"Preview Writeup": In anticipation of a package submission to the NIST Threshold Call

Title: Tanuki

Subtitle: Two-round Threshold Signatures from Lattices

Version: 0.1 (2026-01-21)[1]

Team name: Tanuki Team: Tanuki Team

Team members: 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

Abstract: Tanuki is a novel lattice-based threshold signature scheme based on the Raccoon scheme. It features a two-round signing protocol with preprocessing, compact signatures and verification keys, and achieves unforgeability in the dishonest majority setting under the random oracle model. The scheme is designed to provide post-quantum security. In this preview writeup, we provide a technical overview of Tanuki and outline our plan to submit a package for consideration in the NIST Threshold Call.

Proposed crypto-systems: Signing (Category S1)

Keywords: Threshold Cryptography; NIST Threshold Call

[1] Preliminary version submitted to NIST-MPTC for review

Preview writeup. This document is provided to NIST for online publication, to foster public awareness and support public discussion within the scope of the NIST First Call for Multi-Party Threshold Schemes [NIST-IR8214C]. This "preview writeup" represents a good-faith plan for a subsequent "package submission". However, until the deadline for package submission, the team may still modify its own composition and the submission plan, including possible changes to the technical scope, and/or the used techniques or achieved results.

Team members: Cecilia Boschini [i1,a1], Thomas Espitau [i2,a2], Aaron Kaiser [i3,a7], Shuichi Katsumata [i4,a2,a3], Darya Kaviani [i5,a9], Russell W.F. Lai [i6,a6], Giulio Malavolta [i7,a5], Thomas Prest [i8,a2], Peter Schwabe [i9,a7,a10], Akira Takahashi [i10,a4], Kaoru Takemure [i11,a2,a3], Mehdi Tibouchi [i12,a8]

Open Researcher and Contributor Identifiers (ORCID):

i1 (0000-0003-0956-1616); i2 (0000-0002-7655-9594); i3 (0009-0004-6141-4861); i4 (0000-0002-8496-0476); i5 (0000-0002-1458-0769); i6 (0000-0001-9126-1887); i7 (0009-0009-9737-0094); i8 (0000-0003-1445-6212); i9 (0000-0002-1310-0997); i10 (0000-0001-8556-3053); i11 (0000-0002-9288-1911); i12 (0000-0002-2736-2963)

Affiliations:

[a1] ETH Zurich

[a2] PQShield

[a3] National Institute of Advanced Industrial Science and Technology

[a4] J.P. Morgan Chase & Co.

[a5] Bocconi University

[a6] Aalto University

[a7] Max Planck Institute for Security and Privacy

[a8] NTT

[a9] University of California, Berkeley

[a10] Radboud University Nijmegen

Main contacts:

- **Team mailing list:** tanuki-threshold@googlegroups.com
- **Primary technical contact person:** Akira Takahashi, takahashi.akira.58s@gmail.com
- **Secondary contact person 1:** Kaoru Takemure, kaoru.takemure@pqshield.com
- **Secondary contact person 2:** Russell W.F. Lai, russell.lai@aalto.fi

Produced by humans. The team hereby confirms that the content in this preview writeup: (i) was produced by the team members, and (ii) was not produced by generative artificial intelligence (AI), with the possible exception of AI-proposed grammar improvements, minor integrated suggestions, or some well-identified and short localized portions of auxiliary content (e.g., some illustration); and (iii) was proofread by the team members.

1. Introduction

In this submission, we propose Tanuki, a post-quantum threshold signature scheme based on lattices. Tanuki builds upon the interactive signing process of the EKT scheme [EKT24] and leverages the codebase of Ringtail [BKLMTT25]. Both of these prior works provide scalable distributed signing protocols that produce Raccoon signatures [DKMMPS24; PKPR24]—a class of lattice-based Fiat-Shamir signatures specifically designed to be masking- and threshold-friendly. In summary, Tanuki offers the following key features:

- **Two-round signing protocol with preprocessing:** Tanuki features a two-round signing protocol that enhances efficiency and reduces latency in wide-area network (WAN) settings. This round complexity is nearly optimal for Fiat-Shamir-based threshold signatures, as thresholdization of the Fiat-Shamir paradigm requires at least one round of interaction to collect and aggregate commitments before producing a challenge. Moreover, the first round of communication can be securely preprocessed offline, allowing signers to exchange their first-round protocol messages before knowing the message to be signed or the signing set, which further reduces online signing latency. This design is particularly beneficial in scenarios where rapid signature generation is critical.
- **Compact verification keys and signatures:** Thanks to the underlying Raccoon scheme, Tanuki produces compact signatures that remain verifiable by the Raccoon-like verification process, making it suitable for resource-constrained environments.
- **Scalability to large number of signers:** Similar to TRaccoon [DKMMPS24], Tanuki is designed to support up to 1024 signers, addressing the needs of large-scale distributed systems and applications.
- **Unforgeability based on well-studied lattice assumptions:** Thanks to the rigorous security analysis of Raccoon as well as recent progress in the literature [EKT24; BKLMTT25; ZT25], Tanuki inherits unforgeability guarantees in the random oracle model based on the hardness of commonly used variants of the Module Short Integer Solution (MSIS) and Module Learning with Errors (MLWE) problems, which are well-studied lattice assumptions believed to be resistant against quantum attacks.

The two-round structure of Tanuki (including a single online round) is the main aspect that differentiates it from TRaccoon, which is a three-round protocol. This is achieved at the cost of somewhat higher communication (especially offline), and more involved security arguments. The performance profile of Tanuki, the underlying security assumptions, etc., are otherwise fairly similar to TRaccoon.

2. Specification

2.1. Organization

In the upcoming submission, we plan to modularize the specification document as follows.

Table 1: Parameters for our threshold signature.

λ	Security parameter
n	Number of signing parties
t	Threshold number of signing parties
$\mathcal{R}$	2φ-th cyclotomic ring $\mathcal{R} = \mathbb{Z}[\zeta] \cong \mathbb{Z}[X]/\langle X^{\varphi}+1\rangle$
$\mathcal{C}$	Challenge space $\{c \in \mathcal{R} \mid \|c\|_{\infty} = 1 \wedge \|c\| = \omega\}$ for Fiat-Shamir transformation
φ	Degree of $\mathcal{R}$ (a power of 2)
q	Ring modulus defining $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$
(k,ℓ)	Dimension of the public matrix $\mathrm{A} \in \mathcal{R}_q^{k\times\ell}$
rep	Width of the first-round message $\mathrm{W}_i \in \mathcal{R}_q^{k\times \mathsf{rep}}$
ν_{t}	Number of low bits discarded from t
ν_{w}	Number of low bits discarded from w
$q_{\nu_{\mathrm{t}}}$	$q_{\nu_{\mathrm{t}}} = q/2^{\nu_{\mathrm{t}}}$, used in verification key rounding
$q_{\nu_{\mathrm{w}}}$	$q_{\nu_{\mathrm{w}}} = q/2^{\nu_{\mathrm{w}}}$, used in commitment rounding
B	ℓ_2-norm bound for a valid signature $(\mathrm{z}, 2^{\nu_{\mathrm{w}}}\cdot\mathrm{h})$

Core Modules: Following the usual syntax of interactive threshold signatures proven secure in the game based model, we specify the core modules of Tanuki as a collection of functions, denoted as $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}_1, \mathsf{Sign}_2, \mathsf{Combine}, \mathsf{Verify})$. In Table 1, we list the parameters used in Tanuki.

KeyGen takes as input the number of signers n and the threshold t, and outputs a verification key vk and a set of signing keys $(\mathsf{sk}_i)_{i\in[n]}$ for the n signing parties. In Tanuki, it internally samples a public matrix $\mathrm{A} \in \mathcal{R}_q^{k\times\ell}$, a secret vector $\mathrm{s} \in \mathcal{R}_q^{\ell}$, and an error vector $\mathrm{e} \in \mathcal{R}_q^{k}$ to form the verification key $\mathsf{vk} = (\mathrm{A}, \mathrm{t})$ where $\mathrm{t} = \lfloor 2\cdot(\mathrm{As}+\mathrm{e})\rceil_{\nu_{\mathrm{t}}}$ is a rounded verification key mapped to $\mathcal{R}_{q_{\nu_{\mathrm{t}}}}$, and uses Shamir secret sharing to split s into shares $(\mathrm{s}_i)_{i\in[n]}$. Each signer's signing key sk_i consists of its share s_i along with vk and pairwise seeds $(\mathsf{sd}_{i,j}, \mathsf{sd}_{j,i})_{j\in[n]}$ for generating one-time random masks during signing.

Sign_1 is executed by any party $i \in [n]$. It takes a signing key sk_i as input and outputs a first-round protocol message $\mathsf{pm}_{1,i}$ along with an internal signing state st_i. In Tanuki, Sign_1 samples random matrices R_i and E_i via SampleR and SampleE (which will be specified as part of supporting functions), computes the commitment message $\mathrm{W}_i = \mathrm{AR}_i + \mathrm{E}_i$, and stores R_i as the internal signing state st_i. We also note that st_i is a temporary value that can and should be deleted after use in Sign_2, allowing the caller to instantiate a protocol with short-lived states. While W_i is sent in the clear in the base schemes [EKT24; BKLMTT25], we are considering discarding low order bits of it in the final submission to reduce communication.

Sign_2 is executed by any signing party $i \in T \subseteq [n]$. It takes as input a signer's signing key sk_i, internal signing state st_i, the set of signers $T \subseteq [n]$ such that $|T| \geq t$, the collection of first-round protocol messages pm_1, and the message msg to be signed; it outputs a second-round protocol message $\mathsf{pm}_{2,i}$. In Tanuki, Sign_2 first obtains an aggregation vector $\mathrm{b} \in \mathcal{R}_q^{\mathsf{rep}}$ by hashing vk and the signing session ID ssid, comprised of T, $\mathrm{pm}_1 = (\mathrm{W}_j)_{j\in T}$, and msg. It then derives the

KeyGen(n, t)

$\mathrm{A} \leftarrow \mathsf{SampleA}(k, \ell)$
$\mathrm{s} \leftarrow \mathsf{SampleD}(\ell)$
$\mathrm{e} \leftarrow \mathsf{SampleD}(k)$
$(\mathrm{s}_1, \ldots, \mathrm{s}_n) \leftarrow \mathsf{Share}(n, t, \mathrm{s})$
$\mathrm{t} \leftarrow \lfloor 2 \cdot (\mathrm{As} + \mathrm{e}) \rceil_{\nu_\mathrm{t}}$
$\mathsf{vk} \leftarrow (\mathrm{A}, \mathrm{t})$
for $i \in [n], j \in [n]$ **do**
    $\mathsf{sd}_{i,j} \leftarrow \{0,1\}^\lambda$
for $i = 1$ to n **do**
    $\mathsf{sk}_i \leftarrow (\mathrm{s}_i, \mathsf{vk}, (\mathsf{sd}_{i,j}, \mathsf{sd}_{j,i})_{j \in [n]})$
return $(\mathsf{vk}, (\mathsf{sk}_i)_{i \in [n]})$

Combine$(\mathsf{vk}, T, \mathsf{msg}, \mathrm{pm}_1, \mathrm{pm}_2)$

$(\mathrm{W}_j)_{j \in T} \leftarrow \mathrm{pm}_1$
$(\mathrm{z}_j)_{j \in T} \leftarrow \mathrm{pm}_2$
$\mathsf{ssid} \leftarrow (T, \mathrm{pm}_1, \mathsf{msg})$
$\mathrm{b} \leftarrow \mathsf{G}(\mathsf{vk}, \mathsf{ssid})$
$\mathrm{W} \leftarrow \sum_{j \in T} \mathrm{W}_j$
$\mathrm{w} \leftarrow \lfloor \mathrm{Wb} \rceil_{\nu_\mathrm{w}}$
$c \leftarrow \mathsf{H}(\mathsf{vk}, \mathrm{w}, \mathsf{msg})$
$\mathrm{z} \leftarrow \sum_{j \in T} \mathrm{z}_j$
$\mathrm{h} \leftarrow \mathrm{w} - \lfloor \mathrm{Az} - 2^{\nu_\mathrm{t}} \cdot c \cdot \mathrm{t} \rceil_{\nu_\mathrm{w}} \bmod q_{\nu_\mathrm{w}}$
$\sigma \leftarrow (c, \mathrm{z}, \mathrm{h})$
return σ

Sign$_1(\mathsf{sk}_i)$

$\mathrm{R}_i \leftarrow \mathsf{SampleR}(\ell, \mathsf{rep})$
$\mathrm{E}_i \leftarrow \mathsf{SampleE}(k, \mathsf{rep})$
$\mathrm{W}_i \leftarrow \mathrm{AR}_i + \mathrm{E}_i \in \mathcal{R}_q^{k \times \mathsf{rep}}$
$\mathrm{pm}_{1,i} \leftarrow \mathrm{W}_i$
$\mathsf{st}_i \leftarrow \mathrm{R}_i$
return $(\mathrm{pm}_{1,i}, \mathsf{st}_i)$

Sign$_2(\mathsf{sk}_i, \mathsf{st}_i, T, \mathrm{pm}_1, \mathsf{msg})$ // assuming $|T| \geq t$

$(\mathrm{W}_j)_{j \in T} \leftarrow \mathrm{pm}_1$
$\mathrm{R}_i \leftarrow \mathsf{st}_i$
$\mathsf{ssid} \leftarrow (T, \mathrm{pm}_1, \mathsf{msg})$
$\mathrm{b} \leftarrow \mathsf{G}(\mathsf{vk}, \mathsf{ssid})$
$\mathrm{W} \leftarrow \sum_{j \in T} \mathrm{W}_j$
$\mathrm{w} \leftarrow \lfloor \mathrm{Wb} \rceil_{\nu_\mathrm{w}}$
$c \leftarrow \mathsf{H}(\mathsf{vk}, \mathrm{w}, \mathsf{msg})$
$\mathrm{m}_i \leftarrow \sum_{j \in T} \mathsf{PRF}(\mathsf{sd}_{i,j}, \mathsf{ssid})$
$\mathrm{m}'_i \leftarrow \sum_{j \in T} \mathsf{PRF}(\mathsf{sd}_{i,j}, \mathsf{ssid})$
$\mathrm{z}_i \leftarrow c \cdot \lambda_{T,i} \cdot \mathrm{s}_i + \mathrm{R}_i \mathrm{b} + \mathrm{m}'_i - \mathrm{m}_i \bmod q$
$\mathrm{pm}_{2,i} \leftarrow \mathrm{z}_i$
return $\mathrm{pm}_{2,i}$

Verify$(\mathsf{vk}, \mathsf{msg}, c, \mathrm{z})$

$\mathrm{w}' \leftarrow \lfloor \mathrm{Az} - 2^{\nu_\mathrm{t}} \cdot c \cdot \mathrm{t} \rceil_{\nu_\mathrm{w}} + \mathrm{h} \bmod q_{\nu_\mathrm{w}}$
assert $c = \mathsf{H}(\mathsf{vk}, \mathrm{w}', \mathsf{msg})$
assert $\|(\mathrm{z}, 2^{\nu_\mathrm{w}} \cdot \mathrm{h})\|_2 \leq B$

Figure 1: Draft of the core modules for our two-round threshold signature scheme.

Fiat-Shamir challenge $c \in \mathcal{C}$ by hashing vk, the aggregated commitment $\mathrm{w} = \lfloor \sum_{j \in T} \mathrm{W}_j \mathrm{b} \rceil_{\nu_\mathrm{w}}$, and msg. Finally, it computes the signature share $\mathrm{z}_i = c \cdot \lambda_{T,i} \cdot \mathrm{s}_i + \mathrm{R}_i \mathrm{b} + \mathrm{m}'_i - \mathrm{m}_i$ and outputs $\mathrm{pm}_{2,i} = \mathrm{z}_i$, where $\lambda_{T,i} = \prod_{j \in T \setminus \{i\}} \frac{-i}{j-i}$ is the Lagrange coefficient for party i with respect to the signing set T, and m_i and m'_i are one-time random masks generated by applying PRF to the pairwise seeds in sk_i and ssid.

Combine is executed by any party $i \in [n]$ or the designated coordinator node. It takes as input a verification key vk, a set of signers T, a message msg, the collection of first-round protocol messages pm_1, and the collection of second-round protocol messages pm_2; it outputs a signature σ. In Tanuki, it computes Fiat-Shamir challenge c as in Sign$_2$, and then aggregates the signature shares $\mathrm{z} = \sum_{j \in T} \mathrm{z}_j$. It finally computes the hint vector $\mathrm{h} = \mathrm{w} - \lfloor \mathrm{Az} - 2^{\nu_\mathrm{t}} \cdot c \cdot \mathrm{t} \rceil_{\nu_\mathrm{w}} \bmod q_{\nu_\mathrm{w}}$ and outputs the signature $\sigma = (c, \mathrm{z}, \mathrm{h})$.

Verify takes as input a verification key vk, a message msg, and a signature σ. In Tanuki, it first derives Fiat-Shamir challenge c' by hashing vk, the reconstructed commitment $\mathrm{w} =$

$\lfloor \mathrm{Az} - 2^{\nu_\mathrm{t}} \cdot c \cdot \mathrm{t} \rceil_{\nu_\mathrm{w}} + \mathrm{h}$, and msg. It then checks whether $\|(\mathrm{z}, 2^{\nu_\mathrm{w}} \cdot \mathrm{h})\|_2 \leq B$ and returns true if the check passes and the derived challenge c' matches the challenge c in σ; otherwise, it returns false. Note that this verification procedure is the same as that of the underlying (non-threshold) Raccoon scheme.

Supporting functions originated from TRaccoon **[DKMMPS24]:** Since Tanuki shares some design principles with TRaccoon, we will reuse several supporting functions defined as part of TRaccoon as follows.

$\underline{\mathsf{PRF}(\mathsf{sd}, \mathsf{ssid})}$ is a pseudorandom function that takes as input a seed sd and ssid, and outputs a uniformly random vector in $\mathcal{R}_q^\ell$.

$\underline{\mathsf{H}}$ is a hash function that maps its inputs to a challenge in $\mathcal{C}$.

$\underline{\lfloor \cdot \rceil_{\nu_\mathrm{t}}}$ is a rounding function that takes as input a vector in $\mathcal{R}_q^k$ and outputs a vector in $\mathcal{R}_{q_{\nu_\mathrm{t}}}^k$.

$\underline{\lfloor \cdot \rceil_{\nu_\mathrm{w}}}$ is a rounding function that takes as input a vector in $\mathcal{R}_q^k$ and outputs a vector in $\mathcal{R}_{q_{\nu_\mathrm{w}}}^k$.

$\underline{\mathsf{Share}(n, t, \mathrm{s})}$ is a Shamir secret sharing function that takes as input the number of signers n, the threshold t, and a secret vector $\mathrm{s} \in \mathcal{R}_q^\ell$, and outputs n shares $(\mathrm{s}_i)_{i \in [n]}$. Note that it is defined over the non-field $\mathcal{R}_q$, and thus we set the modulus q such that $i - j$ is invertible modulo q for any distinct $i, j \in [n]$ to ensure that the Lagrange coefficients are well-defined.

Supporting functions unique to Tanuki**:** In addition to the core modules and supporting functions from TRaccoon, Tanuki also defines several supporting functions enabling its two-round signing protocol as follows.

$\underline{\mathsf{SampleA}(k, \ell)}$ samples a uniformly random matrix $\mathrm{A} \in \mathcal{R}_q^{k \times \ell}$.

$\underline{\mathsf{SampleD}(d)}$ samples a vector in $\mathcal{R}^d$ with small coefficients.

$\underline{\mathsf{SampleR}(\ell, \mathsf{rep})}$ samples a random matrix $\mathrm{R} \in \mathcal{R}_q^{\ell \times \mathsf{rep}}$ with small coefficients.

$\underline{\mathsf{SampleE}(k, \mathsf{rep})}$ samples a random matrix $\mathrm{E} \in \mathcal{R}_q^{k \times \mathsf{rep}}$ with small coefficients.

$\underline{\mathsf{G}}$ is a hash function that maps its inputs to an aggregation vector $\mathrm{b} \in \mathcal{R}_q^{\mathsf{rep}}$ with small coefficients.

Distributions of signing keys, errors, randomness and aggregation vectors: By the time of the final submission, we will specify in detail the distributions used for sampling the signing key s, error e, randomness $(\mathrm{R}_i, \mathrm{E}_i)$ in signing, and aggregation vector b. These choices will be informed by future developments in the academic literature as well as our own further analysis and experimentation. A plausible choice for s, e, R_i, and E_i is the discrete Gaussian distribution over $\mathcal{R}$ as in [EKT24; BKLMTT25], which enables us to leverage the Hint-MLWE-to-MLWE reduction [KLSS23]. However, we are also exploring alternative distributions, such as the sum of uniform distribution over sets of small-norm polynomials, to improve efficiency and side-channel resilience. For the coefficients of b, we plan to fix its first coefficient to 1 and sample each of the remaining coefficients uniformly from the set of signed monomials $\{\pm 1, \pm X, \ldots, \pm X^{\varphi-1}\}$, as in [EKT24].

Serialization and deserialization: To convert the various data structures used in Tanuki into byte strings for network transmission and storage, we plan to define serialization and deserialization functions for keys, protocol messages, and signatures.

2.2. System model

Threshold profiles: We plan to support flexible (t, n)-threshold signing for any $1 < n \leq 1024$ and $1 \leq t \leq n$, allowing users to choose parameters that best suit their security and performance requirements. The signature size is expected to only increase modestly regardless of the supported threshold size and party number. We use Shamir's secret sharing over $\mathcal{R}_q$ to distribute the signing key. The unforgeability of Tanuki holds under a dishonest majority.

Trusted setup: We assume a trusted setup phase where a trusted party generates the public parameters and executes KeyGen to produce the signing keys and public verification key. Although the design of a DKG protocol is outside the scope of the current submission, we plan to include a list of candidate DKG protocols that can serve as drop-in replacements for KeyGen in the final submission.

Network model: The unforgeability of Tanuki only requires asynchronous, unauthenticated, and peer-to-peer communication channels between parties. To retain liveness under adversarially controlled network conditions, we assume a synchronous network model with authenticated channels where messages sent between honest parties are eventually delivered without modification.

2.3. Security

Formulation of security: We plan to prove that Tanuki is unforgeable against adaptive chosen-message attacks in the random oracle model, assuming that the adversary can corrupt up to $t - 1$ signing parties. Formally, we guarantee that it satisfies the TS-UF-0 security notion defined in [BTZ22], and we will discuss the achievability of the stronger notion TS-UF-4. While the scheme is currently proven secure under static corruption, we are actively working on extending the proof to handle adaptive corruption.

Assumptions and security strength estimation: The unforgeability of Tanuki will be based on the security of the PRF and the hardness of MLWE and commonly used variants of MSIS problems. We will provide a detailed security analysis in the final submission, including parameter choices that achieve various security levels (e.g., 128-bit, 192-bit, and 256-bit security) against known attacks on MLWE and MSIS problems.

Other security properties: In addition to unforgeability, we will discuss further security properties relevant to Tanuki, including unforgeability under adaptive corruptions, identifiable aborts, strong unforgeability, and resistance to side-channel attacks, together with potential mitigation strategies. Achieving these additional properties may require supplementary protocol modules, stronger network assumptions, or impose restrictions on the supported threshold profiles.

3. Implementation, Performance Evaluation, Licensing, and Patent Claims

Code structure: We plan to implement the core modules and supporting functions mentioned in Section 2 in Rust. Our implmentation will depend on the `sha3` crate for hash functions.[2] Since the Sign_1 algorithm is randomized, we plan to modularize it by separating `sign1_internal` that takes as input the random bytes used for sampling $(\mathrm{R}_i, \mathrm{E}_i)$, and a wrapper function `sign1` that generates the random bytes before calling `sign1_internal`.

Code progress and availability: A proof-of-concept Go implementation is publicly available at https://github.com/daryakaviani/ringtail, which implements Ringtail [BKLMTT25], a close variant of Tanuki. By the time of the final submission, we plan to release a complete and optimized implementation of Tanuki in Rust with a faster polynomial arithmetic backend, more efficient samplers for both ephemeral randomness $(\mathrm{R}_i, \mathrm{E}_i)$ and aggregation vector b, and optimized wrapper modules for concurrency management and network communication.

Implementation of the networking model: As Tanuki is an interactive protocol executed by multiple parties over a network, we plan to provide wrapper modules that manage multiple concurrent signing sessions and communication over standard networking channels.

Testing: We provide Known Answer Tests (KATs) for each core module and supporting function to verify the correctness of our implementation, by fixing msg, T, and the random bytes for sampling all randomness used in the protocol.

Performance evaluation: The aforementioned Go implementation of Ringtail demonstrates practical signing latency and communication for up to $(n, t) = (8, 8)$ in a WAN setting. In [BKLMTT25], they report the end-to-end signing latency is about 2.5 seconds, including network delays, when the signing protocol is executed among 8 AWS `c5.4xlarge` instances across 5 continents, with 16 vCPUs and 32 GB of memory.

Licensing and Patent Claims: The code of our submission will be licensed under the Apache License, Version 2.0. Known patents that may potentially cover the contents of the submission includes [MKPPME].

Disclaimer

[2] https://docs.rs/sha3/latest/sha3/

References

[BKLMTT25] Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. "Ringtail: Practical Two-Round Threshold Signatures from Learning with Errors". In: *2025 IEEE Symposium on Security and Privacy*. Ed. by Marina Blanton, William Enck, and Cristina Nita-Rotaru. IEEE Computer Society Press, May 2025, pp. 149–164. DOI: 10.1109/SP61157.2025.00070. Also at ia.cr/2024/1113.

[BTZ22] Mihir Bellare, Stefano Tessaro, and Chenzhi Zhu. *Stronger Security for Non-Interactive Threshold Signatures: BLS and FROST*. Cryptology ePrint Archive, Report 2022/833. 2022. URL: https://eprint.iacr.org/2022/833.

[DKMMPS24] Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. "Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions". In: *EUROCRYPT 2024, Part II*. Ed. by Marc Joye and Gregor Leander. Vol. 14652. LNCS. Springer, Cham, May 2024, pp. 219–248. DOI: 10.1007/978-3-031-58723-8_8. Also at ia.cr/2024/184.

[EKT24] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. "Two-Round Threshold Signature from Algebraic One-More Learning with Errors". In: *CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. LNCS. Springer, Cham, August 2024, pp. 387–424. DOI: 10.1007/978-3-031-68394-7_13. Also at ia.cr/2024/496.

[KLSS23] Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. "Toward Practical Lattice-Based Proof of Knowledge from Hint-MLWE". In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. LNCS. Springer, Cham, August 2023, pp. 549–580. DOI: 10.1007/978-3-031-38554-4_18. Also at ia.cr/2023/623.

[MKPPME] Fabrice Mouhartem, Shuichi Katsumata, Thomas Prest, Rafael Del Pino, Mary Maller, and Thomas Espitau. *Lattice-based threshold signature method and threshold decryption method*. URL: https://patents.google.com/patent/WO2024228005A1/.

[PKPR24] Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. "Raccoon: A Masking-Friendly Signature Proven in the Probing Model". In: *CRYPTO 2024, Part I*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14920. LNCS. Springer, Cham, August 2024, pp. 409–444. DOI: 10.1007/978-3-031-68376-3_13. Also at ia.cr/2024/1291.

[ZT25] Chenzhi Zhu and Stefano Tessaro. "The Algebraic One-More MISIS Problem and Applications to Threshold Signatures". In: *CRYPTO 2025, Part I*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16000. LNCS. Springer, Cham, August 2025, pp. 548–581. DOI: 10.1007/978-3-032-01855-7_18. Also at ia.cr/2025/436.

[NIST-IR8214C] Luís T. A. N. Brandão and René Peralta. *NIST First Call for Multi-Party Threshold Schemes*. (National Institute of Standards and Technology) NIST Internal Report (NISTIR) 8214C. 2025. DOI: 10.6028/NIST.IR.8214C.