

Tanuki: Two-round Threshold Signatures from Lattices

NIST Workshop on Multi-Party Threshold Schemes

Jan 28th, 2026

Cecilia Boschini, Thomas Espitau, Aaron Kaiser, Shuichi Katsumata, Darya Kaviani, Russell W.F. Lai, Giulio Malavolta, Thomas Prest, Peter Schwabe, **Akira Takahashi**, Kaoru Takemure, Mehdi Tibouchi

ETH zürich

 **PQ SHIELD**



MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

 **AIST**

UC Berkeley

A?
Aalto University

Bocconi

Radboud Universiteit



J.P.Morgan

 **NTT**

Introduction

- Class S1: Threshold signature scheme
- Lattice-based scheme based on **Raccoon** signature
- FROST-style two-round threshold signing with preprocessing
- Merge of two previously published works [EKT24, BKLMTT24] with similar design patterns + additional optimizations



[EKT24] T. Espitau, S. Katsumata, and K. Takemure. Two-round threshold signature from algebraic one-more learning with errors. *CRYPTO 2024*. <https://ia.cr/2024/496>

[BKLMTT24] C. Boschini, D. Kaviani, R. W.F. Lai, G. Malavolta, A. Takahashi, M. Tibouchi. Ringtail: Practical Two-Round Threshold Signatures from Learning with Errors. *Oakland 2025*. <https://ia.cr/2024/1113>

Tanuki: Design Rationale

Enables two-round signing

FROST thresholdization [KG20]

Schnorr-like ID from Lattices

Fiat-Shamir

Raccoon Signature [dPKPR24]

T-Raccoon masking [dPK+24]

Enables secure instantiation with Shamir sharing



Tanuki: Design Rationale

Enables two-round signing

FROST thresholdization [KG20]

Schnorr-like ID from Lattices

Fiat-Shamir

Raccoon Signature [dPKPR24]

T-Raccoon masking [dPK+24]

Enables secure instantiation with Shamir sharing

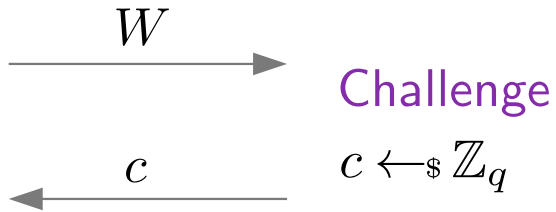


Starting Point: Schnorr ID from Lattices

Commit

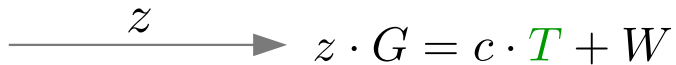
$$r \leftarrow \mathbb{Z}_q$$

$$W := r \cdot G$$



Response

$$z := c \cdot s + r$$



Prover(s)

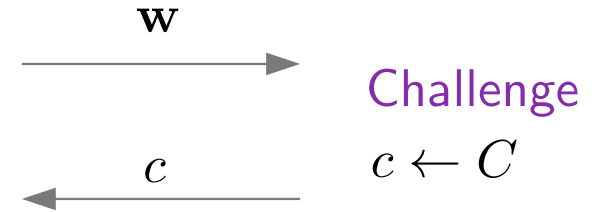


Verifier($T = s \cdot G$)

Commit

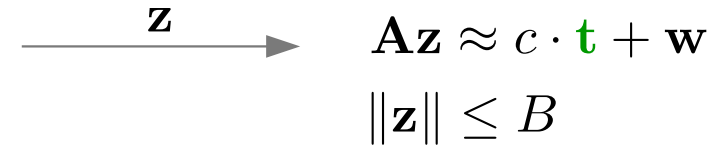
$$(\mathbf{r}, \mathbf{e}^*) \leftarrow \bar{D}^{k+l}$$

$$\mathbf{w} := \mathbf{A}\mathbf{r} + \mathbf{e}^*$$



Response

$$\mathbf{z} := c \cdot \mathbf{s} + \mathbf{r}$$



Prover(\mathbf{s})



Verifier($\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$)

Starting Point: Schnorr ID from Lattices

Setup & Keys

- Power-of-2 cyclotomic ring: $R = \mathbb{Z}[X]/(X^\varphi + 1)$
- Quotient ring: $R_q = R/qR$
- Random public matrix: $\mathbf{A} \in R_q^{k \times \ell}$
- Small secret and error: $(\mathbf{s}, \mathbf{e}) \leftarrow D^{k+\ell}$
- LWE public key: $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
- Sparse and ternary challenge space: $C \subset R$

Why does verification work?

$$\begin{aligned} c \cdot \mathbf{t} + \mathbf{w} &= c(\mathbf{A}\mathbf{s} + \mathbf{e}) + \mathbf{A}\mathbf{r} + \mathbf{e}^* \\ &= \mathbf{A}(c \cdot \mathbf{s} + \mathbf{r}) + \underbrace{c \cdot \mathbf{e} + \mathbf{e}^*}_{\text{small!}} \approx \mathbf{A}\mathbf{z} \end{aligned}$$

Commit

$$(\mathbf{r}, \mathbf{e}^*) \leftarrow \bar{D}^{k+\ell}$$

$$\mathbf{w} := \mathbf{A}\mathbf{r} + \mathbf{e}^*$$

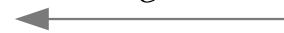
\mathbf{w}



Challenge

$$c \leftarrow C$$

c



Response

$$\mathbf{z} := c \cdot \mathbf{s} + \mathbf{r}$$

\mathbf{z}



Check

$$\mathbf{A}\mathbf{z} \approx c \cdot \mathbf{t} + \mathbf{w}$$

$$\|\mathbf{z}\| \leq B$$



Prover(\mathbf{s})



Verifier($\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$)

Raccoon Signature [dPE+23]

Setup & Keys

- Power-of-2 cyclotomic ring: $R = \mathbb{Z}[X]/(X^\varphi + 1)$
- Quotient ring: $R_q = R/qR$
- Random public matrix: $\mathbf{A} \in R_q^{k \times \ell}$
- Small secret and error: $(\mathbf{s}, \mathbf{e}) \leftarrow D^{k+\ell}$
- LWE public key: $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
- Sparse and ternary challenge space: $C \subset R$

Why does verification work?

$$\begin{aligned} c \cdot \mathbf{t} + \mathbf{w} &= c(\mathbf{A}\mathbf{s} + \mathbf{e}) + \mathbf{A}\mathbf{r} + \mathbf{e}^* \\ &= \mathbf{A}(c \cdot \mathbf{s} + \mathbf{r}) + \underbrace{c \cdot \mathbf{e} + \mathbf{e}^*}_{\text{small!}} \approx \mathbf{A}\mathbf{z} \end{aligned}$$

Commit

$$(\mathbf{r}, \mathbf{e}^*) \leftarrow \bar{D}^{k+\ell}$$

$$\mathbf{w} \leftarrow \lfloor \mathbf{A}\mathbf{r} + \mathbf{e}^* \rfloor$$

Challenge

$$c \leftarrow H(\mathbf{A}, \mathbf{t}, m, \mathbf{w})$$

Response

$$\mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{r}$$

$$\mathbf{h} \leftarrow \mathbf{w} - \lfloor \mathbf{A}\mathbf{z} - \xi \cdot c \cdot \mathbf{t} \rfloor$$

Check

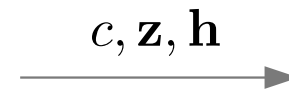
$$\mathbf{w}' \leftarrow \lfloor \mathbf{A}\mathbf{z} - \xi \cdot c \cdot \mathbf{t} \rfloor + \mathbf{h}$$

$$c = H(\mathbf{A}, \mathbf{t}, m, \mathbf{w}')$$

$$\|(\mathbf{z}, \nu \cdot \mathbf{h})\|_2 \leq B$$



Signer(\mathbf{s})



Verifier($\mathbf{t} = \lfloor \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \rfloor$)

Raccoon Signature [dPE+23]

Setup & Keys

- Power-of-2 cyclotomic ring: $R = \mathbb{Z}[X]/(X^\varphi + 1)$
- Quotient ring: $R_q = R/qR$
- Random public matrix: $\mathbf{A} \in R_q^{k \times \ell}$
- Small secret and error: $(\mathbf{s}, \mathbf{e}) \leftarrow D^{k+\ell}$
- LWE public key: $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
- Sparse and ternary challenge space: $C \subset R$

Why Raccoon?

- ✓ No rejection sampling
- ✓ $\mathbf{A}\mathbf{r} + \mathbf{e}^*$ before rounding is not sensitive

Commit

$$(\mathbf{r}, \mathbf{e}^*) \leftarrow \bar{D}^{k+\ell}$$

$$\mathbf{w} \leftarrow \lfloor \mathbf{A}\mathbf{r} + \mathbf{e}^* \rfloor$$

Challenge

$$c \leftarrow H(\mathbf{A}, \mathbf{t}, m, \mathbf{w})$$

Response

$$\mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{r}$$

$$\mathbf{h} \leftarrow \mathbf{w} - \lfloor \mathbf{A}\mathbf{z} - \xi \cdot c \cdot \mathbf{t} \rfloor$$

Check

$$\mathbf{w}' \leftarrow \lfloor \mathbf{A}\mathbf{z} - \xi \cdot c \cdot \mathbf{t} \rfloor + \mathbf{h}$$

$$c = H(\mathbf{A}, \mathbf{t}, m, \mathbf{w}')$$

$$\|(\mathbf{z}, \nu \cdot \mathbf{h})\|_2 \leq B$$



Signer(\mathbf{s})

$c, \mathbf{z}, \mathbf{h}$



Verifier($\mathbf{t} = \lfloor \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \rfloor$)

*ML-DSA relies on a similar blueprint, with several non TS-friendly optimizations

Tanuki: Design Rationale

Enables two-round signing

FROST thresholdization [KG20]

Schnorr-like ID from Lattices

Fiat-Shamir

Raccoon Signature [dPKPR24]

Enables secure instantiation with Shamir sharing

T-Raccoon masking [dPK+24]



Tanuki Signing (Draft)

Offline

- 1: $\mathbf{R}_i \leftarrow \text{SampleR}$
- 2: $\mathbf{E}_i \leftarrow \text{SampleE}$
- 3: $\mathbf{W}_i \leftarrow \mathbf{A}\mathbf{R}_i + \mathbf{E}_i$

Online

- 1: $\text{ssid} \leftarrow (T, (\mathbf{W})_{j \in T}, m)$
- 2: $\mathbf{b} \leftarrow G(\text{vk}, \text{ssid})$
- 3: $\mathbf{W} \leftarrow \sum_{j \in T} \mathbf{W}_j$
- 4: $\mathbf{w} \leftarrow \lfloor \mathbf{W}\mathbf{b} \rfloor$
- 5: $c \leftarrow H(\text{vk}, m, \mathbf{w})$
- 6: $\mathbf{m}_i \leftarrow \text{MaskGen}(\text{sd}_i, \text{ssid})$
- 7: $\mathbf{z}_i \leftarrow c \cdot \lambda_{T,i} \cdot \mathbf{s}_i + \mathbf{R}_i\mathbf{b} + \mathbf{m}_i$

Finalize

- 1: $\mathbf{z} = \sum_{j \in T} \mathbf{z}_j$; compute hint \mathbf{h}
- 2: output $\sigma = (c, \mathbf{z}, \mathbf{h})$



$$\mathbf{W}_i \in R_q^{k \times \text{rep}}$$

$$\mathbf{W}_j \text{ for } j \in T \setminus \{i\}$$

$$\mathbf{z}_i \in R_q^\ell$$

$$\mathbf{z}_j \text{ for } j \in T \setminus \{i\}$$

KeyGen

- 1: $(\mathbf{s}, \mathbf{e}) \leftarrow \text{SampleKey}$
- 2: $\mathbf{t} \leftarrow \lfloor \mathbf{A}\mathbf{s} + \mathbf{e} \rfloor$
- 3: $(\mathbf{s}_1, \dots, \mathbf{s}_n) \leftarrow \text{ShamirShare}(\mathbf{s})$
- 4: $(\text{sd}_1, \dots, \text{sd}_n) \leftarrow \text{SeedGen}$

Design Choices

1. FROST-style aggregation [KG20]
2. One-time mask for secure aggregation [dPK+24]
 - $\sum_{j \in T} \mathbf{m}_j = \mathbf{0}$
 - MaskGen can be instantiated with PRF
 - Why? – $\mathbf{R}_i\mathbf{b}$ doesn't mask large $c \cdot \lambda_{T,i} \cdot \mathbf{s}_i$!
 - Allows standard Shamir-sharing of \mathbf{s} with large Lagrange coefficients and shares

Tanuki Signing (Draft)

Offline

- 1: $\mathbf{R}_i \leftarrow \text{SampleR}$
- 2: $\mathbf{E}_i \leftarrow \text{SampleE}$
- 3: $\mathbf{W}_i \leftarrow \mathbf{A}\mathbf{R}_i + \mathbf{E}_i$

Online

- 1: $\text{ssid} \leftarrow (T, (\mathbf{W})_{j \in T}, m)$
- 2: $\mathbf{b} \leftarrow G(\text{vk}, \text{ssid})$
- 3: $\mathbf{W} \leftarrow \sum_{j \in T} \mathbf{W}_j$
- 4: $\mathbf{w} \leftarrow \lfloor \mathbf{W}\mathbf{b} \rfloor$
- 5: $c \leftarrow H(\text{vk}, m, \mathbf{w})$
- 6: $\mathbf{m}_i \leftarrow \text{MaskGen}(\text{sd}_i, \text{ssid})$
- 7: $\mathbf{z}_i \leftarrow c \cdot \lambda_{T,i} \cdot \mathbf{s}_i + \mathbf{R}_i \mathbf{b} + \mathbf{m}_i$

Finalize

- 1: $\mathbf{z} = \sum_{j \in T} \mathbf{z}_j$; compute hint \mathbf{h}
- 2: output $\sigma = (c, \mathbf{z}, \mathbf{h})$



$$\mathbf{W}_i \in R_q^{k \times \text{rep}}$$

$$\mathbf{W}_j \text{ for } j \in T \setminus \{i\}$$

$$\mathbf{z}_i \in R_q^\ell$$

$$\mathbf{z}_j \text{ for } j \in T \setminus \{i\}$$

KeyGen

- 1: $(\mathbf{s}, \mathbf{e}) \leftarrow \text{SampleKey}$
- 2: $\mathbf{t} \leftarrow \lfloor \mathbf{A}\mathbf{s} + \mathbf{e} \rfloor$
- 3: $(\mathbf{s}_1, \dots, \mathbf{s}_n) \leftarrow \text{ShamirShare}(\mathbf{s})$
- 4: $(\text{sd}_1, \dots, \text{sd}_n) \leftarrow \text{SeedGen}$

Possible Instantiations

1. Discard low order bits of \mathbf{W}_i to save bandwidth
2. Distributions of secrets $\mathbf{s}, \mathbf{e}, \mathbf{R}, \mathbf{E}$
 - Discrete Gaussian [EKT24, BKLMTT24]
 - Sum of uniform [dPKPR24]
3. Aggregation vector \mathbf{b}
 - Signed monomial (and multiply \mathbf{t} by 2) [EKT24]
 - Discrete Gaussian [BKLMTT24]

Tanuki Signing (Draft)

Offline

- 1: $\mathbf{R}_i \leftarrow \text{SampleR}$
- 2: $\mathbf{E}_i \leftarrow \text{SampleE}$
- 3: $\mathbf{W}_i \leftarrow \mathbf{A}\mathbf{R}_i + \mathbf{E}_i$

Online

- 1: $\text{ssid} \leftarrow (T, (\mathbf{W})_{j \in T}, m)$
- 2: $\mathbf{b} \leftarrow G(\text{vk}, \text{ssid})$
- 3: $\mathbf{W} \leftarrow \sum_{j \in T} \mathbf{W}_j$
- 4: $\mathbf{w} \leftarrow \lfloor \mathbf{W}\mathbf{b} \rfloor$
- 5: $c \leftarrow H(\text{vk}, m, \mathbf{w})$
- 6: $\mathbf{m}_i \leftarrow \text{MaskGen}(\text{sd}_i, \text{ssid})$
- 7: $\mathbf{z}_i \leftarrow c \cdot \lambda_{T,i} \cdot \mathbf{s}_i + \mathbf{R}_i \mathbf{b} + \mathbf{m}_i$

Finalize

- 1: $\mathbf{z} = \sum_{j \in T} \mathbf{z}_j$; compute hint \mathbf{h}
- 2: output $\sigma = (c, \mathbf{z}, \mathbf{h})$



$$\mathbf{W}_i \in R_q^{k \times \text{rep}}$$

$$\mathbf{W}_j \text{ for } j \in T \setminus \{i\}$$

$$\mathbf{z}_i \in R_q^\ell$$

$$\mathbf{z}_j \text{ for } j \in T \setminus \{i\}$$

KeyGen

- 1: $(\mathbf{s}, \mathbf{e}) \leftarrow \text{SampleKey}$
- 2: $\mathbf{t} \leftarrow \lfloor \mathbf{A}\mathbf{s} + \mathbf{e} \rfloor$
- 3: $(\mathbf{s}_1, \dots, \mathbf{s}_n) \leftarrow \text{ShamirShare}(\mathbf{s})$
- 4: $(\text{sd}_1, \dots, \text{sd}_n) \leftarrow \text{SeedGen}$

Security

- Model: game-based TS-UF-X in the ROM [BTZ22]
- Assumptions: MLWE and (variants of) MSIS
- Security of base constructions
 - [EKT24] satisfies TS-UF-4 [ZT25]
 - [BKLMTT24]: weaker variant of TS-UF-0
- TBD: adaptive security

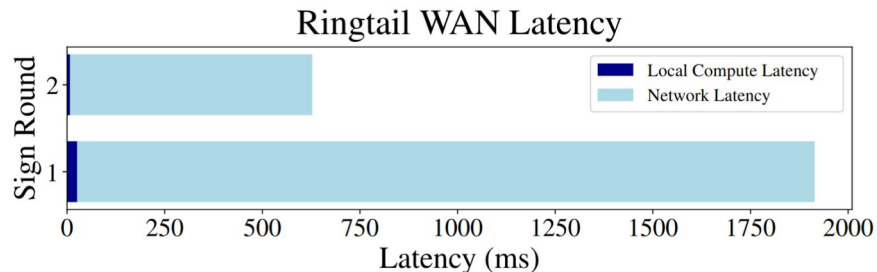
Example Instantiations

Table 1: Preliminary instantiation supporting $t \leq 1024$ and 128-bit security. Sizes are in bytes

	$ vk $	$ sig $	Sign (Total)	Sign (Online)
EKT	5,632	11,059	282,311	14,400
Ringtail	4,608	13,702	$612,864 + 16t$	10,752

Table 1: Global WAN latency (ms) for $t = 8$.

Round	Local Compute	Network Latency	Combine	End-to-End
Sign ₁	26.248	1888.147	—	1914.395
Sign ₁	7.497	620.513	0.173	628.183



Preliminary Experiments

- Implemented in Go with Lattigo library
- WAN experiments with AWS c5.4xlarge
- 8 regions across 5 continents
- Average RTT: 170.11ms
- Average network throughput: 183.23 Mbps
- See eprint 2024/1113 and

<https://github.com/daryakaviani/ringtail>

Future Directions

- Reimplement the core modules in Rust
- Update parameters
- More network experiments

Conclusion

- Two-round threshold signature scheme with 1 offline + 1 online rounds
 - Reasonable signature sizes and communication
- vs. **threshold ML-DSA (Mithril and Quorus)**
 - Raccoon enables highly efficient and scalable threshold signatures with no MPC subroutines
 - At the cost of larger keys and signatures
- vs. **Hermine** (next talk!)
 - Both are based on the Raccoon+FROST paradigm
 - Tanuki: one-time masks + standard Shamir \Rightarrow enables **larger threshold** ~ 1024 **without IA**
 - Hermine: short secret sharing \Rightarrow easy to realize **IA** with **smaller thresholds**
- Ongoing
 - Reference implementation in Rust
 - New parameter sets with refined security analysis

References

- [EKT24] T. Espitau, S. Katsumata, and K. Takemure. Two-round threshold signature from algebraic one-more learning with errors. *CRYPTO 2024*. <https://ia.cr/2024/496>
- [BKLMTT24] C. Boschini, D. Kaviani, R. W.F. Lai, G. Malavolta, A. Takahashi, M. Tibouchi. Ringtail: Practical Two-Round Threshold Signatures from Learning with Errors. *Oakland 2025*. <https://ia.cr/2024/1113>
- [KG20] C. Komlo and I. Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. SAC 2020.
- [ZT25] C. Zhu and S. Tessaro. The Algebraic One-More MISIS Problem and Applications to Threshold Signatures. *CRYPTO 2025*.
- [dPK+24] R. del Pino, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, M. Saarinen. Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions. Eurocrypt 2024.
- [dPKPR24] R. del Pino, S. Katsumata, T. Prest, M. Rossi. Raccoon: A Masking-Friendly Signature Proven in the Probing Model. *CRYPTO 2024*.
- [BTZ22] M. Bellare, S. Tessaro, C. Zhu. Stronger Security for Non-Interactive Threshold Signatures: BLS and FROST. *CRYPTO 2022*.