Fault-Induced Nonce Repetition in Threshold Lattice Signatures: A Reveal-Structure Criterion, Cross-Scheme Analysis, and Detection Boundaries

Authors: Yulong Meng **Affiliation**: Henan Normal University

Abstract: Threshold ML-DSA is regarded as a deployment model that eliminates single-point key risk, and the NIST threshold call (IR 8214C) has likewise attracted many proposals. Existing security analyses, however, generally assume malicious or compromised protocol participants. For the threat of "physical faults in the devices of honest participants," single-party lattice signatures have been studied systematically for over a decade, whereas threshold schemes, to our knowledge, have not yet been systematically evaluated. We show that fault-induced cross-session sample replay by one party leaks that party's share, and that mechanisms such as RNG state rollback can trigger it. In a t-of-n deployment, the attacker need only fault t devices to recover the complete signing secret, while every signature is fully valid when examined individually; the hedged nonce, commitment/session binding, and fail-closed final verification all remain untriggered. We verify the attack end to end on Mithril (ePrint 2026/013, USENIX Security 2026): two implementation versions, six (t,n) configurations, and 96/96 statistical scan groups all achieve exact recovery, with residual error identically 0 and no lattice reduction. The secret recovered solely from the coordinator-visible transcript can be used to forge a signature on the spot that passes the implementation's FIPS 204 verifier. We further elevate the attack to a class-level criterion. The attack surface is determined by the reveal structure, that is, by whether the scheme reveals per-party responses to a party visible to the attacker, rather than by the distributional properties of the nonce. Quorus's structural immunity thus comes from non-reveal and is unrelated to uniformity. TALUS can be broken via a double-spend path in which "one-time consumption of preprocessing is skipped". Trilithium's signing algorithm hands the Server's response share to the Client in plaintext, and [TALUS] classifies it as internally uniform ($I = 0$), making it an instance of the "statistically immune and fault-fragile" quadrant; the attack requires only 1 device fault. The simulator message "(ss, sid, [z']_S)" in the original proof, together with [Jarv26]'s verbatim copy of this algorithm, is the evidence for this determination; see Appendix D for the evidence tier. This yields two independent evaluation dimensions: the statistical axis I and the reproducibility axis (reveal structure). Different design levers allow the two axes to take values independently. All three quadrants already have instances: (reveal, $I > 0$), (non-reveal, $I = 0$), and (reveal, $I = 0$). Two cases lie on the anti-diagonal, [Trilithium] and [Kao26]. We design detection as two layers of transcript-level checks. The early-warning surface alarms on reveal-hash liveness and achieves 305/305 exact hits over the full corpus with zero false positives. The full-coverage surface runs the identification computation shared with the attack; within the feasible range of identification, there is no window in which the attack is usable but undetectable. §6.1 explains the difference in alarm timing. For imperfect faults, we prove the threshold law for full-key recovery $m^* = (\ell - 1)N + \lceil N/K \rceil$ and verify it coordinate by coordinate; for ML-DSA-44 the law gives $768 + \lceil 256/K \rceil$, namely 896 at K=2 and 787 at K=14. This reveals the trade-off nature of K. All claims have undergone a tool-assisted verification workflow with three analysis paths that are independent of one another. The authors direct the workflow; §8.2 describes its execution mode and artifact disclosure.

Keywords: post-quantum cryptography; ML-DSA; threshold signatures; fault attacks; rejection sampling; implementation security

---

1 Introduction

1.1 Background

ML-DSA, originally CRYSTALS-Dilithium, was released as the NIST standard FIPS 204 in 2024 and adopted by CNSA 2.0. Thresholdization distributes signing capability across n parties: any t parties can jointly sign, while no single party holds the full key. It is widely regarded as a necessary deployment form for scenarios such as key escrow, institutional co-management, and blockchain validators. NIST has launched the threshold-scheme call IR 8214C, and the community has proposed a series of constructions in 2025–2026, including Quorus, TALUS, Mithril, Pebble, and Trilithium. Recent additions include Shamir-nonce DKG ([Kao26]) and Mithril's TSaaS variant ML-DSaaS ([TSaaS26]).

Security analyses of these constructions, whether in the UC model or in bespoke games, generally model the threat as coming from malicious or compromised protocol participants. They ask how a corrupted participant can bias the nonce, forge messages, or infer honest parties' secrets when rejection sampling fails. By contrast, the threat of physical faults in the devices of honest participants has been studied systematically for more than a decade in single-party lattice signatures. That work covers fault-induced nonce reuse under deterministic signing, skipped rejection checks, faulty-signature correction, and verifier faults, yet to our knowledge the corresponding threat has not been systematically evaluated in threshold schemes. One direct piece of evidence is the full text of Mithril, widely regarded as one of the most practical constructions: the word "fault" appears only once in 140,916 characters, and only in the fault-tolerance phrase "resilience against faults"; "RNG" appears only in one reference. The design documentation neither analyzes this threat nor clarifies the coverage boundary of existing defenses under faults.

1.2 Contributions of This Paper

This paper presents a systematic study of the threat of "fault-induced nonce repetition" in threshold lattice signatures. The underlying attack idea is simple: if, through mechanisms such as RNG state rollback or a frozen sampling buffer, a party's round-1 samples recur across two sessions, its response satisfies $\mathbf{z}_i^{(2)} - \mathbf{z}_i^{(1)} = (c_2 - c_1) \cdot \mathbf{s}_{1,i}$. The attacker, controlling or bypassing the coordinator, can then recover that party's share directly from the transcript, and t devices suffice to recover the full key. The difficulty and the value lie in advancing this idea to publication-strength evidence and conclusions:

- **(1) Criterion**: The attack surface is determined by the reveal structure, that is, by whether the scheme reveals per-party responses to a party visible to the attacker. The distributional properties of the nonce do not determine the attack surface.
- **(2) End-to-end attack**: The attack is fully verified on Mithril: two implementation versions, six configurations, and 96/96 scan groups, with residual error identically 0 and no lattice reduction or MIP required. The recovered secret can be used to forge a signature on the spot that passes the implementation's FIPS 204 verifier. We separately verify both paths: vanilla semantics (an unmodified build) and a sampler-level fault (a single fault with zero protocol deviation).
- **(3) Class-level analysis**: The partial-signature aggregation family, comprising DOTT22, del Pino–Niot, Mithril, and Pebble, shares this attack surface, each member requiring t devices. TALUS admits a "preprocessing double-spend" attack path, against which BCC, verify-before-release, and fault attribution are all blind. Quorus's structural immunity comes from non-reveal rather than uniformity. Trilithium's signing algorithm hands the Server's response share to the Client in plaintext. [TALUS] classifies it as internally uniform ($I = 0$), making it an instance of the "statistically immune and fault-fragile" quadrant; the attack requires only 1 device fault. [Jarv26] Algorithm 3 reproduces this step verbatim, and the simulator message in the original proof corroborates it; see Appendix D for the evidence tier. This yields two independent evaluation dimensions: the statistical axis I and the reproducibility axis (reveal structure). Different design levers set the two axes independently, and existing schemes and this instance together cover three quadrants.
- **(4) Detection, two-layer transcript-level checks**: The early-warning surface monitors reveal-hash liveness and alarms on recurrence. It achieves 305/305 exact hits over the full corpus with zero false positives, and all controls are clean. The full-coverage surface runs an identifiability check that shares the identification computation with the attack. Within the feasible range of identification, there is no window in which the attack is usable but undetectable. One design lesson follows: input-freshness hardening (counters or entropy checks) is not equivalent to protection.
- **(5) Fault-resolution law**: When faults are imperfect and only some sampled coordinates recur,

we prove the full-key recovery threshold $m^* = (\ell - 1)N + \lceil N/K \rceil$ and verify it coordinate by coordinate. For ML-DSA-44 the law gives $768 + \lceil 256/K \rceil$, namely 896 at K=2 (87.5%) and 787 at K=14. Solving requires only multi-slot joint linear algebra and takes under 0.2 seconds at K=2. This reveals the trade-off nature of K: it amplifies the leakage surface by a factor of K while providing the redundancy needed for solving.

1.3 Verification Rigor

All key claims in this paper have been tested and revised in light of the test results. First, the recovery results are independent of the ground-truth files at the function level, as the "truth-destruction" experiment demonstrates. Second, the recovered secret is self-consistent with the public key, and the consistency check relies on no self-generated ground truth. Third, the forged signature passes the implementation's built-in standard verifier and is reproduced by an independent, from-scratch FIPS 204 verifier. Fourth, three verification workflows with independent analysis paths were all executed in a tool-assisted manner: from-scratch recomputation, upstream comparison, and byte-for-byte reproduction. All reproduce the archived experiments byte for byte. The problems they exposed, including fault-class wording, path classification, and detection row-selection rules, were corrected and written back (§8.2). Known boundaries are detailed in §8. Faults are simulated injections, with the methodology and precedents in §3.3. The recovery target is the synthesized threshold secret used by the implementation, and the vanilla path is classified as a hybrid "fault + malicious coordinator" model.

1.4 Organization

§2 reviews background and related work. §3 defines the threat model and fault classes, including the realism argument. §4 presents the complete attack on Mithril and the experiments. §5 establishes the reveal-structure criterion and the cross-scheme analysis. §6 sets out detection and its precise boundary; §7 establishes the fault-resolution law. §8 states verification rigor and limitations, §9 discusses design and deployment implications, and §10 concludes.

---

2 Background and Related Work

2.1 ML-DSA and Rejection Sampling

ML-DSA is defined over the ring $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$ ($q = 2^{23} - 2^{13} + 1$). The coefficients of the private key $(\mathbf{s}_1, \mathbf{s}_2)$ are bounded by η, and the public key is $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. Signing uses Fiat–Shamir with Aborts. The signer samples a mask $\mathbf{y}$, computes $\mathbf{w} = \mathbf{A}\mathbf{y}$ and the challenge $c = H(\mu \| \text{HighBits}(\mathbf{w}))$, and forms the response $z = y + c\mathbf{s}_1$. The signer then performs three checks: (1) $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$; (2) $\text{HighBits}(\mathbf{w} - c\mathbf{s}_2)$ is consistent with the derivation from c, and $\mathbf{r}_0$ is within bounds; (3) the hint can be recomputed. If any check fails, the signer rejects the candidate and resamples. Rejected candidate signatures carry information about the private key; this is the security role of the rejection-sampling mechanism.

FIPS 204 uses **hedged signing** by default; its design intent is directly relevant to this paper:

> "The use of fresh randomness during signing helps mitigate side-channel attacks, while the use of precomputed randomness protects against the possibility that there may be flaws in the random number generator used by the signer at signing time." — FIPS 204 §3.4 [primary quotation, verified against the local original]

The standard thus lists "the RNG may be flawed at signing time" as an explicit design threat. This paper examines the coverage boundary of this protection in thresholdized implementations.

2.2 Landscape of Threshold ML-DSA

Scheme	Online rounds	Party scale	Assumption	Nonce structure	Per-party response reveal
Quorus (ePrint 2025/1163)	16–29 (concrete broadcast)	Large	Honest majority	Assembled within MPC, internally uniform ($I = 0$)	**Non-reveal** (only w_1 and the aggregate signature; UF-CMA$\perp$)
TALUS (arXiv 2603.22109)	TEE 1 / MPC 2	Any N	MLWE+SelfTargetMSIS	Pooled Shamir + boundary clearance (BCC)	**Revealed to the coordinator** ($\mathbf{z}_h = \hat{\mathbf{y}}_h + c\mathbf{s}_{1,h}$)
Mithril (ePrint 2026/013)	3	N$\leq$ 6	Standard	Partial-signature aggregation (hypersphere local rejection)	**Revealed to the coordinator**
Pebble (Silence Labs)	—	—	Standard	Discrete Gaussian partial-signature aggregation	Reveal (combiner; described in the original text as partial-signature aggregation)
Trilithium (ePrint 2025/675)	14	2	UC + trusted CRP	Additive shares (BeDOZa)	**Releases per-party responses**: in the signing algorithm, S sends $[\![z]\!]_S$ to C. C receives the plaintext share already at Step 7, and assembly and verification follow. See Appendix D for the evidence tier.
Kao26 (arXiv 2601.20917)	P1 3 rounds / P2 5 rounds / P3+ 2 rounds	Any N	Standard (P1 relies on an ideal TEE)	Shamir nonce DKG + PRF masking; statistical protection (conditional min-entropy, no computational assumptions)	The combiner receives per-party $\mathbf{z}_i$ (broadcast to the signing group under P2). Original text: "partial responses ... are visible to the coordinator or broadcast to the signing group"

The above facts have all been checked against the local original texts; TALUS, Pebble, Mithril, Kao26, and Trilithium each have a corresponding original, and the ML-DSA scheme in the SplitForge submission is Trilithium. Itemized sources appear in Appendix D. ML-DSaaS ([TSaaS26], ePrint 2026/814), the TSaaS-optimized variant of Mithril, centralizes communication between the client and the parties through a Relayer. Under the "coordinator-visible" model it inherits Mithril's reveal structure; that determination still needs to be checked against its main text and is listed as a tracking item (Appendix D). P1, P2, and P3+ of [Kao26] denote the TEE, broadcast MPC, and 2PC deployment tiers, respectively.

2.3 A Genealogy of Fault Attacks on Lattice Signatures

- **Fault-induced nonce reuse under deterministic signatures** (Bruinderink–Pessl; see Krahmer et al. 2024 for a survey citation): "a fault-induced nonce-reuse scenario is achieved, allowing trivial key recovery using a single faulty signature" [primary quotation]. The chain "fault $\rightarrow$ nonce reuse $\rightarrow$ key recovery" is an attack paradigm already accepted by the community; Krahmer et al. further note that the design goal of the hedged mode is "robustness against nonce-reuse attacks." This paper studies how this paradigm generalizes to the **thresholdized** setting and where its protection

boundary lies.

- Skipping rejection checks (Polytope, PKC 2025): physically blocking a single rejection test suffices to exploit all rejected signatures and recover the key with an LP. This establishes that "the execution of rejection-sampling checks must itself be protected."
- **Correction attacks and hardening** (TCHES 2024; ePrint 2026/1448): use of the public verifier as a correction oracle, and MILP-based hardening.
- **Verifier faults** (ePrint 2025/2029): a faulty signature-verification routine can accept forged signatures.
- **Nonce-redundancy class** (*An attack on ML-DSA using an implicit hint*, Azevedo-Oliveira–Beraud–Goubin, SAC 2025; same author group as Polytope): implicit or explicit redundancy at the level of nonce coefficients (including the side-channel perspective that "each signature leaks only one coefficient") is exploited to recover the key by intersecting non-full-rank lattices (on the order of 4×160 signatures). Countermeasures of the "transform the generation order" type are insufficient, and the authors recommend masking the entire sampling algorithm. In threshold variants, the non-uniformity of conjugate-sum nonces yields a passive statistical lower bound (TALUS Theorem 3, $\sim 2^{30}$ signatures).
- **Threshold-protocol-level attacks**: pooled-nonce reuse in TALUS v4 (Jo, ePrint 2026/1668) and Niot's lattice-based analysis (ePrint 2026/1386; after disclosure TALUS withdrew some claims, and a revised version is in preparation). The fault path in this paper is model-orthogonal to both (§2.4), but the TALUS determination in §5.2 must be re-checked against its revised version.
- Fault surfaces of new schemes: HAETAE challenge sampling (2026/1344), among others.
- **Methodology**: a considerable portion of the above work is validated by **simulated injection** (Bauer & De Santis simulate on a Cortex-M4 and localize the fault to an instruction; Kosuge & Xagawa analyze the fault class). Simulated injection is the standard methodology in this field.
- **Defenses**: several defense proposals against rejection-sampling faults exist in the single-party setting (e.g., the "third rejection test" of Thales patent WO2025261784A1, and the probabilistic protection framework of Mishra–Mukhopadhyay), all sharing **redundant checking**. Any coverage evaluation of such defenses must compare a defense's stated goal with its actual coverage. The detection surface in §6 of this paper supplies the complement: it checks the liveness of the reveal rather than recomputation. ETSI TR 104 239-3 (V1.1.1, 2026-09), an implementation-guidance document, explicitly includes ML-DSA-specific side-channel and fault issues and cites [Bauer+25]. At the standards level, then, faults are already treated as a first-class threat to implementation security. Yet the public literature has not yet evaluated the fault liveness of the reveal channel under threshold deployments.

2.4 Positioning

Compared with the above work, the contribution of this paper is not yet another single-party attack. First, we advance this paradigm to threshold protocols, where the reveal structure determines the attack surface. Second, we provide a class-level criterion and a cross-scheme evaluation. Third, to our knowledge, we provide the only mechanism-independent detection surface and its precise boundary. Fourth, we give a resolution law for imperfect faults. Unlike the protocol-level or lattice-based attacks on TALUS ([Jo26], [Niot26]), which target malicious participants or the commitment structure, the vanilla path in this paper and the TALUS double-spend path both draw on responses already leaked in the transcript and are independent of whether the session ultimately succeeds.

---

3 Threat Model and Fault Classes

3.1 Attacker Model and Deployment Positioning

Main model. The attacker (1) controls or bypasses the **coordinator** and thereby obtains all protocol messages visible to it, specifically the round-2 reveals and round-3 per-party responses of threshold ML-DSA; (2) can subject **one party's device** in the signing protocol to **repeatable physical faults** such as clock or voltage glitching (verified here by simulated injection; see §3.3); and (3) does not deviate from the protocol or act as a malicious participant.

Deployment positioning. Hardware-isolated participant devices (HSM, TEE, or a dedicated secure

chip) combined with an independently compromised coordinator. Natural scenarios for this combination include data-center and hosted threshold deployments, where all parties are servers or hosted devices. One caveat deserves mention: if the deployment instead takes the form of "a hoster controlling the host virtual machine", the counterargument of "reading memory directly" stands. The target form of this attack is one in which the fault point lies in the isolated device while the visibility point for per-party responses lies at the coordinator. We state this modeling assumption explicitly and list it among the limitations in §8.

The vanilla variant (§4.3, path B) allows the coordinator to replay the session_id. This contradicts the Mithril specification's "use a fresh session id for each attempt", so the variant is in fact a "**fault + malicious coordinator**" hybrid model. Paths A and D, the main results of this paper, rely on no protocol deviation.

Attacker composition and positioning of the main results (summary).

Path	Attacker composition	Protocol deviation	Positioning in this paper
A (binding skip)	fault injector + coordinator observer	none	Main result: all end-to-end experiments in §4 (96/96 scan) and the class-level, detection, and resolution conclusions of §5–§7 take A or D as the reference
D (sampler level)	fault injector + coordinator observer	none	One of the main results: minimal fault budget (single effect, §4.3)
B (vanilla)	fault injector + coordinator replayer	yes (replay sid)	Control and extension: verifies "zero deviation is not necessary" and the fault-budget trade-off

Positioning in one sentence. Paths A and D are the main results: all end-to-end experiments in §4 and the class-level, detection, and resolution conclusions of §5–§7 depend only on them. Path B serves only to show that "zero protocol deviation is not a necessary condition for the attack to hold"; it is a robustness control, and no conclusion of this paper depends on it.

3.2 Fault Effect and Mechanism

The attack relies on a single **effect**:

> E: the round-1 sample $(\mathbf{r}_i, \mathbf{e}_i)$ of some party is replayed across two sessions, while the remaining parties are fresh.

Any physical mechanism that produces E is admissible, because the mechanism layer and the effect layer are decoupled:

Mechanism	Description	Simulation in this paper
i. Entropy state rollback	the device's PRNG or entropy-source state is rolled back or stuck, so the entropy consumed in round 1 recurs	`rollback_to` (RNG state restore)
ii. Sampler-level replay	the sampling output buffer or state is frozen or reused, independently of the input	a sampling hook returns a stale buffer (direct connection)
iii. Constant output	the entropy source always returns the same byte stream; this is coarser and requires orchestration adaptation (see Appendix B)	control experiment in the adversarial audit

Fault budget (from minimal to strongest):

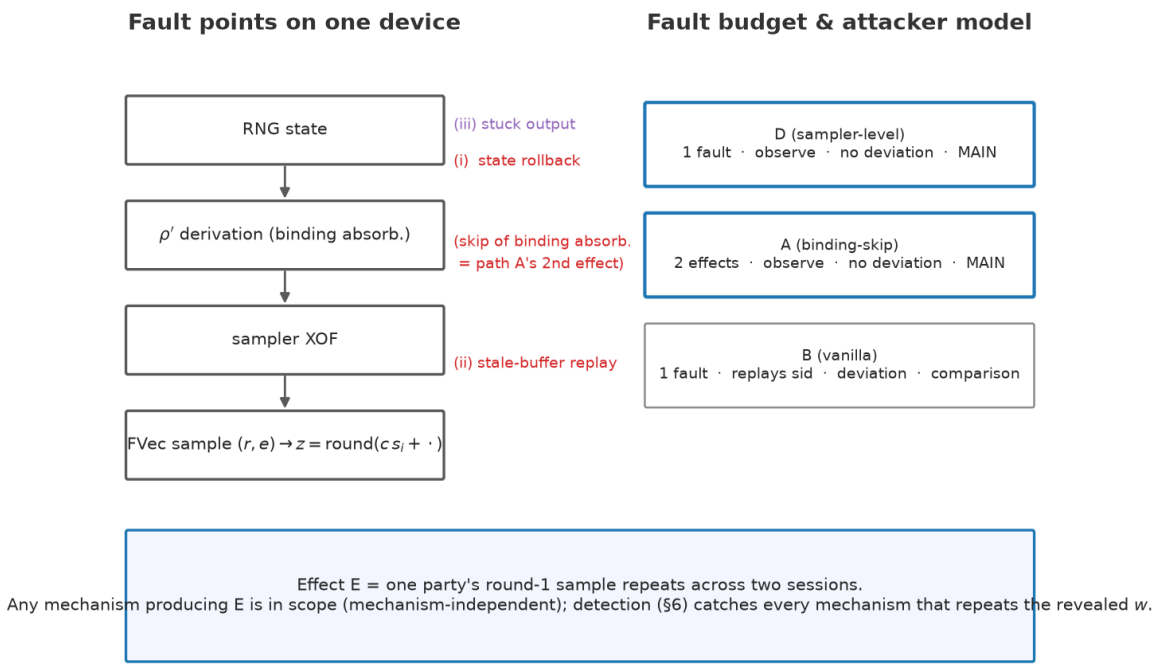

Figure 2: Left: single-device fault point (pipeline from top to bottom); right: fault budgets and attacker models of the three attack paths (MAIN = main result).

Path	Device-side fault	Coordinator side	Protocol deviation
D (sampler level)	1 effect	observation only	none
B (vanilla)	1 (entropy rollback)	replay session_id	yes (fresh-sid rule)
A (binding skip)	2 effects (entropy rollback plus a single-point skip of one absorption in the derivation chain)	observation only	none

Path A's "two effects in exchange for zero protocol deviation" is the trade-off that keeps the paper's main results clean; control experiments confirm that both conditions are indispensable, since rollback alone and replay alone both fail.

3.3 Simulation Methodology and Realism Argument (summary; full version in Appendix A)

Simulated injection is the standard methodology in this field (§2.3). At the effect layer the attack is robust: its algebra depends only on E and is insensitive to the mechanism's underlying implementation. Four realism anchors support this choice. (1) The first-hand quotation from FIPS 204 §3.4 (§2.1) states that RNG failure is a design threat explicitly written into the standard. (2) Krahmer et al. give a first-hand formulation of "fault-induced nonce reuse" (§2.3). (3) Physical faults of RNG/SHAKE have been demonstrated by actual attacks: at ACISP 2025, two faulty signatures recover the key, and the reported figure for a masked implementation is 48.9%, according to the publisher's official abstract (full text by subscription; Appendix D). (4) The assumption framework of "the same fault in ≥ 2 executions" has been adopted by ePrint 2025/606 and others. **An honest boundary.** This paper does not include real glitching experiments (§8).

4 End-to-End Attack on Mithril

4.1 Protocol Facts and Visibility

Mithril is a 3-round protocol with K parallel slots per round. Round 1 is the commitment, and the commitment hash is sent to the coordinator; round 2 reveals the full $\mathbf{w}$, again to the coordinator; round 3 sends the per-party response $\{\mathbf{z}_{i,k}\}$ **to the coordinator**. The per-party response has the form

$$\mathbf{z}_i = \mathrm{round}\big(\underbrace{c \cdot \mathbf{s}_{I,i}}_{\text{integer}} + \underbrace{\text{FVec sampling}}_{\text{floating point}}\big),$$

where $\mathbf{s}_{I,i}$ is the effective secret of party i, namely the sum of its shares, and sampling means hyperball sampling followed by rounding. The coordinator aggregates the responses and performs final verification in a fail-closed manner. The visibility facts, verified against the original text, are as follows: the per-party responses and the $\mathbf{w}$ reveals are **sent only to the coordinator**, and the $\|\mathbf{z}\|_\infty$ check is performed only on the aggregated value.

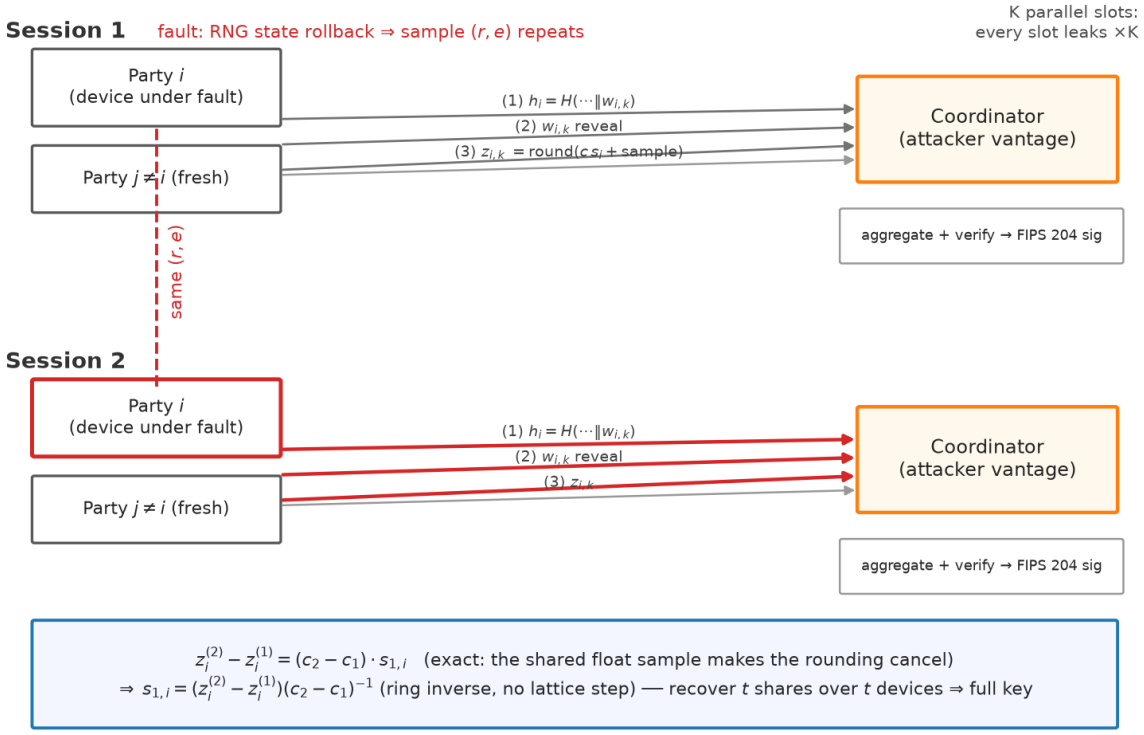

Figure 1: Two-session flow of the threshold signature and the attack path (coordinator's view; red = faulty session; dashed = same sample).

4.2 Attack Derivation

Suppose the sample of some party is replayed across two sessions with challenges $c_1 \neq c_2$. Then

$$\mathbf{z}_i^{(2)} - \mathbf{z}_i^{(1)} = (c_2 - c_1) \cdot \mathbf{s}_{1,I,i} \;+\; \big[\mathrm{round}(\cdot)^{(2)} - \mathrm{round}(\cdot)^{(1)}\big].$$

Key observation. The bracketed term is identically zero. The integer part $(c \cdot \mathbf{s})$ of the response differs across sessions, but rounding acts only on the sum of "the integer part + the reused floating-point sample". Because the reused sample makes the floating-point part bit-for-bit identical across sessions, the rounding error cancels exactly. In our measurements, the residual error over all 4,096+ coefficients is identically 0. Therefore

$$\mathbf{s}_{1,I,i} = (\mathbf{z}_i^{(2)} - \mathbf{z}_i^{(1)}) \cdot (c_2 - c_1)^{-1} \pmod q,$$

and the secret is obtained in a single step of ring inversion, with no lattice reduction and no error handling. The probability that the challenge difference is non-invertible is negligible: 0 times in 10,000 measured random samples, against a theoretical expectation of $256/q \approx 0.3$ times per ten thousand (§7.1). Summing the shares of t parties yields the complete signing secret. **A falsifiable point.** If an implementation's rounding carries an input-dependent bias, the residual will be non-zero; the implementation in this paper (integer part plus reused floating point) is the mechanism by which the residual is zero.

4.3 Experiment Matrix and Results

Implementation. We use two versions of the official Mithril Rust implementation: v0.3.1 (effect-level) and v0.3.4 (direct connection). The patch list is public and affects only visibility and feature gating; it makes no change to protocol logic, and the default build is identical to the upstream build artifacts. **Fault simulation.** All three paths (table in §3.2) use deterministic seeds and are reproducible byte for byte, as verified by archive fingerprints.

Experiment	Configuration	Result
Path A (binding skip + entropy rollback)	2-of-2 / 2-of-3 / 3-of-5	all (party, slot) pairs exact, 1,024/1,024; for 3-of-5, all 42/42 pairs
Statistical scan	6 configurations × 2 modes × seeds × messages × active sets (including non-prefix sets)	**96/96 success**; all 378 sessions succeeded on the **first attempt**
Path B (vanilla: rollback + replay)	2-of-2, 3-of-5 (non-prefix active set)	1,024/1,024; byte-for-byte stable in regression
Path D (sampler level)	2-of-2, 3-of-5	1,024/1,024 (all entropy and session_id values fresh; no rollback, replay, or skip)
Control: rollback only / replay only	2-of-2	0/1,024 (residual ~3.4–3.7×10^4); both conditions are indispensable
Forced failure	2-of-2	recovery from the **failed-attempt** row, 1,024/1,024 (old row-selection behavior, 0/1,024)

Note. The main results do not use the "constant output" semantics (mechanism iii): its retry semantics and multi-device orchestration require adaptation (Appendix B). The fault semantics behind the main results is "**RNG state rollback (snapshot restore)**".

4.4 End-to-End Closure: Public-Key Consistency and Forgery

The recovered secret can be verified without relying on any ground-truth file, in two ways. (1) **Public-key consistency.** From the public key one can compute $\mathbf{s}_2 - \mathbf{t}_0 = \mathbf{t}_1 2^d - \mathbf{A}\mathbf{s}_1$ (coefficient bound $\leq$ 4,098); the measured maximum is 4,094 for 2-of-2 and 4,098 for 3-of-5. (2) **Forgery.** A FIPS 204 signature can be constructed on the spot from the recovered secret alone, with hints built coefficient by coefficient. It succeeds on the 2nd attempt ($|h|_1 = 50/67 \leq \omega$), is accepted by the standard verifier bundled with the implementation, and is independently reproduced by a **from-scratch independent verifier**. Acceptance holds for verifiers that share the same A derivation convention.

4.5 Cost and Amplification Factor

Scenario	Number of devices	Number of sessions	Recovery target
Single-party baseline (ePrint 2025/606)	1	2	full key
This attack (2-of-2)	2	3	full key
Generalization (t-of-n)	t	t+1	full key

Threshold amplification factor = number of parties t (the device dimension). Thresholdization does upgrade a single-point fault to a t-device fault, but it does not eliminate the harm of a single-point fault. A single-device fault leaks one share, which is uninformative on its own; this is precisely

the quantifiable "threshold benefit". Moreover, the attack does not require the signature to be valid: leakage has already occurred once the round-3 message reaches the coordinator, and failed sessions and non-adopted slots apply equally, as the forced-failure experiment in §4.3 demonstrates empirically.

---

5 Reveal-Structure Criterion and Cross-Scheme Analysis

5.1 Criterion

Two necessary conditions emerge from the attack structure of §4, and they are independent of the distributional properties of the nonce:

- **(A) Reveal.** The scheme provides a channel through which some party can obtain the **per-party response $\mathbf{z}_i$**, or a decomposable equivalent.
- **(B) Visible.** This channel is visible to the attacker, who controls or bypasses the coordinator, and the fault injector can pin down one party's sampling.

The intuition is that the leakage equation $\Delta\mathbf{z}_i = \Delta c \cdot \mathbf{s}_{1,i}$ is **per-party**. In schemes that release only the aggregate result, a single party's sampling replay aliases with the fresh samples of the remaining parties, $\Delta\mathbf{z} = \Delta c \cdot \mathbf{s}_1 + \Delta(\sum \mathbf{y})$, and the residual cannot be controlled. No exploitable equation results.

Proposition (criterion within this attack family; transcription-algebra scope). Given effect E, (A) is **both necessary and sufficient** for the differential-equation attack under consideration. For sufficiency, if some party can obtain $\mathbf{z}_i$ and effect E holds, the derivation of §4.2 yields $\mathbf{s}_{1,i} = (\Delta\mathbf{z}_i)(\Delta c)^{-1}$. For necessity, if the per-party response is unavailable and only the aggregate value is present, the difference between the two sessions is $\Delta\mathbf{z} = \Delta c \cdot \mathbf{s}_{1,i} + \Delta(\sum_j \mathbf{y}_j)$; the aliasing term contains the fresh samples of the remaining parties and is inseparable on the difference of a single pair of sessions, while multi-session statistical exploitation lies outside the single-difference algebraic scope of this proposition. **Boundary.** The proposition is limited to algebraic separability, excluding measure-zero events such as "the aggregate and the fresh samples cancel exactly". Side channels, discussed in the boundary statement of §5.2, are not covered. The proposition characterizes this attack family; it does not claim to exhaust all fault attacks.

5.2 Cross-Scheme Determination

Boundary statement. The following determinations are limited to the transcription-algebra scope, that is, to the equations an attacker can extract after accessing protocol messages. Quorus's MPC internal messages do not expose per-party responses to any party, but side channels such as communication timing, error or abort messages, and timeout behavior belong to an independent side-channel threat domain. If any such channel leaks decomposable information about response shares, condition (A) holds again, and the attack and its detection framework apply accordingly. This direction requires a concrete MPC implementation and channel assumptions, which we have not evaluated; we list it as future work.

Likewise, the TALUS determination in the table targets its published online signing algorithm. If a revised version changes the visibility of per-party responses, the determination updates accordingly; all determinations in the table are based on the published version.

In addition, we screened the 60 preview manuscripts of the NIST IR 8214C call (TCall-1) and checked them by name and category. The four threshold ML-DSA submissions in that call are Mithril, Quorus, TALUS, and SplitForge. The first three each have a table row. SplitForge is the submission package from Cybernetica, and its ML-DSA scheme is Trilithium, which the Trilithium row covers. Trilithium, Kao26, and Pebble with its family [DOTT22]/[dPN25] are assessed from public literature outside the list, with the source tier noted row by row. The remaining lattice-based signature submissions in the list belong to category S1, that is, non-NIST-designated primitives, and lie outside the scope of this determination.

Scheme	(A) Reveal	(B) Visible (observer)	Conclusion and fault point
Mithril	✓ ($\mathbf{z}_i$ sent to coordinator)	✓ (coordinator)	Attackable; round-1 sampling replay; t devices (§4, empirical)

Scheme	(A) Reveal	(B) Visible (observer)	Conclusion and fault point
Pebble / [DOTT22] / [dPN25] (the partial-signature structure of DOTT22/dPN25 is paraphrased from §1–§3 of the original Pebble text; the bibliographic data and abstracts of [DOTT22]/[dPN25] have been verified against ePrint 2020/1110 and 2025/872; tiers are given in Appendix D)	✓ (partial-signature aggregation)	✓ (combiner)	Isomorphic algebra ($\mathbf{z}_i = c\mathbf{s}_i + \mathbf{r}_i$), t devices
TALUS	✓ ($\mathbf{z}_h = \hat{\mathbf{y}}_h + c\mathbf{s}_{1,h}$ sent to coordinator, verified against the original text)	✓ (coordinator)	**Relocated fault point**: the pooled-nonce structure absorbs single-point sampling replay, because any evaluation depends on all contributions. Preprocessing, however, is a **one-time-consumable object**: "consumption/deletion being skipped" (double-spend) makes the same evaluation recur across sessions $\Rightarrow \Delta\mathbf{z}_h = \Delta c \cdot \mathbf{s}_{1,h}$; t devices. BCC (statistical properties), verify-before-release (a repeated evaluation is still a legitimate evaluation), and fault attribution (oriented toward malicious behavior) are all blind. Moreover, $\mathbf{z}_h$ has already been transmitted, independent of session success or failure.
Quorus	× (releases only w_1 and the aggregate signature; the UF-CMA$\perp$ proof shows that incomplete signatures leak nothing)	—	**Structurally immune** (under this model): no per-party equations can be collected

Scheme	(A) Reveal	(B) Visible (observer)	Conclusion and fault point
Trilithium	✓ (signing algorithm, steps 7–10: S→C sends "$[\![z]\!]_S$" in plaintext (step 7); C assembles (step 8); the Verify check runs (step 9; if it fails, C aborts without returning); only after passing is it returned to S (step 10). The plaintext share reaches C before signature verification, so the attack holds whether verification passes or aborts. The simulator's expected message "(ss, sid, [z']_S)" in the original security proof corroborates this; [Jarv26] Algorithm 3 copies it verbatim. The declassification subprotocol also first sends the counterparty's share to C in plaintext. See Appendix D for tiers.)	✓ (to the Client/Phone)	**Occupies a new quadrant**: $I = 0$ ([TALUS] groups it as internally uniform) + per-party reveal ⇒ a fault in **1 device** recovers the counterparty's share (observer = the other participant within the protocol; prerequisites: a malicious/controlled Client + one sampling-replay fault on the Server device; see §8.3, item 7, for the physical-access assumption); see §5.3 for details
[Kao26]	✓ (the per-party response $\mathbf{z}_i = \mathbf{y}_i + c\,\mathbf{s}_{1,i}$ is visible to the combiner; Remark 19 of the original text: "or broadcast to the signing group" (all parties under P2))	✓ (combiner; all parties under P2)	**Second instance (same cell as Trilithium)**: protection in the statistical direction (conditional min-entropy, no computational assumption) + per-party reveal. The fault point moves earlier, to its offline nonce DKG (sampling replay ⇒ the same $\mathbf{y}_i$ across sessions ⇒ $\Delta\mathbf{z}_i = \Delta c \cdot \mathbf{s}_{1,i}$); details are listed as a tracking item (Appendix D).

Note that Quorus's immunity comes from non-reveal, not from its "internally uniform nonce". Uniformity characterizes the distribution and defines the statistical axis, while the repetition attack exploits reproducibility and defines the reproducibility axis.

5.3 Two Independent Evaluation Dimensions and Quadrant Occupancy (Core Insight)

Axis	Measure	Threat defended against
Statistical axis	Fisher information I of the nonce marginal	Passive statistical recovery (the lower bound of TALUS Theorem 3)
Reproducibility axis (this paper)	**Reveal structure**	**Faulty nonce repetition**

For comparison, implementation-side side-channel and leakage threats, such as the [606] perspective that "each signature leaks only one coefficient", exploit execution-side observables. They belong to the implementation layer and are orthogonal to the two protocol-layer axes of this paper; their countermeasures, masking and hiding, are mutually independent of the reveal-structure checks.

Existing literature groups schemes only along the statistical axis; the TALUS trilemma, for instance, classifies Quorus as $I = 0$, "escaping the statistical lower bound". This grouping is easily misread as "uniformity provides protection". In fact, the two axes are controlled by different design levers, as discussed below, and among the scheme family we survey (§2.2), every scheme except Trilithium and Kao26 has its (reveal, I) values lying exactly along the diagonal. The literature's convention of grouping by the statistical axis thus makes two explanations, "uniformity provides protection" and "non-reveal provides protection", appear indistinguishable. The task of this paper is precisely to decouple the two axes explicitly into **two independent evaluation dimensions**.

Concrete example. The new quadrant is already occupied, not predicted. Once two easily misread points are clarified, the conclusion is that the current scheme family already covers three quadrants of the two axes:

- **Pebble is not in the new quadrant.** Its partial nonces are discrete Gaussian, and the aggregate remains DG rather than uniform modulo q; its original text states "masked over the integers rather than uniform modulo q". Pebble therefore carries $I > 0$ along the statistical axis. Together with TALUS and Mithril, it sits in the "**fragile on both axes**" cell (reveal + non-uniform).
- To our knowledge, Trilithium is the first anti-diagonal instance. The literature, including the grouping of TALUS Theorem 3, classifies it as $I = 0$, an internally uniform nonce, "immune" on the statistical axis. However, its signing algorithm itself hands the Server's response share to the Client in plaintext. Steps 7–10 proceed in order: S→C: $[\![z]\!]_S$; C: $z \leftarrow [\![z]\!]_S + [\![z]\!]_C$; stop if ¬Verify(…); C→S: z. Here "stop" means that C aborts this signing and does not return z, and S observes only the session failure. The scope of the abort does not affect the attack's validity, only its stealth. C obtains the plaintext share already at step 7; assembly and verification both come afterward, and the return comes only after verification passes. Therefore, whether verification passes or the session aborts, the share is already in hand, and the attack does not depend on the verification result, consistent with "no valid signature is needed". The simulator's expected message "(ss, sid, [z']_S)" in the original security proof corroborates this. The thesis [Jarv26] from the same group copies the algorithm verbatim and states explicitly in §3.1.2 that "the client gets to know the server's share", with $[\![z]\!] = [\![y]\!] + c\,[\![s_1]\!]$. Per-party reveal to the Phone therefore holds, and the differential equation of this paper applies directly. An attacker who controls the Phone and applies one sampling-replay fault to the Server device can recover the Server share from $\Delta z_S = \Delta c \cdot s_{1,S}$, and thereby obtain the full 2-of-2 key with a fault in only 1 device. Here the observer is precisely a legitimate role within the protocol. The statistical-axis "immunity" does not cover this threat: it targets the passive statistical lower bound, which requires about 2^{30} signatures, and is unrelated to reproducibility. Compared with Mithril and TALUS, the attack has a further structural advantage: it does not require the attacker to control the coordinator. The malicious Client is itself a legitimate protocol participant, the observation point is endogenous to the protocol, and the attacker need only control one participant and apply one sampling-replay fault to the other party's device. The evidence tier has been verified against the original full text; see Appendix D for details. We checked the full original text (Springer LNCS edition, 67 pages) on the local machine. Steps 7–10 of Algorithm 14 (`sign_trilithium`) read verbatim "S→P: $[\![z]\!]_S$", "P: $z \leftarrow [\![z]\!]_S + [\![z]\!]_P$", "P: stop if ¬Verify((A,t),μ,(z,c))", "P→S: z", matching line by line the verbatim copy in [Jarv26] (its Algorithm 3, where P is C). The simulator message "(ss, sid, $[\![z']\!]_S$)" in the proof section agrees with the above. The original text of the declassification protocol (Algorithm 5) states "asymmetric: P learns the value v first"; [TALUS] independently groups it as $I = 0$. The main text states directly: "if the declassified result of rejection check is 'OK', then we first declassify z only to the Phone. The Phone verifies the signature (z, c) and only on success sends it to the Server". Tracking item (2) is thereby closed, the line-by-line comparison is complete, and the network restriction on the original ePrint no longer constitutes an evidence gap.
- A second instance, [Kao26], occupies the same cell as Trilithium. This construction comes from an independent team (arXiv 2601.20917, v6, 98 pages) and likewise lies on the anti-diagonal; unlike Trilithium's two-party BeDOZa, however, it targets arbitrary N and generates the signing nonce via Shamir sharing. The combiner collects per-party responses $\mathbf{z}_i = \mathbf{y}_i + c\,\mathbf{s}_{1,i}$ in the P1/P3+ tiers, and under the P2 broadcast mode the responses are exposed directly to the signing group. Remark 19 of the original text states "partial responses … are visible to the coordinator or broadcast to the signing group". Its protection likewise falls in the statistical direction, self-described as "conditional min-entropy, no computational assumption". Two independent teams and two different constructions, namely two-party BeDOZa and arbitrary-N Shamir-DKG, fall into the same cell.

This is further existence evidence that the two axes can take values independently through different design levers. Under the P2 broadcast mode, the observation point is likewise endogenous to the protocol: any signing party sees the per-party responses, and no coordinator control is needed. Its fault path requires the sampling replay to occur in the presignature set of the offline nonce DKG, and a complete derivation must also check the consumption mechanism: if the "consumed" flag is skipped, the construction degenerates into a two-fault combination isomorphic to Mithril. Details await review against its formal version, are listed as a tracking item, and appear in Appendix D.

- With **Quorus** added ($I = 0$, non-reveal), all three quadrants of the two axes have instances among existing schemes. This is not merely a grouping convention but existence evidence for the two axes as independent design dimensions: I is determined by the sampling and assembly algorithm, and the reveal structure by the communication topology and opening semantics. The two can take values independently. Design review should list "reveal structure" and "distribution parameters" as two independent check items.

The basis for this determination is as follows. Step 7 of the signing algorithm sends $[\![z]\!]_S$ from S to C, a reading corroborated by the simulator message "(ss, sid, [z']_S)" in the original proof, and [Jarv26] Algorithm 3 copies this step verbatim. The opening semantics of the declassification subprotocol are a plaintext share plus MAC verification (§3.1.2). The authors' public lecture notes (Laud, SantaCrypt 2025 and NIST MPTS 2026) serve as auxiliary material. The line-by-line comparison against the original has been completed (Appendix D): steps 7–10 of Algorithm 14, Algorithm 5, and the simulator messages in the proof section of the full original text (Springer LNCS edition, 67 pages) were all checked verbatim, and the copy in [Jarv26] matches line by line. The threat-model tension, namely that the malicious observer needs physical access to the other party's device, is consistent with the deployment positioning of §3.1 and is faithfully listed in §8.3.

5.4 Device Budget

Revealing schemes, whether based on partial-signature aggregation, MPC, or a combiner architecture, each require t devices; Mithril, TALUS, Pebble, Trilithium, and [Kao26] all belong to this class. Non-revealing schemes such as Quorus expose no per-party equations and are unreachable under this model. Whether reveal occurs is thus determined not by the implementation family but by the communication topology and opening semantics (§5.3). The single-party baseline uses 1 device, so the gain from thresholdization can be quantified as $\times$t devices.

---

6 Detection and Its Precise Boundary

6.1 Principle: Two-Layer Transcript-Level Checks (Early-Warning Layer and Shared Identification Computation)

The attacker needs "the same party's sampling to be used twice"; the detector needs only "**the same party's secret to be observed twice**". We present two mutually complementary transcript-level checks. The auditor needs neither ground truth nor trusted execution and observes only protocol messages:

- **Check A (reveal-hash liveness)**: the hash of the same party's round-2 reveal data recurs across sessions $\Rightarrow$ alarm. The check has zero cost (32 bytes per party per session) and is mechanism-agnostic, covering **full-buffer replay**, in which the reveal $\mathbf{w} = \mathbf{Ar} + \mathbf{e}$ is repeated byte for byte, including mechanism iii.
- **Check B (identifiability check)**: for paired sessions, the auditor computes the implied secret slot by slot, $\hat{\mathbf{s}}_k = \Delta\mathbf{z}_k \cdot \Delta\mathbf{c}_k^{-1}$, without knowing the key, and then executes the attacker's identification procedure. (1) It counts identical coordinates: if all 256 coordinates of a polynomial agree across all slots, the replay is complete. Under randomness the per-coordinate agreement probability is $\sim 1/q$, and with a threshold of ≥ 4 the false-positive rate is $< 10^{-20}$. (2) It performs **multi-slot joint identification**, running the same solver as the attack in this paper; a box-constrained integer solution with exactly zero residual for every equation is the **certificate** of a replay. Check B therefore covers partial replay, including the "r-only replay" case invisible to Check A.

  Shared identification computation (the core of this section; a descriptive statement, not an independent theorem): Check B runs exactly the attacker's identification procedure, so

its coverage coincides with the attack's recoverable range, with the same equation count $k_p \leq 256(K-1)/K$. Their alarm timings, however, must be distinguished. Check A alarms at the first replayed reveal, earlier than the equations take shape; Check B alarms only once identification has become feasible, at the same moment as the attack becomes solvable, not before. The two layers are thus complementary: A provides the **early-warning layer**, and B provides the **full-coverage layer**, including the "r-only replay" case invisible to Check A.

6.2 Coverage and False Positives (Measured)

Group	Check A (reveal hash)	Check B (implied secret)
Full-replay directories (vanilla/0.3.1/forced failure/mechanism ii direct/m1024)	All hit	Identical coordinates **1024/1024**
partial-m512 / partial-m256	No hit (the reveal has changed)	Identical **512 / 256**
partial-m128 (only poly0 half-clean)	No hit	Identical 0; **multi-slot identification solves poly0 256/256**
3-of-5 (partial-35-m800; the session pairs of the three active parties are verified separately, as mutually independent attack instances, and both attack and detection need only any one of them)	No hit	**768** identical coordinates per pair (each an independent instance)
Controls (rollback only / replay only / all fresh)	**0**	**0**

The full-corpus audit of Check A covers 100+ directories and 1,250 (party, session) records, yielding 305/305 exact hits with zero false positives; a collision would require two independent reveals to be byte-for-byte identical, $\sim 2^{9000}$ bits of entropy. The false positives of Check B are determined by random agreement (with a threshold of ≥ 4, $< 10^{-20}$), and the measured controls are all zero.

Measured computational cost: Check A performs one hash comparison per record, and the full corpus of 1,250 records takes under 1 second. Check B's cross-slot consistency scan traverses all 32 archived directories, covering the transcript, closure, and partial-replay families in 2-of-2 through 4-of-6 and 3-of-5 configurations, and takes 4 seconds. The identification stage runs only when a determination requires it: about 0.1 seconds per session pair at K=2 and about 15 seconds at K=14, the same order as the attack's identification computation (§6.4). In total, the deployment-side cost of the two checks is on the order of seconds and does not constitute a bottleneck. At 10^6 determinations per day, the expected annual cumulative false positives of Check B are below $3.65 \times 10^8 \times 10^{-20} = 3.7 \times 10^{-12}$, which is negligible.

6.3 Deployment: Choosing the Detection Observation Point (Depending on "Who the Attacker Is")

Core constraint: in this threat model the attacker is the coordinator, so the detection observation point must not pass through the attacker. "External audit with transcripts provided by the coordinator" is circular under this model. We therefore distinguish the following tiers:

Scenario	Feasible deployment	Cost	Positioning
Attacker = coordinator (main model)	(1) **in-protocol broadcast reveal** (each party self-checks) or (2) **public log/receipt** (an operational contract mandatorily delivers transcripts to a neutral third party)	Communication ×N / logging infrastructure	Observation point independent of the coordinator; the preferred choice under the main model

Scenario	Feasible deployment	Cost	Positioning
Coordinator is not the attacker (external eavesdropper/device fault), or retrospective regulatory audit is required	**External audit/monitoring** (run Checks A+B on existing transcripts)	32B/party/session	Minimal change; applicability depends on the scenario

Detection under the main model is therefore not a purely technical solution. It requires either protocol-layer cooperation (an in-protocol broadcast reveal, which needs protocol changes and $\times$N communication) or operations-layer cooperation (a public log/receipt that mandatorily delivers transcripts to a neutral third party). Pure external transcript audit, which requires minimal change, applies only when the coordinator is not the attacker.

Deployment example: the TSaaS variant of Mithril (ML-DSaaS, [TSaaS26]) centralizes communication between the client and the parties through a Relayer. When the Relayer is operated by an entity different from the signing parties, a common arrangement in the Wallets-as-a-Service business model, it naturally lies on the communication path and can run Checks A+B directly. The cost is 32B/party/session plus second-scale identification, and the determination is to be rechecked against the text of [TSaaS26], tracking item (4). If the Relayer is itself the attacker, we return to the two tiers in the table above (in-protocol broadcast or public log).

Trust assumptions for the auditor: the effectiveness of any tier depends on the audit or observation party being independent of the attacker, sharing neither the operator nor the fault surface. Collusion or joint compromise renders detection ineffective; this is the same trust structure as any audit system, such as the log operators of CT logs. The auditor observes only transcripts and needs no trusted execution of any cryptographic function. Enhancements, ordered by benefit, are: (1) connecting audit results to an automated key-rotation process triggered upon detection; (2) multi-party independent audit with cross-verification; (3) publicly verifiable logs, which are tamper-proof and facilitate attribution. The **device-side sampling fingerprint** is a low-cost reinforcement for the residual window of §6.4. Stateful hardware (counters) is not recommended as the sole measure, since it covers only mechanism i.

Unified positioning: for fault attacks with "attacker = coordinator", detection does not block a leak that has already occurred, since the adversary has already seen the reveal of the first faulty session. It does, however, provide detection signals and the fault attribution (key change, repair, upgrade) that existing defenses entirely lack.

6.4 Boundary: Check B and the Attack Share the Identification Computation (Coincident Coverage); the Residual Window

Considering only Check A, a blind window $[768 + \lceil 256/K \rceil, 2048)$ formerly existed in which "the attack succeeds but is undetectable". It arose because r is replayed while e is fresh, so the reveal does not repeat. Within the range where identification is feasible, Check B has no such blind spot. For every polynomial the attacker can recover by the equation count, the auditor can obtain a "box-constrained integer solution + zero residual" certificate using the same identification computation. The shared identification computation thus makes the two boundaries coincide. Likewise, the alarm timing is not earlier than the feasibility point (§6.1):

$$\text{detectable region} \;\equiv\; \text{recoverable region} \;=\; \{\text{poly } p:\ \text{clean}_p \geq 256(K-1)/K\}.$$

Residual window: only polynomials with $\text{clean}_p < 256(K-1)/K$ remain; for K=2, these have fewer than 128 clean coordinates. Here the attack is not recoverable (§7 measures that m=128 is exactly the boundary of poly0 and m=120 is already underdetermined), and detection likewise has no clean signal. This interval leaks at most 127 coarse equations, which does not constitute recovery. The **device-side sampling fingerprint** is a low-cost reinforcement for this residual window, of limited value.

Design lesson (revised): input-freshness hardening (counters, entropy-source health checks) $\neq$ protection, because sampler-layer replay bypasses all input-side measures. By contrast, the output-side identifiability check has the same boundary as the attack: it applies directly to every scheme that makes

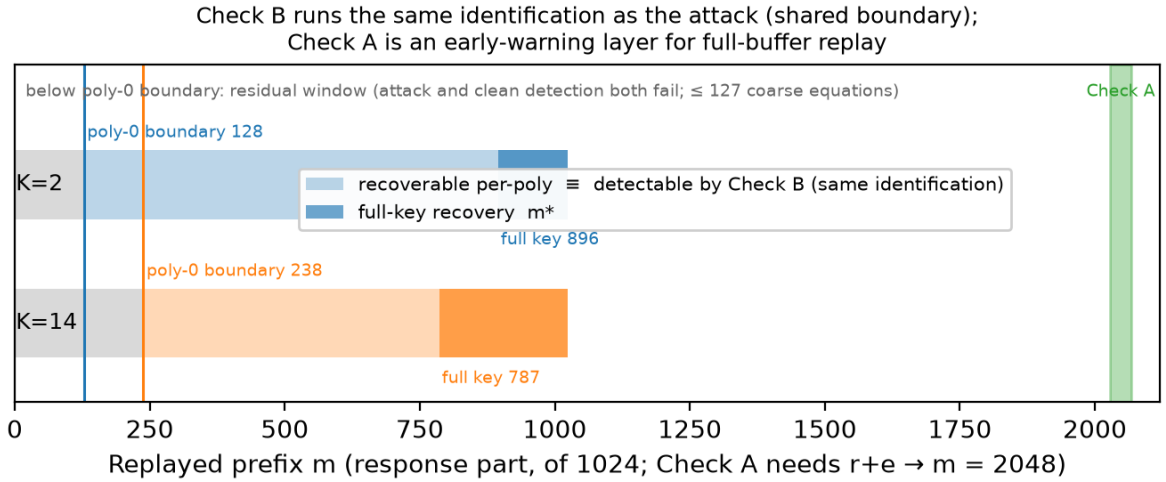

Figure 4: Check B and the attack share the same identification computation (coincident boundaries); Check A is the early-warning layer, covering full-buffer replay.

per-party responses obtainable via the transcript, which is the criterion of §5, and is a deployable complete detection surface.

---

7 Fault Resolution: The Recovery Law under Imperfect Faults

7.1 Model and Solution

Real faults are often imperfect. We model the fault as **prefix stickiness**: the first m coordinates of a sample, out of 1,024 (the response part of dimension LN), are reproduced while the remaining coordinates are fresh. The differences on dirty coordinates are then no longer clean: $\Delta z_j = (\Delta c \otimes \mathbf{s})_j + \delta_j$, where δ_j is the difference of two fresh samples, of magnitude 10^4. The K parallel slots of each party share the same secret, but their challenges and their δ are independent; they are solved jointly per polynomial, all over the integers with no mod wraparound:

$$K \cdot 256 \text{ equations} \geq 256\ (x) + K \cdot k_p\ (\delta) \implies k_p \leq 256 \cdot \tfrac{K-1}{K},$$

that is, the dirty-coordinate tolerance. The full-key threshold is given by the general form $m^* = (\ell - 1)N + \lceil N/K \rceil$, which takes the dirtiest polynomial. For ML-DSA-44 this is $768 + \lceil 256/K \rceil$, with $N = 256$ and $\ell = 4$; for 65/87, $\ell = 5/7$, and the form is unchanged. The solve is a multi-slot joint linear algebra step: column scaling, lstsq, and integer refinement. The condition number of the K=2 critical instance is $\sim 3 \times 10^4$ under the 2^{11} column-scaling convention and $\sim 5 \times 10^2$ under the column-normalization convention, with a runtime of <0.2s.

Modeling remarks and position-independence argument. Prefix stickiness is a natural model of "buffer or register block freeze" faults, and it is also how the test instances are generated. The counting criterion depends only on the total number of dirty coordinates, not on their positions, as argued below. Let the dirty set D_s of slot s be an arbitrary support. The unknowns of the joint system are x and the δ_s of each slot, where δ_s is supported on D_s. Full column rank is equivalent to the non-existence of $x \neq 0$ such that $\mathrm{supp}(\Delta c_s \otimes x) \subseteq D_s$ holds for all slots.

We first make an invertibility assumption. Since $512 \mid q - 1$, R_q splits completely via NTT, and Δc_s is invertible if and only if all its NTT coordinates are nonzero. The probability that a single coordinate vanishes is about $1/q$, so by the union bound the probability that the entire polynomial is non-invertible is about $256/q \approx 3 \times 10^{-5}$. This gives an expected failure count of about 0.3 per 10^4 samples, with 0 measured (§4.2). When an individual slot is non-invertible, the attacker and the auditor can simply switch slots or discard that slot. Under the main model the attacker controls the coordinator and can choose slots, while the auditor observes all slots. Feasibility is therefore not lost.

Under the conditions that the per-slot challenges are independent and each Δc_s is invertible, each constraint $x \in (\Delta c_s \otimes)^{-1} V_{D_s}$ defines a generic subspace of dimension $|D_s|$. The intersection of these subspaces has dimension $\max\bigl(0,\ \sum_s |D_s| - (K-1)N\bigr)$, which contains only cardinalities, not positions. Therefore the feasibility boundary is $\sum_s |D_s| \le (K-1)N$; when all slots have the same dirty count, this becomes $k_p \le 256(K-1)/K$. At the critical point the system is a generic invertible square matrix, and the solution is unique.

Empirical validation (`experiments/scatter_solve.py`; coordinate-level grafting: clean coordinates take measured differences from full-replay pairs, and dirty coordinates take measured differences from independent fresh-session pairs with the same distribution as the dirty rows of a genuine partial replay). For 2-of-2, at the critical point, a randomly scattered dirty set with k=128 and $\sum_s |D_s|$=256 achieves exactly 256/256 in all 10 random-support instances, generated with different random seeds rather than hand-picked (0.05–0.10s). The cases k=136 (272) and heterogeneous 100+161 (261) are count-underdetermined, consistent with the behavior of the prefix $m = 888$. The heterogeneous case 100+156 (256) succeeds, with the non-critical polynomial poly1 scattered 6/6. The condition number fluctuates between $\sim 4 \times 10^4$–4×10^5 with the support positions under the 2^{11} column-scaling convention, without changing the feasibility boundary; this agrees item by item with "depending only on $\sum_s |D_s|$".

For 3-of-5 (K=14), a randomly scattered k=237 with $\sum_s |D_s|$=3318 $\le$ 3328, again critical, yields exactly 256/256 for all three parties (13.7–19.3s), whereas k=238 is underdetermined. We rechecked the heterogeneous boundary in the same way: with slot 0 dirty at 240 and each of the others at 237, so that $\sum_s |D_s|$=3321 $\le$ 3328, all three parties, ×2 seeds, recover exactly 256/256 (14.7–19.1s); 240+13×238 (3334) is underdetermined. The criterion "depends only on $\sum_s |D_s|$" has thus been verified heterogeneously in both the K=2 and K=14 configurations.

7.2 Law Verification (Coordinate-by-Coordinate Exactness)

Configuration	Predicted threshold	Measured
2-of-2 (K=2, ML-DSA-44)	$m^* = 896$ (87.5%)	896 → 1,024/1,024 (0.10s); 888 (dirty 136>128) → underdetermined
3-of-5 (K=14, ML-DSA-44)	$m^* = 787$	787 (dirty 237) → all three parties 1,024/1,024 (17–25s); 786 (dirty 238) → underdetermined

Both thresholds are hit exactly, coordinate by coordinate, with an error of 0 coordinates. The counting argument of the law, whose general form is $m^* = (\ell - 1)N + \lceil N/K \rceil$, does not depend on the specific parameter set, only on ℓ and K. The experiments in this paper use ML-DSA-44, because the public Rust implementation of Mithril, crate 0.3.x, includes only 44; parameter-level reproduction for 65/87 is left to future work.

The general form also holds for 65/87. The same counting criterion, $\sum_s |D_s| \le (K-1)N$, applies independently to each polynomial, and the full key requires the dirtiest polynomial to be solvable. Under **prefix stickiness**, the clean count of the p-th polynomial is $m - pN$, so solvability of the dirtiest one ($p = \ell - 1$) means $m - (\ell - 1)N \ge \lceil N/K \rceil$. Under scattered support, by the position independence of §7.1, it suffices to read "clean count" as the minimum of the clean counts across polynomials, and the conclusion keeps its form. Only ℓ enters the argument, independently of parameters such as η and γ_1; hence 65/87 ($\ell = 5/7$) have the same form.

7.3 The Trade-off Property of K

K-way parallelism was originally a design choice for the signing success rate. From the attack perspective it acts in both directions: it multiplies the leakage surface by ×K (leakage per slot), and at the same time it provides the redundancy needed for solving, since the equations from 256 independent challenges per slot share the same x ⇒ the larger K is, the stronger the fault tolerance. The threshold approaches $(\ell - 1)N$, a fraction $(\ell - 1)/\ell$ of the coefficient space ℓN, which is 75% for ML-DSA-44. Read together with §6.4, K enlarges the leakage surface while also raising the equation redundancy shared by attack and detection, so both boundaries shift down in tandem to a lower threshold. Under the fault threat, the choice of K is a set of design parameters that require a trade-off.

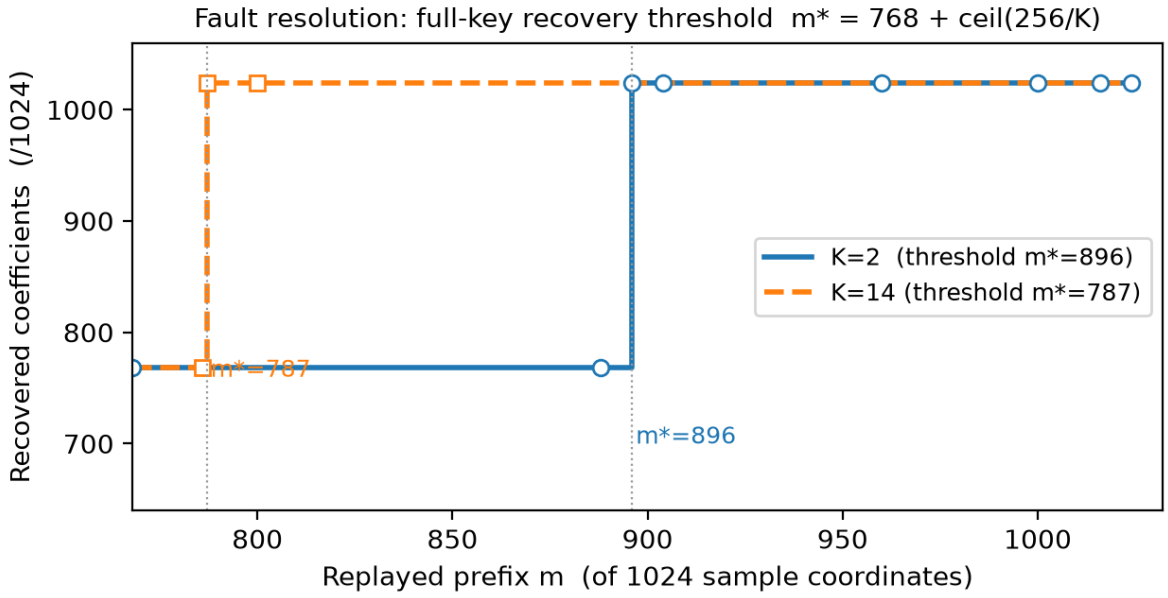

Figure 3: Fault-resolution curve (hollow points = measured; staircase = threshold law).

7.4 Methodological Note (condensed)

Two transferable lessons emerge from the solving process: (1) **flat-objective pathology**. For a feasibility MIP with the secret as variable, when the clean equations have slack, the LP relaxation can reach zero residual directly, and branch and bound loses its direction. (2) **redundancy > optimizer**. What ultimately solved the identification was the multi-slot equation redundancy of §7.1, not a stronger optimizer. The complete record, including the debugging of a matrix-block construction error, is in Appendix C.

---

8 Validation Rigor and Limitations

8.1 Validation Chain

Claim	Independent check	Result
Recovery result independent of the ground-truth file	Replace the entire ground-truth file with constants and rerun	Recovery output **byte-for-byte unchanged**
The recovered secret is genuine	**Public-key consistency** (coefficient bound of $\mathbf{s}_2 - \mathbf{t}_0$)	4,094/4,098, within bound
Recovery capability = forgery capability	The implementation's built-in FIPS 204 verifier; additionally reproduced by a **from-scratch independent** verifier	**ACCEPTED** (2-of-2 and 3-of-5)
Experiments reproducible	An independent auditor rebuilds on a clean copy and reproduces all archives	**Byte-for-byte identical** (all fingerprints match)
Attack algebra independent of mechanism	Three mechanisms (entropy fallback / sampler layer / constant control) verified separately in archives/experiments	See §3.2, §4.3

8.2 Adversarial Validation Workflows (Three Methods) and Execution Disclosure

We subjected all claims to three validation workflows with independent analysis paths. All three used automated tools in fresh contexts under author supervision, without using the author's existing analysis code

or conclusions. All artifacts have been made public on GitHub (https://github.com/M892549874/fault-nonce-threshold-mldsa), so any third party can rerun them step by step. The workflows were as follows. (1) **From-scratch recomputation**: without using any analysis code from this paper, this workflow independently rebuilt the challenge expansion, ring inversion, and packing pipeline and recomputed all results, with bit-for-bit agreement. It also produced discriminating experiments: unpaired-combination residuals of 3.4–5.2$\times 10^4$ with recovery 0/1024, and the truth-destruction experiment, in which the entire ground-truth file is replaced and the recovery output remains byte-for-byte unchanged. (2) **Upstream comparison**: it performed a full-tree diff against the clean source code and checked item by item that the patch list and protocol logic show zero modifications. (3) **Byte-for-byte reproduction**: it rebuilt the implementation in an independent copy and reproduced all archives with all fingerprints identical; an independent control experiment separately reproduced the constant-semantics boundary. The workflows exposed several problems, including fault-class wording, path classification, row-selection rules, and statistical conventions; all were corrected and written back. The workflows consistently found that all core numbers were independently reproduced and agree bit for bit. The attackable surface is concentrated in threat-model assumptions and fault feasibility, which are explicitly declared in §1.3/§3, rather than in code or data consistency.

Execution disclosure. The three workflows above ran in a tool-assisted manner under author supervision. The tools were scripts and LLM-assisted analysis sessions, not human reviewers. Independence at the execution level is limited to methodological path isolation. Each workflow ran in an independent working directory and took only public artifacts as input: data files, upstream clean source code, and transcribed copies. No intermediate code or conclusions were shared between workflows; importing or copying the author's existing analysis code and conclusions was prohibited, as was network access. The author set the validation objectives and the adversarial settings.

Different sessions of the same model still share training biases and reasoning patterns, so this isolation does not constitute independence at the executor level. "Independence" here refers only to independence of analysis paths: from-scratch recomputation, upstream comparison, and byte-for-byte reproduction do not depend on one another. At the execution level, then, this isolation does not constitute independent review by human peers. All problems the workflows exposed were rechecked item by item, corrected, and written back, and all input artifacts and reports are rerunnable (Appendix E). We recommend and welcome third-party reruns of all conclusions.

8.3 Limitations (an Honest List)

1. **No real glitch experiments**: faults are simulated injections, the standard methodology in the field; for precedents see §2.3. The engineering parameters of real injection are left to future work.
2. The recovery target is a synthesized threshold secret. The recovered value is the sum of the shares actually used by the implementation. Its coefficient bound is $C(n, n-t+1)\eta$, which may exceed the η bound that FIPS 204 imposes on a single private key. However, the public key is generated from this secret and is self-consistent with it, and signatures forged from it pass the standard verifier. "Functional equivalence" therefore means signing-capability equivalence, not that the secret distribution matches that of the original FIPS 204 key generation.
3. **A derivation convention**: verifier acceptance holds for verifiers that share this convention; the implementation and the independent reproduction are listed.
4. **sid provisioning assumption**: the replay capability of the vanilla path is a harness modeling assumption. In a real deployment, the protocol layer must specify the supplier of session_id through a broadcast consensus input or a coordinator proposal.
5. **Orchestration boundary of mechanism iii (constant output)**: see Appendix B. Covering the detection residual window of §6.4, namely polynomials with clean<128 for which attack and uniformity detection fail simultaneously, requires device-side fingerprint recording and is of limited value.
6. **Citation checking** (item-by-item source stratification in Appendix D): primary sources (FIPS §3.4, Krahmer et al.) were checked verbatim. For the remaining entries, bibliographic data were verified against publication records, and content sources are noted item by item (publisher's official abstract, indexing abstract, or availability of the original text).
7. The evidence hierarchy and the two kinds of premises for the Trilithium determination are addressed in turn (§5.3). First, the evidence hierarchy has been verified against the original full text: steps 7–10 of Algorithm 14 in the full original text (Springer LNCS edition, 67 pages) were checked verbatim against Algorithm 5, and the simulator messages in the proof agree with and match the [Jarv26] copy

line by line, with the independent grouping $I = 0$ of [TALUS] as corroborating evidence (Appendix D). Second, its threat model is a malicious observer plus one fault on the other party's device, that is, an instance of the §3.1 combination model in the 2-party setting; the assumption that the malicious party has physical contact with the other party's device requires separate justification according to the deployment context.

8. The multi-fault combination surface is not evaluated. The attacks and detection in this paper target a single effect E, the cross-session reproduction of one party's sampling. Two points about combinability can be made. (a) If t devices each apply E, the effects compose directly; this is exactly the $\times t$ amplification in §4.5. (b) Checks A/B are decided independently per party and per pair, so simultaneous reproduction on multiple devices is hit individually in full-corpus auditing (§6.2), and detection coverage does not degrade with the number of faults. **Joint exploitation**, such as the superposition of "skipping the rejection check" [Polytope] and sampling reproduction, and "multiple combinations of faults + protocol-level deviations" belong to an independent threat domain and are left to future work. A preliminary analysis suggests that skipping the rejection check acts on the distribution of "rejected signatures", whereas sampling reproduction acts on "cross-session response differences". The two have different algebraic origins and can be implemented independently, and their combination does not reduce the "single-device effect" budget required by either. Note, however, that the detection surface of this paper (the two-layer checks of §6) targets reveal liveness and does not cover "skip-rejection-check"-type faults, whose countermeasure belongs to rejection-check execution protection (§2.3). Therefore, multi-vector deployments should stack the two classes of protection.

---

9 Discussion: Design and Deployment Recommendations

For threshold protocol designers:

1. Protect the reveal channel or acknowledge it. If per-party responses are necessarily revealed to the coordinator, then "a single-device fault ⇒ share leakage" is a structural fact. The defenses, namely a hedged nonce, commitment/session binding, and fail-closed final verification, are all blind to it. Design documents should therefore assess this threat explicitly rather than rely on the intuition that "threshold means more secure."
2. Liveness and identifiability detection should become a standard monitoring item. We recommend adopting **Check A + B** (§6.1) as a standard monitoring item for threshold ML-DSA deployments, analogous to CT logs in the certificate ecosystem: the observer is independent of the signer, and the value lies in discovery and attribution. The deployment mode is chosen according to the trust model (§6.3). When the attacker is the coordinator, protocol- or operational-layer cooperation is required, such as in-protocol broadcast of reveals or public logs; this is not a purely technical solution. When the coordinator is trusted, an external audit with minimal modification is adopted. Detection and attack share the same identification computation (§6.4), and the entire attack-usable range is covered without introducing new cryptographic assumptions.
3. Input freshness ≠ protection. Counter and entropy-source health checks cover only entropy fallback mechanisms. Sampler-layer reproduction bypasses all input-side hardening, so the **output** liveness of one-time randomness should be independently verified.
4. State management of one-time randomness requires redundant protection. The TALUS "preprocessing double-spend" path shows that the state update itself, which marks a one-time object as consumed, is the fault point; this is isomorphic to "the execution of the check must be protected." Deletion and count updates should be subject to redundant or atomic protection.
5. K (parallel slots) is a trade-off parameter under the fault threat. K amplifies the leakage surface while increasing tolerance of imperfect faults (§7), so the optimization direction of monotonically increasing K must be evaluated together with the fault model.
6. Write the threshold gain under the fault model quantitatively. The gain of thresholdization against fault attacks is $\times$t devices (§4.5), a number that can be written into deployment documents and can also be computed by attackers.

For standards and validation frameworks: The hedged design of FIPS 204 already lists "RNG flaws" as a threat (§2.1). Thresholdization specifications in the NIST IR 8214C context, and validation programs such as FIPS 204 conformance testing, should incorporate (1) coverage statements for the fault model and (2) liveness requirements for the reveal channel.

Future work: real glitching experiments; a low-cost implementation and threat model of the device-side sampling fingerprint; original-text-level confirmation of the Quorus reveal structure (tracking in progress, Appendix D); full-text re-verification of [TSaaS26] and [Kao26]; cost-benefit analysis of embedding liveness checks into the protocol rather than relying on external audit; and Pareto analysis of K and (t,n) under "efficiency × fault safety".

10 Conclusion

This paper shows that the resistance of threshold lattice signatures to "honest-device physical faults" is far lower than their fault-tolerance narrative suggests. A single party's sampling recurring across sessions suffices to recover its share exactly: the residual error is identically zero, and no error handling is required. In a t-of-n deployment, faults in t devices constitute a complete break, and each signature viewed in isolation is entirely valid, so none of the existing defenses is triggered. We elevate this observation to a class-level conclusion: the reveal-structure criterion, two independent evaluation dimensions, and existence evidence for quadrant occupancy, with two instances already on the anti-diagonal, [Trilithium] and [Kao26]. The detection scheme is a two-layer transcript-level check whose layers provide the early-warning surface and the shared identification computation, respectively. The quantitative results are the fault-resolution threshold $m^* = 768+\lceil 256/K \rceil$ and the trade-off property of K. These claims are supported by end-to-end experiments, including two implementation versions, a 96/96 sweep, and forged signatures accepted by the standard verifier, and are re-checked by a verification process executed independently along three analytical paths with tool assistance. What thresholdization buys is a quantifiable fault-safety budget, namely ×t devices. That budget should be written into design documents rather than obscured by the word "threshold".

References

- [**FIPS204**] NIST. *Module-Lattice-Based Digital Signature Standard.* FIPS 204, 2024.
- [**IR8214C**] L. T. A. N. Brandão, R. Peralta. *NIST IR 8214C: NIST First Call for Multi-Party Threshold Schemes.* NIST, 2026-01.
- [**BDK+21**] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, D. Stehlé. *CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation.* Round-3 submission.
- [**Mithril**] D. Celi, R. del Pino, T. Espitau, T. Niot, T. Prest. *Efficient Threshold ML-DSA.* ePrint 2026/013; USENIX Security 2026.
- [**Quorus**] A. Bienstock, L. de Castro, D. Escudero, A. Polychroniadou, A. Takahashi. *Quorus: Efficient, Scalable Threshold ML-DSA Signatures from MPC.* ePrint 2025/1163.
- [**SplitForge**] Cybernetica AS. *SplitKey Preview Writeup (v0.1).* NIST Multi-Party Threshold Schemes Call (TCall-1), 2026.
- [**TALUS**] L. Kao, R. Chang. *TALUS: FIPS-204-Exact Threshold ML-DSA via Boundary Clearance.* arXiv:2603.22109 (NIST threshold call submission TALUS-PW02).
- [**Trilithium**] A. Dufka, S. Kravtšenko, P. Laud, N. Snetkov. *Trilithium: Efficient and Universally Composable Distributed ML-DSA Signing.* ePrint 2025/675; ESORICS 2026 (Springer LNCS; local full-text verification, 67 pages).
- [**Kao26**] L. Kao. *FIPS 204-Compatible Threshold ML-DSA via Shamir Nonce DKG.* arXiv:2601.20917 (v6), 2026.
- [**TSaaS26**] M. Rambaud, S. Roth, A. Urban. *Threshold Signatures as-a-Service: Achieving Threshold ML-DSA in One Online Round.* ePrint 2026/814.
- [**Pebble**] O. Kondi, V. Lapiha, V. Leontiadis, N. K. Sen. *Pebble: Normal Threshold ML-DSA.* Silence Laboratories, 2026.
- [**Polytope**] P. Azevedo-Oliveira, A. Calle Viera, B. Cogliati, L. Goubin. *Finding a Polytope: A Practical Fault Attack Against Dilithium.* PKC 2025; ePrint 2025/195.
- [**DOTT22**] I. Damgård, C. Orlandi, A. Takahashi, M. Tibouchi. *Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices.* Journal of Cryptology 35(2):14, 2022.
- [**dPN25**] R. del Pino, G. Niot. *Finally! A compact lattice-based threshold signature.* PKC 2025, pp. 169–199.
- [**Jarv26**] P. Järv. *Adding Garbled Circuits to the Trilithium Post-quantum Signature Protocol.* MSc

thesis, University of Tartu / Cybernetica, 2026. (Its §3.1–3.3 copy Trilithium's signing algorithm and declassification protocol verbatim, and explicitly record "the client gets to know the server's share"; step 7 of Algorithm 3 is exactly "S→C: $[\![z]\!]_S$" (the simulator message in the original proof corroborates this, and the author's public lecture notes are also listed—see Appendix D for the tier).)

- **[BP18]** L. Groot Bruinderink, P. Pessl. *Differential Fault Attacks on Deterministic Lattice Signatures.* IACR TCHES 2018(3):21–43.
- **[Krahmer+24]** E. Krahmer, P. Pessl, G. Land, T. Güneysu. *Correction Fault Attacks on Randomized CRYSTALS-Dilithium.* IACR TCHES, 2024.
- **[Bauer+25]** S. Bauer, F. De Santis. *Forging Dilithium and Falcon Signatures by Single Fault Injection.* ePrint 2025/2029.
- **[KX25]** H. Kosuge, K. Xagawa. *The Security of ML-DSA against Fault-Injection Attacks.* ePrint 2025/904.
- **[606]** P. Azevedo-Oliveira, J. Beraud, L. Goubin. *An attack on ML-DSA using an implicit hint.* SAC 2025; ePrint 2025/606.
- **[MILP26]** H. Ouyang, C. Wang, G. Liu, L. Wu, M. Wang, Y. Fan. *Improving Skipping Fault Correction Attacks on Randomized Dilithium via MILP.* ePrint 2026/1448 (SAC 2026).
- **[HAETAE-FI26]** W. Shin, S. Jeon, D. Bae, S. Park, H. Kim. *Public Coefficient Matters: A Practical Differential Fault Attack on ML-DSA and HAETAE.* ePrint 2026/1344.
- **[Jo26]** S. Jo. *A Key-Recovery Attack on TALUS-MPC in TALUS v4.* ePrint 2026/1668.
- **[Niot26]** G. Niot. *Key-Recovery Attacks on TALUS: A Cryptanalytic Note.* ePrint 2026/1386.
- **[ACISP25]** D. Du, Y. Liu, Y. Gao, J. Ming, H. Yuan, Y. Zhou. *Breaking the Shield: Novel Fault Attacks on CRYSTALS-Dilithium.* ACISP 2025, pp. 42–63 (page numbers per the NSTL index).
- **[Patent]** Thales Dis France SAS (Azevedo Oliveira, Calle Viera, Cogliati, Vigilant). *Method for protecting against fault attacks an execution of a generation of a CRYSTALS-Dilithium signature.* WO2025261784A1; same family EP 24305977.1 (priority date 2024-06-20).
- **[551]** N. Mishra, D. Mukhopadhyay. *Probabilistic Algorithms with applications to countering Fault Attacks on Lattice based Post-Quantum Cryptography.* ePrint 2024/551.
- **[ToB]** Trail of Bits. *Don't let TEEs break your MPC.* 2026-09-25.
- **[ETSI]** ETSI TR 104 239-3 V1.1.1: *Quantum-Safe Cryptography (QSC); Secure Implementation Guidance for KEMs and Digital Signature Schemes; Part 3: ML-DSA.* 2026-09.

Appendices

Appendix A: Realism Argument for the Fault Class (Condensed Version)

The full argument is as follows. (1) FIPS 204 §3.4 provides a first-hand quotation (§2.1): the standard itself states RNG flaws as a design threat. (2) The BP18 paradigm is that "fault-induced nonce reuse = trivial key recovery using a single faulty signature" (§2.3, Krahmer+24's restatement checked verbatim). (3) Physical faults of the RNG/derivation chain have been practically demonstrated (ACISP 2025, two faulty signatures recover the key, per the publisher's official abstract). (4) The assumption framework of "the same fault across ≥ 2 executions" has been adopted (606). (5) Device-level state rollback has been weaponized in the wallet/TEE context (ToB). The fault budget is given in §3.2: the simplest path is a single sampler-layer fault effect with zero protocol deviation.

Appendix B: Orchestration Boundary of Constant-Output Semantics

Under the semantics of "constant output", in which the entropy source always returns the same byte stream, a stuck sampling by a single party causes the aggregation commitments in a multi-device orchestration to become fixed. Challenge collisions then arise in subsequent sessions, so the two-session pairing must use different messages or a different active set to restore pairing. The byte-by-byte comparison experiment is detailed in artifact `AUDIT_R4b_C`.

Frequency and coverage of mechanism iii (supplementary): A stuck entropy source or repeated output belongs to the natural fault class; entropy starvation, device aging, and entropy-pool failures have a long industrial record. This class does not require attacker injection. For iii, the "attacker" need only observe, which in a sense makes it closer to "device compromise" than to "an attacker requiring

proximity to the device." We are not aware of a public direct demonstration of "pinning the entropy source by injection", so we anchor the simulation for the main results on the two more conservative, precisely simulable mechanisms, i (RNG state rollback) and ii (sampler-layer reproduction). Regarding coverage, iii falls within the complete coverage of the two-layer checks: full-buffer reproduction makes the revealed **w** repeat byte-for-byte across sessions, so Check A hits directly. Even if its reveal structure is altered by a variation in which e is fresh, Check B covers it with the same identification computation (§6.1–6.4). Thus, whether iii is more common does not weaken the coverage of the main results: the more common iii is, the more complete the coverage of the detection surface in §6.

Appendix C: Evolution of the Solver Methodology

The solver methodology evolved through three stages: the flat-objective pathology of the x-variable feasibility MIP, with 248/256 clean and only 92/256 in 120s; δ-codimension parameterization, which was solvable but gave insufficient single-slot identification; and multi-slot simultaneous linear identification, finalized in §7. The complete record includes a debugging episode involving a "matrix block construction error" that manifested as "x exact but residual huge." The lesson is that residual verification must be checked row by row against the matrix; aggregate numbers are not trustworthy. **Scattered-support validation** (`experiments/scatter_solve.py`) used coordinate-level grafting: clean coordinates take measured differences from full-reproduction pairs, and dirty coordinates take measured differences from independent fresh-session pairs. Under 2-of-2 scattered k=128, all ten instances reached 256/256; k=136 and heterogeneous 261 were underdetermined; heterogeneous 256 succeeded, with the non-critical polynomial poly1 scattered 6/6. Under 3-of-5, k=237 gave 256/256 for all three parties, while k=238 was underdetermined; both agree item by item with the full-column-rank criterion $\sum_s |D_s| \le (K-1)N$. The 3-of-5 heterogeneous boundary was additionally tested: 240+13×237=3321 succeeds (three parties × 2 seeds), while 240+13×238=3334 is underdetermined.

Appendix D: Citation Verification Status (2026-10-20; Source Tiers Distinguished Item by Item)

A. Verified against the original full text (quotations checked verbatim or key points checked): [FIPS204] (§3.4 quotation verbatim), [Krahmer+24] (local paper full text, quotation verbatim), [Mithril] (local full text), [TALUS] (local full text), [Pebble] (local full text), [Polytope] (local full text), [Patent] (WIPO family status report, title checked verbatim), [Jarv26] (local paper full text; Algorithm 3 and §3.1.2 quotations verified verbatim), [Kao26] (local arXiv full text v6; Remark 19 on per-party response visibility and the statistical-protection statement verified verbatim), [Trilithium] (original full text, Springer LNCS version, 67 pages; Algorithm 14 steps 7–10, Algorithm 5, and the proof-passage quotations checked verbatim).

B. Bibliographic data verified; content source stated item by item:

Item	Bibliographic source	Content source	Reason the original text was unavailable
[ACISP25]	Springer/ACM publication record (pp. 42–63; DOI 10.1007/978-981-96-9098-5_3)	**Publisher's official abstract** (two faulty signatures; masked implementation 48.9%)	Full text requires subscription
[606]	ePrint record + SAC 2025 publication record	Abstract	ePrint full text unreachable from the local network
[KX25] [Bauer+25] [MILP26] [HAETAE-FI26] [Jo26] [Niot26] [551]	ePrint records	Abstract	As above
[ETSI]	Specification bibliographic data (V1.1.1, 2026-09)	Abstract/table-of-contents page	Official-site PDF returns 403

Item	Bibliographic source	Content source	Reason the original text was unavailable
[ToB]	Blog publication record (2026-09-25)	Abstract	—
[BDK+21] [BP18]	Publication records	Descriptive citation (no specific numerical claims)	—
[DOTT22]	ePrint 2020/1110 page + J. Cryptology record	Bibliographic data and abstract verified; the partial-signature aggregation structure is restated from [Pebble] original §1–§3	—
[dPN25]	ePrint 2025/872 page + PKC 2025 record	Bibliographic data and abstract verified ("T-way parallel Dilithium executions"); structure restated from [Pebble] original §1–§3	—
[TSaaS26]	ePrint abstract page (2026/814)	Abstract: TSaaS model, Relayer-centralized communication, Mithril variant (ML-DSaaS); full text 403	ePrint PDF unreachable from the local machine

Tracking items: (1) Re-check the §5.2 determination once [TALUS] releases its revised construction; the construction is being revised following disclosure of the Niot attack. (2) [Trilithium] is completed and closed: the original full text (Springer LNCS version, 67 pages) was checked verbatim locally. Algorithm 14 steps 7–10, Algorithm 5, the direct statement in the body, and the simulator message in the proof passage all agree with the determinations in §5.2/§5.3; the copy in [Jarv26] matches line by line, and P is C in its Algorithm 3. (3) The details of the fault path of [Kao26] await re-verification against its formal version, including the sampling reproduction semantics of the offline nonce DKG, one-time consumption, and the effect of the mask on $\Delta\mathbf{z}_i$; its reveal structure and statistical-protection determination have been verified against the local full text. (4) Once the full text of [TSaaS26] (ML-DSaaS) is obtained, re-examine the reveal structure, including Relayer visibility; the abstract-level determination is that it inherits Mithril's coordinator-visible model. The remaining items, namely (1)(3)(4), remain open.

Appendix E: Artifacts

The reference implementation patch list and examples (the crates can be restored according to `PATCHES.md`), all transcript archives (including the fingerprint manifest, provided with the repository via `transcripts.tar.gz`), the attack, detection, and solver scripts, the audit reports, and the developer's complete record are publicly available on GitHub: https://github.com/M892549874/fault-nonce-threshold-mldsa. All core experiments can be reproduced byte-for-byte from deterministic seeds.