Lemur: Scalable Post Quantum Synchronized Multi-Signatures

Muhammed F. Esgin[1] Yini Lin[1] Markku-Juhani O. Saarinen[2] Amin Sakzad[1]
Ron Steinfeld[1]

[1]Monash University

[2]Tampere University

Presented on 2026-July-8 at TCPT2

NIST Event for Threshold Call Preview Talks # 2

Non-Interactive Multi-Signatures

Why Non-Interactive?

- Validators sign **independently**; no extra communication rounds.
- Any *untrusted aggregator* can combine signatures publicly.
- Essential for **proof-of-stake blockchains** (Ethereum, Dfinity, etc.).

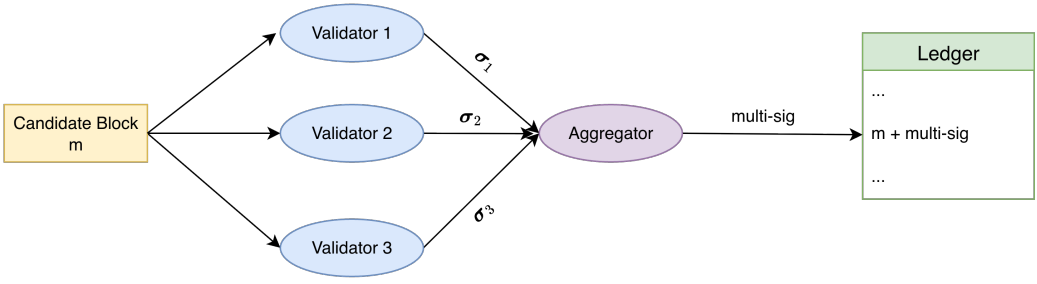

Figure 1: Non-interactive multi-signature for blockchain.

Existing Non-interactive PQ Aggregation

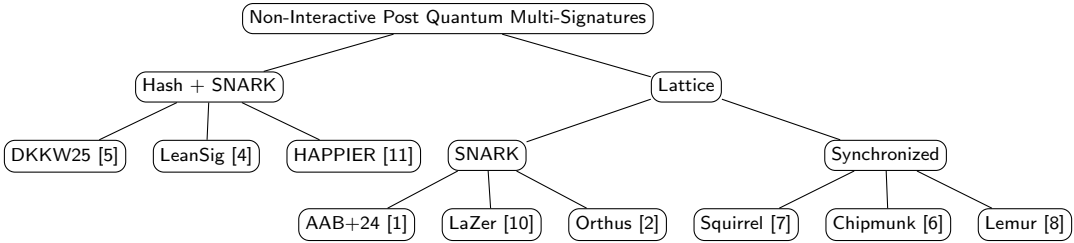

Figure 2: Existing non-interactive post quantum multi-signatures.

- **Efficiency.** Aggregate signature size, verification time, aggregation time, RAM
- **Scalability.** Aggregate signature size grows significantly with N?
- **Other Features.** Batch verifiable, practical enough to be implemented, ...

Existing Non-interactive PQ Aggregation

Feature	**SNARK-Based** e.g., LaZer [10], Orthus [2]	**Synchronized** e.g., Chipmunk [6], Lemur [8]
Distinct-message aggregation	✓	✗
Homomorphic aggregation	✗	✓
Public-key aggregation	✗	✓
Verification efficiency	linear/sublinear in N	independent of N after pk agg.
Computational overhead	High	Low
Implementation complexity	High	Low

Table 1: Feature level comparison of existing non-interactive multi-signatures.

Existing Non-interactive PQ Aggregation

Scheme	# signers	$\lvert\sigma_{\text{agg}}\rvert$	Sig. Agg. Time	PK Agg. Time	Vrfy Time	RAM
Chipmunk [6]	2^{10}	237	?	?	?	?
	2^{16}	>425	?	?	?	?
LaZer [10]	2^{10}	74	0.6	N/A	≈ 0.36	200
	10^5	72	65	N/A	≈ 39	26112
Orthus [10]	2^{13}	≈ 100	5	N/A	1.4	?
	2^{16}	≈ 100	32.4	N/A	4.6	?
HAPPIER [11]	2^{10}	2396	221	N/A	0.14	<16384
	2^{16}	2959	795	N/A	0.16	<16384

Table 2: Performance comparison of current non-interactive multi-signature schemes. Aggregate signature size $\lvert\sigma_{\text{agg}}\rvert$ is reported in KB, RAM usage in MB, and all execution times (signature aggregation, public key aggregation, and verification) in seconds. Chipmunk [6] signature sizes are sourced from its re-evaluation in [8]. Note that reported runtimes may not be directly comparable due to varying implementation-specific optimizations, such as multi-threading or AVX acceleration. "?": not (explicitly) reported. "N/A": not applicable.

Existing Non-interactive PQ Aggregation

Scheme	# signers	$\lvert\sigma_{\text{agg}}\rvert$	Sig. Agg. Time	PK Agg. Time	Vrfy Time	RAM
Chipmunk [6]	2^{10}	237	?	?	?	?
	2^{16}	>425	?	?	?	?
LaZer [10]	2^{10}	74	0.6	N/A	≈ 0.36	200
	10^5	72	65	N/A	≈ 39	26112
Orthus [10]	2^{13}	≈ 100	5	N/A	1.4	?
	2^{16}	≈ 100	32.4	N/A	4.6	?
HAPPIER [11]	2^{10}	2396	221	N/A	0.14	<16384
	2^{16}	2959	795	N/A	0.16	<16384
Lemur [8]	2^{10}	185	1.16	0.01	0.0033	≈ 0.11
	2^{16}	<304	22.4	0.12	0.0033	≈ 0.11

Table 2: Performance comparison of current non-interactive multi-signature schemes. Aggregate signature size $\lvert\sigma_{\text{agg}}\rvert$ is reported in KB, RAM usage in MB, and all execution times (signature aggregation, public key aggregation, and verification) in seconds. Chipmunk [6] signature sizes are sourced from its re-evaluation in [8]. Note that reported runtimes may not be directly comparable due to varying implementation-specific optimizations, such as multi-threading or AVX acceleration. "?": not (explicitly) reported. "N/A": not applicable.

Lemur: Key Features

Lemur is a lattice-based synchronized multi-signature scheme that is secure against rogue-key attack in the random oracle model.

- **Post Quantum**. Lemur's security is rooted in Module-LWE / Module-SIS.
- **Aggregatable**. Sublinear-size signatures; 1M sigs $\rightarrow$ 380 KB (vs >728 KB Chipmunk).
- **Batch verifiable**. Single-pass verification; no per-signer cost.
- **Non-interactive**. Restricted to the synchronized setting.
- **Threshold Friendly**. Threshold policy is enforced at verification time.

Lemur: High Level Idea

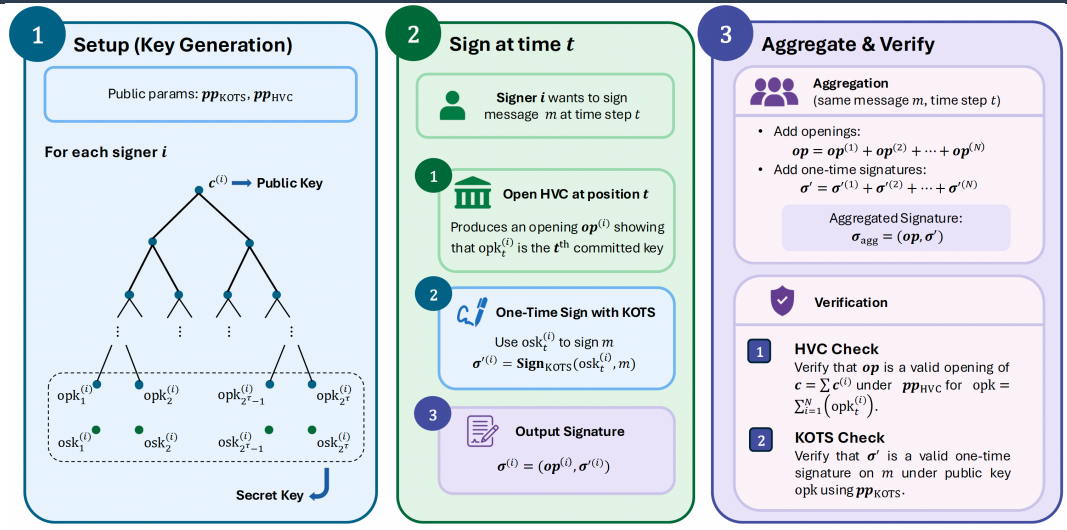

Figure 3: High Level Idea of Lemur / Chipmunk / Squirrel. For simplicity, we ignore the rogue-key attack in the figure. HVC: Homomorphic Vector Commitment. KOTS: Key-Homomorphic One-Time Signature.

Limitation of Chipmunk KOTS

- Chipmunk KOTS follows LM08 [9] / BK20 [3]
- Security relies on **statistical arguments**
  - **entropy loss**: public key + one signature
  - secret key is information-theoretically hidden
- **Key issues: Requires large parameters**
  - larger signatures
  - less flexibility for aggregation
- **Takeaway: statistical security bottleneck limits efficiency**

From Chipmunk to Lemur

Chipmunk KOTS
Statistical security

Problem: Use large parameters to retain enough entropy.

Lemur KOTS
Dual Hint-MLWE

Idea: Model this leakage computationally instead of statistically.

KOTS: Lemur vs Chipmunk

Chipmunk

- Statistical proof
- Secret key must retain absolute entropy
- Large parameters
- Agg. KOTS: 20–110 KB

Lemur

- Dual Hint-MLWE
- Computational security
- Tighter parameters
- Agg. KOTS: < 8 KB (Better scalabilty)

The key innovation: statistical proof $\rightarrow$ computational one.

New assumption $\Longrightarrow$ Better efficiency

Lemur KOTS

Key Generation

- Secret key: short matrix $\mathbf{S}$
- Public key: $\mathbf{T} = \mathbf{SA} \pmod{q'}$

Signing

$$\mu \xrightarrow{\text{Hash}} \mathbf{h}$$

$$\boxed{\mathbf{z}^\top = \mathbf{h}^\top \mathbf{S}}$$

- Signature is a linear projection of $\mathbf{S}$

Verification

Accept iff

$$\boxed{\|\mathbf{z}\|_\infty \leq \beta \text{ and } \mathbf{z}^\top \mathbf{A} = \mathbf{h}^\top \mathbf{T} \pmod{q'}}$$

How to Aggregate?

Both public keys and signatures are linear in $\mathbf{S}$.

$$\sum_i \mathbf{z}_i = \mathbf{h}^\top \left(\sum_i \mathbf{S}_i \right)$$

$$\sum_i \mathbf{T}_i = \left(\sum_i \mathbf{S}_i \right) \mathbf{A}$$

Security of Lemur KOTS

KOTS construction. $\boxed{\mathbf{T} = \mathbf{SA}}$ $\boxed{\mathbf{z}^\top = \mathbf{h}^\top \mathbf{S}}$

The verifier simultaneously sees

$$(\mathbf{A},\ \mathbf{SA},\ \mathbf{h},\ \mathbf{h}^\top \mathbf{S}).$$

Does revealing the linear projection $\mathbf{h}^\top \mathbf{S}$ leak enough information to break security?

Dual Hint-MLWE

Distinguish between

$$D_0 = (\mathbf{A},\ \mathbf{SA},\ \mathbf{h},\ \mathbf{h}^\top \mathbf{S})$$

and

$$D_1 = (\mathbf{A},\ (\mathbf{S} + \mathbf{U})\mathbf{A},\ \mathbf{h},\ \mathbf{h}^\top \mathbf{S}),$$

where every column of $\mathbf{U}$ is sampled uniformly from the right kernel of $\mathbf{h}$.

Security of Lemur KOTS

- We prove a formal reduction

$$\text{Module-LWE} \Longrightarrow \textbf{Dual Hint-MLWE}.$$

- We prove the unforgeability of Lemur's KOTS under

$$\textbf{Dual Hint-MLWE} + \textbf{MSIS}.$$

- This replaces Chipmunk's statistical proof with a computational one.

Lemur HVC

Inherited from Chipmunk

- Merkle-tree framework
- Ajtai hash
- Homomorphic aggregation

Our improvements

- Generalize message space

  Vector $\rightarrow$ **Matrix**

- Replace security assumption

  Ring-SIS $\rightarrow$ **Module-SIS**

Generalized HVC $\Longrightarrow$ More flexible parameters $\Longrightarrow$ Better efficiency

Performance

N (signers)	Chipmunk (KB)		Lemur (KB)	
	One-Time Sig	Multi-Sig	One-Time Sig	Multi-Sig
2^{10}	26	237	5.6	185
2^{15}	$>$90	$>$411	6.9	265
2^{20}	$>$110	$>$728	7.8	380

Table 3: Aggregated signature sizes at 128-bit security (τ=20). Chipmunk: theoretical, from their scripts. Lemur: measured in implementation for N=2^{10}; Rice-encoded for $N \in \{2^{15}, 2^{20}\}$.

Open Question.

How to extend from one-time signing to many-time signing (beyond HVC)?

Summary

Lemur replaces statistical arguments with computational ones, leading to a **more scalable** post-quantum synchronized multi-signature scheme.

Highlight. 1M sigs $\rightarrow \approx$380 KB (vs >728 KB Chipmunk); KOTS $<$ 8 KB; constant batch verification time $\approx$ 3.3 ms.

Contributions.

- A more compact and scalable KOTS, based on Dual Hint-MLWE and MSIS
- A formal reduction from Dual Hint-MLWE to standard Module-LWE
- A generalized HVC under Module-SIS
- Open-source implementation: `https://github.com/lemur-sig`
- Lemur Paper: `ia.cr/2026/1161`

Thank you!

References I

Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi.
Aggregating falcon signatures with labrador.
In *CRYPTO 2024*, LNCS, pages 71–106. Springer, 2024.

Madalina Bolboceanu, Jonathan Bootle, Vadim Lyubashevsky, Antonio Merino-Gallardo, and Gregor Seiler.
Orthus: Practical sublinear batch-verification of lattice relations from standard assumptions.
To appear in CRYPTO 2026, ia.cr/2026/1161.

Dan Boneh and Sam Kim.
One-time and interactive aggregate signatures from lattices.
https://api.semanticscholar.org/CorpusID:221364120.

References II

Justin Drake, Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner.
Technical note: Leansig for post-quantum ethereum.
Preprint, ia.cr/2025/1332.

Justin Drake, Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner.
Hash-based multi-signatures for post-quantum ethereum.
CIC 2025, 2(1):13, 2025.

Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang.
Chipmunk: Better synchronized multi-signatures from lattices.
In *CCS 2023*, pages 386–400. ACM, 2023.

Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang.
Squirrel: Efficient synchronized multi-signatures from lattices.
In *CCS 2022*, pages 1109–1123. ACM, 2022.

References III

Yini Lin, Muhammed F. Esgin, Amin Sakzad, Ron Steinfeld, and Markku-Juhani O. Saarinen.
Lemur: Scalable post-quantum synchronized multi-signatures.
Preprint, ia.cr/2026/1161.

Vadim Lyubashevsky and Daniele Micciancio.
Asymptotically efficient lattice-based digital signatures.
In *TCC 2008*, LNCS, pages 37–54. Springer, 2008.

Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer.
The lazer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy.
In *CCS 2024*, pages 3125–3137. ACM, 2024.

References IV

Arda Saygan, Muhammed Said Gündogan, Atakan Arslan, and Mehmet Emin Gönen.
Hash-based, aggregatable, practical post-quantum signatures implemented efficiently with risc0.
In *LightSec 2025*, LNCS, pages 3–22. Springer, 2025.