinct honest input lengths: the J role (1120-byte inputs, 32-byte outputs; idealized as $\mathcal{O}_J$), the draft-10 seed expansion (32-byte inputs, 96-byte outputs; the PRG of Definition 3.3; concrete), and the PRF_η noise derivation inside deterministic K-PKE (33-byte inputs, 128-byte outputs for ML-KEM-768; concrete). Only the J role is idealized. The model makes no ideality claim about concrete SHAKE256 on any input: $\mathcal{O}_J$ is an independent ideal object, and its values, including on 32- or 33-byte query strings, are unrelated to concrete SHAKE256 by definition of the model. Because the three roles' honest input lengths are pairwise distinct, no honest evaluation point is both idealized and concrete. The separation is by input length alone; output-length differences separate nothing for an extendable-output function, whose shorter output is a prefix of its longer output at the same input, and no such separation is claimed. This per-role idealization is the model's declared heuristic residue: a real adversary evaluates one concrete SHAKE256 everywhere, and the model does not capture cross-role consistency between $\mathcal{O}_J$ and the concrete seed expansion. Remark 3.6 records why the PRG assumption composes soundly with the ideal oracles, and the alternative treatment that is deliberately not taken.

Not idealized. HMAC-SHA256 (PRF assumption), SHA-384 (collision resistance, Lemma 11.1 only), X25519 (nominal-group distance and SDH), SHAKE128 (the K-PKE matrix XOF, a different primitive, inside deterministic K-PKE), and the two concrete SHAKE256 roles above.

Query counting. q_H , q_G , and q_J count every query to $\mathcal{O}_H$, $\mathcal{O}_G$, and $\mathcal{O}_J$ in the experiment under consideration, regardless of originator: the adversary's direct queries, the queries honest algorithms make while answering protocol interfaces, and the queries reductions make internally while simulating. A statement that counts a strict subset says so in place; Proposition 6.7 counts the C2PRI adversary's queries, direct and induced, and carries the win-check decapsulation's two queries in its additive constant, so substituting any larger count is conservative. Each exact draft-10 operation induces a fixed pattern: one decapsulation queries $\mathcal{O}_G$ once, $\mathcal{O}_J$ once, and $\mathcal{O}_H$ once (the combiner); one honest encapsulation queries $\mathcal{O}_G$ once and $\mathcal{O}_H$ twice (the 1184-byte $H(ek)$ input and the 134-byte combiner input); one key generation queries $\mathcal{O}_G$ once and $\mathcal{O}_H$ once. Draft-10's literal decapsulation re-expands the packed seed, repeating the key-generation queries at their original points. The constructed reductions run an extensionally equivalent memoized implementation that retains expanded keys, so those repeated key-expansion calls are not performed, and q_H , q_G , and q_J count the actual oracle invocations of that implementation (a repeated query at a previously queried point would in any case change no lazily sampled answer). Section 13 turns these patterns into the per-reduction totals used by Theorems 8.2 and 9.4.

Joint-model composition. The composed classical theorems hold in this one joint model. Each imported or local random-oracle statement is proved for adversaries that may also access the other, independent oracles: independent ideal oracles can be lazily sampled by a simulator, so their presence changes no advantage term. The imported combiner-oracle proof is oblivious to whether its C2PRI term is realized in the standard model or by Proposition 6.7's ($\mathcal{O}_G, \mathcal{O}_J$) bound.

Remark 3.6 (The SHAKE PRG advantage is oracle-free). *In the model of Definition 3.5, the SHAKE PRG*

assumption of the classical theorems is formally an assumption about distinguishers that may query $\mathcal{O}_H$, $\mathcal{O}_G$, and $\mathcal{O}_J$. This access is harmless: the three oracles are ideal objects sampled independently of the PRG challenge distribution, which is a statement about concrete SHAKE256 only, so a distinguisher can lazily sample all three itself. For every oracle distinguisher $\mathcal{P}$ there is an oracle-free $\mathcal{P}'$ with identical advantage whose running time exceeds $\mathcal{P}$'s only by the lazy-sampling bookkeeping. The classical theorems therefore charge the plain PRG-cc advantage of Definition 3.3, and writing the assumption oracle-free loses no generality. The alternative treatment, idealizing SHAKE256 on the 32-byte seed-expansion domain as well and replacing the assumption by a hidden-input query bound of the shape $q/2^{256}$, is deliberately not taken: the deployed seed expansion is a concrete function, and this paper keeps its security an explicit named assumption rather than an artifact of a wider idealization.

3.7 Single-user pre-use root game

Definition 3.7 (Staged single-user contextual root game RR_b). **Setup.** The challenger generates an exact draft-10 X-Wing key pair (sk, pk) , computes one honest challenge encapsulation $(K^*, ct^*) \leftarrow \text{XW}^{10}.\text{Enc}(pk)$, gives (pk, ct^*) to the adversary, and sets $\text{tested} \leftarrow \text{false}$.

Oracles, available throughout the experiment, in any order and any interleaving:

1. $\text{RootDec}(ct, \tau)$: requires $ct \neq ct^*$ and $|ct| = 1120$ bytes (the one ciphertext length exact draft-10 decapsulation accepts); returns $R(\text{XW}^{10}.\text{Dec}(sk, ct), \tau)$. Available both before and after TestRoot .
2. $\text{TestRoot}(\tau^*)$: permitted at most once; τ^* ranges over arbitrary byte strings, subject only to containing the exact bytes of ct^* (byte-substring containment: the same containment relation the multi-user Test check uses; no canonical-parseability precondition). Sets $\text{tested} \leftarrow \text{true}$ and returns $Y_0 = R(K^*, \tau^*)$ if $b = 0$, or uniform $Y_1 \leftarrow \{0, 1\}^{256}$ if $b = 1$.

Totality. A call violating a stated precondition ($\text{RootDec}(ct^*, \cdot)$, a RootDec whose ct is not exactly 1120 bytes long, a second TestRoot , or a τ^* without the required containment) returns $\perp$ and changes no state. A “second TestRoot ” means a call made while $\text{tested} = \text{true}$: an unsatisfied call (one whose τ^* lacks the required containment) neither consumes nor blocks the single permission, since it changes no state. (This is deliberately asymmetric to the multi-user Test rule, where the first call consumes the permission whether or not its checks pass; each reduction matches its own game’s convention, and the fixed-slot reduction forwards only a containment-satisfying TestRoot , so the asymmetric corner is unreachable in the lift.) R and H_T are total on arbitrary byte strings, and $\text{XW}^{10}.\text{Dec}$ is total on 1120-byte ciphertexts (only a wrong length is rejected), so no other input is invalid.

Output. The adversary halts with one bit, which the game returns. If TestRoot was never called (or never called with a satisfied precondition), the game instead returns an independent uniform bit.

The exact challenge ciphertext is excluded from RootDec for all contexts, both before and after TestRoot ; this is the same challenge-ciphertext exclusion used by KEM IND-CCA. The game exposes no target-root-dependent downstream object.

The transcript may be adversarially selected after seeing ct^* and after arbitrarily many RootDec queries. This is stronger than requiring an honest public context, and strictly generalizes a sequential game in which the adversary must commit to τ^* immediately after receiving the challenge: any sequential adversary is a staged adversary that happens to call TestRoot first. The staging exists so that the multi-user reduction of Lemma 9.2 may legally forward pre-test root-decapsulation queries, and the no-test convention matches the multi-user experiment so that the guessing identity of Section 9 is exact. The challenge exclusion is essential: allowing arbitrary contextual evaluations for ct^* requires the more expensive two-PRF/two-KEM transform described in Remark 7.3.

4 Exact construction

4.1 X-Wing draft-10 key generation

The packed decapsulation key is one 32-byte seed. The exact expansion is:

```

Expand(sk):
    require len(sk) == 32
    z = SHAKE256(sk, 96 bytes)
    d    = z[0:32]
    zeta = z[32:64]
    sk_X = z[64:96]
    (pk_M, sk_M) = ML-KEM-768.KeyGen_internal(d, zeta)
    pk_X = X25519(sk_X, X25519_BASE)
    return (sk_M, sk_X, pk_M, pk_X)

GenerateKeyPair():
    sk = random(32)
    (sk_M, sk_X, pk_M, pk_X) = Expand(sk)
    return (sk, pk_M || pk_X)

```

The public key is 1216 bytes, the ciphertext 1120 bytes, and the shared secret 32 bytes [2].

4.2 Exact combiner

Let

$$L_{XW} = 5c2e2f2f5e5c.$$

For 32-byte values K_M, K_X, ct_X, pk_X , draft-10 computes

$$K_{XW} = \text{SHA3-256}(K_M \parallel K_X \parallel ct_X \parallel pk_X \parallel L_{XW}). \quad (4)$$

Define the exact keyed family used by the post-quantum theorem:

Definition 4.1 (Draft-10 SHA3 keyed family). *For $k \in \{0, 1\}^{256}$ and $x \in \{0, 1\}^{768}$,*

$$F_k^{10}(x) = \text{SHA3-256}(k \parallel x \parallel L_{XW}).$$

In X-Wing, $k = K_M$ and $x = K_X \parallel ct_X \parallel pk_X$.

This definition is required because the peer-reviewed paper writes its concrete label before the key, whereas draft-10 places the 32-byte ML-KEM shared secret first and the six-byte label last. The generic theorem applies to any keyed family satisfying its PRF premise; the concrete family must be named exactly.

4.3 Auris direct adapter

The 23-byte root label, one zero byte, four-byte suite ID, and 48-byte transcript hash form a 76-byte **info** string. RFC 5869 Expand with $L = 32$ performs exactly one HMAC call:

$$\text{info}(\tau) = \text{"auris/root/v5/xwing-d10"} \parallel 00 \parallel \text{u32}_{BE}(0x00050001) \parallel H_T(\tau), \quad (5)$$

$$R(K, \tau) = \text{HKDF-Expand-SHA256}(K, \text{info}(\tau), 32) \quad (6)$$

$$= \text{HMAC-SHA256}(K, \text{info}(\tau) \parallel 01). \quad (7)$$

The earlier public-salt Extract design is not used. In the selected design, the pseudorandom X-Wing value occupies HMAC's secret-key position.

5 Explicit assumption dispositions

The proof separates reduction correctness from primitive-instantiation confidence. The following rows are premises, not conclusions hidden inside the theorem.

Assumption	Disposition	Exact meaning and evidence boundary
SHAKE256 : $\{0, 1\}^{256} \rightarrow \{0, 1\}^{768}$ PRG-cq / PRG-cc	Explicit unproved primitive assumption	The shared-seed hybrid argument explains where this term enters. FIPS 202 [7] fixes SHAKE256 bytes, but neither the standard nor this paper proves a concrete 256-to-768-bit PRG advantage for all QPT/PPT distinguishers. Besides the two security hops of Lemma 6.1, the correctness transfer of Lemma 10.3 charges one further, separately constructed distinguisher $\mathcal{P}_{corr}$. In the classical branch the assumption is read inside the model of Definition 3.5, where oracle access does not strengthen the distinguisher (Remark 3.6), so the plain notion is charged.
Final MLKEM768 IND-CCA-pq	Imported primitive assumption	The post-quantum branch reduces to the standard IND-CCA notion. FIPS 203 fixes the algorithm and supplies correctness parameters; standardization is not a proof that every concrete attack has the assumed cost.
Exact draft-10 family $F_k^{10}(x) = \text{SHA3-256}(k \parallel x \parallel L_{XW})$ PRF-cq	Exact construction-specific assumption	Barbosa et al. Theorem 2 is generic in a keyed family. Draft-10 moves the six-byte label and fixes the exact byte order, so this paper names the resulting family rather than treating the paper's concrete order as identical. No separate peer-reviewed theorem for this exact family is claimed.
HMAC-SHA256 PRF-cq / PRF-cc	Explicit adapter assumption	The adapter uses one classical 77-byte evaluation under the hidden 32-byte X-Wing key. Hosoyamada-Iwata gives a rigorous quantum-random-oracle research basis for HMAC/NMAC, but this paper does not convert that result into a complete concrete bound for the deployed SHA-256 instantiation. The cq interface here is weaker than granting superposition access to the HMAC evaluation oracle.
ML-KEM correctness δ_M	Imported statistical term	The post-quantum X-Wing theorem contributes $2\delta_M$. The responder-agreement path uses δ_{seed} for the exact shared-seed distribution (Definition 10.1), transferred to δ_M by Lemma 10.3. The numerical failure estimate is a parameter statement, not a substitute for implementation conformance.
X25519 nominal-group distance and strong Diffie-Hellman (SDH)	Classical-only assumptions	These terms support the classical fallback through Barbosa et al. Theorem 1. They are not used in the post-quantum branch.
Independent G, J random oracles for ML-KEM C2PRI	Classical ROM assumption	Proposition 6.7 maps the published C2PRI proof to final FIPS 203 algorithms; the oracles are $\mathcal{O}_G, \mathcal{O}_J$ of the global model (Definition 3.5), which fixes their independence, role domains, and query counting. It is not a standard-model statement about concrete SHA3/SHAKE.

A reviewer may reject an instantiation while accepting every reduction step. Such a disposition narrows or blocks the theorem; it does not justify silently deleting the assumption.

6 Exact X-Wing transfer lemmas

6.1 Shared-seed endpoint hops

The published QSF theorem samples the ML-KEM and X25519 component keys independently. Draft-10 derives both from one seed. The following lemma gives the exact cost under the standard two-game IND-CCA advantage.

Lemma 6.1 (Shared-seed to independent-key transfer). *Let XW_{seed}^{10} be draft-10 key generation and let XW_{ind}^{10} use independent uniform $(d, \zeta, sk_X) \in \{0, 1\}^{768}$, with the same public algorithms thereafter. For every IND-CCA adversary $\mathcal{A}$, there exist PRG distinguishers $\mathcal{P}_0, \mathcal{P}_1$ such that*

$$\text{Adv}_{XW_{seed}^{10}, \mathcal{A}}^{\text{IND-CCA}} \leq \text{Adv}_{XW_{ind}^{10}, \mathcal{A}}^{\text{IND-CCA}} + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_0}^{\text{PRG}} + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_1}^{\text{PRG}}. \quad (8)$$

The mode of each PRG advantage matches the adversary mode. If both reductions are bounded by the same value ϵ_{PRG} , the additive cost is at most $2\epsilon_{\text{PRG}}$.

Proof. For each IND-CCA bit b , define a hybrid H_b that replaces the 96-byte draft-10 expansion by a uniform 96-byte string. A PRG distinguisher receives Z , parses it as its three 32-byte segments (d, ζ, sk_X) , constructs the ML-KEM and X25519 keys, and runs the complete IND-CCA_b experiment. If $Z = \text{SHAKE256}(s, 96)$, this is the exact shared-seed experiment. If Z is uniform, the component-generation randomness is independent and has the distribution required by XW_{ind}^{10} . The distinguisher stores expanded components rather than the packed seed; IND-CCA never reveals the decapsulation key, so the simulation is perfect.

Thus, for $b = 0, 1$,

$$\left| \Pr[\text{IND-CCA}_b^{XW_{seed}^{10}, \mathcal{A}} \Rightarrow 1] - \Pr[\text{IND-CCA}_b^{XW_{ind}^{10}, \mathcal{A}} \Rightarrow 1] \right| \leq \text{Adv}_{\text{SHAKE256}, \mathcal{P}_b}^{\text{PRG}}.$$

Apply the triangle inequality across both endpoints of the IND-CCA advantage. Two endpoint hops are necessary under this advantage convention; writing one ordinary PRG term would omit one of the two public-key-distribution changes. $\square$

Remark 6.2. *The lemma requires that the packed 32-byte seed have no protocol-visible use beyond deriving the complete X-Wing key. A seed-derived identifier, backup handle, commitment, deterministic nonce, or cross-protocol component export would invalidate the simulation.*

6.2 Post-quantum QSF instantiation

Lemma 6.3 (Exact draft-10 post-quantum X-Wing bound). *Assume MLKEM768 is post-quantum IND-CCA secure with correctness failure probability at most δ_M , and assume the exact family F^{10} from Definition 4.1 is a PRF against QPT adversaries making classical evaluation queries. Then, for every classical-interface QPT adversary $\mathcal{A}$ against XW_{ind}^{10} , there exist reductions $\mathcal{B}_1, \mathcal{B}_2, \mathcal{D}, \mathcal{E}$ such that*

$$\text{Adv}_{XW_{ind}^{10}, \mathcal{A}}^{\text{IND-CCA}} \leq \text{Adv}_{\text{MLKEM768}, \mathcal{B}_1}^{\text{IND-CCA}} + \text{Adv}_{\text{MLKEM768}, \mathcal{B}_2}^{\text{IND-CCA}} + \text{Adv}_{F^{10}, \mathcal{D}}^{\text{PRF-cq}} + \text{Adv}_{F^{10}, \mathcal{E}}^{\text{PRF-cq}} + 2\delta_M. \quad (9)$$

Each ML-KEM reduction makes at most Q_D decapsulation queries when $\mathcal{A}$ makes at most Q_D X-Wing decapsulation queries. The first PRF reduction makes at most $Q_D + 1$ evaluations and the second at most Q_D . No X25519 hardness, Ed25519 security, or C2PRI assumption appears.

Proof. Barbosa et al. Theorem 2 proves the displayed QSF bound for any keyed family H that is a PRF when keyed by the PQ KEM shared secret, including the two ML-KEM reductions, two PRF reductions, and $2\delta_M$ correctness term [1]. Instantiate their abstract keyed family with F^{10} . Equation (4) is exactly

$$F_{K_M}^{10}(K_X \parallel ct_X \parallel pk_X).$$

The proof requires only that the keyed family be efficiently computable and PRF-secure on the protocol's classical messages. It does not require the label to precede the key.

The nested decapsulation simulation remains valid. For an outer query $(c_M, c_X) \neq (c_M^*, c_X^*)$, the ML-KEM reduction queries its decapsulation oracle only when $c_M \neq c_M^*$. When $c_M = c_M^*$ but $c_X \neq c_X^*$, it reuses the held ML-KEM challenge key K_M^* and computes the changed X25519 and F^{10} inputs itself. It never submits c_M^* to the inner ML-KEM decapsulation oracle. This is the exact challenge-component simulation in the published proof, and changing the byte order inside F^{10} does not affect it.

The reductions and their query counts therefore transfer unchanged after the exact instantiation. The X25519 branch may be fully known or cryptographically broken; it enters only the PRF message. $\square$

6.3 Classical random-oracle input order

Lemma 6.4 (Random-oracle order transfer). *Let ρ rotate a 134-byte string left by six bytes, and act as the identity on strings of every other length. Then ρ is a permutation on $\{0,1\}^*$. If $\mathcal{R}$ is a random oracle, $\mathcal{R} \circ \rho$ is also a random oracle. Therefore the QSF classical proof written for*

$$L_{XW} \parallel K_M \parallel K_X \parallel ct_X \parallel pk_X$$

applies without loss to draft-10's

$$K_M \parallel K_X \parallel ct_X \parallel pk_X \parallel L_{XW}.$$

Proof. A fixed-length byte rotation is a bijection, with inverse given by the opposite rotation. Composing a lazily sampled random function with a domain permutation preserves the joint distribution of answers to every adaptive query sequence. The four 32-byte fields plus six-byte label always form a 134-byte combiner input, so the protocol evaluations correspond exactly under ρ . $\square$

Remark 6.5 (Scope of the rotation in the shared-oracle model). *In the model of Definition 3.5 the rotation acts on the shared oracle $\mathcal{O}_H$, which also serves the FIPS 203 role H . Because ρ is the identity on every input length other than 134 bytes, $\mathcal{O}_H \circ \rho$ and $\mathcal{O}_H$ agree on every H -role input (1184-byte encapsulation keys): the reordering touches only the combiner role, and ML-KEM's SHA3-256 evaluations are unchanged by it.*

Lemma 6.6 (Exact draft-10 classical X-Wing bound). *In the classical random-oracle model of Definition 3.5, with the combiner role of $\mathcal{O}_H$ as the theorem's random oracle, for every classical adversary $\mathcal{A}$ against XW_{ind}^{10} , there exist reductions $\mathcal{B}, \mathcal{C}$ such that*

$$\text{Adv}_{XW_{ind}^{10}, \mathcal{A}}^{\text{IND-CCA}} \leq 2\Delta_X + \text{Adv}_{X25519, \mathcal{B}}^{\text{SDH}} + \text{Adv}_{\text{MLKEM768}, \mathcal{C}}^{\text{C2PRI}}. \quad (10)$$

The SDH reduction makes at most $2q_H$ DH-oracle queries when q_H bounds the $\mathcal{O}_H$ queries of the experiment, counted per Definition 3.5; the count is conservative, since the reduction's extraction concerns only 134-byte combiner-form queries. For the X25519 nominal group, the imported analysis gives $\Delta_X < 2^{-126}$, hence $2\Delta_X < 2^{-125}$.

Proof. Apply Barbosa et al. Theorem 1 to the exact draft-10 combiner through Lemma 6.4. The theorem gives the nominal-group distance, SDH, and C2PRI terms with the stated query translation [1]. No ML-KEM confidentiality assumption is used. $\square$

6.4 Final FIPS 203 C2PRI

Proposition 6.7 (Final-FIPS-203 C2PRI bound). *Model the final FIPS 203 functions $G : \{0,1\}^* \rightarrow \{0,1\}^{512}$ and $J : \{0,1\}^* \rightarrow \{0,1\}^{256}$ as the independent random oracles $\mathcal{O}_G$ and $\mathcal{O}_J$ of Definition 3.5. Treat K-PKE.Encrypt(ek, m, r) as deterministic in its inputs. For a C2PRI adversary making at most q_G and q_J queries, counted per Definition 3.5 and including every query the adversary's own computation induces, while the two queries induced by the final win-check decapsulation are carried by the additive constant,*

$$\text{Adv}_{\text{MLKEM768}, \mathcal{C}}^{\text{C2PRI}} \leq \frac{q_G + q_J + 2}{2^{256}}. \quad (11)$$

Proof. FIPS 203 Algorithm 17 computes $(K, r) = G(m \parallel H(ek))$ and $c = \text{K-PKE.Encrypt}(ek, m, r)$. Algorithm 18 decrypts a submitted c' to m' , computes $(K', r') = G(m' \parallel H(ek))$, computes the implicit-rejection value $\bar{K} = J(z \parallel c')$, re-encrypts $c'' = \text{K-PKE.Encrypt}(ek, m', r')$, and returns K' if $c' = c''$, otherwise $\bar{K}$ [4].

Let (K^*, c^*) be an honest challenge. For $c' \neq c^*$ to decapsulate to K^* , one of three cases must occur.

1. If the valid branch uses the same input to G as the challenge, then $m' = m^*$ and $r' = r^*$. Deterministic re-encryption gives $c'' = c^*$. The validity test $c' = c''$ contradicts $c' \neq c^*$.
2. If the valid branch uses a different input to G , the first 256 bits of a fresh independent 512-bit random-oracle output must equal K^* . Across q_G adversarial queries and the one query induced by final decapsulation, this has probability at most $(q_G + 1)/2^{256}$.
3. If implicit rejection is selected, an independent J output must equal K^* . Across q_J adversarial queries and the one decapsulation-induced query, this has probability at most $(q_J + 1)/2^{256}$.

A union bound gives Equation (11). Final FIPS input checks do not increase success: a malformed-length ciphertext is not passed to the internal algorithm and cannot return the challenge key. $\square$

7 Root-adapter transform

The adapter lemma is not stated for an arbitrary IND-CCA KEM: its proof uses an interface property that an arbitrary KEM need not have. The class is named first; the statement then records which clause the proof consumes and which clauses align the game’s constants with the analyzed KEM.

Definition 7.1 (Fixed-length total KEM). *A KEM is fixed-length total with parameters (ℓ_{ct}, ℓ_K) if (i) every honestly generated ciphertext is exactly ℓ_{ct} bytes long; (ii) decapsulation is a total function $\text{Dec} : SK \times \{0, 1\}^{\ell_{ct}} \rightarrow \{0, 1\}^{\ell_K}$, returning an ℓ_K -byte shared secret on every ℓ_{ct} -byte input, never $\perp$ and never an error, whatever the input’s provenance; and (iii) an input of any other length is rejected, and the rejection is decidable from public information (the length alone). Exact draft-10 X-Wing, with either key-generation distribution of Lemma 6.1, is fixed-length total with $(\ell_{ct}, \ell_K) = (1120, 32)$: honest ciphertexts are 1120 bytes (Section 4), decapsulation returns a 32-byte secret on every 1120-byte input (ML-KEM implicit rejection plus the total X25519 evaluation; Section 15), and only a wrong length is rejected.*

Lemma 7.2 (Direct one-block adapter). *Let KEM be IND-CCA secure and fixed-length total with $(\ell_{ct}, \ell_K) = (1120, 32)$ (Definition 7.1); exact draft-10 X-Wing qualifies. Let $A_k(m) = \text{HMAC-SHA256}(k, m)$ be a PRF on 77-byte messages in the relevant adversary mode. Then the single-user contextual root game of Definition 3.7, executed with this KEM, satisfies*

$$\text{Adv}_{\text{KEM} \rightarrow \mathcal{A}, \mathcal{A}}^{\text{RR}} \leq \text{Adv}_{\text{KEM}, \mathcal{B}}^{\text{IND-CCA}} + \text{Adv}_{\mathcal{A}, \mathcal{E}}^{\text{PRF}}. \quad (12)$$

The HMAC reduction makes one evaluation query for the target message. The context may be chosen adaptively after the adversary sees the challenge ciphertext and after arbitrarily many RootDec queries; RootDec remains available after TestRoot. Of Definition 7.1’s clauses, the proof consumes (ii): totality makes the simulated RootDec well-defined on every 1120-byte ciphertext. Clauses (i) and (iii) are interface alignment rather than proof steps: they record that the analyzed KEM’s honest ciphertexts occupy exactly the game’s hard-coded length and that nothing is ever decapsulated outside it, while the game’s wrong-length rejection is a public check on the byte string itself and requires no property of the KEM.

Proof. Let G_0 be the staged real game ($b = 0$) and let G_2 be the staged random-root game ($b = 1$).

KEM hop. Construct a KEM adversary $\mathcal{B}$. Its IND-CCA challenger gives it (pk, ct^*, K_β) at the start of the experiment together with a decapsulation oracle that is available for the whole experiment and refuses exactly ct^* (the same interface shape as the staged game). $\mathcal{B}$ gives (pk, ct^*) to $\mathcal{A}$, maintains a flag *valid_tested* $\leftarrow$ false, and answers, at any time and in any interleaving: each $\text{RootDec}(ct, \tau)$ whose 1120-byte ct differs from ct^* with $\text{HMAC-SHA256}(\text{Dec}(ct), M(\tau))$ through its decapsulation oracle (legal because $ct \neq ct^*$, and well-defined because exact decapsulation returns a 32-byte secret for every 1120-byte ciphertext, so HMAC is invoked only on a returned 32-byte secret); and the single *precondition-satisfying* $\text{TestRoot}(\tau^*)$ (the first call whose τ^* contains ct^*) with $\text{HMAC-SHA256}(K_\beta, M(\tau^*))$, setting *valid_tested*. Every other call is answered $\perp$ from public information with no state change: a wrong-length or challenge-naming RootDec, a TestRoot without the required containment, and any TestRoot after the valid one. If $\mathcal{A}$ never makes a precondition-satisfying TestRoot call (*valid_tested* = false at halt), $\mathcal{B}$ outputs an independent uniform bit, matching the game’s no-test convention (Definition 3.7’s output rule counts only a satisfied-precondition test); otherwise it outputs $\mathcal{A}$ ’s bit. When $\beta = 0$ this is exactly G_0 . When $\beta = 1$ it is the hybrid G_1 , in which the challenge ciphertext remains honestly distributed but the target adapter key is an independent uniform 256-bit string. Therefore

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \text{Adv}_{\text{KEM}, \mathcal{B}}^{\text{IND-CCA}},$$

and $\mathcal{B}$ makes at most one decapsulation query per RootDec query.

PRF hop. Construct a PRF adversary $\mathcal{E}$ against $A_k(m) = \text{HMAC-SHA256}(k, m)$ under a hidden uniform key. $\mathcal{E}$ generates its own exact key pair (sk, pk) , computes one honest challenge encapsulation, discards the encapsulated shared secret, and gives (pk, ct^*) to $\mathcal{A}$. Holding sk locally and maintaining the same *valid_tested* flag, it answers every $\text{RootDec}(ct, \tau)$ with a 1120-byte $ct \neq ct^*$ (before and after the test, in any order) by ordinary decapsulation and ordinary HMAC on the returned 32-byte secret. It answers the single precondition-satisfying $\text{TestRoot}(\tau^*)$ with $\text{Eval}(M(\tau^*))$, its only evaluation query; an invalid TestRoot (no containment, or any call after the valid one) is answered $\perp$ and spends no evaluation query, and wrong-length or challenge-naming RootDec calls are answered $\perp$; if $\mathcal{A}$ never makes a precondition-satisfying test, it outputs an independent uniform bit. When Eval is the real keyed function, the hidden uniform PRF key plays the role of G_1 ’s independent uniform adapter key, so the simulation is exactly G_1 . When Eval is a random function, the never-otherwise-queried point $M(\tau^*)$ receives a uniform independent 256-bit value, which is exactly G_2 . Hence

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \text{Adv}_{\mathcal{A}, \mathcal{E}}^{\text{PRF}}.$$

Combining the two hops proves the lemma. Both reductions are insensitive to the interleaving of **RootDec** and **TestRoot**: the KEM decapsulation oracle and the locally held secret key are available for the entire experiment, and the single PRF evaluation is made whenever the precondition-satisfying τ^* arrives. Calls violating a stated precondition (**RootDec**($ct^*, \cdot$), a **RootDec** whose ct is not 1120 bytes long, a **TestRoot** after the valid one, or a τ^* without the required containment) are answered $\perp$ by each reduction from the public value ct^* and the public length alone, identically in every game and hybrid, with no state change and nothing consumed; the no-test convention fires exactly when no precondition-satisfying **TestRoot** occurred (*valid_tested* = false), never merely because some invalid call was made. The no-test convention maps to the same independent-coin output in every game and hybrid, so the branch probabilities compare exactly. $\square$

Remark 7.3 (General contextual oracle). *If the game allowed **RootDec**(ct^*, τ) for $\tau \neq \tau^*$, a single HMAC hop would not restore the random-root endpoint’s real same-ciphertext oracle. A complete proof of that stronger game uses two KEM and two HMAC PRF transitions, analogous to the two-KEM/two-PRF structure of the QSF theorem. Auris instead relies on exact ciphertext binding, signatures, and replay handling so that the abstract target game can use the standard challenge-ciphertext exclusion.*

8 Complete single-user theorems

Theorem 8.1 (Post-quantum pre-use root theorem). *For every QPT adversary $\mathcal{A}$ with classical protocol and primitive interfaces in Definition 3.7, there exist reductions such that*

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{\text{RR-PQ}} \leq & \text{Adv}_{\text{SHAKE256}, \mathcal{P}_0}^{\text{PRG-cq}} + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_1}^{\text{PRG-cq}} + \text{Adv}_{\text{MLKEM768}, \mathcal{B}_1}^{\text{IND-CCA-pq}} + \text{Adv}_{\text{MLKEM768}, \mathcal{B}_2}^{\text{IND-CCA-pq}} \\ & + \text{Adv}_{F^{10}, \mathcal{D}}^{\text{PRF-cq}} + \text{Adv}_{F^{10}, \mathcal{E}}^{\text{PRF-cq}} + 2\delta_M + \text{Adv}_{\text{HMAC-SHA256}, \mathcal{H}}^{\text{PRF-cq}}. \end{aligned} \quad (13)$$

X25519 hardness, Ed25519 security, and C2PRI are not premises.

Proof. The literal composition order is adapter-first: Lemma 7.2 gives $\text{Adv}_{\mathcal{A}}^{\text{RR}} \leq \text{Adv}_{\text{XW}^{10}_{\text{seed}}}^{\text{IND-CCA}} + \text{Adv}_{\text{HMAC-SHA256}}^{\text{PRF}}$; Lemma 6.1 then splits the shared-seed IND-CCA term into the independent-key term plus the two SHAKE endpoint hops, and Lemma 6.3 bounds the independent-key term. (Equivalently, the appendix B.1 ledger realizes the same bound as a direct hybrid chain on the staged game, whose $G_0 \rightarrow G_1$ and $G_7 \rightarrow G_8$ endpoint hops use the RR-flavored SHAKE distinguisher, the same construction as Lemma 6.1’s, except that it parses its 96-byte challenge into component keys and simulates the whole staged game from the expanded components, which it holds. Both routes are sound and yield the identical bound; as inequalities the three lemmas compose by substitution in either order.) The target game excludes the full challenge ciphertext from root-decapsulation queries, so one outer HMAC PRF term is sufficient. The transcript may depend on the public challenge ciphertext; it does not depend on the hidden challenge bit before the target root is returned. Collecting the terms gives Equation (13). $\square$

Theorem 8.2 (Active-classical pre-use fallback theorem). *For every classical PPT adversary $\mathcal{A}$ in Definition 3.7, in the classical random-oracle model of Definition 3.5 there exist reductions such that*

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{\text{RR-C}} \leq & \text{Adv}_{\text{SHAKE256}, \mathcal{P}_0^C}^{\text{PRG-cc}} + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_1^C}^{\text{PRG-cc}} + 2\Delta_X + \text{Adv}_{X^{25519}, \mathcal{B}}^{\text{SDH}} \\ & + \frac{q_G + q_J + 2}{2^{256}} + \text{Adv}_{\text{HMAC-SHA256}, \mathcal{H}_C}^{\text{PRF-cc}}. \end{aligned} \quad (14)$$

Here $2\Delta_X < 2^{-125}$ under the imported X25519 nominal-group mapping; the two PRG terms are the plain classical notion, which oracle access does not strengthen (Remark 3.6); and q_G, q_J are the terminal C2PRI adversary’s query counts in the sense of Proposition 6.7, satisfying $q_G \leq q_G^A + Q_D$ and $q_J \leq q_J^A + Q_D$ when $\mathcal{A}$ makes at most Q_D root-decapsulation queries and q_G^A, q_J^A direct oracle queries (Section 13). ML-KEM confidentiality is not assumed. The theorem is classical only.

Proof. As in Theorem 8.1, the literal composition is adapter-first: the classical form of Lemma 7.2, then Lemma 6.1’s two endpoint PRG hops splitting the shared-seed IND-CCA term, then Lemma 6.6 with its C2PRI term instantiated by Proposition 6.7 (the appendix B.2 ledger is the equivalent direct-hybrid narration, with the RR-flavored SHAKE distinguisher at its endpoint hops). No step grants a quantum adversary or quantum random-oracle access, so the conclusion is not a post-quantum fallback theorem. $\square$

9 Static-slot adaptive multi-user lift

9.1 Static slot space

Fix public bounds $N_K \geq 1$ and $N_S \geq 1$. Define the potential target-slot set

$$\mathcal{S} = [N_K] \times [N_S], \quad N_T = |\mathcal{S}| = N_K N_S.$$

The i -th successful honest-key creation receives key index i . The s -th successful honest encapsulation under key i receives session index s . These are historical indices. Reveal, corruption, erasure, restore, replay, failure, or later loss of freshness never rennumbers a key or session. A potential slot that is never created remains $\perp$.

No dynamic filtering

If keys K_1, K_2 are created and K_1 is later revealed, a session under K_2 remains in a slot $(2, s)$. It does not become the first slot. A reduction that guessed a slot under K_1 simply loses its guess; that event is exactly what the $1/N_T$ probability pays for.

9.2 Complete multi-user experiment

Definition 9.1 (Static-slot multi-user contextual root game MU-RR_b). *The challenger maintains honest-key tables (sk_i, pk_i) , per-key session tables $(K_{i,s}, ct_{i,s})$, a per-slot eligibility flag, a per-key erasure flag $erased_i$ (initially false), and a single target variable $T \in \mathcal{S} \cup \{\perp\}$. Slots start eligible, and ineligibility is monotone: once a slot is marked ineligible, no later call makes it eligible again. Eligibility flags live on the static potential-slot space $\mathcal{S}$ itself, including slots not yet occupied: “mark every slot under i ineligible” always includes unoccupied and future slots, so a session created later under i occupies an already-ineligible slot. A key is created once NewKey has assigned its index, and live while it is created and $erased_i = \text{false}$. The challenger exposes exactly the following interfaces; RevealComponent’s type argument ranges over the fixed public enumeration $\{\text{mlkem}, \text{x25519}\}$, and any other value violates its stated precondition; RootDec’s ct argument is a byte string of exactly 1120 bytes (the draft-10 ciphertext length, the one length that exact decapsulation accepts), and any other length violates its stated precondition. **Every interface is total:** a call naming an out-of-range index, an uncreated key or slot, or otherwise violating a stated precondition returns $\perp$ and changes no state, except where a rule below states otherwise. **Erasure alone never changes any slot’s eligibility:** erasure models the honest deletion a forward-secrecy claim relies on, while disclosure of retained or restored material is what disqualifies. A post-erasure RevealSeed or RevealComponent returns only the public marker **ERASED**, discloses nothing, and does not change any slot’s eligibility.*

1. **NewKey()**: if fewer than N_K honest keys exist, generate an exact draft-10 key pair, assign the next irrevocable index i , and return (i, pk_i) . Otherwise return $\perp$.
2. **Encaps(i)**: for a live honest key with fewer than N_S sessions, run exact honest draft-10 encapsulation, assign the next irrevocable index s , store $(K_{i,s}, ct_{i,s})$, and return $(i, s, ct_{i,s})$. Otherwise return $\perp$.
3. **RootDec(i, ct, τ)**: a ct whose length is not exactly 1120 bytes violates the stated precondition: return $\perp$ with no state change, regardless of key state or test phase; the call marks nothing and enters no Test history. Otherwise, if key i is not live, return $\perp$. Otherwise return $R(\text{XW}^{10}.\text{Dec}(sk_i, ct), \tau)$, except that if (i, ct) is the tested target ciphertext the game ends with an independent random output bit. A pre-Test call whose ct byte-equals the stored ciphertext of one or more occupied slots under key i is answered normally and marks every slot (i, s) with $ct_{i,s} = ct$ ineligible; a $\perp$ -answered call (key not live) is not answered normally: it marks nothing and, per the totality rule, changes no state, including any later Test’s query history. (Distinct honest encapsulations share ciphertext bytes only on an encapsulation collision; the rule is total regardless, and no proof step relies on non-collision.)
4. **RevealSeed(i)** and **RevealComponent(i, type)**: a type outside $\{\text{mlkem}, \text{x25519}\}$ violates the stated precondition: return $\perp$ with no state change and no marking, regardless of the key’s lifecycle state or the test phase (this check precedes the erasure branch). Otherwise, if $erased_i = \text{true}$, return only the public marker **ERASED** with no state change. Otherwise return the requested retained material and mark every slot under i ineligible.
5. **Erase(i)**: for a created key, set $erased_i \leftarrow \text{true}$ and acknowledge; repeated calls are idempotent. The packed seed and expanded components become unavailable to every reveal interface while $erased_i = \text{true}$. Erasure changes no static index and no slot’s eligibility.

6. **Restore**(i): if key i is created and $\text{erased}_i = \text{true}$, set $\text{erased}_i \leftarrow \text{false}$, return the complete packed seed and expanded components (modelling rollback restoration), and mark every slot under i ineligible. In every other case (key uncreated, never erased, or currently not erased) return $\perp$ with no state change. After a successful **Restore** the key is live again and every interface behaves accordingly; the disqualification of its slots is permanent (monotonicity). **Erase** and **Restore** may alternate arbitrarily; each successful **Restore** is a disclosure event.
7. **RevealRoot**(i, s, τ): for an occupied slot, return $R(K_{i,s}, \tau)$ and mark slot (i, s) ineligible; for an unoccupied or out-of-range slot, return $\perp$ with no state change.
8. **Continue**(i, s): mark that a root or descendant was handed to the excluded downstream protocol; the slot becomes ineligible. No AEAD, ratchet, header, or acknowledgement oracle otherwise exists in this R2 game.
9. **Test**(i, s, τ): permitted at most once. The slot must be occupied and still eligible; τ must be fixed by this query before the challenge root is returned and must contain the exact pk_i and $ct_{i,s}$; the key and encapsulation must be honest; no seed/component/restore/root/continue disclosure may have marked the slot ineligible; and no normally answered **RootDec**($i, ct, \cdot$) call with $ct = ct_{i,s}$ (byte equality) may have occurred (a $\perp$ -answered call never enters this history). The checks inspect slot eligibility and the disclosure history, never the key's lifecycle state: erasure alone never disqualifies (only disclosure does). In particular a merely-erased key remains a valid test target, while a key whose material was restored has every slot already ineligible through the restore disclosure; the checks still consult only eligibility, not the lifecycle flags. Set $T = (i, s)$. Return $R(K_{i,s}, \tau)$ when $b = 0$, or uniform $Y \leftarrow \{0, 1\}^{256}$ when $b = 1$.

The first **Test** call consumes the single **Test** permission whether or not its checks pass; a failed **Test** ends the game immediately with an independent random output bit. A first **Test** call is therefore never answered by the totality rule's $\perp$: every first **Test** (including one naming an out-of-range or unoccupied slot) is a consumed-permission failed test, not a precondition violation. A later **Test** call violates the at-most-once precondition and returns $\perp$ with no state change (the game continues; such a call can only occur after a passed test). After a successful **Test**, any target-disqualifying query causes an immediate independent random output bit. If no valid **Test** occurs, the game also returns an independent random bit. Otherwise it returns the adversary's final bit. Malicious keys are not target-eligible and may be generated by the adversary outside these honest-key interfaces.

The slot T , or the fact $T = \perp$, is selected from the pre-challenge view. The challenge root is returned only after the adversary has supplied (i, s, τ) . Therefore the distribution of T is independent of b .

9.3 Fixed-slot reduction

Lemma 9.2 (Static-slot target guessing). *For every adversary $\mathcal{A}$ in Definition 9.1, there exists a single-user adversary $\mathcal{B}$ such that*

$$\text{Adv}_{\mathcal{A}}^{\text{MU-RR}} \leq N_K N_S \cdot \text{Adv}_{\mathcal{B}}^{\text{RR}}. \quad (15)$$

Proof. Before setup, $\mathcal{B}$ samples a static pair $(I, S) \leftarrow \mathcal{S}$. It receives the single-user challenge public key pk^* and ciphertext ct^* ; in the staged game of Definition 3.7, its **RootDec** oracle is available from this point on, and $\mathcal{B}$ does not call **TestRoot** yet.

$\mathcal{B}$ maintains *exactly the challenger's bookkeeping*: creation counters, per-slot eligibility flags, per-key erasure flags erased_i , the disclosure history, and the single test permission. Every flag transition is triggered by an adversary-visible call, so $\mathcal{B}$ tracks all of them without any secret material. Its key material is: for every key $i \neq I$, the complete locally generated draft-10 key; for key I , only pk^* ; for every session (I, s) with $s \neq S$, the locally chosen encapsulation randomness and shared secret $K_{I,s}$; for (I, S) , only ct^* .

Abort principle. $\mathcal{B}$ halts with an independent fair coin ("coin-abort") exactly at the queries whose game-prescribed reply requires the challenge material: the secret key behind pk^* , its packed seed or expanded components, or the challenge secret K^* . These are precisely the following, all pre-test: **RootDec**($I, ct^*, \cdot$) on the live key; **RevealSeed**(I)/**RevealComponent**($I, \text{type} \in \{\text{mlkem}, \text{x25519}\}$) on the created, non-erased key (the material-returning reveals; an out-of-domain type is a $\perp$, never an abort); material-returning **Restore**(I) (key created and currently erased); and **RevealRoot**($I, S, \cdot$) on the occupied slot. Each of these, by the game's own rules, disqualifies the guessed slot (I, S) : the reveals, restores, root-reveals, and continues mark it ineligible, and a normally answered **RootDec**($I, ct^*, \cdot$) enters the **Test** history condition for $ct_{I,S}$ whether or not (I, S) is yet occupied (the byte-collision rule additionally marks it when it is). Therefore no run containing one can end with a valid test on (I, S) . Every other lifecycle or argument case of these same oracles (the uncreated, unoccupied,

out-of-range, not-live, never-erased, erased-marker, out-of-domain-*type*, and wrong-length-*ct* cases) has a game-prescribed reply ($\perp$ or **ERASED**) that requires no challenge material and is answered exactly; after a passed test on (I, S) , the game itself ends every target-disqualifying query with an independent coin, which $\mathcal{B}$ mirrors with its own fair coin. Every other query is answered exactly as the challenger would, from public values, local material, and the single-user oracles. The complete case analysis below is *first-match ordered and, after the capacity, successful-creation, type-domain, and ciphertext-length qualifiers its rows carry, pairwise disjoint*: every row names the created/occupied/liveness/test-phase state (and, where the oracle assigns an index or takes a *type* or *ct* argument, the capacity/successful-creation, domain, and length conditions) it requires; “the call performing the i -th successful creation” always means creation counting (the slot-space rule), never call counting, so an at-capacity call matches no success row; the rows jointly cover every syntactically possible query in every reachable state; and a query is answered by its first matching row, which under the stated qualifiers is also its only matching row, so totality is checkable row by row against the game text:

Query and case	$\mathcal{B}$'s action	Exactness
NewKey at key capacity (N_K keys already created); checked first	$\perp$; no state change	exact: public bookkeeping
NewKey, capacity free, the call performing the i -th <i>successful</i> creation, $i \neq I$	local draft-10 key generation	exact: identical distribution
NewKey, capacity free, the call performing the I -th <i>successful</i> creation	publish pk^*	exact: pk^* is honestly distributed
Encaps(i), key uncreated, out of range, not live, or at session capacity; any i incl. I	$\perp$ from public bookkeeping; no state change	exact (totality)
Encaps($i \neq I$), key live, capacity free	local honest encapsulation	exact: identical distribution
Encaps(I), key live, capacity free, the call performing the s -th <i>successful</i> session creation, $s \neq S$	local honest encapsulation under pk^* ; store $K_{I,s}$	exact: identical distribution
Encaps(I), key live, capacity free, the call performing the S -th <i>successful</i> session creation	publish ct^*	exact: honest challenge encapsulation under pk^*
RootDec($i, ct, \cdot$) with $ ct \neq 1120$ bytes; any i incl. I , any key state, pre- or post-test	$\perp$ from the public length alone; no state change, marks nothing, enters no Test history	exact (totality)
RootDec($i, \cdot, \cdot$), 1120-byte ct , key uncreated or out of range	$\perp$; no state change, enters no Test history	exact (totality)
RootDec($i, \cdot, \cdot$), 1120-byte ct , key created, not live; any i incl. I , any such ct incl. ct^* , pre- or post-test	$\perp$ from the public erasure flag; marks nothing, enters no Test history	exact
RootDec($i \neq I, ct, \tau$), 1120-byte ct , key live	local: sk_i held; a pre-test normally answered call applies the game's byte-collision marking	exact
RootDec(I, ct, τ), key live, 1120-byte $ct \neq ct^*$, pre- or post-test	forward (ct, τ) to the single-user RootDec oracle (legal at every stage of the staged game; the forwarded ct satisfies the single-user length and challenge-exclusion preconditions); a pre-test call marks byte-matching occupied slots ineligible	exact
RootDec(I, ct^*, τ), key live, pre-test	<i>coin-abort</i> : the game's normal reply needs Dec(sk^*, ct^*)	coin-safe: the normally answered query enters the Test history for $ct_{I,S}$ (and, when (I, S) is occupied, also marks it ineligible), so no later test on (I, S) is valid
RootDec(I, ct^*, τ), key live, after the passed test on (I, S)	output a fair coin	exact: the game ends with an independent coin

Query and case	$\mathcal{B}$'s action	Exactness
RevealSeed/RevealComponent(i), key uncreated or out of range, or $type$ outside $\{\text{mlkem}, \text{x25519}\}$ (any key state, any phase); any i incl. I	$\perp$; no state change, no marking	exact (totality)
RevealSeed($i \neq I$) / RevealComponent($i \neq I, type \in \{\text{mlkem}, \text{x25519}\}$), key created	local material per the game rule: ERASED marker when erased; otherwise return the material and mark every slot under i , incl. unoccupied ones, ineligible	exact
RevealSeed(I) / RevealComponent($I, type \in \{\text{mlkem}, \text{x25519}\}$), key created, $erased_I = \text{true}$	return ERASED ; no material involved, no marking	exact
RevealSeed(I) / RevealComponent($I, type \in \{\text{mlkem}, \text{x25519}\}$), key created, $erased_I = \text{false}$, pre-test (the material-returning reveals)	<i>coin-abort</i> : needs the seed or components of pk^*	coin-safe: marks every slot under I , incl. unoccupied ones, ineligible
RevealSeed(I) / RevealComponent($I, type \in \{\text{mlkem}, \text{x25519}\}$), key created, $erased_I = \text{false}$, after the passed test on (I, S)	output a fair coin	exact: the game ends with an independent coin
Erase(i), key uncreated or out of range; any i incl. I	$\perp$; no state change, <i>no flag is set</i>	exact (totality)
Erase(i), key created; any i incl. I ; repeated calls idempotent	set $erased_i$, acknowledge; nothing is returned	exact
Restore(i), key uncreated or out of range; any i incl. I	$\perp$; no state change	exact (totality)
Restore($i \neq I$), key created, every case	local per the lifecycle rule	exact
Restore(I), key created, not currently erased (incl. never erased)	$\perp$, no state change; no material involved, <i>no abort</i>	exact: matches the game's never-erased-restore rule
Restore(I), key created, $erased_I = \text{true}$, pre-test	<i>coin-abort</i> : the game's reply returns the packed seed	coin-safe: marks every slot under I , incl. unoccupied ones, ineligible
Restore(I), key created, $erased_I = \text{true}$, after the passed test on (I, S)	output a fair coin	exact: the game ends with an independent coin
RevealRoot(i, s, τ), slot unoccupied or out of range; any (i, s) incl. (I, S)	$\perp$; no state change	exact (totality)
RevealRoot($i \neq I, s, \tau$), occupied	local	exact
RevealRoot(I, s, τ), $s \neq S$, occupied	$R(K_{I,s}, \tau)$ from the stored secret; mark (I, s) ineligible	exact
RevealRoot(I, S, τ), occupied, pre-test	<i>coin-abort</i> : needs K^*	coin-safe: marks (I, S) ineligible
RevealRoot(I, S, τ), occupied, after the passed test on (I, S)	output a fair coin	exact: the game ends with an independent coin
Continue(i, s), slot unoccupied or out of range; any (i, s) incl. (I, S)	$\perp$; no state change, <i>no ineligibility is recorded</i>	exact (totality)
Continue(i, s), occupied, pre-test; any (i, s) incl. (I, S)	record ineligibility; nothing is returned, so no abort is needed	exact: a later $\text{Test}(I, S, \cdot)$ fails the eligibility check on both sides
Continue(i, s), occupied, $(i, s) \neq (I, S)$, after the passed test	record ineligibility; nothing is returned	exact
Continue(I, S), occupied, after the passed test on (I, S)	output a fair coin	exact: the game ends with an independent coin
Test($\cdot, \cdot, \cdot$), permission already consumed	$\perp$; no state change (the game continues)	exact (totality)

Query and case	$\mathcal{B}$'s action	Exactness
$\text{Test}(i, s, \tau)$, first call, $(i, s) \neq (I, S)$	output a fair coin without calling TestRoot	coin-safe: $T \neq (I, S)$ whether or not the checks pass
$\text{Test}(I, S, \tau)$, first call, all checks pass ($\mathcal{B}$ evaluates every check from its bookkeeping, incl. the normally-answered RootDec history)	call $\text{TestRoot}(\tau)$, forward the returned root; keep simulating under these same rows	exact conditioned on the guess: the single-user challenge root has exactly the branch- b distribution
$\text{Test}(I, S, \tau)$, first call, some check fails	output a fair coin; permission consumed	exact: independent coin on both sides
$\mathcal{A}$ never calls Test	halt without TestRoot	exact: both games output an independent coin by the no-test convention

All exact draft-10 public keys have the same distribution, and all honest encapsulations under a fixed public key have the same distribution, so the public simulation is independent of the hidden guess (I, S) . Static indices are assigned at creation and never renumbered. By the table, $\mathcal{B}$ deviates from the challenger only at coin-aborts, and every coin-abort query disqualifies (I, S) under the game's own rules, by marking it ineligible or by entering the Test history condition for $\text{ct}_{I,S}$. Consequently: in any run whose single test is valid and lands on (I, S) , no coin-abort query ever occurred, the simulation was exact end-to-end, and $\mathcal{B}$'s output equals the multi-user game's output (the adversary's bit, or the game's independent coin after a post-test disqualification, which $\mathcal{B}$ mirrors with its own fair coin). In every other run (a different test slot, a failed or absent test, or a coin-abort), $\mathcal{B}$'s output is an independent fair coin, and when $\mathcal{B}$ never called TestRoot the staged game's no-test convention produces that coin itself. Conditioned on $T = (I, S)$, the simulation is perfect in both challenge branches.

Let $p_b = \Pr[\text{MU-RR}_b^{\mathcal{A}} \Rightarrow 1]$ and, for $t \in \mathcal{S}$, let $p_b(t) = \Pr[T = t \wedge \text{output } 1]$ in branch b ; the no-test convention gives $p_b = \sum_{t \in \mathcal{S}} p_b(t) + \Pr[T = \perp] \cdot \frac{1}{2}$. Couple $\mathcal{B}$'s execution with a multi-user execution that shares all randomness except the guess, which is uniform on $\mathcal{S}$ and independent of the run. The embedded challenge objects are distributed exactly as honest ones, so the coupled run is distributed exactly as MU-RR_b ; and T is fixed before the challenge root is shown, so the distribution of T is independent of b . If the coupled run's single valid test equals the guess (I, S) , then no query disqualifying (I, S) (by ineligibility marking or by the Test history condition) ever occurred; by the abort principle $\mathcal{B}$ never coin-aborted, its simulation was exact to the end, and its output equals the multi-user game's output ($\mathcal{A}$'s bit, or the game's post-test independent coin, which $\mathcal{B}$ mirrors with its own fair coin). In every other case (a different test slot, a failed or absent test, or a coin-abort at a query that marked the guessed slot ineligible), $\mathcal{B}$ outputs an independent fair coin, produced by $\mathcal{B}$ itself or by the staged game's no-test convention. Note the harmless cases the table certifies $\mathcal{B}$ survives without aborting: a never-erased or uncreated $\text{Restore}(I)$ (answered $\perp$), reveals or Erase calls on uncreated keys (answered $\perp$, no flag set), an out-of-domain $\text{RevealComponent type}$ in any key state (answered $\perp$, marks nothing), post-erasure reveals under I (answered ERASED), $\text{Erase}(I)$ itself, Continue on any occupied slot (recorded, nothing returned) or unoccupied slot ($\perp$, nothing recorded), a second $\text{Test}(\perp)$, RootDec under a not-live key ($\perp$), and a wrong-length RootDec in any key state ($\perp$, enters no Test history). Averaging over the guess, for each b ,

$$\begin{aligned}
\Pr[\text{RR}_b^{\mathcal{B}} \Rightarrow 1] &= \frac{1}{N_T} \sum_{t \in \mathcal{S}} p_b(t) + \left(1 - \frac{\Pr[T \neq \perp]}{N_T}\right) \frac{1}{2} = \frac{1}{N_T} \left(p_b - \frac{\Pr[T = \perp]}{2}\right) + \frac{1}{2} - \frac{1 - \Pr[T = \perp]}{2N_T} \\
&= \frac{1}{N_T} p_b + \left(1 - \frac{1}{N_T}\right) \frac{1}{2}.
\end{aligned} \tag{16}$$

Subtracting the two branch equations and taking absolute values yields $\text{Adv}_{\mathcal{B}}^{\text{RR}} = \text{Adv}_{\mathcal{A}}^{\text{MU-RR}} / N_T$, which is Equation (15). $\square$

Theorem 9.3 (Complete static-slot multi-user post-quantum bound). *Under the assumptions of Theorem 8.1,*

$$\begin{aligned}
\text{Adv}_{\mathcal{A}}^{\text{MU-RR-PQ}} &\leq N_T \left(\text{Adv}_{\text{SHAKE256}, \mathcal{P}_0}^{\text{PRG-cq}} + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_1}^{\text{PRG-cq}} + \text{Adv}_{\text{MLKEM768}, \mathcal{B}_1}^{\text{IND-CCA-pq}} + \text{Adv}_{\text{MLKEM768}, \mathcal{B}_2}^{\text{IND-CCA-pq}} \right. \\
&\quad \left. + \text{Adv}_{F^{10}, \mathcal{D}}^{\text{PRF-cq}} + \text{Adv}_{F^{10}, \mathcal{E}}^{\text{PRF-cq}} + 2\delta_M + \text{Adv}_{\text{HMAC-SHA256}, \mathcal{H}}^{\text{PRF-cq}} \right).
\end{aligned} \tag{17}$$

Proof. Combine Lemma 9.2 with Theorem 8.1. The target factor is charged once around the complete single-user bound. $\square$

Theorem 9.4 (Complete static-slot multi-user classical bound). *Under the assumptions and classical random-oracle model (Definition 3.5) of Theorem 8.2,*

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{\text{MU-RR-C}} \leq N_T \Big(& \text{Adv}_{\text{SHAKE256}, \mathcal{P}_0^C}^{\text{PRG-cc}} + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_1^C}^{\text{PRG-cc}} + 2\Delta_X + \text{Adv}_{\text{X25519}, \mathcal{B}}^{\text{SDH}} \\ & + \frac{q_G + q_J + 2}{2^{256}} + \text{Adv}_{\text{HMAC-SHA256}, \mathcal{H}_C}^{\text{PRF-cc}} \Big). \end{aligned} \quad (18)$$

Here the composed experiment additionally simulates the multi-user interfaces, so $q_G \leq q_G^A + Q_D + (N_K - 1) + (N_K N_S - 1)$ and $q_J \leq q_J^A + Q_D$, with Q_D now the multi-user RootDec total (Section 13).

Proof. Combine Lemma 9.2 with Theorem 8.2. □

10 System realization corollary

The preceding theorems are cryptographic games. Mapping a running messenger into those games requires exact event obligations. One obligation deserves its own definitions and lemmas: the games test the *encapsulator's* stored secret $K_{i,s}$, while the real responder derives its root from $\text{XW}^{10}.\text{Dec}(sk_i, ct_{i,s})$. The two agree except for a correctness failure. Deployed key generation derives its component randomness from one SHAKE-expanded seed, so the failure probability must first be *named* for that exact distribution and then transferred to the FIPS 203 parameter by an explicit PRG-correctness hop; the security hops of Lemma 6.1 live inside a different experiment and cannot serve as this transfer.

Definition 10.1 (Shared-seed correctness parameter). *Let (sk, pk) be generated by exact draft-10 shared-seed key generation (uniform 32-byte sk ; (d, ζ, sk_X) parsed from $\text{SHAKE256}(sk, 96 \text{ bytes})$) and let $(K, ct) \leftarrow \text{XW}^{10}.\text{Enc}(pk)$ be an honest encapsulation. Define*

$$\delta_{\text{seed}} = \Pr[\text{XW}^{10}.\text{Dec}(sk, ct) \neq K]$$

over the key-generation and encapsulation randomness: the correctness failure probability of the exact deployed distribution.

Lemma 10.2 (Draft-10 root agreement). *For every correctly paired draft-10 key, one whose public X25519 value is derived from its private scalar ($pk_X = \text{X25519}(sk_X, B)$; every output of draft-10 **Expand** has this form, whatever 32-byte seed produced it), and every honest encapsulation, X-Wing disagreement implies ML-KEM disagreement: if $\text{XW}^{10}.\text{Dec}(sk, ct) \neq K$, then the decapsulated ML-KEM secret differs from the encapsulated one. The X25519 component agrees identically for all inputs. In particular δ_{seed} is bounded by the probability of the ML-KEM correctness event under shared-seed key generation.*

Proof. Encapsulation computes ss_M by ML-KEM encapsulation, samples an ephemeral scalar seed e , and sets $ct_X = \text{X25519}(e, B)$, $ss_X = \text{X25519}(e, pk_X)$, where B is the base point and $pk_X = \text{X25519}(sk_X, B)$. Decapsulation computes $ss'_M = \text{MLKEM768}.\text{Dec}(sk_M, ct_M)$ and $ss'_X = \text{X25519}(sk_X, ct_X)$.

The X25519 component agrees *identically*, not merely with high probability. For every input coordinate, the X25519 function evaluates the u -coordinate of Montgomery-ladder scalar multiplication by the clamped scalar, and the ladder is correct on Curve25519 and on its quadratic twist, so for all scalars a, b :

$$\text{X25519}(a, \text{X25519}(b, B)) = u([\text{clamp}(a)][\text{clamp}(b)] P_B) = \text{X25519}(b, \text{X25519}(a, B)),$$

because scalar multiplications commute in the endomorphism ring. With $a = sk_X$, $b = e$ this gives $ss'_X = ss_X$ always.

The combiner (4) is a deterministic function of (ss_M, ss_X, ct_X, pk_X) ; ct_X, pk_X are byte-identical on both sides, and $ss'_X = ss_X$ always. The two shared secrets can therefore differ only if $ss'_M \neq ss_M$ (the ML-KEM correctness event [4]), so the X-Wing disagreement event is *contained in* the ML-KEM disagreement event and bounded by its probability. (The converse containment is not claimed: it would require injectivity of the combiner in ss_M , and a SHA3 collision could make distinct ML-KEM secrets yield the same X-Wing secret.) The FIPS 203 parameter δ_M is stated for ML-KEM's *native* key-generation randomness (independent uniform seed material) while shared-seed generation supplies SHAKE-derived bytes; the next lemma pays for exactly that distribution gap. □

Lemma 10.3 (Seed-derived correctness transfer). *There is an explicit distinguisher $\mathcal{P}_{\text{corr}}$ such that*

$$\delta_{\text{seed}} \leq \delta_M + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_{\text{corr}}}^{\text{PRG}},$$

where δ_M is the FIPS 203 ML-KEM-768 correctness failure probability over native key-generation randomness. $\mathcal{P}_{corr}$ is a fixed classical polynomial-time algorithm, distinct from the security distinguishers $\mathcal{P}_0, \mathcal{P}_1$ of Lemma 6.1; its classical PRG advantage suffices, and any post-quantum PRG bound dominates it.

Proof. $\mathcal{P}_{corr}$ receives a 96-byte challenge Z , parses (d, ζ, sk_X) as its three 32-byte segments, constructs the ML-KEM key pair from (d, ζ) by the deterministic internal key generation of FIPS 203 and the X25519 public key from sk_X , performs one honest X-Wing encapsulation and one decapsulation, and outputs 1 exactly when the decapsulated secret differs from the encapsulated one.

If $Z = \text{SHAKE256}(s, 96 \text{ bytes})$ for a uniform 32-byte s , the constructed key pair and the encapsulation have exactly the shared-seed distribution of Definition 10.1, so $\mathcal{P}_{corr}$ outputs 1 with probability exactly δ_{seed} . If Z is uniform, (d, ζ) are independent uniform 32-byte strings (precisely ML-KEM’s native key-generation randomness), and by Lemma 10.2 the output-1 event is contained in the ML-KEM correctness event, so its probability is at most δ_M ; the value of sk_X is irrelevant because the X25519 component agrees for every scalar. Hence

$$\delta_{seed} \leq \delta_M + |\Pr[\mathcal{P}_{corr}(\text{SHAKE256}(s, 96)) = 1] - \Pr[\mathcal{P}_{corr}(U_{768}) = 1]| = \delta_M + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_{corr}}^{\text{PRG}}.$$

The reductions of Lemma 6.1 run inside the IND-CCA experiment and bound advantage differences, not correctness probabilities; they cannot substitute for this hop, which is why $\mathcal{P}_{corr}$ is constructed separately. $\square$

When ML-KEM decapsulation fails, implicit rejection returns $\bar{K} = J(z \parallel ct_M)$, and the responder derives a root unrelated to the encapsulator’s. In the intended composition, this event should cause confirmation failure and session termination. Establishing that it has no confidentiality or authentication consequence is a separate post-use obligation: the post-use protocol is excluded here, so distinguishable error behavior, retries, state desynchronization, an oracle created by later messages, and handling under mismatched roots are not analyzed by this paper. Inside this paper the event means that the tested object is the encapsulator’s root, and the responder’s root inherits the theorem only through the agreement event above. The system lift makes this explicit as Bad_{corr} .

Define the following branch-indexed bad events in the real protocol experiment $b \in \{0, 1\}$:

Event	Exact failure condition
Bad_{key}	A target-eligible complete X-Wing key was not generated by exact draft-10 shared-seed generation, or a component was imported, exported, reused, or paired independently.
Bad_{parse}	A transcript reaching RootReady is not the unique canonical re-encoding of one 44-field semantic object, or an accepted field has a wrong length, duplicate ID, alias, unknown critical ID, or trailing data.
Bad_{map}	A target-eligible RootReady event lacks a corresponding honest encapsulation under the exact public key and exact ciphertext represented in the transcript.
Bad_{replay}	The exact challenge ciphertext reaches a second unrestricted root computation, or an altered replay reaches RootReady under the same replay identity.
$\text{Bad}_{boundary}$	Before Test , any target root, raw X-Wing secret, or root-derived AEAD, header, acknowledgement, ratchet, confirmation, or other descendant is exposed.
Bad_{conc}	A concurrency or crash window violates the RootReady ordering without matching another row: interleaved tasks, duplicate completion callbacks, a crash, restart, or transaction rollback allow a target root or descendant to be emitted, or a slot’s state to be recorded, outside the happens-before order (transcript finalization, then authorization, then root creation, then freshness recording, then any downstream use) that the identical-until-bad coupling assumes.
Bad_{seed}	The target packed seed or expanded component is exposed while the target remains classified fresh, or an erased value is restored by rollback.
Bad_{rng}	Any honest seed, ML-KEM encapsulation randomness, X25519 ephemeral seed, or handshake identifier deviates from the specified distribution.

Event	Exact failure condition
Bad_{corr}	The tested slot's decapsulated secret disagrees with its encapsulated secret: $\text{XW}^{10}.\text{Dec}(sk_i, ct_{i,s}) \neq K_{i,s}$ for the <i>adaptively selected</i> tested slot T . The adversary selects T from the public view among up to N_T honest slots, and no independence between that selection and the per-slot correctness events is assumed, so the charge is the union bound over every slot the selection could land on: $\Pr[\text{Bad}_{\text{corr}}^{(b)}] \leq N_T \delta_{\text{seed}}$ per branch (Definition 10.1, Lemmas 10.2 and 10.3). A realization requiring <i>every</i> honest responder root to agree carries the same $N_K N_S \delta_{\text{seed}}$ bound.

Let $\text{Bad}^{(b)}$ be their union in branch b , and define

$$\epsilon_{\text{sys}} = \Pr[\text{Bad}^{(0)}] + \Pr[\text{Bad}^{(1)}]. \quad (19)$$

In particular the responder-correctness contribution to ϵ_{sys} is at most $2N_T \delta_{\text{seed}}$ ($N_T \delta_{\text{seed}}$ per branch from the Bad_{corr} row), not $2\delta_M$.

Corollary 10.4 (Real-system lift). *Suppose for each branch b the real experiment and the abstract multi-user game are defined on a common probability space (coupled) such that outside the event $\text{Bad}^{(b)}$ the two produce identical adversary views and outputs (identical-until-bad). Then*

$$\text{Adv}_{\mathcal{A}}^{\text{real}} \leq \text{Adv}_{\mathcal{B}}^{\text{MU-RR}} + \epsilon_{\text{sys}}. \quad (20)$$

Substituting Theorem 9.3 or Theorem 9.4 gives the corresponding system bound.

Proof. For each branch the coupled experiments produce different outputs only inside $\text{Bad}^{(b)}$; the difference lemma gives

$$|\Pr[\text{Real}_b \Rightarrow 1] - \Pr[\text{Abstract}_b \Rightarrow 1]| \leq \Pr[\text{Bad}^{(b)}].$$

Apply the triangle inequality across $b = 0$ and $b = 1$, then substitute the abstract-game theorem. $\square$

Exact scope

Equation (19) is a definition, not a hidden claim that these probabilities are negligible. The authentication and identity layers must bound Bad_{map} ; the state-handling layer must bound $\text{Bad}_{\text{replay}}$ and Bad_{seed} ; parser and conformance work must bound $\text{Bad}_{\text{parse}}$ and Bad_{key} . Bad_{conc} is a code-level obligation on the implementation and its runtime, and this paper claims no bound on it: discharging it requires a state relation over the running code that fixes the atomic definition of RootReady ; the happens-before rule among transcript finalization, authorization, root creation, freshness recording, and first use; the behavior of that order under concurrent tasks, duplicate callbacks, crashes, restarts, and transaction rollback; how historical key and session indices survive restart; the concrete horizon defining N_K and N_S with an injective mapping of production identifiers onto static slots; and which component owns and erases each copy of the root and the X-Wing secret. Bad_{corr} is the one event this paper bounds itself (Definition 10.1 with Lemmas 10.2 and 10.3: $N_T \delta_{\text{seed}}$ per branch); it is listed so the mapping claim is honest about which party's root the modular theorems certify. In the stopped-prefix experiment, $\text{Bad}_{\text{boundary}}$ is excluded by terminating the target at RootReady . In a continuing messaging protocol, root-derived output is expected rather than negligible, so no bound on $\text{Bad}_{\text{boundary}}$ can create an end-to-end theorem. A separate post-use composition game and proof are required.

11 Transcript and context binding

Lemma 11.1 (Exact adapter-message separation). *Assume the transcript parser is injective on accepted semantic objects and SHA-384 is collision resistant against the adversary class of the branch in which the lemma is read: classical adversaries when it accompanies the classical results, quantum adversaries when it accompanies the post-quantum results. For two accepted transcript objects (s, τ) and (s', τ') , where τ is the accepted byte string and s denotes the semantic transcript object the parser recovers from τ , equality of their 77-byte HMAC messages implies either $s = s'$ and $\tau = \tau'$, or a SHA-384 collision.*

Proof. The message consists of a fixed 23-byte label, one fixed zero byte, one fixed-width four-byte suite identifier, one 48-byte transcript digest, and one fixed counter byte. Equality implies equal suite identifiers and equal transcript digests. If the transcript byte strings differ, the two distinct domain-separated SHA-384 inputs collide. If the byte strings are equal, parser injectivity gives equal semantic objects. $\square$

This lemma is a context-binding statement, not a primitive term required by Theorems 8.1 and 8.2. Those theorems define freshness on the exact 77-byte target message. Context binding is needed to interpret that byte-level statement as agreement on semantic fields.

12 Post-use raw-root claim is false

Proposition 12.1 (AEAD verification distinguishes the raw root). *Suppose a visible artifact C verifies under an AEAD key $K_D(\text{root})$ deterministically derived from the target root. Let ϵ_{honest} bound honest verification failure, and define the wrong-root verification probability*

$$\epsilon_{wkv} = \Pr_{R' \leftarrow \{0,1\}^{256}} [\text{Verify}(K_D(R'), C) = 1].$$

A distinguisher given a candidate root has advantage at least

$$1 - \epsilon_{\text{honest}} - \epsilon_{wkv}$$

between the real root and an independent uniform root.

Proof. The distinguisher derives K_D from the candidate root and outputs “real” exactly when the visible artifact verifies. With the real root, verification succeeds with probability at least $1 - \epsilon_{\text{honest}}$. With an independent uniform root, success has probability exactly ϵ_{wkv} by definition. Subtracting the two probabilities proves the claim. $\square$

For an ideal t -bit authentication tag with unique key derivation, ϵ_{wkv} is typically near 2^{-t} . That estimate is an extra wrong-key-verification premise; it does not follow from tag length or ordinary AEAD authenticity alone.

No proof can make the raw root real-or-random hidden after such a verification artifact while still giving the distinguisher the raw candidate root. A post-use theorem must instead define a protocol confidentiality/authenticity game that never reveals the raw root.

13 Concrete loss and query ledger

Term	Branch	Exact role
N_T	both	One conservative static-slot target guess, charged once around the full single-user bound.
Two SHAKE PRG terms	both	One key-distribution hop for each IND-CCA endpoint. May be written $2\epsilon_{\text{PRG}}$ when both have the same resource bound.
Two ML-KEM IND-CCA terms	PQ	Imported QSF Theorem 2; each reduction makes at most Q_D decapsulation queries.
Two F^{10} PRF terms	PQ	Imported QSF Theorem 2; first uses at most $Q_D + 1$ evaluations, second at most Q_D .
$2\delta_M$	PQ	Two correctness transitions in the QSF proof. FIPS 203 describes a decapsulation-failure estimate near $2^{-164.8}$ for ML-KEM-768, obtained under a heuristic random-function analysis; it is an estimate, not a proved bound for deployed systems, and δ_M stays symbolic in every formal statement of this paper.
One HMAC PRF term	both	Direct adapter challenge with full challenge-ciphertext exclusion; one target evaluation.

Term	Branch	Exact role
$2\Delta_X$	classical	X25519 nominal-group distribution distance; imported bound $< 2^{-125}$.
SDH	classical	X25519 strong Diffie-Hellman reduction; at most $2q_H$ DH-oracle queries, q_H counted per Definition 3.5.
$(q_G + q_J + 2)/2^{256}$	classical	Final-FIPS ML-KEM C2PRI in the independent oracles $\mathcal{O}_G, \mathcal{O}_J$ of Definition 3.5; the query formulas are in Section 13.1.
$N_T \delta_{seed}$ root agreement	system only	Encapsulator–responder root agreement for the adaptively tested slot, $\delta_{seed} \leq \delta_M + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_{corr}}^{\text{PRG}}$ (Definition 10.1, Lemmas 10.2, 10.3); charged $N_T \delta_{seed}$ per branch inside Bad_{corr} , never an extra term of the modular theorems. The X25519 component agrees identically; only ML-KEM contributes.
ϵ_{sys}	system only	Sum of the two branch bad-event probabilities in Equation (19); not part of the modular cryptographic theorem.

No deployment-wide “bit security” number follows until N_T, Q_D, q_H, q_G, q_J , adversary running times, primitive advantages, and system-event probabilities are fixed. In particular, the classical statistical term alone is strictly weaker than an unqualified 128-bit slogan.

13.1 Reduction resource ledger

Fix an adversary $\mathcal{A}$ with running time $T_{\mathcal{A}}$ that makes at most Q_D root-decapsulation queries (single-user RootDec, or the multi-user RootDec total) and, in the classical model, at most q_H^A, q_G^A, q_J^A direct queries to $\mathcal{O}_H, \mathcal{O}_G, \mathcal{O}_J$. Let Σ_{τ} bound the total byte length of every context string $\mathcal{A}$ supplies across all queries. One adapter evaluation R on an ℓ -byte context costs one SHA-384 pass over $\ell + 30$ bytes plus one HMAC call on the fixed 77-byte message; every reduction reads each supplied context exactly once, in that single pass, and retains none beyond it, so the entire context-hashing cost of any reduction is one linear scan of Σ_{τ} bytes. The canonical honest profile bounds honest transcripts by 8192 bytes; the games accept arbitrary context lengths, so bounds are stated in Σ_{τ} . Write t_{KG}, t_E, t_D for the cost of one exact draft-10 key generation, encapsulation, and decapsulation. Each reduction’s running time is $T_{\mathcal{A}}$ plus its displayed operation counts at these unit costs, plus the Σ_{τ} -byte scan, plus constant per-query bookkeeping except where a row states a larger marking cost.

Reduction (constructed in)	Oracle and challenge budget	Work added to $T_{\mathcal{A}}$
$\mathcal{B}$, KEM hop (Lemma 7.2)	at most Q_D decapsulation-oracle queries, one per RootDec	at most $Q_D + 1$ adapter evaluations; no key or encapsulation generated (challenger objects)
$\mathcal{E} / \mathcal{H} / \mathcal{H}_C$, PRF hop (Lemma 7.2; the theorems’ HMAC terms)	one evaluation query, on one 77-byte message	one key generation, one encapsulation, at most Q_D local decapsulations, at most Q_D adapter evaluations
$\mathcal{P}_0, \mathcal{P}_1$ and $\mathcal{P}_0^C, \mathcal{P}_1^C$ (Lemma 6.1)	one 96-byte PRG challenge	component-key construction from the challenge (one ML-KEM internal key generation, one X25519 base evaluation; no seed expansion), one honest encapsulation, at most Q_D decapsulations, at most $Q_D + 1$ adapter evaluations
$\mathcal{P}_{corr}$ (Lemma 10.3)	one 96-byte PRG challenge	fixed and adversary-independent: one component-key construction, one encapsulation, one decapsulation, one comparison

Reduction (constructed in)	Oracle and challenge budget	Work added to T_A
$\mathcal{B}_1, \mathcal{B}_2$, ML-KEM IND-CCA (Lemma 6.3, imported)	at most Q_D inner decapsulation queries each	at most Q_D+1 F^{10} evaluations (one per query and one for the challenge secret), at most Q_D+3 X25519 evaluations (one per query plus the one-time challenge setup), plus the outer at most Q_D+1 adapter evaluations
$\mathcal{D}, \mathcal{E}$, F^{10} PRF (Lemma 6.3, imported)	at most Q_D+1 and Q_D evaluation queries	the same simulation work as the row above, with the ML-KEM key held and decapsulated locally
$\mathcal{B}$, SDH (Lemma 6.6, imported)	at most $2q_H$ DH-oracle queries	at most Q_D simulated decapsulations, the one-time challenge assembly, the outer at most Q_D+1 adapter evaluations, plus bookkeeping linear in q_H
$\mathcal{C}$, C2PRI (terminal classical adversary)	q_G, q_J as displayed below	at most Q_D ML-KEM decapsulations by the held key with one X25519 and one combiner evaluation per query, the one-time X25519 component setup and challenge combiner evaluation, plus the outer at most Q_D+1 adapter evaluations
$\mathcal{B}$, target guess (Lemma 9.2)	single-user budget: at most Q_D forwarded RootDec queries and one TestRoot	at most N_K-1 local key generations and $N_K N_S-1$ local encapsulations, at most Q_D local decapsulations, one adapter evaluation per answered RootDec or RevealRoot; bookkeeping $O(N_K N_S)$ at setup, $O(N_S)$ per material-returning reveal or restore (slot marking), constant amortized otherwise

The classical oracle totals follow from the per-operation patterns of Definition 3.5. In the composed single-user experiment the terminal C2PRI adversary's counts, in the sense of Proposition 6.7, satisfy

$$q_G \leq q_G^A + Q_D, \quad q_J \leq q_J^A + Q_D,$$

since every one of its $\mathcal{O}_G/\mathcal{O}_J$ queries is a forwarded direct query of $\mathcal{A}$ or one of the two queries induced by one of its at most Q_D simulated decapsulations, and the win-check pair rides the proposition's additive constant. In the composed multi-user experiment the simulation additionally performs at most N_K-1 local key generations and $N_K N_S-1$ local encapsulations, one $\mathcal{O}_G$ query each:

$$q_G \leq q_G^A + Q_D + (N_K-1) + (N_K N_S-1), \quad q_J \leq q_J^A + Q_D.$$

The experiment-wide $\mathcal{O}_H$ totals behind the SDH translation are $q_H \leq q_H^A + Q_D + 3$ for the single-user experiment (the Q_D combiner evaluations, one key generation, and the challenge encapsulation's two inputs) and $q_H \leq q_H^A + Q_D + N_K + 2N_K N_S$ for the multi-user experiment. These totals dominate every adversary-side count, so substituting them anywhere is conservative.

14 Exact implementation obligations

A conforming implementation of the proved suite must:

1. use exact draft-10 key generation, object lengths, combiner order, and six-byte label;
2. use the complete X-Wing key as one typed object and never use a component independently;
3. perform the final FIPS 203 encapsulation-key check before encapsulation;
4. use ML-KEM implicit rejection and expose no component-specific failure;

5. not add an X25519 all-zero abort to this suite, and compute ss_X in decapsulation with the raw RFC 7748 ladder as a total function, the sole conforming path on every backend (Section 15); never derive the value from a caught backend refusal, and never map any backend failure to zeros;
6. never assume a platform-native hybrid-KEM API is byte-compatible with draft-10 without fresh vectors (Apple’s native X-Wing surface has referenced an earlier draft profile); compose from components against the pinned vectors otherwise;
7. compute the 77-byte adapter message exactly and use direct one-block HKDF-Expand with the X-Wing secret as PRK;
8. reject noncanonical transcript encodings before hashing;
9. bind the complete X-Wing public key and complete ciphertext into the canonical transcript;
10. verify authorization and signatures before classifying a responder computation as target-eligible;
11. prevent the challenge ciphertext from reaching an unrestricted second root computation;
12. expose no target-root-dependent output before the pre-use test boundary; and
13. treat any byte, draft, algorithm, validation, or ordering change as a new suite requiring new vectors and a new proof package.

15 X25519 all-zero behavior: statement and evidence

Draft-10 decapsulation computes $ss_X = \text{X25519}(sk_X, ct_X)$ and always passes the result into the combiner. The draft performs no all-zero or small-order check, and RFC 7748 makes such a check optional at the protocol layer. Adversarial ciphertexts are part of the IND-CCA experiment: for a small-order ct_X , the mathematical X25519 output is the all-zero string, and a conforming decapsulation returns

$$\text{SHA3-256}(ss_M \parallel 0^{32} \parallel ct_X \parallel pk_X \parallel L_{XW})$$

rather than aborting. The classical theorem tolerates such queries through the nominal-group model, and the post-quantum theorem makes no X25519 hardness assumption; in both, decapsulation is a total function of the ciphertext. An implementation that aborts on the all-zero output therefore implements a *different KEM* from the one analyzed, and none of the theorems in this paper transfer to it without a new argument.

Common backends enforce RFC 7748’s optional check at the API layer: OpenSSL’s EVP X25519 derivation and Apple CryptoKit both refuse a key agreement whose output would be all-zero. This was confirmed empirically for $u = 0$, $u = 1$, and the order-8 point `e0eb7a7c...b800`: OpenSSL refused all three derivations (OpenSSL 3.6.3 in this package’s sealed evidence run, recorded in `reproduction/reproduce_xwing_full.out`; earlier evidence runs on 3.5.5 and 3.6.2 refused identically), and Apple CryptoKit on macOS 26 threw for all three (`underlyingCoreCryptoError(-7)`); the package preserves both as sealed evidence (`reproduction/`).

The reproducer’s decapsulation therefore computes ss_X with a *raw RFC 7748 Montgomery ladder* (a TweetNaCl-derived public-domain implementation embedded in the source), which is a total deterministic function of the inputs and yields the mathematical value 0^{32} for small-order ct_X without any backend interaction. A previous revision instead caught the backend refusal and substituted zeros; that pattern maps *every* backend failure (allocation, provider, context) to a cryptographic zero and is withdrawn. The backend behavior is now separate evidence, never a computation path: on every honest path the ladder output is asserted byte-equal to OpenSSL’s EVP derivation, and on the three verified low-order inputs the reproducer asserts both that EVP *refuses* and that the ladder yields 0^{32} , which then enters the combiner. Three small-order ct_X cases per pinned vector confirm that decapsulation completes without abort, that the combiner input contains $ss_X = 0^{32}$, that the recomputed shared secret equals an independently computed combiner evaluation, and that it differs from the honest vector value.

The conformance obligation for X-Wing decapsulation is single: *always compute ss_X with the raw RFC 7748 ladder*, on every backend. An earlier revision of this section offered an alternative, catching the backend refusal and substituting 0^{32} after classifying the peer value as low-order; that alternative is withdrawn as a conforming path and survives here only as this deprecation notice. The reason is the shape of RFC 7748 decoding itself: the ladder masks the high bit of the peer coordinate and accepts non-canonical field encodings, reducing them modulo p , so several distinct byte encodings denote one field element, and a substitution classifier would have to be exact, exhaustive, and constant-time over every accepted encoding, including high-bit-set and non-canonical ones; anything less maps some backend failures to zeros, which is the withdrawn defect in another

form. Propagating a refusal as a decapsulation failure likewise implements a different KEM. Any future Auris implementation on Apple CryptoKit must satisfy the raw-ladder obligation explicitly, because CryptoKit’s key-agreement API throws on the all-zero output. Separately: Apple’s native hybrid-KEM API surface must *never* be assumed byte-compatible with draft-10 (its documentation has referenced an earlier X-Wing draft profile), so an implementation either composes the hybrid from components against this paper’s pinned vectors or proves byte compatibility with fresh vectors first.

16 Theorem ledger

Identifier	Status in this paper	Exact disposition
R2-OLD-COMBINED	Refuted	A classical X25519 fallback is not a surviving premise in one active-quantum theorem.
R2-SEED-ST	Proved with correction	Two endpoint PRG hops under the standard IND-CCA advantage; see Lemma 6.1.
R2-XW-PQ	Proved conditionally	Exact draft-10 family F^{10} , two ML-KEM terms, two PRF terms, and $2\delta_M$.
R2-XW-C	Proved conditionally	Exact random-oracle order transfer, X25519 nominal group, SDH, and C2PRI, in the global classical model of Definition 3.5.
R2-C2PRI-F203	Proved in ROM	Proposition 6.7; the independent oracles $\mathcal{O}_G, \mathcal{O}_J$ of Definition 3.5.
R2-OUTER-EXPAND	Proved conditionally	Lemma 7.2; direct one-block HMAC with challenge-ciphertext exclusion, stated for fixed-length total KEMs (Definition 7.1); draft-10 X-Wing qualifies.
R2-CONTEXT-384	Proved conditionally	Lemma 11.1; parser injectivity and SHA-384 collision resistance.
R2-HS-POSTUSE-ROOT	Refuted	Proposition 12.1.
R2-ROOTREADY-PQ-EXPAND	Complete conditional proof	Theorems 8.1 and 9.3; review status as described in Section 1.3.
R2-ROOTREADY-C-EXPAND	Complete conditional proof	Theorems 8.2 and 9.4; review status as described in Section 1.3.
R2-SU-STAGED	Proved conditionally	Definition 3.7 staged interface (root decapsulation before and after the test); Lemma 7.2 re-proved with any-order oracle access and identical terms.
R2-MU-STATIC	Proved conditionally	Definition 9.1 with total oracle semantics and Lemma 9.2; static (i, s) slots over the staged base game, complete per-case simulation table with first-match-ordered, pairwise-disjoint guarded rows (uncreated/unoccupied/not-live/at-capacity/wrong-length/type-domain/second-Test $\perp$ cases explicit; index assignment stated in successful-creation counting), coin-aborts only at guess-disqualifying queries, exact $N_K N_S$ identity in the fixed, static-slot, one-Test experiment.

Identifier	Status in this paper	Exact disposition
R2-MU-RESTORE-FIX	Proved conditionally	An earlier revision of this reduction aborted on every <code>Restore(I)</code> , while the game answers a never-erased <code>Restore</code> with a harmless $\perp$; the present simulation answers that case exactly and coin-aborts only on material-returning restores. The type-domain and ciphertext-length guards are threaded through every re-statement, the <code>NewKey</code> and <code>Encaps</code> index families state capacity-first rows with successful-creation counting, and the mechanical consistency check of Appendix C verifies the table, appendix, and companion restatements against a hand-coded game oracle.
R2-RESP-CORR	Proved	Definition 10.1 and Lemmas 10.2, 10.3: agreement fails with probability at most $\delta_{seed} \leq \delta_M + \text{Adv}_{\text{SHAKE256}, \mathcal{P}_{corr}}^{\text{PRG}}$ per honest session; the X25519 component agrees identically. Charged as $N_T \delta_{seed}$ per branch inside <code>Bad_{corr}</code> ; the modular theorems are encapsulator-scoped.
R2-C2PRI-REMOVAL	Not proved	Transcript binding does not remove C2PRI from the classical X-Wing theorem.
R2-MAL-BIND	Not claimed	Fresh honestly generated target-key theorem only.
COMBINER-1	Open	The project’s requirement that this construction’s security case be independently reviewed before any deployment claim is made. The mathematics of this paper is complete under its assumptions; the obligation stays open until independent expert review has occurred (Section 1.3).

17 Canonical transcript profile

The proof treats accepted transcript bytes as primary. The profile is named `AURIS-T0-V5`, the one identifier used for it throughout this paper; its tag-length-value field layout below is the TLV1 encoding family, named here once. The byte-complete normative source for the profile is executable and published with the paper: the transcript constructor and parser of the artifact bundle (`reproduction/reproduce_transcript.py`, SHA-256 `a85392968554e0d8d3d8436ee06ba5c65ee0b3b1e87499379a106d917e132f67`, with the pinned vector objects, enumerated by hash in the bundle manifests). Acceptance of any candidate transcript is defined by that parser, and the constants displayed here are its constants; this section is a summary of that normative source, not a second authority. The profile uses:

```
TOBytes = ASCII("AURIS-T0-V5") || UINT16_BE(44) || Field[1] || ... || Field[44]
Field   = UINT16_BE(fieldId) || UINT32_BE(valueLength) || value
```

Fields occur exactly once in strictly increasing order. Integers are fixed-width unsigned big-endian. Optional values use `presence:u8 || payload`; absence is exactly `00`, presence exactly `01 || payload`. The parser re-encodes and byte-compares before hashing. Total size is bounded by 8192 bytes.

ID	Field	Encoding	Purpose
1	<code>domain</code>	18-byte ASCII	Cross-protocol separation
2	<code>protocolVersion</code>	<code>u32 = 5</code>	Version separation
3	<code>cipherSuiteId</code>	<code>u32 = 0x00050001</code>	Complete suite binding
4-8	<code>handshake, ratchet, header, transparency, state profile IDs</code>	five <code>u32</code>	No piecemeal negotiation
9	<code>rolePair</code>	two <code>u8</code> enums	Reflection prevention
10	<code>handshakeId</code>	32 bytes	Replay/session uniqueness

ID	Field	Encoding	Purpose
11–13	initiator master, epoch, device commitments	3 x 48 bytes	Authorization context
14	initiator epoch number	u64	Generation agreement
15	initiator deviceId	1–64 bytes	Device agreement
16	initiator Ed25519 key	32 bytes	Classical authentication key
17	initiator ML-DSA-65 key	1952 bytes	PQ authentication key
18–20	responder master, epoch, device commitments	3 x 48 bytes	Authorization context
21	responder epoch number	u64	Generation agreement
22	responder deviceId	1–64 bytes	Device agreement
23	responder Ed25519 key	32 bytes	Credential key
24	responder ML-DSA-65 key	1952 bytes	Credential key
25	prekey-generation commitment	48 bytes	Exact prekey authorization
26	prekey generationId	1–64 bytes	Rotation agreement
27	initial ratchet keyId	1–64 bytes	Separate ratchet selection
28	initial ratchet X25519 key	32 bytes	Ratchet binding only
29	initial ratchet expiry	u64	Freshness policy
30	selected hybrid key type	u8	One-time vs. last-resort
31	selected hybrid keyId	1–64 bytes	Exact lookup
32	selected hybrid generationId	1–64 bytes	Exact generation
33	selected X-Wing public key	1216 bytes	Complete KEM key binding
34	public-key commitment	48 bytes	Lookup/credential cross-check
35	selected key expiry	u64	Time validity
36	single-use policy	u8	OTP/reusable consistency
37	initiator ratchet public key	32 bytes	First ratchet binding
38	X-Wing ciphertext	1120 bytes	Complete encapsulation binding
39–40	capability commitments	2 x 48 bytes	Downgrade context
41–42	transparency commitments	2 x 48 bytes	Directory/log context
43	maximum first-message time	u64	Delay policy
44	extension-set commitment	48 bytes	Explicit extension/absence binding

18 Vector and parser reproduction summary

The supplied reproduction suite verifies:

- authentication of the exact author-repository vector JSON by SHA-256 `409efe197550b22985b4a0419418a0c5f2c2b193426c55bd998399ec8d3e614d`;
- byte-for-byte comparison of all three generated 1216-byte public keys, 1120-byte ciphertexts, and 32-byte shared secrets against that pinned source, plus independent full-object SHA-384 checks from the pinned Auris input packet;
- deterministic encapsulation and independent decapsulation of all three vectors: ML-KEM via OpenSSL (3.6.3 in the sealed evidence run; the toolchain line of `reproduce_xwing_full.out` names the exact version), the X25519 component via the embedded raw RFC 7748 ladder asserted byte-equal to the EVP derivation on every honest path, with all three 32-byte shared secrets matching;
- rejection of 1119-byte and 1121-byte ciphertexts, plus three same-length ciphertext mutations per vector (first and last ML-KEM byte, first X25519 ciphertext byte), each producing a different final shared secret;
- reconstruction of the complete 7516-byte, 44-field canonical `AURIS-T0-V5` transcript (leading magic as displayed in the profile section) from the generated public key and ciphertext rather than from a supplied transcript hash;
- canonical decode/re-encode, the transcript hash, authentication hash, session identifier, 76-byte root information, 77-byte HMAC input, and expected `rootKey0`;
- one binding mutation for each of the 44 fields; every mutation changes both the transcript hash and root;
- rejection of wrong magic, wrong field count, missing, duplicate, unknown, reordered, truncated-header, truncated-value, trailing-byte, oversized, wrong-length, commitment-mismatch, and key-type/policy-mismatch encodings; and

- draft-10 small-order ct_X decapsulation behavior per Section 15: for $u = 0$, $u = 1$, and the order-8 point, decapsulation completes without abort, the raw ladder’s all-zero ss_X enters the combiner (`ladder_zero=1`), and the backend refusal is asserted as separate trap evidence (`evp_refused=1`); no substitution path exists.

The first adapter vector reproduces:

Value	Hex / size
X-Wing secret	d2df0522128f09dd8e2c92b1e905c793 d8f57a54c3da25861f10bf4ca613e384
Transcript bytes	7516 bytes, 44 fields
Transcript-byte SHA-384	02a7cf929011e6c7b2e9347b67048b28164e648342c66efa 032bfaa2680dc101aa78f2ccc4e0bb419d55633c0b14e0fd
Transcript-domain SHA-384	9cc257859a59cfabe01cfca461875e909e9450b5d59e2320 0ff2b9d2e39f9c72a70166de2ea387740328d2e3a1a7135c
Root info / HMAC input <code>rootKey0</code>	76 / 77 bytes 2d30642fe1a075cacb3512f123c0bf09 f0ca3db2c069e4bb5519678e0221beea
Session identifier	5761ab54f019e848424b70789dc6bbf27aeb8015e009819f 24bf5b99d485d5a5d4953025494f5bd1728595e1e64c05fa

Reproduction establishes byte and parser consistency. It does not prove the primitive assumptions in Section 5.

19 Companion result: a machine-checked model of the v5 authentication layer

Authentication is outside this paper’s theorems (Section 5); this section summarizes a companion symbolic artifact because it is the intended separate source of peer-identity evidence for the handshake this paper analyzes. No composition theorem connecting the two is claimed here; establishing one is future work. The companion is a Tamarin 1.12.0 theory of the v5 device-authentication layer over a fully adversarial wire (standard Out/In message deduction, standard signature equations, no bespoke forgery rule, revocation as verifier-side linear state). Nine of its ten lemmas are machine-checked:

- **Five all-traces theorems** (verified by unattended search; 8/51/257/257/129 proof steps): a sources lemma; a session-key secrecy sanity check; non-injective agreement unless *both* of the peer device’s signing keys (Ed25519 and ML-DSA) were compromised before acceptance; and the two split-break survival theorems: authentication survives a complete break of either signature scheme alone, including a reading in which the Ed25519 compromise event is fully unrestricted and authentication rests on the symbolic ML-DSA key.
- **Four exists-trace witnesses** (guided interactive derivations, validated by Tamarin’s non-searching proof-object check; 812/64/118/68 steps): an honest session is reachable with all compromise events excluded, so the all-traces theorems are not vacuous; impersonation *is* derivable once both device signing keys leak (64 steps), so the model is adequate to express the attack it claims to exclude; a key-compromise-impersonation scenario and a crypto-only replay-acceptance scenario are likewise reachable.
- **One lemma remains open**: acceptance-after-visible-revocation impossibility (all-traces). Automated search exhausts resources on it and a first guided attempt met a recurring predecessor-chain obstruction; the documented route is an auxiliary no-reactivation invariant. It is reported open, not assumed.

These are results inside the stated symbolic model only: they establish no computational or quantum-soundness theorem, no statement about any implementation, and no composition with the games of this paper. The theory, the canonical proved artifact with embedded proof objects, per-lemma replay logs, and the exact validation commands are published in the artifact bundle accompanying this preprint (Appendix C). The witnesses replay with Tamarin’s non-searching lemma check; the five theorems re-prove by scoped search (the bundled logs record 359 seconds for the full scoped run on a desktop machine).

20 Final statement

Final mathematical disposition

The exact Auris v5 pre-use root theorem is proved conditionally for both the staged single-user game and the complete static-slot multi-user game with total oracle semantics. The proof uses exact draft-10 byte ordering, two shared-seed PRG endpoint hops, the published QSF reductions, final-FIPS C2PRI in the globally defined classical oracle model, the direct one-block HMAC adapter under any-order oracle access, irrevocable $(i_{key}, i_{session})$ slots, and the exact random-output guessing identity of the fixed static-slot one-Test experiment. The responder's root is covered through the explicit shared-seed correctness charge $N_T \delta_{seed}$ per branch inside the system lift, not by the modular theorems directly. The raw-root post-use theorem is refuted. The review status of these results is stated in Section 1.3: the proofs are conditional, internally reviewed, and await independent expert scrutiny, which this preprint exists to obtain.

A Reduction pseudocode

A.1 Outer KEM-to-root reduction

```

B^(KEM.Dec)(pk, ct_star, K_beta):
  give (pk, ct_star) to A
  valid_tested = false

  on A.RootDec(ct, tau):          # any time, before or after TestRoot
    if len(ct) != 1120: return bot # wrong length: no state change
    if ct == ct_star: return bot  # challenge exclusion
    K = KEM.Dec(ct)               # decapsulation oracle: whole experiment;
                                # returns a 32-byte secret for every
                                # 1120-byte ct != ct_star
    return HMAC_SHA256(K, M(tau)) # HMAC only on that 32-byte K

  on A.TestRoot(tau_star):        # adversary-chosen moment
    if valid_tested or not contains(tau_star, ct_star):
      return bot                  # invalid test: no state change,
                                # nothing consumed
    valid_tested = true
    return HMAC_SHA256(K_beta, M(tau_star)) # single use of K_beta

  if not valid_tested: return fresh random bit # no-test convention:
                                              # fires iff no precondition-satisfying
                                              # TestRoot occurred

  return A.output_bit()

```

A.2 PRF reduction for the direct adapter

```

H^(Eval):
  (sk, pk) = XWing.GenerateKeyPair()
  (K_unused, ct_star) = XWing.Encapsulate(pk)
  give (pk, ct_star) to A
  valid_tested = false

  on A.RootDec(ct, tau):          # any time, before or after TestRoot
    if len(ct) != 1120: return bot # wrong length: no state change
    if ct == ct_star: return bot  # challenge exclusion
    K = XWing.Decapsulate(sk, ct) # local secret key; returns a
                                # 32-byte secret for every
                                # 1120-byte ct
    return HMAC_SHA256(K, M(tau)) # HMAC only on that 32-byte K

```

```

on A.TestRoot(tau_star):
    if valid_tested or not contains(tau_star, ct_star):
        return bot          # invalid test: no state change,
                             # no Eval query is spent
    valid_tested = true
    return Eval(M(tau_star))    # the only Eval query

if not valid_tested: return fresh random bit # no-test convention:
                                     # fires iff no precondition-satisfying
                                     # TestRoot occurred
return A.output_bit()

```

A.3 Static-slot target-guess reduction

B chooses (I, S) uniformly from $[N_K] \times [N_S]$.

Totality bucket -- CHECKED FIRST for every incoming query, at any phase: return bottom, change no state, NO abort:

- any call naming an uncreated key, unoccupied slot, or out-of-range index (Erase, reveals, Restore, RevealRoot, Continue, RootDec, Encaps alike); NewKey at key capacity; Encaps under a created-but-not-live key or at session capacity (any i incl. I); any RootDec whose ct is not exactly 1120 bytes (any i incl. I , any key state, pre- or post-test); a RevealComponent type outside {mlkem, x25519} (any i incl. I , any key state, pre- or post-test); a 1120-byte RootDec under a created-but-not-live key; Restore(I) with the key not currently erased (incl. never erased); a second Test. These queries mark nothing and enter no Test history.

On the NewKey call performing the I -th SUCCESSFUL creation (creation counting, never call counting; capacity handled by the bucket above): publish the single-user challenge pk^* .

On the Encaps(I) call performing the S -th SUCCESSFUL session creation (key live; same counting rule): publish the held challenge ct^* .

Other successful creations: local key generation. Other successful sessions under I (key live): local honest encapsulation under pk^* ; retain $K_-(I, s)$.

Everything under keys $i \neq I$, and occupied RevealRoot($I, s \neq S$): answered locally from held material, with the challenger's exact bookkeeping (markings included).

RootDec(I, ct, τ), key live, 1120-byte $ct \neq ct^*$: forward to $RR.RootDec$ (staged game: legal before and after TestRoot; the forwarded ct satisfies the single-user length and challenge-exclusion preconditions); a pre-test call marks byte-matching occupied slots ineligible.

Pre-test RootDec(I, ct^*, τ), key live: output an independent random bit (the game's reply needs Dec(sk^*, ct^*); the normally answered call enters the Test history for $ct_-(I, S)$, so no later test on (I, S) is valid).

Pre-test RevealSeed(I) / RevealComponent(I , type in {mlkem, x25519}) on the created, non-erased key (the material-returning reveals), and material-returning Restore(I) (created, currently erased): output an independent random bit (each marks every slot under I , including unoccupied ones, ineligible). Erased-state reveals return only the ERASED marker -- no abort.

Pre-test RevealRoot(I, S, τ), occupied: output an independent random bit ((I, S) becomes ineligible).

Continue(i, s), occupied slot: pre-test (any, incl. (I, S)), or post-test with $(i, s) \neq (I, S)$: record ineligibility; nothing returned, NO abort.

Erase(i), created key (any i incl. I): set the flag, acknowledge.

Test(i, s, τ), first call:

- if $(i, s) \neq (I, S)$: output an independent random bit (no

```

    TestRoot; T != (I,S) either way).
    if (i,s) == (I,S) and every check passes (B evaluates them
      all from its bookkeeping): call RR.TestRoot(tau);
      forward Y_b; keep simulating under these same rules.
    if (i,s) == (I,S) and a check fails: output an independent
      random bit (the game itself ends with an independent coin).
After the passed test on (I,S), a target-disqualifying query
(RootDec(I,ct*,.) on the live key; material-returning reveals
or Restore under I; RevealRoot(I,S,.); Continue(I,S)): output
a fair coin (the game ends with an independent coin). Every
other query continues under the rules above.
No Test: halt without TestRoot (the staged game supplies the coin).
Otherwise return A's final bit.

```

B Compact game ledgers

B.1 Post-quantum single-user chain

Game	Change	Charged term
G0	Real shared-seed draft-10 contextual root game	–
G1	Replace shared-seed key distribution in the real endpoint by independent component randomness	first SHAKE PRG term
G2–G5	Apply published QSF post-quantum chain to replace X-Wing challenge secret by uniform	two ML-KEM IND-CCA, two F^{10} PRF, $2\delta_M$
G6	Replace one-block HMAC target output by uniform	one HMAC PRF term
G7	Restore shared-seed public-key distribution at the random endpoint	second SHAKE PRG term
G8	Exact random-root endpoint	zero remaining gap

B.2 Classical single-user chain

Game	Change	Charged term
G0	Real shared-seed draft-10 contextual root game	–
G1	Independent component randomness at real endpoint	first SHAKE PRG term
G2	Apply QSF classical ROM theorem	$2\Delta_X + \text{Adv}^{SDH} + \text{Adv}^{C2PRI}$
G3	Instantiate final-FIPS C2PRI	$(q_G + q_J + 2)/2^{256}$
G4	Replace one-block HMAC target output by uniform	HMAC classical PRF
G5	Restore shared-seed public-key distribution at random endpoint	second SHAKE PRG term
G6	Exact random-root endpoint	zero remaining gap

C Artifact, versioning, and provenance

C.1 Artifact bundle

All artifacts accompanying this paper are available at the following archive identifier, whose record is published together with this preprint:

[doi:10.5281/zenodo.21486656](https://doi.org/10.5281/zenodo.21486656)

The bundle contains the complete source package of this paper (the proof documents, the pinned test vectors, and the reproduction suite, including the mechanical consistency check for the multi-user simulation table) and the companion authentication model (the Tamarin theory, the canonical proved artifact with embedded proof objects, and the per-lemma validation logs). Every file is enumerated with its SHA-256 digest in the bundle’s checksum manifests. Two anchors, one per artifact family:

Object	SHA-256
Canonical proved Tamarin theory	c8b3b831b7875bc8153cf3ac0103b86d d6fadaaf06fd6f01f845f20caa4ef9e7
Pinned X-Wing draft-10 vectors	409efe197550b22985b4a0419418a0c5 f2c2b193426c55bd998399ec8d3e614d

To reproduce: `bash reproduction/run_all.sh` verifies every vector and generated object byte-for-byte; `python3 reproduction/cross_restatement_gate.py` checks the simulation-table restatements; the Tamarin witness proofs validate with `tamarin-prover --lemma=<name> <theory>` (no search), and the five unattended theorems re-prove with the scoped `--prove` command listed in the bundle’s README.

C.2 Versioning and provenance

This is revision 1.16 of the package and its first public edition; earlier internal revisions were never published. Published bytes are never edited: corrections are issued as errata alongside the artifact, and any revised edition takes a new revision number. The development, checking, and pre-publication adversarial review of this work made substantial use of AI assistance; all mechanically checkable artifacts are published as described above, and the internal review archives, including verbatim reports and adjudications, are retained by the project. The review status of the results is stated in Section 1.3.

References

- [1] M. Barbosa, D. Connolly, J. D. Duarte, A. Kaiser, P. Schwabe, K. Varner, and B. Westerbaan. *X-Wing: The Hybrid KEM You’ve Been Looking For*. IACR Communications in Cryptology, 1(1), 2024. DOI: 10.62056/a3qj89n4e. Cryptology ePrint Archive 2024/039.
- [2] D. Connolly, P. Schwabe, and B. Westerbaan. *X-Wing: general-purpose hybrid post-quantum KEM*. Internet-Draft `draft-connolly-cfrg-xwing-kem-10`, 2 March 2026. Work in Progress.
- [3] D. Connolly, R. Barnes, and P. Grubbs. *Hybrid PQ/T Key Encapsulation Mechanisms*. Internet-Draft `draft-irtf-cfrg-hybrid-kems-12`, 6 July 2026. Work in Progress.
- [4] National Institute of Standards and Technology. *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard*. August 2024. DOI: 10.6028/NIST.FIPS.203.
- [5] H. Krawczyk and P. Eronen. *HMAC-based Extract-and-Expand Key Derivation Function*. RFC 5869, May 2010.
- [6] A. Langley, M. Hamburg, and S. Turner. *Elliptic Curves for Security*. RFC 7748, January 2016.
- [7] National Institute of Standards and Technology. *FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. 2015.
- [8] A. Hosoyamada and T. Iwata. *On Tight Quantum Security of HMAC and NMAC in the Quantum Random Oracle Model*. Cryptology ePrint Archive, Report 2021/774, 2021.
- [9] E. Kret and R. Schmidt. *The PQXDH Key Agreement Protocol*. Signal Foundation specification, 2023.
- [10] F. Giacon, F. Heuer, and B. Poettering. *KEM Combiners*. Public-Key Cryptography – PKC 2018.