Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions

Rafael del Pino[1], Shuichi Katsumata[1,2], Mary Maller[1,3], Fabrice Mouhartem[4], Thomas Prest[1], Markku-Juhani Saarinen[1,5]

[1]PQShield

rafael.del.pino@pqshield.com,shuichi.katsumata@pqshield.com,

mary.maller@pqshield.com,thomas.prest@pqshield.com

[2]AIST

[3]Ethereum Foundation

[4]XWiki/CryptPad

fabrice.mouhartem@xwiki.com

[5]Tampere University

markku-juhani.saarinen@tuni.fi

7th February 2024

Abstract

Threshold signatures improve both availability and security of digital signatures by splitting the signing key into N shares handed out to different parties. Later on, any subset of at least T parties can cooperate to produce a signature on a given message. While threshold signatures have been extensively studied in the pre-quantum setting, they remain sparse from quantum-resilient assumptions.

We present the first efficient lattice-based threshold signatures with signature size 13 KiB and communication cost 40 KiB per user, supporting a threshold size as large as 1024 signers. We provide an accompanying high performance implementation. The security of the scheme is based on the same assumptions as Dilithium, a signature recently selected by NIST for standardisation which, as far as we know, cannot easily be made threshold efficiently.

All operations used during signing are due to symmetric primitives and simple lattice operations; in particular our scheme does not need heavy tools such as threshold fully homomorphic encryption or homomorphic trapdoor commitments as in prior constructions. The key technical idea is to use *one-time additive masks* to mitigate the leakage of the partial signing keys through partial signatures.

Contents

1 Introduction

A threshold signature scheme [Des90, DF90] is a specific type of multiparty computation that aims at issuing digital signatures, for which any subset of T parties among N signers are able to sign a message, but $(T-1)$ cannot. This ability to distribute trust among several parties has sparked widespread interest from the blockchain ecosystem, and also from governmental bodies such as NIST [PB23].

In the pre-quantum world, there exist practical threshold signature solutions [Bol03, LJY14, KG20, BCK+22, CKM21, Lin22, RRJ+22, ANO+22, Sho00] based on Schnorr, ECDSA, RSA or BLS [BLS01] signatures. They have reached a high level of maturity and often satisfy advanced functionality and security features such as robustness, identifiable-aborts, small round complexity and backward compatibility with existing constructions.

Unfortunately, there are far fewer *post-quantum* threshold signature schemes. Some solutions have been proposed, but few have been implemented, and those who have suffer from major inefficiencies, such as large signatures, slow signing times, and sometimes both. More details are provided in Section 1.2.

In January 2023, the US agency NIST released a call for multi-party threshold schemes [PB23]. Quantum resistance is repeatedly listed as an important criterion [PB23, Sections 3.2 and 3.3], with a deadline for submissions expected for the first half of 2024. The exploratory state of post-quantum threshold signatures stands in stark contrast with the timeline of NIST's call for proposals. It also contrasts with the comparative maturity of standard post-quantum signatures; NIST has standardised two post-quantum (statefull) signatures [CAD+20], has announced the future standardisation of three other (stateless) signatures [CAD+20, AAC+22], and recently, in June 2023, had an additional call for proposals [NIS22].

The focus of this work is thus to construct a practically efficient post-quantum threshold signature and to close the gap between the classical setting.

1.1 Our Contributions

We propose TRaccoon: a practical three-round lattice-based threshold signature assuming the hardness of the MLWE and MSIS problems.[1] Our threshold signature is based on a variant of the Lyubashevsky's (non-thresholdised) signature scheme [Lyu09, Lyu12] and departs from prior constructions that utilise heavy cryptographic tools such as threshold fully homomorphic encryption (FHE) or trapdoor homomorphic commitments [BGG+18, DOTT21, DOTT22, ASY22]. Moreover, it can be viewed as a thresholdised version of Raccoon [dPEK+23], a lattice-based signature scheme by del Pino et al., submitted to the additional NIST call for proposals [NIS22] (see Section 2 for more detail). In particular, the verification algorithm of both schemes is identical. Meaning that if we were to parametrize TRaccoon with compatibility in mind, obtaining a signature that can be used as a signature of Raccoon is feasible.[2] This is a desirable property for threshold signatures as it allows to seamlessly use TRaccoon in an ecosystem that relies on Raccoon.

At a birds eye's view, we follow the folklore construction of a three-round threshold signature from Schnorr's signature scheme [Sch90, Sch91] and Shamir's secret sharing protocol [Sha79a]. However, as it is well-known in the lattice community, a naive translation does not work since, unlike in the classical setting, the signing key and signatures must satisfy additional size constraints. Indeed, the folklore construction ported to the lattice setting would leak too much information about the (distributed) signing keys and lead to practical attacks. The key technical ingredient we use to mitigate this leakage is the use of pairwise *one-time additive masks* that are *non-interactively* shared between each pair of users at each signing procedure and recombined in a way that allows individual users to hide their response while preserving correctness. More details are provided in Section 2.

We complement our theoretical design by providing concrete parameters, along with a high performance implementation. For example, for a bit-security level $\kappa = 128$, a number of signers $T = 1024$[3], and a

[1] More precisely, we rely on the recent Hint-MLWE problem by Kim et al. [KLSS23] and the SelfTargetMSIS problem underlying the security of Dilithium [KLS18]. Both problems are known to be as hard as the MLWE and MSIS problems, respectively.

[2] We choose not to do so yet as the parameters of both schemes are still subject to improvements.

[3] This is an upper limit of the "large" requirements of NIST preliminary call for threshold [PB23]. Note that our parameters support any $1 \leq T \leq N \leq 1024$.

maximum number of signatures $Q_{\mathsf{Sign}} = 2^{60}$, our verification key and signatures sizes are about 4 and 13 KiB, respectively. In addition, creating a signature requires 116 ms of single-core computation from each signer, ignoring possible communication latencies and enabling 4.5 GHz turbo on an i7-12700. More details are provided in Tables 2 and 3, illustrating that our scheme is very efficient compared to existing works, even when broadening the scope to recent lattice-based multi-signatures [FSZ22, DOTT21, DOTT22, BTT22, Che23a]. This brings us within an order of magnitude of Dilithium, whose verification key and signatures are about 1.3 and 2.4 KiB, respectively. Finally, we note that the communication complexity of our distributed signing protocol is also very competitive; the total communication cost of producing a signature is about 40 KiB per party.

1.2 Related Works

Threshold signatures are an extremely dynamic research topic and several threshold signature schemes have been proposed in the recent years. However, most of these threshold signatures are based on Schnorr signatures, such as FROST [KG20, BCK+22] and ROAST [RRJ+22], or on ECDSA, such as [ANO+22]. As a result, they are not post-quantum. On the other hand, there exist comparatively fewer post-quantum threshold signatures. We survey the main approaches.

Lattice-based Threshold Signature. To our knowledge, the most concrete proposal of a lattice-based threshold signature is a recent work by Agrawal, Stehlé, and Yadav [ASY22]. They optimize the one-round threshold signature by Boneh et al. [BGG+18] based on threshold FHE, using a more fine-grained analysis on the noise growth using Rényi divergence [Rén61]. They further show how to turn the selectively secure scheme by Boneh et al. into a (partially) adaptive scheme. While the construction of [ASY22] is round-optimal, the use of threshold FHE will likely incur significant computation and communication overheads as they require a full FHE evaluation of a standard signature scheme.

Lattice-based Multi-Signatures. A multi-signature is a special type of N-out-of-N threshold signature that allows multiple signatures for the same message but from independent signers to be aggregated into a single signature.

Recent works have built elegant solutions for lattice-based multi-signatures [FSZ22, DOTT21, DOTT22, BTT22, Che23b]. Boschini et al. [BTT22] and Chen [Che23b] are the state-of-the-art for lattice-based multi-signatures following the Fiat-Shamir with aborts paradigm [Lyu09, Lyu12]. They construct a two-round protocol based on the MSIS and MLWE assumptions, achieving a signature size about 107 KiB and 31 KiB, respectively, for 1024 users [Che23b]. While Boschini et al. has a larger signature size, it has an appealing offline-online feature where the first round is independent of the message. Moreover, the verification algorithm of their scheme is the same as the underlying signature scheme (similarly to [BGG+18, ASY22] and ours), while Chen includes an additional commitment randomness in the signature. Fleischhacker et al. [FSZ22] introduced a multi-signature based on the ring SIS assumption that does not require interaction between the signers to aggregate the signatures at the expense of some preprocessing. They achieve a signature size of 572 KiB for 1024 users. We note that none of these multi-signature solutions can be trivially extended to threshold signatures.

Post-quantum Threshold Signatures. Besides lattices, threshold signatures based on other post-quantum families have been designed. Khaburzaniya et al. [KCLM22] proposed a STARK-based approach for aggregating and thresholdising hash-based signatures. For $N = 1024$ and a bit-security $\kappa = 128$, they report signatures of about 170 KiB with an aggregation time of 4 to 20 seconds.

The isogeny-based scheme CSI-FiSh by Beullens et al. [BKV19] has been thresholdised in two works. Cozzo and Smart [CS20] proposed a distributed key generation procedure, while de Feo and Meyer [DM20] proposed a distributed signing procedure. However, both works [CS20, DM20] are inherently sequential and the round complexity scales with the number of users. Moreover, all three works [BKV19, CS20, DM20] rely on the CSIDH cryptographic group action, whose underlying quantum security is subject to debate [BLMP19, Pei20, BS20, CSCJR22].

Concurrent and Independent Work. Very recently, Gur, Katz, and Silde [GKS23] construct a two-round lattice-based threshold signature. It is an optimisation of the threshold signature by Agrawal, Stehlé, and

Yadav [ASY22]. They minimise the computation required by the threshold FHE to only linear computation by tailoring the construction to a specific underlying signature scheme and adding one additional round. They report a signature size of 11.5 KiB and a communication cost of 1.5 MB per users for the 3-out-of-5 case. Considering that they use homomorphic trapdoor commitments, similarly to Damgård et al. [DOTT21, DOTT22], we expect the signature size to become much larger as (T, N) grows. In their work, they further propose a distributed key generation algorithm, removing the need of a trusted setup.

2 Our Techniques

Schnorr's signature scheme has been a successful tool to construct threshold signature schemes in the classical setting. Our goal is to replicate this in the post-quantum setting building on (a variant of) Lyubashevsky's signature scheme [Lyu09, Lyu12], a lattice-based signature scheme based on the Fiat-Shamir transform.

2.1 Recap: Lyubashevsky's Signature *Without* Abort

We first recall Lyubashevsky's signature scheme, the starting point for our threshold signature scheme.

Let $(\mathbf{A}, \mathbf{t})$ be the public key where $\mathbf{A} \in \mathcal{R}_q^{k \times \ell}$ is a matrix, $\mathbf{t} \in \mathcal{R}_q^k$ is a vector, and $\mathcal{R}_q$ denotes some polynomial ring. The signing key is a tuple of vectors $(\mathbf{s}, \mathbf{e}) \in \mathcal{R}_q^\ell \times \mathcal{R}_q^k$ such that $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, and $(\mathbf{s}, \mathbf{e})$ are "short." Specifically, $(\mathbf{s}, \mathbf{e})$ is a solution to the MLWE instance defined by $(\mathbf{A}, \mathbf{t})$. To sign a message msg, the signature scheme proceeds as follows:

(A.1) Sample ephemeral randomness $(\mathbf{r}, \mathbf{e}')$ from an appropriate distribution and compute a *commitment* $\mathbf{w} = \mathbf{A} \cdot \mathbf{r} + \mathbf{e}'$.

(A.2) Next, generate a *challenge* $c \in \mathcal{R}_q$ using a hash function H_c as $c \leftarrow \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$, where c has small coefficients.

(A.3) Compute a *response* $(\mathbf{z}, \mathbf{y}) = (c \cdot \mathbf{s} + \mathbf{r}, c \cdot \mathbf{e} + \mathbf{e}')$.

(A.4) Check if $(\mathbf{z}, \mathbf{y})$ satisfies some size constraints. If not, abort and restart from Item (A.1). Otherwise, output $(c, \mathbf{z}, \mathbf{y})$ as the signature.

(A.5) To verify, check that $(\mathbf{z}, \mathbf{y})$ satisfies some size constraint and that $c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{A} \cdot \mathbf{z} + \mathbf{y} - c \cdot \mathbf{t})$.

Item (A.4) is the so-called *rejection sampling* [Lyu12] step, which happens with a non-negligible probability. This is one of the key differences between Schnorr. Informally, this check ensures that the distribution of the signature does not leak information about the "short" signing key, a concern that is non-existing in classical Schnorr signatures. This has minimal effect on a standard (non-thresholdised) signature since the signer can restart locally until no rejection occurs.[4] In contrast, this becomes problematic when extending it to a threshold signature scheme. Even if some signers succeed (i.e., do not abort), in order to ensure privacy of the shared signing key, everybody must restart the distributed signing protocol in case any single signer aborts. As the signer set grows, such a restart becomes increasingly difficult to handle in practice.

To this end, we follow the common approach of adding a larger commitment noise $(\mathbf{r}, \mathbf{e}')$ to hide the signing key $(\mathbf{s}, \mathbf{e})$ [GKPV10], so that no signer aborts the signing protocol. In the context of threshold signatures such an approach was taken by Agrawal, Stehlé, and Yadav [ASY22], and more recently by Gur, Katz, and Slide [GKS23]. Both works rely on the Rényi divergence to *statistically* bound the leakage of the signing key; compared to the more standard statistical distance, it is known to provide better statistical bounds. In contrast, in our work, we rely on the *hint* MLWE ($\mathsf{Hint\text{-}MLWE}$) assumption recently introduced by Kim et al. [KLSS23] to *computationally* bound the leakage of the signing key. This results in a simpler *and* better bound compared to using the Rényi divergence, which we believe to be of an independent interest.

[4]From a security proof perspective, a proof with rejection sampling requires slightly more work due to subtle issues [DFPS23, BBD+23].

Additionally, for practical parameters, we perform several optimisations such as bit-dropping, similarly to Dilithium [DKL+18], a highly optimised signature based on Lyubashevsky's signature scheme, soon to be standardised by NIST under the name ML-DSA [NIS23]. Looking ahead, the way we perform bit-dropping is based on $\mathsf{Raccoon}$ [dPEK+23], a lattice-based signature scheme by del Pino et al., submitted to the additional NIST call for proposals [NIS22]. Their bit-dropping is more natural and easier to implement compared to Dilithium. See Section 5 for more details. Throughout this section, we ignore these optimisations for simplicity. Moreover, for ease of presentation, we refer to Lyubashevsky's signature scheme without the rejection sampling step as $\mathsf{Raccoon}$ (Item (A.4)).

2.2 Naive Extension to Lattices

Due to the similarity between Schnorr's signature scheme and $\mathsf{Raccoon}$, we can try to apply the common approach used in the classical setting [Sho00, KY02, Bol03] to build threshold signatures starting from Schnorr. Below, let us explain what a naive (insecure) (T, N)-threshold signature from $\mathsf{Raccoon}$ would look like.

We first secret share $\mathsf{Raccoon}$'s signing key $(\mathbf{s}, \mathbf{e})$ using Shamir's secret sharing scheme [Sha79b]. Namely, $\mathbf{s}$ is encoded as the constant term of a degree $T-1$ polynomial P, and the *partial* signing key of user $i \in [N]$ is defined as the evaluation $\mathbf{s}_i = P(i) \in \mathcal{R}_q^\ell$ along with a *freshly sampled* short vector $\mathbf{e}_i$. Each users' partial signing key (implicitly) defines a partial public key $\mathbf{t}_i = \mathbf{A} \cdot \mathbf{s}_i + \mathbf{e}_i$, which is an MLWE instance. Here, note that given any T partial signing keys $(\mathbf{s}_i, \mathbf{e}_i)_{i \in \mathsf{act}}$, where $\mathsf{act} \subset [N]$ and $|\mathsf{act}| = T$, we can use the Lagrange coefficients $(\lambda_{\mathsf{act},i})_{i \in \mathsf{act}}$ (see Section 3.5 for definition) to recompute the signing key as:

$$\mathbf{s} = \sum_{i \in \mathsf{act}} \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i. \tag{1}$$

For any set act of T signers, the distributed signing protocol proceeds as follows:

(B.1) User $i \in \mathsf{act}$ computes $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$. To protect against rushing adversaries it initially only outputs a *hash commitment* $\mathsf{H}_{\mathsf{com}}(\mathbf{w}_i)$.[5]

(B.2) After obtaining the hash commitment from all users in act, user i reveals $\mathbf{w}_i$ and checks the correctness of all other reveals.

(B.3) User i collects all the commitments and *locally* generates a challenge $c \leftarrow \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$, where $\mathbf{w} = \sum_{j \in \mathsf{act}} \mathbf{w}_j$.

(B.4) User i computes a response $(\mathbf{z}_i, \mathbf{y}_i) = (c \cdot \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i + \mathbf{r}_i, c \cdot \mathbf{e}_i + \mathbf{e}'_i)$ and outputs $(\mathbf{z}_i, \mathbf{y}_i)$ as its *partial* signature.

(B.5) The final signature is $(c, \mathbf{z}, \mathbf{y}) = (c, \sum_{j \in \mathsf{act}} \mathbf{z}_j, \sum_{j \in \mathsf{act}} \mathbf{y}_j)$, verified as in $\mathsf{Raccoon}$ by checking the equality $c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{A} \cdot \mathbf{z} + \mathbf{y} - c \cdot \mathbf{t})$.

A routine calculation using Eq. (1) shows that the signature is valid.

Difficulty of Handling Lagrange Coefficients.

While correct, the above construction admits an attack. This stems from the fact that Lagrange coefficients are large and can be chosen adaptively by the adversary.

In more detail, looking at Item (B.4) carefully, we can alternatively view user i as generating a signature with a signing key $(\lambda_{\mathsf{act},i} \cdot \mathbf{s}_i, \mathbf{e}_i)$. Importantly, $\mathbf{s}_i$ is scaled by the Lagrange coefficient $\lambda_{\mathsf{act},i}$. Since the user i provides a valid $\mathsf{Raccoon}$ signature — a partial signature of the threshold scheme — this allows the adversary to obtain information on the corresponding scaled partial public key $\mathbf{t}_{\mathsf{act},i} = \lambda_{\mathsf{act},i} \cdot \mathbf{A} \cdot \mathbf{s}_i + \mathbf{e}_i$. The

[5]This step prevents the adversary from maliciously generating its commitment *after* learning all the honest users commitments (see for instance [BN06]).

adversary can adaptively ask user i to sign on a scaled public key of its choice by specifying a different signer set $\mathsf{act} \subset [N]$. By collecting enough $\mathbf{t}_{\mathsf{act},i}$ with specifically crafted Lagrange coefficients $\lambda_{\mathsf{act},i}$, the partial signing key $\mathbf{s}_i$ can be recovered via simple linear algebra. In the classical setting where the noise vector $\mathbf{e}_i$ does not exist, the above attack does not apply since all the obtained scaled partial public keys are linearly dependent.

This phenomenon is not new to our work. Lattice-based cryptography has always had a hard time handling Lagrange coefficients, see for example [ABV+12, BLMR13, BGG+18]. This has led work to rely on an alternative secret sharing scheme know as the $\{0,1\}$-linear (or $\{-1,0,1\}$-linear) secret sharing scheme, see e.g., [LST18, BGG+18, DLN+21, ASY22, CSS+22, CCK23]. While this gets around the issue with Lagrange coefficients, the downside is that the reconstruction algorithm becomes much more complex and individual shares grow by at least $O(N^4)$ [Val84]. Alternatively, we can blow up the modulus size q to scale with $O(N!^2)$ to argue that large Lagrange coefficients become relatively small to q [ABV+12, BGG+18, CCK23]. However, it is clear that such an approach leads to impractical parameters. Recently, Albrecht and Lai [AL21, Section 3.1] defines the Lagrange interpolating polynomial on specific elements in $\mathcal{R}_q$ to handle the blowup more granularly, however, the concrete gain is unclear for a general T-out-of-N threshold.

2.3 Our Solution: Masking the Commitments

We sidestep all these prior hurdles by using a very simple idea, exploiting the fact that threshold signatures are *interactive*. In more detail, assume for now that every two pairs of users $i, j \in \mathsf{act}$ privately share two *one-time random masks* $(\mathbf{m}_{\mathsf{act},i,j}, \mathbf{m}_{\mathsf{act},j,i}) \in (\mathcal{R}_q^\ell)^2$. We modify the signing protocol of the naive threshold signature scheme as follows:

(C.1) User $i \in \mathsf{act}$ computes a commitment $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$, a (public) *row* mask $\mathbf{m}_{\mathsf{act},i} = \sum_{j\in\mathsf{act}} \mathbf{m}_{\mathsf{act},i,j}$, and outputs $(\mathsf{H}_{\mathsf{com}}(\mathbf{w}_i), \mathbf{m}_{\mathsf{act},i})$.[6]

(C.2) After obtaining the hash commitment and row mask from all users in act, user i reveals $\mathbf{w}_i$.

(C.3) User i collects all the commitments and locally generate a challenge $c := \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$, where $\mathbf{w} = \sum_{i\in\mathsf{act}} \mathbf{w}_i$.

(C.4) User i computes a (private) *column* mask $\mathbf{m}^*_{\mathsf{act},i} = \sum_{j\in\mathsf{act}} \mathbf{m}_{\mathsf{act},j,i}$ and response $(\mathbf{z}_i, \mathbf{y}_i) = (c \cdot \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i + \mathbf{r}_i + \mathbf{m}^*_{\mathsf{act},i}, c \cdot \mathbf{e}_i + \mathbf{e}'_i)$, and outputs $(\mathbf{z}_i, \mathbf{y}_i)$ as its partial signature.

(C.5) The final signature is $(c, \mathbf{z}, \mathbf{y}) = (c, \sum_{j\in\mathsf{act}}(\mathbf{z}_j - \mathbf{m}_{\mathsf{act},j}), \sum_{j\in\mathsf{act}} \mathbf{y}_j)$ and is verified as in Item (A.5).

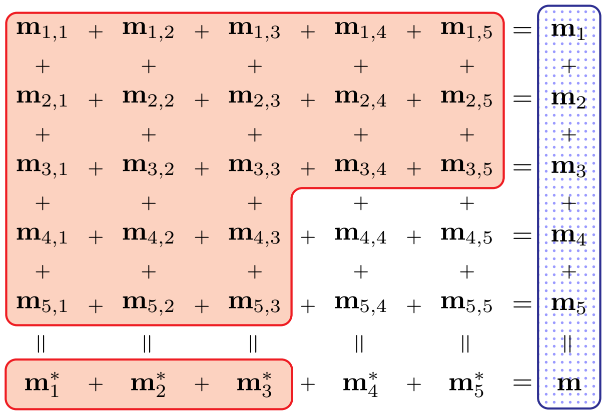
Notice the sum of the row masks and column masks are equal: $\sum_{j\in\mathsf{act}} \mathbf{m}_{j,\mathsf{act}} = \sum_{j\in\mathsf{act}} \mathbf{m}^*_{j,\mathsf{act}}$. When all the users are honest, it can be checked that the aggregated response becomes $\mathbf{z} = \sum_{j\in\mathsf{act}}(\mathbf{z}_j - \mathbf{m}_{\mathsf{act},j}) = \sum_{j\in\mathsf{act}}(c \cdot \lambda_{\mathsf{act},j} \cdot \mathbf{s}_j + \mathbf{r}_j + (\mathbf{m}^*_{\mathsf{act},j} - \mathbf{m}_{\mathsf{act},j})) = c \cdot \mathbf{s} + \sum_{j\in\mathsf{act}} \mathbf{r}_j$, a Raccoon signature as desired (see Item (A.3)).

Intuition of the Security Proof.

A typical security proof of a Lyubashevsky signature consists of invoking honest-verifier zero-knowledge of the (implicit) underlying identification protocol and programming the random oracle. At a high level, the reduction first samples a challenge c and response $(\mathbf{z}, \mathbf{y})$ distributed independently from the signing key, and simulates the commitment $\mathbf{w} = \mathbf{A} \cdot \mathbf{z} + \mathbf{y} - c \cdot \mathbf{t}$ (see Item (A.5)). Informally, if the commitment randomness $(\mathbf{r}, \mathbf{e}')$ are sufficiently larger than the scaled signing key $(c \cdot \mathbf{s}, c \cdot \mathbf{e})$, such a reduction remains indistinguishable from the real world. It is worth highlighting that $(\mathbf{r}, \mathbf{e}')$ cannot be *too* large since the response $(\mathbf{z}, \mathbf{y})$ must be "short," unlike in the classical Schnorr signature. Finally, it programs the random oracle as $\mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$.

Let us consider porting this proof to the threshold setting. To illustrate the effect of our masking idea, we explain what happens without them. Without the mask, user i outputs a partial signature $(\mathbf{z}_i, \mathbf{y}_i) =$

[6]See Fig. 1 for why we call it row and column masks.



$$\begin{array}{ccccccccccc}
\mathbf{m}_{1,1} & + & \mathbf{m}_{1,2} & + & \mathbf{m}_{1,3} & + & \mathbf{m}_{1,4} & + & \mathbf{m}_{1,5} & = & \mathbf{m}_1 \\
+ & & + & & + & & + & & + & & + \\
\mathbf{m}_{2,1} & + & \mathbf{m}_{2,2} & + & \mathbf{m}_{2,3} & + & \mathbf{m}_{2,4} & + & \mathbf{m}_{2,5} & = & \mathbf{m}_2 \\
+ & & + & & + & & + & & + & & + \\
\mathbf{m}_{3,1} & + & \mathbf{m}_{3,2} & + & \mathbf{m}_{3,3} & + & \mathbf{m}_{3,4} & + & \mathbf{m}_{3,5} & = & \mathbf{m}_3 \\
+ & & + & & + & & + & & + & & + \\
\mathbf{m}_{4,1} & + & \mathbf{m}_{4,2} & + & \mathbf{m}_{4,3} & + & \mathbf{m}_{4,4} & + & \mathbf{m}_{4,5} & = & \mathbf{m}_4 \\
+ & & + & & + & & + & & + & & + \\
\mathbf{m}_{5,1} & + & \mathbf{m}_{5,2} & + & \mathbf{m}_{5,3} & + & \mathbf{m}_{5,4} & + & \mathbf{m}_{5,5} & = & \mathbf{m}_5 \\
\parallel & & \parallel & & \parallel & & \parallel & & \parallel & & \parallel \\
\mathbf{m}_1^* & + & \mathbf{m}_2^* & + & \mathbf{m}_3^* & + & \mathbf{m}_4^* & + & \mathbf{m}_5^* & = & \mathbf{m}
\end{array}$$

Figure 1: Relationships between $\mathbf{m}_{i,j}, \mathbf{m}_i$ and $\mathbf{m}_j^*$, where we drop the subscript $\mathsf{act} = \{1,2,3,4,5\}$ for readability.

- The row masks $\mathbf{m}_i$ (blue, dotted pattern) are all public.
- An adversary corrupting the user set $\{1, 2, 3\}$ learns the set $(\mathbf{m}_{i,j})_{\min(i,j)\leq 3}$ and can infer the column masks $(\mathbf{m}_j^*)_{j\leq 3}$ (red).

$(c \cdot \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i + \mathbf{r}_i, c \cdot \mathbf{e} + \mathbf{e}_i')$. To perform the above proof strategy, the reduction must sample the response $(\mathbf{z}_i, \mathbf{y}_i)$ and simulate the commitment as $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{z}_i + \mathbf{y}_i - c \cdot \mathbf{t}_{\mathsf{act},i}$ without the partial signing key $\mathbf{s}_i$, where $\mathbf{t}_{\mathsf{act},i} = \mathbf{A}\mathbf{s}_i + \mathbf{e}_i$ is the (implicit) partial public key. However, notice the above proof strategy falls apart since the scaled partial signing key $c \cdot \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i$ is not guaranteed to be small compared to the commitment randomness $\mathbf{r}_i$ as the Lagrange coefficients $\lambda_{\mathsf{act},i}$ can become arbitrarily large modulo q. Moreover, we cannot just sample $\mathbf{r}_i$ random over $\mathcal{R}_q^\ell$ since this breaks the condition that the response $\mathbf{z}_i$ is short. Recall here that this is not an artifact of the proof strategy since there is a concrete attack, as we explained above.

This brings us to our masking idea. At a high level, the masking allows the reduction to move the partial signing keys around in such a way that the response can be simulated using only the full signing key, without the partial signing key. Effectively, we can remove the Lagrange coefficients in the reduction, and arrive at a reduction similar to the standard non-thresholdised signature scheme. Let us explain via an example. Assume the adversary queries a set $\mathsf{act} = \{1, 2, 3, 4, 5\}$ with two honest users 4 and 5 as in Fig. 1. Let us focus on the four masks $(\mathbf{m}_{\mathsf{act},i,j})_{i,j\in\{4,5\}}$ not known to the adversary. From Item (C.1), recall that the first signing round reveals the sums $\sum_{j\in\{4,5\}} \mathbf{m}_{\mathsf{act},4,j}$ and $\sum_{j\in\{4,5\}} \mathbf{m}_{\mathsf{act},5,j}$ to the adversary since all $(\mathbf{m}_{\mathsf{act},i,j})_{\min(i,j)\leq 3}$ (in red in Fig. 1) are known to the adversary. This leaves us one degree of freedom; the sums $\sum_{j\in\{4,5\}} \mathbf{m}_{\mathsf{act},j,4}$ and $\sum_{j\in\{4,5\}} \mathbf{m}_{\mathsf{act},j,5}$ are distributed uniformly random from the view of the adversary, conditioned on their sum being $\sum_{i,j\in\{4,5\}} \mathbf{m}_{\mathsf{act},i,j}$. Put differently, the column masks $\mathbf{m}^*_{\mathsf{act},4}$ and $\mathbf{m}^*_{\mathsf{act},5}$ are distributed uniformly random, conditioned on their sum being consistent with $\sum_{j\in\{4,5\}} \mathbf{m}_{\mathsf{act},j}$. Using this, in the proof, we can argue that the two responses $(\mathbf{z}_4, \mathbf{z}_5)$ generated as

$$\left(c \cdot \lambda_{\mathsf{act},4} \cdot \mathbf{s}_4 + \mathbf{r}_4 + \mathbf{m}^*_{\mathsf{act},4}, \quad c \cdot \lambda_{\mathsf{act},5} \cdot \mathbf{s}_5 + \mathbf{r}_5 + \mathbf{m}^*_{\mathsf{act},5}\right) \tag{2}$$

are distributed identically to responses generated as

$$\left(\mathbf{r}_4 + \overline{\mathbf{m}}^*_{\mathsf{act},4}, \quad c \cdot \left(\sum_{i\in\{4,5\}} \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i\right) + \mathbf{r}_5 + \overline{\mathbf{m}}^*_{\mathsf{act},5}\right),$$

where $\overline{\mathbf{m}}^*_{\mathsf{act},4}$ is sampled uniformly random and $\overline{\mathbf{m}}^*_{\mathsf{act},5}$ is set as the unique value that guarantees consistency with the verification equations.

Lastly, we use the fact that $\sum_{j\in\{4,5\}} \lambda_{\mathsf{act},j} \cdot \mathbf{s}_j = \mathbf{s} - \sum_{j\in\mathsf{corrupt}} \lambda_{\mathsf{act},j} \cdot \mathbf{s}_j$ (see Eq. (1)), where the adversary (and the reduction) controls the secrets for all users in $\mathsf{corrupt} = \mathsf{act}\backslash\{4,5\}$. Plugging this into the above, the reduction can instead generate the responses as

$$\left(\mathbf{r}_4 + \overline{\mathbf{m}}^*_{\mathsf{act},4}, \quad c \cdot \mathbf{s} - c \cdot \sum_{j\in\mathsf{corrupt}} \lambda_{\mathsf{act},j} \cdot \mathbf{s}_j + \mathbf{r}_5 + \overline{\mathbf{m}}^*_{\mathsf{act},5}\right).$$

Since the reduction can now simulate the response $c \cdot \mathbf{s} + \mathbf{r}_5$ of the base signature scheme only using the full signing key $\mathbf{s}$, we can rely on prior proof techniques at this point to complete the proof.

Subtle Issue with the Proof and a Fix.

While the intuition is simple, the concrete proof requires much care. One important point we glossed over was how we guarantee Eq. (2). Recall users 4 and 5 only *locally* generate the challenge $c := \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$, where $\mathbf{w} = \sum_{i\in\mathsf{act}} \mathbf{w}_i$ is the aggregated commitment (see Item (C.3)). In particular, a malicious adversary can send users 4 and 5 with inconsistent commitments (e.g., malicious user 1 provides distinct $\mathbf{w}_1$ and $\mathbf{w}'_1$ to users 4 and 5), in which case, the locally derived challenges c and c' by users 4 and 5 may differ. Against such an adversary, the reduction cannot argue Eq. (2), and incidentally, the proof breaks down. In fact, it turns out that we can turn this idea into a concrete attack, similarly to those explained prior. This brings us to our final construction, $\mathsf{TRaccoon}$, where we fix this issue by modifying the users to authenticate their views in the second round. One way we achieve this is to let the users add a signature to the hash commitments it received in Item (C.2). Another way is to let the users add a MAC instead. See Section 6 for more details.

Sharing the Masks.

Lastly, we explain how pairs of users $(i, j) \in \mathsf{act}$ share the masks $\mathbf{m}_{\mathsf{act},i,j}$ and $\mathbf{m}_{\mathsf{act},j,i}$ during the signing protocol. We simply generate seeds $(\mathsf{seed}_{i,j})_{i,j\in[N]}$ during the key generation phase and give $(\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[N]}$ to user i as part of their partial signing key. Once the set act is defined, user i can locally compute the random masks $\mathbf{m}_{\mathsf{act},i,j}$ and $\mathbf{m}_{\mathsf{act},j,i}$ by using a PRF on $\mathsf{seed}_{i,j}$ and $\mathsf{seed}_{j,i}$ respectively. For the masks to never be repeated, we assume each signing session has a unique identifier for which the PRF is called upon.

2.4 Future Work

In this work, we provide a 3-round lattice-based threshold signature $\mathsf{TRaccoon}$ that outperforms prior constructions relying on (fully) homomorphic encryptions and/or homomorphic trapdoor commitments. Our work leaves several practical and theoretical open questions. Currently, $\mathsf{TRaccoon}$ requires a trusted key generation process; for many applications, for example cryptocurrency wallet backups, this is a reasonable assumption. However, designing a distributed key generation protocol would be relevant for settings where a trusted setup cannot be assumed. Moreover, in case a distributed signature generation fails due to malicious behaviour, we may require properties like *robustness* or *identifiable abort*, allowing the users to identify at least one malicious signer. Also, as explained above, $\mathsf{TRaccoon}$ requires to maintain the unique identifier for each signing session so as not to reuse the same masks. We leave a stateless signature scheme as an important open problem. Finally, a 2-round lattice-based threshold signature without relying on heavy tools like (fully) homomorphic encryptions seems to require new ideas; in the classical setting, we need to rely on strong one-more-type assumptions (e.g., one-more DL) or idealised models like the algebraic group model, something non-trivial in the lattice setting.

3 Background

We provide the background. Standard definitions are deferred in Appendix A.

3.1 Notations

Sets, distributions and functions.

Given an integer $N \in \mathbb{N}$, we denote by $[N]$ the set $\{i \in \mathbb{N} \mid 1 \leq i \leq N\}$. Given a finite set S, we denote by $\mathcal{U}(S)$ the uniform distribution over S. Whenever possible, we will use $x \leftarrow S$ as shorthand for $x \leftarrow \mathcal{U}(S)$. Given a deterministic (respectively randomised) procedure f and a compatible input x, we note setting the variable y to be the result of $f(x)$ by $y \coloneqq f(x)$ (respectively $y \leftarrow f(x)$). A function $f : \mathbb{N} \to \mathbb{R}$ is said to be negligible (or in $\mathsf{negl}(\kappa)$) if $f(\kappa) = \kappa^{-\omega_{\mathsf{asymp}}(1)}$, where $\omega_{\mathsf{asymp}}(g)$ denotes a class of functions that grow asymptotically faster than g.

Algebra and Representation of $\mathbb{Z}_q$ Elements.

Scalars, vectors and matrices are noted in italic (i.e., x), lowercase bold (i.e., $\mathbf{x}$) and uppercase bold (i.e., $\mathbf{X}$) respectively. Let $n, q \in \mathbb{N}$ be two integers such that n is a power-of-two and the polynomial $X^n + 1$ fully splits over $\mathbb{Z}_q$. We define the ring $\mathcal{R}$ as $\mathbb{Z}[X]/(X^n + 1)$ and $\mathcal{R}_q$ as $\mathcal{R}/q\mathcal{R}$. We may define functions over the domain $\mathbb{Z}$ or $\mathbb{Z}_q$, then extend their domains to $\mathcal{R}$ or $\mathcal{R}_q$ by coefficient-wise application. Similarly, we may extend the domain to vectors or matrices with entries in $\mathcal{R}$ or $\mathcal{R}_q$.

In this work we use the so-called *canonical* unsigned representation of integers modulo q. Given an integer $x \in \mathbb{Z}$, this representation is the unique non-negative element $0 \leq t \leq q - 1$ such that $x = t \mod q$. We will generically note this element $(x \mod q)$. Conversely, given a class $x + q\mathbb{Z} \in \mathbb{Z}_q$, we define the corresponding lift $\bar{x}$ to the unique integer in $x + q\mathbb{Z} \cap [0, \dots q - 1]$.

For any norm $\|\cdot\|$ over $\mathbb{Q}^n$, we define the *length* of a (vector) class $\mathbf{x} + q\mathbb{Z}^n$ to be $\min_{\mathbf{z} \in \mathbf{x} + q\mathbb{Z}^n} \|\mathbf{z}\|$, and overload the notation as $\|\mathbf{x} + q\mathbb{Z}^n\|$, $\|\mathbf{x} \mod q\|$ or even $\|\mathbf{x}\|$ if the context is clear enough. As for the integers, we prefer to write simply $|x|$ when $n = 1$ to refer to the absolute value. We show below that with the choices in this definition, $\|\cdot\|$ is a *F-norm* over free modules over $\mathbb{Z}_q$. The only non trivial point to show is the triangular inequality.

Lemma 3.1. *For any $q, n \in \mathbb{N} \setminus \{0\}$, and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, we have*

$$|\|\mathbf{x}\| - \|\mathbf{y}\|| \leq \|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| .$$

Proof. Take $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$ two vector classes mod q. By triangular inequality over $\mathbb{Z}$, we have:

$$\|\mathbf{x} + \mathbf{y}\| = \min_{u,v \in \mathbf{x} \times \mathbf{y}} \|u + v\| \leq \min_{u,v \in \mathbf{x} \times \mathbf{y}} (\|u\| + \|v\|) = \min_{u \in \mathbf{x}}(\|u\|) + \min_{v \in \mathbf{y}}(\|v\|) = \|\mathbf{x}\| + \|\mathbf{y}\|.$$

We can then use it to write $\|\mathbf{x}\| \leq \|\mathbf{x} + \mathbf{y}\| + \|-\mathbf{y}\|$ and the fact that $\|\mathbf{y}\| = \|-\mathbf{y}\|$ to establish the left hand side inequality in the lemma statement. $\square$

3.2 Modulus Rounding

Let $\nu \in \mathbb{N} \setminus \{0\}$. Any integer $x \in \mathbb{Z}$ can be *uniquely* decomposed as:

$$x = 2^\nu \cdot x_\top + x_\perp, \quad (x_\top, x_\perp) \in \mathbb{Z} \times [-2^{\nu-1}, 2^{\nu-1} - 1], \tag{3}$$

which consists essentially in separating the lower-order bits from the higher-order ones, that is, it drops the ν lower bits. We define the function

$$\lfloor \cdot \rceil_\nu : \mathbb{Z} \to \mathbb{Z} \quad \text{s.t.} \quad \lfloor x \rceil_\nu = \lfloor x/2^\nu \rceil = x_\top, \tag{4}$$

where $\lfloor \cdot \rceil : \mathbb{R} \mapsto \mathbb{Z}$ denotes the rounding operator. More precisely the "rounding half-up" method $\lfloor x \rceil = \lfloor x + \frac{1}{2} \rfloor$ where half-way values are rounded up: e.g. $\lfloor 2.5 \rceil = 3$ and $\lfloor -2.5 \rceil = -2$. With a slight overload of notation, when $q > 2^\nu$, we extend $\lfloor \cdot \rceil_\nu$ to take inputs in $\mathbb{Z}_q$, in which case, we assume the output is an element in $\mathbb{Z}_{q_\nu}$ where $q_\nu = \lfloor q/2^\nu \rfloor$.Formally, we define:

$$\lfloor \cdot \rceil_\nu : \mathbb{Z}_q \mapsto \mathbb{Z}_{q_\nu} = \mathbb{Z}_{\lfloor q/2^\nu \rfloor} \quad \text{s.t.} \quad \lfloor x \rceil_\nu = \lfloor \bar{x}/2^\nu \rceil + q_\nu\mathbb{Z} = (\bar{x})_\top + q_\nu\mathbb{Z},$$

The function $\lfloor\cdot\rceil_\nu$ naturally extends to vectors coefficient-wise.

We provide some useful bounds regarding this modular rounding operation, mainly used for establishing correctness and bounding the SelfTargetMSIS solution size in the security proofs.

Lemma 3.2. *Let ν, q be positive integers such that $q > 2^\nu$ and set $q_\nu = \lfloor q/2^\nu \rfloor$. Moreover, assume q and ν satisfy $q_\nu = \lfloor q/2^\nu \rceil$, that is, q can be decomposed as $q = 2^\nu \cdot q_\nu + q_\perp$ for $q_\perp \in [0, 2^{\nu-1} - 1]$. Then, for any $x \in \mathbb{Z}_q$, we have*

$$\left| x - 2^\nu \cdot \overline{\lfloor x \rceil_\nu} \right| \leq 2^\nu - 1. \tag{5}$$

Moreover, for any $\mathbf{x}, \boldsymbol{\delta} \in \mathbb{Z}_q^n$, we have

$$\left\| 2^\nu \left(\overline{\lfloor \mathbf{x} + \boldsymbol{\delta} \rceil_\nu - \lfloor \mathbf{x} \rceil_\nu} \right) \mod q \right\| \leq \left\| 2^\nu \cdot \overline{\lfloor \boldsymbol{\delta} \rceil_\nu} \mod q \right\| + \|\mathbf{1}\| \cdot f(\nu), \tag{6}$$

where $f(\nu) = \begin{cases} 3 & \textit{if } \nu = 1 \\ 6 & \textit{if } \nu = 2 \\ 12 & \textit{if } \nu = 3 \\ 2^\nu & \textit{elsewise} \end{cases}$.

Proof. We first prove Eq. (5). Let us uniquely write $\bar{x} = 2^\nu \cdot x_\top + x_\perp$, where $(x_\top, x_\perp) \in [0, q_\nu] \times [-2^{\nu-1}, 2^{\nu-1} - 1]$ (using the assumption on the values of q and ν to write $x_\top \leq \left\lfloor \frac{q}{2^\nu} \right\rceil = q_\nu$). Then:

$$\overline{\lfloor x \rceil_\nu} = x_\top \mod q_\nu = \begin{cases} 0 & \text{if } x_\top = q_\nu \\ x_\top & \text{elsewise} \end{cases}.$$

In the first case, we have:

$$\begin{aligned} |x - 2^\nu \overline{\lfloor x \rceil_\nu}| &= |x| && (\text{replacement using that } 0 \mod q_\nu = 0 \mod q) \\ &= |x_\perp + 2^\nu q_\nu| && (x_\top = q_\nu) \\ &= |x_\perp - q_\perp| && (q = 2^n q_\nu + q_\perp) \\ &\leq |x_\perp| + |q_\perp| \leq 2^\nu - 1 && (\text{taken over } \mathbb{Z}) \end{aligned}.$$

In the second case, we simply have $(x - 2^\nu \cdot \lfloor x \rceil_\nu) \mod q = x_\perp \mod q$, concluding the proof.

We next turn to Eq. (6). The problem being cyclic module 2^ν, we can treat the first three cases ($\nu = 1, 2, 3$), by exhausting all possible cases. For larger values as we before, let us uniquely decomposes $\bar{\mathbf{x}} = 2^\nu \cdot \mathbf{x}_\top + \mathbf{x}_\perp$, where $(\mathbf{x}_\top, \mathbf{x}_\perp) \in [0, q_\nu]^n \times [-2^{\nu-1}, 2^{\nu-1} - 1]^n$, and define $\boldsymbol{\delta}_\top$ and $\boldsymbol{\delta}_\perp$ similarly.

Remark that each of the coefficients of $\overline{\mathbf{x} + \boldsymbol{\delta}}$ is bounded by $2q$, so that we can write $\lfloor \mathbf{x} + \boldsymbol{\delta} \rceil_\nu$ as

$$\lfloor 2^\nu \cdot (\mathbf{x}_\top + \boldsymbol{\delta}_\top) + \mathbf{x}_\perp + \boldsymbol{\delta}_\perp - \boldsymbol{\alpha} \cdot q \rceil_\nu = \lfloor 2^\nu \cdot (\mathbf{x}_\top + \boldsymbol{\delta}_\top - \boldsymbol{\alpha} \cdot q_\nu) + \mathbf{x}_\perp + \boldsymbol{\delta}_\perp - \boldsymbol{\alpha} \cdot q_\perp \rceil_\nu$$

where $\boldsymbol{\alpha} \in \{0,1\}^L$ such that $\boldsymbol{\alpha}_i = 1$ if $\mathbf{x}_i + \boldsymbol{\delta}_i > q - 1$ and $\boldsymbol{\alpha}_i = 0$ otherwise. Put differently, if the i-th entry of $\mathbf{x} + \boldsymbol{\delta}$ is larger than q, we perform a modulo q reduction, where note that each entry is smaller than $2q$ by definition. We thus have

$$\overline{\lfloor \mathbf{x} + \boldsymbol{\delta} \rceil_\nu - \lfloor \mathbf{x} \rceil_\nu} = \overline{\left((\mathbf{x}_\top + \boldsymbol{\delta}_\top - \boldsymbol{\alpha} \cdot q_\nu + \boldsymbol{\gamma}) - \mathbf{x}_\top \right)} = (\boldsymbol{\delta}_\top + \boldsymbol{\gamma}) - \boldsymbol{\epsilon} \cdot q_\nu,$$

where the right hand side is over the integers and $(\boldsymbol{\gamma}, \boldsymbol{\epsilon}) \in \{-1, 0, 1\}^L \times \{0, 1\}^L$. More specifically, we have

$$\boldsymbol{\gamma}_i = \begin{cases} -1 & \text{if } \mathbf{x}_{\perp,i} + \boldsymbol{\delta}_{\perp,i} - \boldsymbol{\alpha}_i \cdot q_\perp \leq -2^{\nu-1} - 1 \\ 0 & \text{if } \mathbf{x}_{\perp,i} + \boldsymbol{\delta}_{\perp,i} - \boldsymbol{\alpha}_i \cdot q_\perp \in [-2^{\nu-1}, 2^{\nu-1} - 1] \\ 1 & \text{if } \mathbf{x}_{\perp,i} + \boldsymbol{\delta}_{\perp,i} - \boldsymbol{\alpha}_i \cdot q_\perp \geq 2^{\nu-1} \end{cases}, \tag{7}$$

where note that these are the only values $\boldsymbol{\gamma}_i$ can take since for any $i \in [L]$, $|\mathbf{x}_{\perp,i} + \boldsymbol{\delta}_{\perp,i} - \boldsymbol{\alpha}_i \cdot q_{\perp}| \leq 3 \cdot 2^{\nu-1} - 1$. Moreover, we can be more precise:

$$\epsilon_i = \begin{cases} 0 & \text{when} \quad \boldsymbol{\delta}_{\top,i} + \boldsymbol{\gamma}_i \in \{0, \cdots, q_\nu - 1\} \\ 1 & \text{when} \quad (\boldsymbol{\delta}_{\top,i}, \boldsymbol{\gamma}_i) \in \{(q_\nu - 1, 1), (q_\nu, 0), (q_\nu, 1)\} \end{cases}$$

Notice that we can't have $(\boldsymbol{\delta}_{\top,i}, \boldsymbol{\gamma}_i) = (0, -1)$, i.e., $\boldsymbol{\epsilon}_i = -1$. Namely, when $\boldsymbol{\delta}_{\top,i} = 0$, we must have $\boldsymbol{\delta}_{\perp,i} \geq 0$ since we use the canonical non-negative representative for $\mathbb{Z}_q$. From Eq. (7), we also need $\boldsymbol{\alpha}_i = 1$, $q_{\perp} > 0$ and $\mathbf{x}_{\perp,i} + \boldsymbol{\delta}_{\perp,i} < 0$ for $\boldsymbol{\gamma}_i = -1$. However, this contradicts $\mathbf{x}_i + \boldsymbol{\delta}_i > q - 1$, which is required for $\boldsymbol{\alpha}_i = 1$, since $0 \leq \mathbf{x}_i < 2^\nu \cdot q_\nu + q_{\perp}$.

We thus have, when $\boldsymbol{\delta}_\top \neq q_\nu$:

$$\left\| 2^\nu \cdot \overline{\left(\lfloor \mathbf{x} + \boldsymbol{\delta} \rceil_\nu - \lfloor \mathbf{x} \rceil_\nu \right)} \mod q \right\| = \left\| 2^\nu \cdot \left(\boldsymbol{\delta}_\top + \boldsymbol{\gamma} - \boldsymbol{\epsilon} \cdot q_\nu \right) \mod q \right\|$$
$$= \left\| 2^\nu \cdot \overline{\lfloor \boldsymbol{\delta} \rceil_\nu} + 2^\nu \cdot \boldsymbol{\gamma} + \boldsymbol{\epsilon} \cdot q_{\perp} \mod q \right\|, \tag{8}$$

where the second equality follows from $q = 2^\nu \cdot q_\nu + q_{\perp}$.

Then, when $\boldsymbol{\epsilon}_i = 0$, we can use Lemma 3.1 to bound the $i - th$ coefficient by $|2^\nu \cdot \overline{\lfloor \boldsymbol{\delta}_i \rceil_\nu} \mod q| + |2^\nu \mod q|$ since $\gamma \in \{-1, 0, 1\}$. On the other hand, when $\boldsymbol{\epsilon}_i = 1$, we have three cases: $(\boldsymbol{\delta}_{\top,i}, \boldsymbol{\gamma}_i) \in \{(q_\nu - 1, 1), (q_\nu, 0), (q_\nu, 1)\}$, meaning that $\boldsymbol{\delta}_{\top,i} = q_\nu - 1$, by assumption. As such, we find that the i-th coefficient of Eq. (8) computes to $2^\nu q_\nu + q_{\perp} = q$ which equals 0 taken modulo q.

Now if $\boldsymbol{\delta}_\top = q_\nu$, we have the simple form:

$$\left\| 2^\nu \cdot \overline{\left(\lfloor \mathbf{x} + \boldsymbol{\delta} \rceil_\nu - \lfloor \mathbf{x} \rceil_\nu \right)} \mod q \right\| = \left\| 2^\nu \cdot \boldsymbol{\gamma} + (\boldsymbol{\epsilon} - 1) \cdot q_{\perp} \mod q \right\|,$$

If $\boldsymbol{\epsilon}_i = 1$, since we necessarily have $\boldsymbol{\gamma}_i \geq 0$, the bound holds trivially. And if $\boldsymbol{\epsilon}_i = 0$,

1. either $\boldsymbol{\gamma}_i = 0$, meaning that the i th coeffcient of the right hand side is $-q_\perp$, which will be centerly reduced to a value smaller than 2^ν.

2. either $\boldsymbol{\gamma}_i = 1$, and the coefficient is $2^\nu - q_\perp \mod q$, which is bounded by triangular inequality by $|2^\nu \mod q| + |q_\perp \mod q| = 2^\nu + |2^\nu q_\nu \mod q| = 2^\nu + |2^\nu \delta_{\top,i} \mod q|$.

This concludes the proof. □

Remark 3.3. One can be puzzled by the apparently arbitrary condition $q_\nu = \lfloor q/2^\nu \rfloor = \lceil q/2^\nu \rfloor$. This allows to use the slightly better bound presented in the lemma (the generic bound in Eq. (5) being otherwise $2^\nu + 2^{\nu-1} - 1$), as our parameter choices will always satisfy the desired rounding bound. We note that other choices of q_ν may lead to better bounds but we leave it for future work. For instance, setting $q_\nu = \lfloor (q-1)/2^\nu \rceil + 1$ leads to a tighter bound on Eq. (5), replacing $2^\nu - 1$ with $2^{\nu-1}$. However, it is not obvious whether it leads to a tighter bound on Eq. (6).

3.3 Gaussians

Let $\rho_\sigma(\mathbf{z}) = \exp\left(-\frac{\|\mathbf{z}\|^2}{2\sigma^2}\right)$. The discrete Gaussian distribution of support $S \subseteq \mathcal{R}^k$, centre $\mathbf{v} \in \mathcal{R}^k$ and standard deviation $\sigma \in \mathbb{R}$, is defined by its probability distribution function:

$$\mathcal{D}_{S,\mathbf{v},\sigma}(\mathbf{z}) = \frac{\rho_\sigma(\mathbf{z} - \mathbf{v})}{\sum_{\mathbf{z}' \in S} \rho_\sigma(\mathbf{z}' - \mathbf{v})}.$$

When the support S is clear from context, we may simply note $\mathcal{D}_{\mathbf{z},\sigma}$. When $\mathbf{z} = \mathbf{0}$, we may simply write $\mathcal{D}_{S,\sigma}$. The following lemma is obtained by combining Minkowski's inequality with [Lyu12, Lemma 4.4, Item 3].

Lemma 3.4. *Let $\mathbf{s} \leftarrow \mathcal{D}^\ell_{\mathcal{R},\sigma}$, and $c \in \mathcal{R}$. Then, we have*

$$\mathbb{P}\left[\|c \cdot \mathbf{s}\|_2 \geq e^{1/4} \|c\|_1 \sigma \sqrt{n\ell} \right] \leq 2^{-n\ell/10}.$$

3.4 Hardness Assumptions

The standard definitions of the MLWE and MSIS problems and their reductions to worst-case module lattice problems are recalled in Appendix A.1. In this work, rather than directly relying on these problems, we use two hard problems related to the MLWE and MSIS problems. By outsourcing some of the proofs into these problems, we are able to simplify the proof of our threshold signature scheme.

We first recall the *hint* MLWE ($\mathsf{Hint\text{-}MLWE}$) problem [KLSS23]. It is defined similarly to MLWE, except that the adversary also obtains some *noisy leakage* of the MLWE secrets. Looking ahead, this noisy leakage corresponds to the response (i.e., third round flow) of a Lyubashevsky-type signature scheme [Lyu09, Lyu12]. While it is typical to use statistical arguments based on rejection sampling or the Rényi divergence to simulate the response, $\mathsf{Hint\text{-}MLWE}$ naturally provides this through a computational argument. Kim et al [KLSS23] showed that for specific parameter choices, $\mathsf{Hint\text{-}MLWE}$ is as hard as MLWE, see Lemma A.5. In our case, the reduction of [KLSS23] is tighter than Rényi divergence-based arguments such as the ones used by [ASY22, dPEK$^+$23]. This effectively allows us to set better parameters.

Definition 3.5 ($\mathsf{Hint\text{-}MLWE}$)**.** *Let ℓ, k, q, Q be integers, $\mathcal{D}, \mathcal{G}$ be probability distributions over $\mathcal{R}_q$, and $\mathcal{C}$ be a set over $\mathcal{R}_q$. The advantage of an adversary $\mathcal{A}$ against the* Hint Module Learning with Errors *$\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q,\mathcal{D},\mathcal{G},\mathcal{C}}$ problem is defined as:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hint\text{-}MLWE}}(\kappa) = \left| \Pr\left[1 \leftarrow \mathcal{A}\Big(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}, (c_i, \mathbf{z}_i, \mathbf{z}'_i)_{i \in [Q]}\Big)\right] - \Pr\left[1 \leftarrow \mathcal{A}\Big(\mathbf{A}, \mathbf{b}, (c_i, \mathbf{z}_i, \mathbf{z}'_i)_{i \in [Q]}\Big)\right] \right|$$

where $(\mathbf{A}, \mathbf{b}, \mathbf{s}, \mathbf{e}) \leftarrow \mathcal{R}_q^{k \times \ell} \times \mathcal{R}_q^k \times \mathcal{D}^\ell \times \mathcal{D}^k$, $c_i \leftarrow \mathcal{C}$ for $i \in [Q]$. Moreover, $(\mathbf{z}_i, \mathbf{z}'_i) = (c_i \cdot \mathbf{s} + \mathbf{r}_i, c_i \cdot \mathbf{e} + \mathbf{e}'_i)$ where $(\mathbf{r}_i, \mathbf{e}'_i) \leftarrow \mathcal{G}^\ell \times \mathcal{G}^k$ for $i \in [Q]$. The $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q,\mathcal{D},\mathcal{G},\mathcal{C}}$ assumption states that any efficient adversary $\mathcal{A}$ has negligible advantage. We may write $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q,\sigma_\mathcal{D},\sigma_\mathcal{G},\mathcal{C}}$ as a shorthand when $\mathcal{D}$ and $\mathcal{G}$ are the discrete Gaussian distributions of standard deviation $\sigma_\mathcal{D}$ and $\sigma_\mathcal{G}$, respectively.

We further rely on the *self-target* MSIS ($\mathsf{SelfTargetMSIS}$) problem [DKL$^+$18, KLS18]. This is a variant of the standard MSIS problem, where the problem is defined relative to some hash function modeled as a random oracle. Following a standard proof using the forking lemma [FS87, BN06], when the range of the hash function is exponentially large, $\mathsf{SelfTargetMSIS}$ is shown to be as hard as MSIS in the random oracle model.[7] In our work, we directly work with $\mathsf{SelfTargetMSIS}$ instead since it allows for a simpler proof compared to using MSIS, while putting a focus on concrete security, ignoring the reduction loss incurred by the forking lemma. Indeed, $\mathsf{SelfTargetMSIS}$ also underlies the hardness of the signature scheme Dilithium [DKL$^+$18], recently selected by NIST for standardisation, and widely understood to be as concretely secure as MSIS. Formally, $\mathsf{SelfTargetMSIS}$ is defined as follows. The concrete hardness of $\mathsf{SelfTargetMSIS}$ is analysed in Section 8.1. For completeness, we include more details on the asymptotic hardness of $\mathsf{SelfTargetMSIS}$ in Appendix B.1.

Definition 3.6 ($\mathsf{SelfTargetMSIS}$)**.** *Let ℓ, k, q be integers and $B_{\mathsf{stmsis}} > 0$ be a real number. Let $\mathcal{C}$ be a subset of $\mathcal{R}_q$ and let $\mathsf{G} : \mathcal{R}_q^k \times \{0,1\}^{2\kappa} \to \mathcal{C}$ be a cryptographic hash function modeled as a random oracle. The advantage of an adversary $\mathcal{A}$ against the* Self Target MSIS *problem, noted $\mathsf{SelfTargetMSIS}_{q,\ell,k,C,B_{\mathsf{stmsis}}}$, is defined as:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SelfTargetMSIS}}(\kappa) = \Pr\Big[\mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}, (\mathsf{msg}, \mathbf{z}^*) \leftarrow \mathcal{A}^{\mathsf{G}}(\mathbf{A}), (\mathsf{msg}, \mathbf{z}) \in \{0,1\}^{2\kappa} \times \mathcal{R}_q^{\ell+k} :$$
$$\left(\mathbf{z} = \begin{bmatrix} c \\ \mathbf{z}' \end{bmatrix}\right) \wedge (\|\mathbf{z}\|_2 \leq B_{\mathsf{stmsis}}) \wedge \ \mathsf{G}\left(\left[\mathbf{A} \mid \mathbf{I}\right] \cdot \mathbf{z}, \ \mathsf{msg}\right) = c\Big].$$

The $\mathsf{SelfTargetMSIS}_{q,\ell,k,C,B_{\mathsf{stmsis}}}$ assumption states that any efficient adversary $\mathcal{A}$ has no more than negligible advantage.

[7] As with any invocation of the forking lemma, the reduction comes with a reduction loss dependent on the number of random oracle queries the adversary performs. We note the reduction loss can be tuned using alternative forking strategies [MR02, OO98, PS00].

Lastly, we define a distribution and a property that will be useful in the security proof.

Definition 3.7. *For any $\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$, let $\mathcal{D}^{\textsf{bd-MLWE}}_{q,\ell,k,\sigma,\nu}(\mathbf{A})$ be the distribution defined as $\left\{ \lfloor \mathbf{A}\cdot\mathbf{s}+\mathbf{e} \rceil_\nu \mid (\mathbf{s},\mathbf{e}) \leftarrow \mathcal{D}_\sigma^\ell \times \mathcal{D}_\sigma^k \right\}$. That is, it samples an $\mathsf{MLWE}_{q,\ell,k,\sigma}$ instance with ν bits dropped.*

Lemma 3.8. *For any $\sigma > 2n \cdot q^{\frac{1}{k+\ell}+\frac{2}{n\ell}}$ and $\nu < \log(q) - 2$, we have:*

$$\Pr_{\mathbf{A}\leftarrow\mathcal{R}_q^{k\times\ell}}\left[H_\infty\left(\mathcal{D}^{\textsf{bd-MLWE}}_{q,\ell,k,\sigma,\nu}(\mathbf{A})\right) \geq n-1\right] \geq 1 - 2^{-n+1}.$$

Proof. Let $q_\nu = \lfloor q/2^\nu \rfloor$. We first consider a restricted distribution in which we only output the first polynomial in $\mathcal{D}^{\textsf{bd-MLWE}}_{q,\ell,k,\sigma,\nu}(\mathbf{A})$, i.e for $\mathbf{a} \in \mathcal{R}_q^{1\times\ell}$ the first row of $\mathbf{A}$ and $(\mathbf{r}, e') \leftarrow D_{\sigma_\mathbf{w}}^{\ell+1}$, we consider the distribution $\phi = \lfloor \langle \mathbf{a}, \mathbf{r}\rangle + e' \rceil_\nu$. A direct consequence of the Data Processing inequality is that applying a deterministic function to a distribution can only reduce its min-entropy, Hence:

$$H_\infty\left(\mathcal{D}^{\textsf{bd-MLWE}}_{q,\ell,k,\sigma,\nu}(\mathbf{A})\right) \geq H_\infty(\phi).$$

We now use the regularity lemma of Lyubashevsky et al. [LPR13, Corollary 7.5] to state that the distribution of $\langle \mathbf{a}, \mathbf{r}\rangle + e'$ is statistically close to uniform with probability $1 - \mathsf{negl}(\kappa)$ over the choice of $\mathbf{a}$. Making the constant in the $\Omega(n)$ asymptotic of the aforementioned paper explicit we get:

$$\Delta(\langle \mathbf{a}, \mathbf{r}\rangle + e', \mathcal{U}(\mathcal{R}_q)) \leq 2^{-n\,\ell+1} + \ell\, 2^{1-2\,n} \leq 2^{-n}.$$

Using the data processing inequality for the statistical distance we have:

$$\Delta(\phi, \lfloor \mathcal{U}(\mathcal{R}_q) \rceil_\nu) \leq \Delta(\langle \mathbf{a}, \mathbf{r}\rangle + e', \mathrm{Unif}(\mathcal{R}_q)) = 2^{-n}.$$

By definition of the min-entropy:

$$\begin{aligned}
2^{-H_\infty(\phi)} &= \max_{y\in\mathcal{R}_{q_\nu}} \phi(y) \\
&\leq \max_{y\in\mathcal{R}_{q_\nu}} \lfloor \mathcal{U}(\mathcal{R}_q) \rceil_\nu (y) + 2^{-n} \\
&\leq \left(\frac{2^\nu + 1}{q}\right)^n + 2^{-n} \\
&\leq 2^{-n(\log q_\nu - 1)} + 2^{-n} \\
&\leq 2^{-n+1}.
\end{aligned}$$

Where the third line comes from the fact that the probability of $\lfloor \mathcal{U}(\mathcal{R}_q) \rceil_\nu (y)$ is equal to the number of elements in $\mathcal{R}_q$ mapped to y by rounding, divided by q. Which is at most $2^\nu + 1$. Collecting all the bounds gives the desired result. ☐

3.5 Linear Secret Sharing

In our (T, N)-threshold signature construction, the secret is distributed as the constant term of a degree $T-1$ polynomial over $\mathcal{R}_q$. The reconstruction is later done via polynomial interpolation using Lagrange polynomials evaluated at zero. This methodology is also known as *Linear Shamir Secret Sharing* [Sha79b].

Let $N < q$ be an integer such that for distinct $i, j \in [N]$, $(i - j)$ is invertible over $\mathbb{Z}_q$. Let $S \subseteq [N]$ be a set of cardinality at least T. Then, given $i \in S$, we define the Lagrange coefficient $\lambda_{S,i}$ as $\lambda_{S,i} := \prod_{j\in S\setminus\{i\}} \frac{-j}{i-j}$.

Let $s \in \mathcal{R}_q$ be a secret to be shared, $P \in \mathcal{R}_q[X]$ a degree $T-1$ polynomial such that $P(0) = s$. Given any set of evaluation points $E = \{(i, y_i)\}_{i\in S}$ such that $y_i = P(i)$ for all $i \in S$, we note that: $s = \sum_{i\in S} \lambda_{S,i} \cdot y_i$. The notations naturally extend to secrets that are in vector form. With a slight abuse of notation, we say that $\mathbf{P} \in \mathcal{R}_q^\ell[X]$ is of degree $T-1$ if each entry of $\mathbf{P}$ is a degree $T-1$ polynomial. Moreover, $\mathbf{P}(x)$ denotes the evaluation of each entry of $\mathbf{P}$ on the point x.

4 Definitions of Threshold Signature

Here we provide the definition of threshold signatures. In Section 4.1, we first prepare some notations that allows simplifying the notations of threshold signatures. We then define the syntax and security notions in Section 4.2.

4.1 User States and Session States

For clarity and compactness sake, the definitions of threshold signatures use different objects to structurally store information. A user state (Definition 4.1) is an internal state that each user maintains during their lifetime in a distributed system. Most of the stored information is set at KeyGen: the total number of users, the threshold bound, the index of the user and their signing key and the global verification key. Meanwhile, the last part contains the list of previously run sessions by user idx, mostly to avoid the reuse of nonce parts (as per the session id sid that should not be reused).

This session state (Definition 4.1) is updated during the lifetime of a signing session. It consists of an agreed-on session ID, the information about the current session, and the contributions of every signing user for each round.

Definition 4.1 (User and session states)**.** *The* $\mathsf{UserState}$ *(resp.* $\mathsf{Session}$*) class defines a blueprint for storing all information relative to a specific user in a set of signers (resp. a specific user's view of a specific signing session).*

An object $\mathsf{state} \in \mathsf{UserState}$ *(resp.* $\mathsf{session} \in \mathsf{Session}$*) contains these fields:*

$\underline{\mathsf{state}}$*:*

- N*: The total number of parties*
- T*: The signing threshold*
- idx*: The index of the current user*
- sk*: The signing key* $\mathsf{sk}_{\mathsf{idx}}$ *of* idx
- vk*: The public key*
- $\mathsf{sessions}$*: A dictionary* $(\mathsf{session}_{\mathsf{sid}})_{\mathsf{sid}}$ *of past and active sessions*

$\underline{\mathsf{session}}$*:*

- sid*: ID of the signing session*
- $\mathsf{act} \subseteq [N]$*: Set of active signers*
- msg*: The message being signed*
- $\mathsf{round} \in [\mathsf{rnd}]$*: The current round*
- $\mathsf{internal}$*: The internal state of* idx
- $\mathsf{contrib}_i$*: The contribution of all users in* act *during round* i

4.2 Threshold Signatures

A threshold signature scheme is parameterised with a number $N \geq 1$ of signers and a reconstruction threshold T. While we construct a 3-round threshold signature, below we define a general rnd-round scheme with $\mathsf{rnd} \in \mathbb{N}$. Formally, an rnd-round (T, N)-*threshold signature* scheme is a tuple of PPT algorithms (KeyGen, $\{\mathsf{ShareSign}_i\}_{i \in [\mathsf{rnd}]}$, Verify, $\mathsf{Combine}$) such that:

$(\mathsf{vk}, (\mathsf{sk}_i)_{i \in [N]}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, T, N)$:

The key generation algorithm takes as inputs common public parameters, a reconstruction threshold T, and a total number of signers N such that $1 \leq T \leq N$. It returns a full public key vk and secret key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$. It also initializes any hash functions modeled as a random oracle in the security proof. Signatures under vk can only be generated by parties that control at least T secret key shares.

$(\mathsf{state}_j, \mathsf{session}_{\mathsf{sid}}, \mathsf{contrib}_1[j]) \leftarrow \mathsf{ShareSign}_1(\mathsf{state}_j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$:

The first round signing algorithm run by user j takes as inputs an internal state state_j (including a secret key share), a session id sid, an active signing set $\mathsf{act} \subseteq [N]$ of size at least T, and a message

msg. It returns its first round contribution $\mathsf{contrib}_1[j]$, initializes a new session $\mathsf{session}_{\mathsf{sid}}$ and updates its internal state state_j. Here, we assume sid includes act and msg.

$(\mathsf{state}_j, \mathsf{contrib}_i[j]) \leftarrow \mathsf{ShareSign}_i(\mathsf{state}_j, \mathsf{contrib}_{i-1})$:

The i-th round signing algorithm for $i \geq 2$ run by user j takes as inputs an internal state (including a secret key share and all contributions from the previous rounds) and a contribution of all the users from round $i-1$. It returns its own part of the i-th round contribution $\mathsf{contrib}_i[j]$ and updates its internal state state_j.

$\mathsf{sig} \leftarrow \mathsf{Combine}(\mathsf{vk}, \mathsf{sid}, \mathsf{msg}, (\mathsf{contrib}_i)_{i\in[\mathsf{rnd}]})$:

This algorithm takes as inputs the public key vk, the session identifier sid, the message msg, and the contributions from all rounds. It outputs a signature sig for the message msg under public key vk.

$0/1 \leftarrow \mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$:

This deterministic algorithm takes as inputs the public key vk, a message msg, and a signature sig. It outputs 1 (resp. 0) if sig is deemed to be a valid (resp. invalid) signature.

In our construction, we use different hash functions parameterised by a value $\mathsf{HashParams}$. This information is part of public parameters pp that is implicitly given as input to all algorithms (besides KeyGen, where it is explicitly stated). For completeness, correctness is defined in Appendix A.4.

For unforgeability, we focus on interactive schemes where the set of active signers (act) and the message (msg) are determined in the first round of the protocol. We formally define unforgeability in Definition 4.2, allowing the adversary to individually query each partial signing oracle following [BCK+22]. We assume that parties maintain states and that the security game updates party states implicitly during algorithm calls. The command **assert**(x) immediately outputs $\perp$ if x is false and does nothing otherwise.

Definition 4.2 (Unforgeability)**.** *Let* $\mathsf{pp}(\kappa)$ *be a parameter generating algorithm that takes as input the security parameter* 1^κ. *We define* $\mathsf{Game}^{\mathsf{ts\text{-}uf}}_{\mathcal{A}}$ *in Fig. 2 for an adversary* $\mathcal{A}$. *We say a threshold signature scheme is* unforgeable *if for any efficient* $\mathcal{A}$*, the following advantage of* $\mathcal{A}$ *is* $\mathsf{negl}(\kappa)$*:* $\mathsf{Adv}^{\mathsf{ts\text{-}uf}}_{\mathcal{A}} = \Pr[\mathsf{Game}^{\mathsf{ts\text{-}uf}}_{\mathcal{A}}(\kappa) = 1]$.

Remark 4.3 (Q_{Sign}-Bounded Scheme). While our construction of threshold signature supports an unbounded polynomially many signing queries, we would require a super-polynomial sized modulus q, making the scheme impractical. To this end, we consider a more fined grained bounded scheme where unforgeability holds against any adversary making at most $Q_{\mathsf{Sign}} = \mathsf{poly}(\kappa)$ signing queries to OSgn_1. As per NIST's 2022 call for additional (post-quantum) signatures [NIS22], we set $Q_{\mathsf{Sign}} \approx 2^{64}$ for our concrete instantiation.

Some discussion on the assumed communication channel between the signers is provided in Appendix A.5.

5 Underlying Signature Scheme

In this section, we describe a standard non-thresholdised signature, underlying our threshold signature in Section 6. As explained in Section 2.1, the signature scheme is a variant of Lyubashevsky's signature scheme [Lyu09, Lyu12]. Conceptually, the signature scheme we consider is closest to Raccoon [dPEK+23], a lattice-based signature scheme by del Pino et al., submitted to the additional NIST call for proposals [NIS22].[8] The main difference between Lyubashevsky's signature scheme is that Raccoon no longer performs rejection sampling, similarly to [ASY22, GKS23], while simultaneously performs several optimisations, similarly to Dilithium.

In this work, we consider a slight variant of Raccoon where the noise distribution is a discrete Gaussian, rather than a sum of uniforms. del Pino et al. considered sum of uniforms as they are simpler to sample in a context where side-channel attacks are important. As thresholdisation is the main focus of our work,

[8]This is not to be confused with [dPPRS23], having the same signature name Raccoon. While [dPPRS23] also considers a variant of the Lyubashevsky's signature scheme without rejection sampling, [dPEK+23] can be viewed as a major simplification and refinement of it.

```
$\mathsf{Game}^{\mathsf{ts\text{-}uf}}_{\mathcal{A}}(1^\kappa)$
1: $L_{\mathsf{Sign}}, L_{\mathsf{H}} := \emptyset$
2: $(N, T, \mathsf{corrupt}) \leftarrow \mathcal{A}(\mathsf{pp}(1^\kappa))$
3: assert{ $\mathsf{corrupt} \subseteq [N]$ }                                  ▷ Set of corrupt parties
4: assert{ $|\mathsf{corrupt}| < T$ }
5: $\mathsf{honest} := [N] \backslash \mathsf{corrupt}$                                  ▷ Set of honest parties
6: $(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, T, N)$                 ▷ Trusted key generation
7: for $i \in \mathsf{honest}$ do                                               ▷ Maintaining state
8:     $\mathsf{state}_i.\mathsf{sk} := \mathsf{sk}_i$
9:     $\mathsf{state}_i.\mathsf{vk} := \mathsf{vk}$
10: $(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathsf{OSgn}_i(\cdot))_{i\in[\mathsf{rnd}]}}(\mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{corrupt}})$
                                ▷ Run adversary, which eventually outputs a forgery
11: if $(\mathsf{msg} \in L_{\mathsf{Sign}})$ or $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) = 0$ then       ▷ Check msg not queried, sig valid
12:     return $0$
13: return $1$
```

```
$\mathsf{OSgn}_1(j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$
1: assert{ $j \in \mathsf{honest} \cap \mathsf{act} \ \wedge\ \mathsf{act} \subseteq [N]$ }      ▷ User $j$ is honest and signing set act is valid
2: $L_{\mathsf{Sign}} := L_{\mathsf{Sign}} \cup \{\mathsf{msg}\}$                                ▷ Add msg to query set
3: $(\mathsf{state}_j, \mathsf{session}_{\mathsf{sid}}, \mathsf{contrib}_1[j]) \leftarrow \mathsf{ShareSign}_1(\mathsf{state}_j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$     ▷ Compute contribution
4: return $\mathsf{contrib}_1[j]$
```

```
$\mathsf{OSgn}_k(j, \mathsf{sid}, \mathsf{contrib}_{i-1})$, for $k \in \{2, \ldots, \mathsf{rnd}\}$
1: assert{ $j \in \mathsf{honest} \cap \mathsf{act} \ \wedge\ \mathsf{act} \subseteq [N]$ }
2: $(\mathsf{state}_j, \mathsf{contrib}_k[j]) \leftarrow \mathsf{ShareSign}_i(\mathsf{state}_j, \mathsf{contrib}_{k-1})$       ▷ Compute contribution
3: return $\mathsf{contrib}_k[j]$
```

```
$\mathsf{H}(\mathsf{str}, \mathsf{digest})$
1: assert{ $\mathsf{str} \in \mathsf{pp}.\mathsf{HashParams}$ }                          ▷ Check domain string
2: if $(\mathsf{str}, \mathsf{digest}, r) \in L_{\mathsf{H}}$ then
3:     return $r$                                                   ▷ Check if already queried
4: else
5:     Sample $r$ randomly
6:     $L_{\mathsf{H}} := L_{\mathsf{H}} \cup \{(\mathsf{str}, \mathsf{digest}, r)\}$                        ▷ Add to query set
7:     return $r$
```

Figure 2: Unforgeability game for threshold signatures. The adversary $\mathcal{A}$ wins if $\mathsf{Game}^{\mathsf{ts\text{-}uf}}_{\mathcal{A}}$ returns 1.

we use discrete Gaussians instead. Using nice properties of discrete Gaussians, such as convolution lemmas, we are able to achieve a more efficient scheme. Specifically, we provide a new security proof using the recently introduced *hint* MLWE (Hint-MLWE) assumption by Kim et al. [KLSS23]. This allows to sidestep the complex Rényi divergence argument in the proof of Raccoon, making the proof much simpler as an added bonus. The new proof may be of an independent interest as it can be used to optimise other variants of Lyubashevsky's signature scheme not relying on rejection sampling [ASY22, GKS23].

```
Alg. 1: KeyGen($1^\kappa$)
1: $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}$                      ▷ Uniform matrix
2: $(\mathbf{s},\mathbf{e}) \leftarrow \mathcal{D}_{\mathbf{t}}^{\ell} \times \mathcal{D}_{\mathbf{t}}^{k}$          ▷ Small secret and noise
3: $\mathbf{t} := \lfloor \mathbf{A}\cdot\mathbf{s}+\mathbf{e} \rceil_{\nu_{\mathbf{t}}}$           ▷ Part of public key in $\mathcal{R}_{q_{\mathbf{t}}}^{k}$
4: return $\mathsf{vk} := (\mathbf{A},\mathbf{t})$, $\mathsf{sk} := \mathbf{s}$

Alg. 2: Sign(vk, sk, msg)
1: $(\mathbf{r},\mathbf{e}') \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell} \times \mathcal{D}_{\mathbf{w}}^{k}$         ▷ Small randomness and noise
2: $\mathbf{w} := \lfloor \mathbf{A}\cdot\mathbf{r}+\mathbf{e}' \rceil_{\nu_{\mathbf{w}}}$         ▷ (Rounded) commitment in $\mathcal{R}_{q_{\mathbf{w}}}^{k}$
3: $c := \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$            ▷ Challenge
4: $\mathbf{z} := c\cdot\mathbf{s}+\mathbf{r}$                    ▷ Response in $\mathcal{R}_q^{\ell}$
5: $\mathbf{y} := \lfloor \mathbf{A}\cdot\mathbf{z} - 2^{\nu_{\mathbf{t}}}\cdot c\cdot\mathbf{t} \rceil_{\nu_{\mathbf{w}}}$     ▷ Intermediate value in $\mathcal{R}_{q_{\mathbf{w}}}^{k}$
6: $\mathbf{h} := \mathbf{w} - \mathbf{y}$                      ▷ Hint in $\mathcal{R}_{q_{\mathbf{w}}}^{k}$
7: return $\sigma := (c, \mathbf{z}, \mathbf{h})$

Alg. 3: Verify(vk, msg, $\sigma$)
1: $(c,\mathbf{z},\mathbf{h}) := \mathsf{parse}(\sigma)$
2: $c' := \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \lfloor \mathbf{A}\cdot\mathbf{z} - 2^{\nu_{\mathbf{t}}}\cdot c\cdot\mathbf{t} \rceil_{\nu_{\mathbf{w}}} + \mathbf{h})$
3: if $\{c = c'\}$ and $\{\|(\mathbf{z}, 2^{\nu_{\mathbf{w}}}\cdot\mathbf{h})\|_2 \leq B_2\}$ then
4:     return 1
5: return 0
```

Figure 3: A variant of the Raccoon signature scheme [dPEK+23] where the noise is sampled from a discrete Gaussian, rather than from a sum of uniforms.

Construction Overview.

Let $q_{\mathbf{t}} = \lfloor q \rceil_{\nu_{\mathbf{t}}}$ and $q_{\mathbf{w}} = \lfloor q \rceil_{\nu_{\mathbf{w}}}$. The public parameters include the security parameter κ, a ring $\mathcal{R}$, and a hash function

$$\mathsf{H}_c : \mathcal{V} \times \mathcal{M} \times \mathcal{R}_{q_{\mathbf{w}}}^{k} \rightarrow \mathcal{C},$$

that maps a public key $\mathsf{vk} \in \mathcal{V}$, message $\mathsf{msg} \in \mathcal{M}$ and a commitment $\mathbf{w} \in \mathcal{R}_{q_{\mathbf{w}}}^{k}$ to the challenge space $\mathcal{C}$ defined as:

$$\mathcal{C} := \{c \in \mathcal{R}_q \mid \|c\|_\infty = 1 \wedge \|c\|_1 = \omega\}. \tag{9}$$

In the security proof, H_c is modeled as a random oracle.

The key generation algorithm takes as input the public parameters and randomly selects a public matrix $\mathbf{A}$ and secret vectors $(\mathbf{s},\mathbf{e})$. The public key is given by $\mathsf{vk} := (\mathbf{A},\mathbf{t})$ for $\mathbf{t} := \lfloor \mathbf{A}\cdot\mathbf{s}+\mathbf{e} \rceil_{\nu_{\mathbf{t}}} \in \mathcal{R}_{q_{\mathbf{t}}}^{k}$ where $\mathbf{e}$ is a small noise and we perform bit dropping to compress the public key. It is worth noting that, while conceptually similar, the way Raccoon performs bit-dropping is much simpler and easier to implement compared to those performed by Dilithium.

To sign, the signer samples small ephemeral errors $(\mathbf{r}, \mathbf{e}')$ and generates an MLWE commitment $\mathbf{w} := \lfloor \mathbf{A} \cdot \mathbf{r} + \mathbf{e}' \rceil_{\nu_{\mathbf{w}}} \in \mathcal{R}_{q_{\mathbf{w}}}^k$. A challenge is then computed by hashing the public key, message, and $\mathbf{w}$ together. Finally, a response $\mathbf{z} := c \cdot \mathbf{s} + \mathbf{r} \in \mathcal{R}_q^\ell$ is computed together with a *hint* $\mathbf{h} := \mathbf{w} - \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathbf{t}}} \cdot c \cdot \mathbf{t} \rceil_{\nu_{\mathbf{w}}} \in \mathcal{R}_{q_{\mathbf{w}}}^k$. Here, the hint $\mathbf{h}$ is used to compensate for the bit dropping in the commitment and we multiply $c \cdot \mathbf{t}$ by $2^{\nu_{\mathbf{t}}}$ to compensate for the bit dropping in the public key. The signature is given by $(c, \mathbf{z}, \mathbf{h})$, where note that we send c instead of $\mathbf{w}$ because it is smaller. Importantly, we do not perform rejection sampling and always output the response $\mathbf{z}$.

Lastly, the verifier checks that $\|(\mathbf{z}, 2^{\nu_{\mathbf{w}}} \cdot \mathbf{h})\|_2$ is small and that the challenge is consistent with $\mathbf{z}$ and $\mathbf{h}$:

$$c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathbf{t}}} \cdot c \cdot \mathbf{t} \rceil_{\nu_{\mathbf{w}}} + \mathbf{h}).$$

To be more precise, when we check the size of $2^{\nu_{\mathbf{w}}} \cdot \mathbf{h}$, we check the size of $(2^{\nu_{\mathbf{w}}} \cdot \bar{\mathbf{h}} \mod q)$. That is, we first lift $\mathbf{h}$ over $\mathcal{R}_{q_{\mathbf{w}}}$ to $\{0, 1, \cdots, q_{\mathbf{w}-1}\}$ (i.e., it's canonical unsigned representation), multiply it by $2^{\nu_{\mathbf{w}}}$ over the integers, map it to $\mathcal{R}_q$, and finally consider the norm over modulo q. A similar comment for $2^{\nu_{\mathbf{t}}} \cdot c \cdot \mathbf{t}$ holds, where we compute $2^{\nu_{\mathbf{t}}} \cdot c \cdot \bar{\mathbf{t}}$ to be more precise. We opt using the simplified notation for better readability. As noted in Section 3.2, these will be handled with much more care in the relevant security proofs.

Correctness and Security Proof.

We prove the correctness and security of the signature scheme given in Fig. 3 under the $\mathsf{Hint\text{-}MLWE}$ and $\mathsf{SelfTargetMSIS}$ assumptions as shown in Theorem 5.1. The proof and the asymptotic parameters for which the reduction hold are deferred in Appendix C.

Theorem 5.1. *The signature scheme in Fig. 3 is correct. Moreover, for any efficient adversary making at most Q_{Sign} signing queries, it is unforgeable under the $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ and $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ assumptions.*

6 TRaccoon: Our Threshold Signature Scheme

Our 3-round threshold signature, named $\mathsf{TRaccoon}$, is given formally in Figs. 4 and 5 and proven secure in Theorem 7.2. We assume the presence of a trusted centralised party to run the key generation algorithm KeyGen. This can also be achieved with a distributed key generation algorithm; the design of a suitable DKG is outside of the scope of this work. The key generation runs Shamir's Secret Sharing algorithm in order to derive a $\mathsf{Raccoon}$ public key along with N secret key shares such that any T are sufficient to sign.[9] It takes as inputs the system parameters $\mathsf{pp}(\kappa)$, a threshold T, and a total number of parties N.

6.1 Key Generation

The key generation algorithm is defined formally in Fig. 4. As a threshold version of the plain $\mathsf{Raccoon}$ signature, the key generation algorithm (see Fig. 3) generates the public key in the same manner. A short secret $(\mathbf{s}, \mathbf{e}) \leftarrow \mathcal{D}_{\mathbf{t}}^\ell \times \mathcal{D}_{\mathbf{t}}^k$ is sampled and the public key is $(\mathbf{A}, \lfloor \mathbf{A}\mathbf{s} + \mathbf{e} \rceil_{\nu_{\mathbf{t}}})$. The main changes are the use of secret sharing and pairwise shared seeds.

Secret Sharing.

The secret $\mathbf{s}$ is be shared between users in a threshold-friendly manner. To achieve (T, N)-threshold signatures, $\mathbf{s}$ is shared using Shamir Secret Sharing as the evaluations of a polynomial of degree $T - 1$ over the set $[N]$.

[9]Strictly speaking, the underlying signature scheme is not $\mathsf{Raccoon}$ as we use discrete Gaussians instead of sum of uniforms. However, we attribute $\mathsf{Raccoon}$ as the core features (i.e., removing rejection sampling and optimisations) are the same.

Alg. 4: $\mathsf{KeyGen}(\mathsf{pp}, T, N)$

```
1:  $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}$                                  ▷ Sample matrix
2:  $(\mathbf{s}, \mathbf{e}) \leftarrow \mathcal{D}_{\mathbf{t}}^{\ell} \times \mathcal{D}_{\mathbf{t}}^{k}$              ▷ Small secret and noise
3:  $\mathbf{t} := \lfloor \mathbf{A}\cdot\mathbf{s} + \mathbf{e} \rceil_{\nu_{\mathbf{t}}}$                ▷ Part of public key in $\mathcal{R}_{q_{\mathbf{t}}}^{k}$
4:  $\mathsf{vk} := (\mathbf{A}, \mathbf{t})$
5:  $\mathbf{P} \leftarrow \mathcal{R}_q^{\ell}[X]$ with $\deg(\mathbf{P}) = T-1$, $\mathbf{P}(0) = \mathbf{s}$      ▷ Shamir Secret Sharing
6:  $(\mathbf{s}_i)_{i\in[N]} := (\mathbf{P}(i))_{i\in[N]}$                          ▷ Secret shares
7:  for $i \in [N]$ do
8:      $(\mathsf{vk}_{\mathsf{sig},i}, \mathsf{sk}_{\mathsf{sig},i}) \leftarrow \mathsf{KeyGen}_{\mathsf{sig}}(1^{\kappa})$     ▷ Standard signature keys for each user
9:      for $j \in [N]$ do
10:         $\mathsf{seed}_{i,j} \leftarrow \{0,1\}^{\kappa}$                       ▷ Pairwise-shared seeds
11: for $i \in [N]$ do
12:     $\mathsf{sk}_i := (\mathbf{s}_i, (\mathsf{vk}_{\mathsf{sig},i})_{i\in[N]}, \mathsf{sk}_{\mathsf{sig},i}, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[N]})$
13: return $(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]})$
```

Figure 4: Centralised key generation for TRaccoon. In above, we assume the key generation algorithm initialises the state of each user j.

Pairwise Shared Seeds.

Moreover, to ensure the unforgeability of the signing procedure, the individual responses by the users are additively hidden with private column mask vectors $(\mathbf{m}_i^*)_{i\in\mathsf{act}}$. These are later subtracted from the aggregated response with the publicly computable row mask vectors $\sum_{i\in\mathsf{act}} \mathbf{m}_i$. These mask vectors can be viewed as a T-out-of-T shared secret that is *non-interactively* computed *on-the-fly* during the individual ShareSign protocols, and its shared values are recomputed by the combine algorithm using the communication transcript. These shares are to be pairwise shared between users and have to be unique between sessions. To achieve this, we generate them as the result of a pseudorandom function PRF from a seed and the session id: $\mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{sid})$ with the seeds $(\mathsf{seed}_{i,j})_{(i,j)\in[N]\times[N]}$ that are generated and given to the corresponding users during the key generation.

Signing Keys.

To ensure that users agree on the view of the signing session in Round 2, they sign their view under a personal signing key. The key generation chooses a personal verification and signing key for all parties, $(\mathsf{vk}_{\mathsf{sig},i}, \mathsf{sk}_{\mathsf{sig},i})$. Alternatively, key generation can use N^2 shared symmetric keys so that all parties are pairwise linked. Then the view is authenticated using T MACs per party. This is more efficient when T is small because MACs are much smaller than post quantum digital signatures. Such symmetric keys may be derived from the pairwise shared seeds explained above.

6.2 Distributed Signing Procedure

Signing proceeds in 3 rounds. In essence, we use two T-out-of-T secret sharings. The first one is of the commitment $\mathbf{w}$ and the second is a masking term $\mathbf{m}$ that is used to mask the distributions of the partial responses in the Fiat-Shamir transform underlying Raccoon. Over the first two rounds, this commitment $\mathbf{w}$ is exchanged in a commit-reveal manner to prevent potential attacks from rushing adversaries. We note that some important yet tedious consistency checks (e.g., check whether session for sid exists) in our signing protocol is outsourced to Appendix D, Fig. 19 for better readability.

Alg. 5: $\mathsf{ShareSign}_1(\mathsf{state}, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$

```
1: assert{ $\mathsf{ConsistCheck}_1(\mathsf{state}, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$ }    ▷ Consistency checks
2: $(\mathbf{r}_j, \mathbf{e}'_j) \leftarrow \mathcal{D}^\ell_{\mathbf{w}} \times \mathcal{D}^k_{\mathbf{w}}$    ▷ Sample small ephemeral randomness and small noise
3: $\mathbf{w}_j := \mathbf{A} \cdot \mathbf{r}_j + \mathbf{e}'_j$    ▷ MLWE commitment in $\mathcal{R}^k_q$ without rounding
4: $\mathsf{cmt}_j := \mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_j)$    ▷ Hash commitment
5: Fetch $(\mathsf{seed}_{j,i})_{i \in \mathsf{act}}$ from $\mathsf{state.sk}$
6: $\mathbf{m}_j := \sum_{i \in \mathsf{act}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{sid})$    ▷ Compute row blinder in $\mathcal{R}^\ell_q$
7: $\mathsf{state.session}[\mathsf{sid}] := \{\mathsf{sid}, \mathsf{act}, \mathsf{msg}, 1, \{\mathbf{r}_j, \mathbf{w}_j, \mathsf{cmt}_j, \mathbf{m}_j\}, \emptyset\}$    ▷ New session state
8: return $\mathsf{contrib}_1[j] := (\mathsf{cmt}_j, \mathbf{m}_j)$
```

Alg. 6: $\mathsf{ShareSign}_2(\mathsf{state}, \mathsf{sid}, \mathsf{contrib}_1)$

```
1: assert{ $\mathsf{ConsistCheck}_2(\mathsf{state}, \mathsf{sid}, \mathsf{contrib}_1)$ }    ▷ Consistency checks
2: Fetch $\mathsf{sk}_{\mathsf{sig},j}$ from $\mathsf{state.sk}$
3: $\sigma_j \leftarrow \mathsf{Sign}_{\mathsf{sig}}(\mathsf{sk}_{\mathsf{sig},j}, \mathsf{sid} \,||\, \mathsf{act} \,||\, \mathsf{msg} \,||\, \mathsf{contrib}_1)$
        ▷ Sign first-round contribution with standard signature
4: Fetch $\mathbf{w}_j$ from $\mathsf{state.sessions}[\mathsf{sid}].\mathsf{internal}$    ▷ Recall $\mathbf{w}_j$ from $\mathsf{ShareSign}_1$
5: $\mathsf{state.session}[\mathsf{sid}] := \{\mathsf{sid}, \mathsf{act}, \mathsf{msg}, 2, \{\mathbf{r}_j, \mathbf{w}_j, \mathsf{cmt}_j, \mathbf{m}_j\}, \mathsf{contrib}_1\}$    ▷ Update session state
6: return $\mathsf{contrib}_2[j] := (\mathbf{w}_j, \sigma_j)$
```

Alg. 7: $\mathsf{ShareSign}_3(\mathsf{state}, \mathsf{sid}, \mathsf{contrib}_2)$

```
1: assert{ $\mathsf{ConsistCheck}_3(\mathsf{state}, \mathsf{sid}, \mathsf{contrib}_2)$ }    ▷ Consistency checks
2: Let $\mathsf{session} = \mathsf{state.sessions}[\mathsf{sid}]$
3: Fetch $(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$ from $\mathsf{session}$
4: Fetch $\mathbf{r}_j$ from $\mathsf{session.internal}$ and $\mathbf{s}_j, (\mathsf{vk}_{\mathsf{sig},i})_{i \in [N]}, (\mathsf{seed}_{i,j})_{i \in \mathsf{act}}$ from $\mathsf{state.sk}$
5: Fetch $\mathsf{contrib}_1 = (\mathsf{cmt}_i, \mathbf{m}_i)_{i \in \mathsf{act}}$ from $\mathsf{session.contrib}_1$
6: Parse $\mathsf{contrib}_2 = (\mathbf{w}_i, \sigma_i)_{i \in \mathsf{act}}$
7: for $i \in \mathsf{act}$ do
8:     assert{ $\mathsf{cmt}_i = \mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{msg}, \mathsf{act}, \mathbf{w}_i)$ }
            ▷ Check consistency of hash commitments
9:     assert{ $\mathsf{Verify}_{\mathsf{sig}}(\mathsf{vk}_{\mathsf{sig},i}, \mathsf{sid} \,||\, \mathsf{act} \,||\, \mathsf{msg} \,||\, \mathsf{contrib}_1, \sigma_i) = 1$ }
            ▷ Check users used same first-round contribution
10: $\mathbf{w} := \lfloor \sum_{i \in \mathsf{act}} \mathbf{w}_i \rceil_{\nu_{\mathsf{w}}}$    ▷ Aggregated rounded commitment in $\mathcal{R}^k_{q_{\mathsf{w}}}$
11: $c := \mathsf{H}_c(\mathsf{state.vk}, \mathsf{msg}, \mathbf{w})$    ▷ Global challenge
12: $\mathbf{m}^*_j := \sum_{i \in \mathsf{act}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{sid})$    ▷ Compute column blinder in $\mathcal{R}^\ell_q$
13: $\mathbf{z}_j := c \cdot \lambda_{\mathsf{act},j} \cdot \mathbf{s}_j + \mathbf{r}_j + \mathbf{m}^*_j$    ▷ Individual response in $\mathcal{R}^\ell_q$
14: return $\mathsf{contrib}_3[j] := \mathbf{z}_j$
```

Alg. 8: $\mathsf{Combine}(\mathsf{vk}, \mathsf{sid}, \mathsf{msg}, \mathsf{contrib}_1, \mathsf{contrib}_2, \mathsf{contrib}_3)$

```
1: Parse $\mathsf{contrib}_1 = (\mathsf{cmt}_i, \mathbf{m}_i)_{i \in \mathsf{act}}$, $\mathsf{contrib}_2 = (\mathbf{w}_i, \sigma_i)_{i \in \mathsf{act}}$, $\mathsf{contrib}_3 = (\mathbf{z}_i)_{i \in \mathsf{act}}$
2: Parse $\mathsf{vk} = (\mathbf{A}, \mathbf{t})$
3: $\mathbf{w} := \lfloor \sum_{i \in \mathsf{act}} \mathbf{w}_i \rceil_{\nu_{\mathsf{w}}}$    ▷ Aggregated rounded commitment in $\mathcal{R}^k_{q_{\mathsf{w}}}$
4: $\mathbf{z} := \sum_{i \in \mathsf{act}} (\mathbf{z}_i - \mathbf{m}_i)$    ▷ Aggregated response shifted by column blinders in $\mathcal{R}^\ell_q$
5: $c := \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$    ▷ Global challenge
6: $\mathbf{y} := \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathsf{t}}} \cdot c \cdot \mathbf{t} \rceil_{\nu_{\mathsf{w}}}$    ▷ Intermediate value in $\mathcal{R}^k_{q_{\mathsf{w}}}$
7: $\mathbf{h} := \mathbf{w} - \mathbf{y}$    ▷ Hint in $\mathcal{R}^k_{q_{\mathsf{w}}}$
8: return $\mathsf{sig} := (c, \mathbf{z}, \mathbf{h})$
```

Figure 5: Signing procedure for TRaccoon. In above, we omit the subscript and assume state is the state of party $j \in \mathsf{act}$. The consistency checks are described in Fig. 19.

First round.

Every party j inside the signing set act generates their (rounded) MLWE commitment share $\mathbf{w}_j$ encoding the ephemeral randomness $\mathbf{r}_j$. In parallel, they use their pairwise-shared seeds $(\mathsf{seed}_{j,i})_{i\in\mathsf{act}}$ and the session id sid to compute a public *row* mask $\mathbf{m}_j = \sum_{i\in\mathsf{act}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{sid})$. We recall Fig. 1 for a pictorial explanation of the mask term. They then publish $\mathbf{m}_j$, as well as a hash commitment cmt_j of $\mathbf{w}_j$.

Second round.

Each party j reveals their MLWE commitment share $\mathbf{w}_j$. Additionally they sign their current view of the signing session under their personal secret keys $\mathsf{sk}_{\mathsf{sig},j}$ (or MAC key). The commit-reveal is a standard technique so that the adversary does not generate its commitments in accordance with those of the honest users (see for instance [BN06]).

Third round.

All parties checks that the received commitment share $\mathbf{w}_j$ is consistent with the hash commitments in the first round and that all the signatures from the second round verify. It then computes the resulting global commitment $\mathbf{w}$ as $\lfloor \sum_{i\in\mathsf{act}} \mathbf{w}_i \rceil_{\nu_\mathbf{w}}$ using the messages they received from the first two rounds. Then, parties compute the signature challenge $c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$ for themselves.

The parties further compute a secret *column* mask $\mathbf{m}^*_j = \sum_{i\in\mathsf{act}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{sid})$ using their pairwise-shared seeds $(\mathsf{seed}_{i,j})_{i\in\mathsf{act}}$ and the session id sid. They use this to define their response share $\mathbf{z}_j = c \cdot \lambda_{\mathsf{act},j} \cdot \mathbf{s}_j + \mathbf{r}_j + \mathbf{m}^*_j$ for $\mathbf{s}_j$ their secret share and $\lambda_{\mathsf{act},j}$ a Lagrange coefficient corresponding to the active signing set act. Since $\sum_{i\in\mathsf{act}} \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i = \mathbf{s}$ sums to the full secret $\mathbf{s}$, these shares sum to a valid response shifted by the column masks. Here, the main observation is that $\sum_{i\in\mathsf{act}} \mathbf{m}_i = \sum_{i\in\mathsf{act}} \mathbf{m}^*_i$ (see Fig. 1). Lastly, they return the response share $\mathbf{z}_j$.

Combination.

Once all parties have completed all rounds, the coordinator runs a combine algorithm to compute the signature. This algorithm simply rounds the sum of the MLWE commitments to get the full commitment $\mathbf{w} = \lfloor \sum_{i\in\mathsf{act}} \mathbf{w}_i \rceil_{\nu_\mathbf{w}}$. The challenge is $c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$. The response is the sum of the response shares, subtracted with the sum of the public column masks: $\mathbf{z} = \sum_{i\in\mathsf{act}} (\mathbf{z}_i - \mathbf{m}_i)$. Finally, the hint is computed as $\mathbf{h} = \mathbf{w} - \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_\mathbf{t}} \cdot c \cdot \mathbf{t} \rceil_{\nu_\mathbf{w}}$ where $\mathsf{vk} = (\mathbf{A}, \mathbf{t})$. It returns a signature $(c, \mathbf{z}, \mathbf{h})$ of $\mathsf{Raccoon}$.

Verification.

We do not explicitly define the verification algorithm since it is identical to those of the plain $\mathsf{Raccoon}$ signature scheme in Fig. 3.

Remark 6.1 (Statefullness). The signing algorithm requires signers never to respond with respect to the same session ID twice. They must store all session IDs that they have used previously and abort if they receive a repeated request.

7 Correctness and Security Reduction

In this section, we provide the correctness and security proofs of our threshold signature scheme $\mathsf{TRaccoon}$ as described in Theorem 7.2.

7.1 Asymptotic Parameters

We first give a asymptotic parameter for which the scheme can be proven correct and secure. For reference, we recall the set of parameters used by the scheme in the following Table 1. Note the restriction on $(q_\mathbf{t}, q_\mathbf{w})$ allows us to perform rounding operations nicely (see Lemma 3.2).

Parameter	Explanation
$\mathcal{R}_q$	Polynomial ring $\mathcal{R}_q = \mathbb{Z}[X]/(q, X^n + 1)$
(k, ℓ)	Dimension of public matrix $\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$
$(\mathcal{D}_{\mathbf{t}}, \sigma_{\mathbf{t}})$	Gaussian distribution with width $\sigma_{\mathbf{t}}$ used for the verification key $\mathbf{t}$
$(\mathcal{D}_{\mathbf{w}}, \sigma_{\mathbf{w}})$	Gaussian distribution with width $\sigma_{\mathbf{w}}$ used for the commitment $\mathbf{w}$
$\nu_{\mathbf{t}}$	Amount of bit dropping performed on verification key
$\nu_{\mathbf{w}}$	Amount of bit dropping performed on (aggregated) commitment
$(q_{\mathbf{t}}, q_{\mathbf{w}})$	Rounded moduli satisfying $(q_{\mathbf{t}}, q_{\mathbf{w}}) := (\lfloor q/2^{\nu_{\mathbf{t}}} \rceil, \lfloor q/2^{\nu_{\mathbf{w}}} \rceil) = (\lfloor q/2^{\nu_{\mathbf{t}}} \rceil, \lfloor q/2^{\nu_{\mathbf{w}}} \rceil)$
$(\mathcal{C} \subset R_q, \omega)$	Challenge set $\{c \in \mathcal{R}_q \mid \|c\|_\infty = 1 \wedge \|c\|_1 = \omega\}$ s.t. $|\mathcal{C}| \geq 2^\kappa$
B_2	Two-norm bound on the signature

Table 1: Overview of parameters used in the (variant of) Raccoon signature.

For unforgeability, we require the $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ and $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ problems to be hard. More specifically, we require the following constraints.

- $\sigma \geq \sqrt{\ell} \cdot \omega_{\mathsf{asymp}}(\sqrt{\log n})$ for Lemma A.3 (hardness of MLWE).
- $B_{\mathsf{hint}} = Q_{\mathsf{Sign}} \cdot \omega \cdot \left(1 + n\frac{1}{\sqrt{Q_{\mathsf{Sign}}}}(\kappa + 1 + 2\log(n))\right)$ and $\frac{1}{\sigma^2} = 2 \cdot \left(\frac{1}{\sigma_{\mathbf{t}}^2} + \frac{B_{\mathsf{hint}}}{\sigma_{\mathbf{w}}^2}\right)$ for Lemmas A.5 and B.2 (reduction from $\mathsf{Hint\text{-}MLWE}$ to MLWE).
- $q > B_{\mathsf{msis}}\sqrt{nk} \cdot \omega_{\mathsf{asymp}}(\log(nk))$ for Lemma A.4 (hardness of MSIS).
- $B_{\mathsf{msis}} = 2B_{\mathsf{stmsis}}$ for Lemma B.1 (reduction from $\mathsf{SelfTargetMSIS}$ to MSIS).

In the above, note that Q_{Sign} denotes the maximum signature query an adversary can perform.

Candidate Asymptotic Parameters. We give a set of asymptotic parameters which fit the above constraints. The only difference between the asymptotic parameter selections from the non-thresholdised Raccoon signature is that we have a factor of $\sqrt{T}$ in the bound B (see Appendix C.1). This is caused by aggregating T commitments $(\mathbf{w}_i)_{i\in\mathsf{act}}$ for $\mathsf{act} \subseteq [N] \wedge |\mathsf{act}| = T$. In particular, we recover the same parameters as the non-thresholdised Raccoon signature by setting $T = 1$.

- $n, \ell, k = \mathsf{poly}(\kappa)$ such that $n \geq \kappa$.
- $\omega = \omega_{\mathsf{asymp}}(1)$ for $|\mathcal{C}| \geq 2^\kappa$ for Lemma B.1 (hardness of MSIS).
- $(\sigma_{\mathbf{t}}, \sigma_{\mathbf{w}}) = \left(2\sqrt{\ell} \cdot \log n, 2\sqrt{B_{\mathsf{hint}} \cdot \ell} \cdot \log n\right)$.
- $\nu_{\mathbf{t}}, \nu_{\mathbf{w}} = O(\log \lambda)$, where $\nu_{\mathbf{w}} \geq 4$ for correctness (see Section 7.2).
- $B_{2,T} = e^{1/4} \cdot (\omega\,\sigma_{\mathbf{t}} + \sqrt{T}\,\sigma_{\mathbf{w}})\sqrt{n(k+\ell)} + (\omega \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1}) \cdot \sqrt{nk}$ for correctness (see Section 7.2).
- $B_{\mathsf{stmsis}} = B_{2,T} + \sqrt{\omega} + (\omega \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1}) \cdot \sqrt{nk}$ for Lemma 7.4.
- q is the smallest prime larger than $2B_{\mathsf{stmsis}} \cdot \sqrt{nk} \cdot \log(nk)^2$ such that $(q, \nu_{\mathbf{t}}, \nu_{\mathbf{w}})$ satisfy the condition in Table 1.

7.2 Correctness

The following establishes the correctness of the threshold signature scheme $\mathsf{TRaccoon}$ as described in Theorem 7.2

Lemma 7.1 (Correctness)**.** *The threshold signature scheme* $\mathsf{TRaccoon}$ *in Theorem 7.2 is correct if* $\nu_{\mathbf{w}} \geq 4$ *and:*

$$B_{2,T} = e^{1/4} \cdot (\omega\, \sigma_{\mathbf{t}} + \sqrt{T}\, \sigma_{\mathbf{w}}) \sqrt{n(k+\ell)} + (\omega \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1}) \cdot \sqrt{nk}. \tag{10}$$

Proof. It is clear that the check $\{c = c'\}$ holds. In the proof, we focus on the check of the L_2-norm of the signature. First of all, the aggregated response satisfies

$$\mathbf{z} = \sum_{i \in \mathsf{act}} (\mathbf{z}_i - \mathbf{m}_i) = \sum_{i \in \mathsf{act}} (c \cdot \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i + \mathbf{r}_i + \mathbf{m}_i^* - \mathbf{m}_i) = c \cdot \mathbf{s} + \underbrace{\sum_{i \in \mathsf{act}} \mathbf{r}_i}_{=:\mathbf{r}_{\mathsf{sid}}},$$

where the third equality follows from the correctness of the Shamir secret sharing scheme and the fact that $\sum_{i \in \mathsf{act}} \mathbf{m}_i = \sum_{i \in \mathsf{act}} \mathbf{m}_i^*$. Moreover, the aggregate commitment satisfies

$$\mathbf{w} = \left\lfloor \sum_{i \in \mathsf{act}} \mathbf{w}_i \right\rceil_{\nu_{\mathbf{w}}} = \left\lfloor \mathbf{A} \cdot \sum_{i \in \mathsf{act}} \mathbf{r}_i + \underbrace{\sum_{i \in \mathsf{act}} \mathbf{e}_i'}_{=:\mathbf{e}_{\mathsf{sid}}'} \right\rceil_{\nu_{\mathbf{w}}}.$$

It is now easy to see that $\mathsf{sig} = (c, \mathbf{z}, \mathbf{h})$ can be viewed as a non-thresholdised standard (Raccoon) signature where the noise is amplified. In particular, the threshold signature scheme outputs a signature that is identical to the Raccoon signature with the only difference that the commitment $\mathbf{w}$ is generated from a *sum* of T discrete Gaussians $\mathcal{D}_{\mathbf{w}}$. Using convolution of discrete Gaussian distributions, we can simply replace $\sigma_{\mathbf{w}}$ with $\sqrt{T}\, \sigma_{\mathbf{w}}$ in Lemma C.1. We note that the standard deviation is well above the smoothing parameter for convolution of discrete Gaussians to be justified. This completes the proof. □

7.3 Unforgeability

In this section, we prove the security of our threshold signature scheme $\mathsf{TRaccoon}$ as described in Theorem 7.2. The statement assumes the asymptotic parameter selections in Section 7.1.

Theorem 7.2. *The threshold signature scheme* $\mathsf{TRaccoon}$ *described in Fig. 5 is unforgeable under the unforgeability of the (non-thresholdised) signature scheme, pseudorandomness of* PRF*, the* $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ *and* $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ *assumptions.*

Formally, for any adversary $\mathcal{A}$ *against the unforgeability game making at most* Q_{H} *and* Q_{Sign} *queries to the random oracles* $\mathsf{H}_c, \mathsf{H}_{\mathsf{com}}$ *and the signing oracle, respectively, there exists adversaries* $\mathcal{B}_{\mathsf{Sign}}$*,* $\mathcal{B}_{\mathsf{PRF}}$*,* $\mathcal{B}$*, and* $\mathcal{B}'$ *against the unforgeability of the signature scheme, pseudorandomness of* PRF*, and* $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ *and* $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ *problems, respectively, such that*

$$\begin{aligned}\mathsf{Game}_{\mathcal{A}}^{\mathsf{ts\text{-}uf}}(\kappa) \leq N \cdot \mathsf{Adv}_{\mathcal{B}_{\mathsf{Sign}}}^{\mathsf{sig\text{-}uf}}(\kappa) + \mathsf{Adv}_{\mathcal{B}_{\mathsf{PRF}}}^{\mathsf{PRF}}(\kappa) + \frac{(Q_{\mathsf{H}}+1) \cdot Q_{\mathsf{Sign}}}{2^{n-1}} \\ + \frac{Q_{\mathsf{H}} + Q_{\mathsf{H}}^2}{2^{2\kappa}} + \mathsf{Adv}_{\mathcal{B}}^{\mathsf{Hint\text{-}MLWE}}(\kappa) + \mathsf{Adv}_{\mathcal{B}'}^{\mathsf{SelfTargetMSIS}}(\kappa)\end{aligned}$$

where $\mathsf{Time}(\mathcal{B}_{\mathsf{Sign}}), \mathsf{Time}(\mathcal{B}_{\mathsf{PRF}}), \mathsf{Time}(\mathcal{B}), \mathsf{Time}(\mathcal{B}') \approx \mathsf{Time}(\mathcal{A})$*.*

To prove Theorem 7.2, we proceed using a series of hybrid games. The aim is to arrive at a hybrid in which a reduction can fully simulate the adversaries view given just a $\mathsf{SelfTargetMSIS}$ challenge. The techniques used in this reduction are new to this work. Before providing the full formal proof of Theorem 7.2, let us provide an overview.

Proof Overview. The aim of the hybrids is to switch into a setting where the masking sums $\mathbf{m}_j$, commitments $\mathbf{w}_j$ and responses $\mathbf{z}_j$ effectively contain no useful information. We proceed as follows.

1. Hybrid_1 corresponds to the real unforgeability game.

2. Hybrid_2 asserts that all honest parties in a session have signed the state. This switch is indistinguishable assuming the signature scheme is unforgeable. In Hybrid_2 an adversary cannot cause honest parties to have a differing view of the signing transcript during the second round of the signing procedure. Here, we note that we can instead use MACs to authenticate their states (see Remark 7.5 for detail).

3. Hybrid_3 switched to a game in which the masks $\mathbf{m}_{i,j}$ are randomly sampled from the full domain whenever both i and j are honest. Distinguishing this game from the previous one allows breaking the pseudorandomness of the PRF used during seed generation.

4. Hybrid_4 samples the honest parties full masks $\mathbf{m}_j$ and $\mathbf{m}_j^*$ uniformly at random (apart from the last honest $\mathbf{m}_j^*$ which is chosen to be consistent). This hybrid is statistically indistinguishable from the previous.

5. Hybrid_5 switches to a game in which the reduction only chooses the full nonce $\mathbf{w}$ in OSgn_2, after the adversary has committed itself to its partial nonces during the first signing round. We program $\mathsf{H}_{\mathsf{com}}$ for consistency. This will later allow us to program the signature challenge inside the signing oracle before the adversary can learn $\mathbf{w}$.

   This game remains identical to the previous game unless we have to reprogram a value that has already been queried by the adversary. The value being programmed is sampled from the $\mathcal{D}^{\mathsf{bd\text{-}MLWE}}_{q,\ell,k,\sigma,\nu}(\mathbf{A})$ distribution (see Definition 3.7); the chance of collision is small because the distribution has high min-entropy.

6. Hybrid_6 selects the challenge c in OSgn_1, before revealing the full nonce $\mathbf{w}$. The full nonce is determined only after the last honest party in the signing session, party h, reveals its partial nonce $\mathbf{w}_h$. We program H_c for consistency. This is indistinguishable from Hybrid_5 provided that the adversary has not queried H_c on the nonce $\mathbf{w}$ before and provided that the adversary is uniquely committed to its own partial nonces $\mathbf{w}_i$. Both are true provided the $\mathcal{D}^{\mathsf{bd\text{-}MLWE}}_{q,\ell,k,\sigma,\nu}(\mathbf{A})$ distribution has high min-entropy and the image of $\mathsf{H}_{\mathsf{com}}$ is large enough.

7. Hybrid_7 changes how the reduction is simulating the signatures such that it chooses $\mathbf{z}_h$ before $\mathbf{w}_h$. Here the hth party is the last honest party in the signing session, $\mathbf{w}_h$ is their partial nonce, and $\mathbf{z}_h$ is their partial response. This is statistically identical to Hybrid_6.

8. Hybrid_8 chooses the partial responses $\mathbf{z}_j$ for all honest $j \neq h$ randomly and edits how the $\mathbf{m}_h^*$ are sampled for consistency. Here the hth party is the last honest party in the signing session. This hybrid crucially relies on the fact that all honest parties have a consistent view of the signing transcript, which is guaranteed due to the modification we made in Hybrid_2. This is statistically identical to Hybrid_7.

9. Hybrid_9 is more interesting; in essence we simulate the honest signers such that they only use the secret key in the form of an $\mathsf{Hint\text{-}MLWE}$ sample. In the Hybrid_8 we have

$$\begin{aligned}
\mathbf{w}_h &\coloneqq \mathbf{A} \cdot (\mathbf{z}_h - \mathbf{m}_h^*) - c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{A} \cdot \mathbf{s}_h + \mathbf{e}'_h \\
\mathbf{z}_h &\coloneqq c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{s}_h + \mathbf{r}_h + \mathbf{m}_h^*
\end{aligned}$$

   for h the last honest party in the signing session and $\lambda_{\mathsf{act},h}$ the h-th Lagrange coefficient. Recall that Lagrange coefficients are not small. The reduction does not have the means to compute $c \cdot \mathbf{A} \cdot \mathbf{s}_h$ without also introducing a large error term $\lambda_{\mathsf{act},h} \cdot c \cdot \mathbf{e}$. This is where our key insight comes in. In Hybrid_9 we instead compute

$$\begin{aligned}
\mathbf{w}_h &\coloneqq \mathbf{A} \cdot \mathbf{z}_{\mathsf{sid}} - c \cdot \widehat{\mathbf{t}} + \mathbf{z}'_{\mathsf{sid}} \\
\mathbf{z}_h &\coloneqq \mathbf{z}_{\mathsf{sid}} - c \cdot \sum_{i \in \mathsf{corrupt}_{\mathsf{sid}}} \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i + \text{terms independent of } \mathbf{s}_h
\end{aligned}$$

where $(c, \mathbf{z}_{\mathsf{sid}}, \mathbf{z}'_{\mathsf{sid}})$ is a $\mathsf{Hint\text{-}MLWE}$ sample.

This alternative computation is statistically equivalent. The private masking $\mathbf{m}^*_h$ term set in Hybrid_8 includes a $\sum_{i \in \mathsf{honest}, i \neq h} c \cdot \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i$ term where i is taken over honest parties with $i \neq h$. We thus move the interesting dependence on $\mathbf{s}_i$ into a single honest signing party for each session. This means that $\mathbf{w}_h$ and $\mathbf{z}_h$ can be alternatively computed to have a dependence on $\sum_{i \in \mathsf{act}} (\lambda_{\mathsf{act},i} \cdot c \cdot \mathbf{A} \cdot \mathbf{s}_i)$. By design this is equal to $c \cdot \mathbf{A} \cdot \mathbf{s}$.

10. The final Hybrid_{10} samples the public key and corrupt secret key shares randomly, and in particular it does not retain secret key shares for the honest parties. The $\mathsf{Hint\text{-}MLWE}$ samples required for computing OSgn_i responses are generated using some $(\mathbf{s}, \mathbf{e})$ which are independent from the public key. In particular this means that all challenger responses in the hybrid are now independent from the public key. This is indistinguishable from Hybrid_9 by the $\mathsf{Hint\text{-}MLWE}$ assumption and we formally show indistinguishability in Lemma 7.3.

    Finally, we reduce Hybrid_{10} from $\mathsf{SelfTargetMSIS}$ directly. We formally give the reduction in Lemma 7.4.

This completes the proof overview. □

We now provide the full formal proof of Theorem 7.2 following the proof structure explained above.

Proof. Let $\mathcal{A}$ be an adversary against the unforgeability of the threshold Raccoon signature scheme. We use a series of hybrid games where Hybrid_1 corresponds to the original unforgeability game. The final Hybrid_{10} is designed such that a reduction $\mathcal{B}'$ can simulate the hybrid game given a $\mathsf{SelfTargetMSIS}$ instance, and extract a solution of this problem from a successful adversary.

Hybrid_1. This is the unforgeability security game.

Hybrid_2. In this hybrid, the challenger checks all the signatures observed in the third round was generated by the challenger. This is formally depicted in Fig. 6.

The asserted condition can be checked to hold in a straightforward manner by invoking the unforgeability of the (non-thresholdised) signature scheme. In particular, we can construct an adversary $\mathcal{B}_{\mathsf{Sign}}$ against the unforgeability of the signature scheme such that

$$\left| \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}(\kappa) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1}(\kappa) \right| \leq N \cdot \mathsf{Adv}_{\mathcal{B}_{\mathsf{Sign}}}^{\mathsf{sig\text{-}uf}}(\kappa).$$

The reduction $\mathcal{B}_{\mathsf{Sign}}$ takes as input a challenge public key $\mathsf{vk}_{\mathsf{sig}}$ and aims to output a forgery. It has access to a signing oracle. Then $\mathcal{B}_{\mathsf{Sign}}$ simulates the view of the Hybrid_1 challenger to $\mathcal{A}$ by:

1. Computing $\mathsf{vk} = (\mathbf{A}, \mathbf{t})$ and $(\mathbf{s}_i)_{i \in [N]}$ the same as in Hybrid_1;

2. Choosing random $i^* \in \mathsf{honest}$ and setting $\mathsf{vk}_{\mathsf{sig},i^*} = \mathsf{vk}_{\mathsf{sig}}$.

3. Sampling $(\mathsf{vk}_{\mathsf{sig},i}, \mathsf{sk}_{\mathsf{sig},i}) \leftarrow \mathsf{KeyGen}_{\mathsf{sig}}(1^\kappa)$ for $i \in \mathsf{honest}, i \neq i^*$;

and running $\mathcal{A}^{\mathsf{H},(\mathsf{OSgn}_i)_{i\in[3]}}(\mathsf{vk}, (\mathsf{sk}_i)_{i \in \mathsf{corrupt}})$.

When $\mathcal{A}$ queries its signing oracles it responds the same as in Hybrid_1 except that signatures from i^* are generated by querying the signing oracle. If $\mathcal{A}$ provides a signature σ on some message $(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1) \notin \mathsf{Signed}[i^*]$ then $\mathcal{B}_{\mathsf{Sign}}$ returns $((\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1), \sigma)$ as its forgery.

Since i^* is information theoretically hidden from $\mathcal{A}$, the probability that $\mathcal{B}$ correctly guesses which public key $\mathcal{A}$ provides a forgery for is $\frac{1}{N}$.

```
Hybrid_2
1: $\mathsf{Signed}[\cdot] := \bot$                                ▷ List to maintain standard signature of honest users
2: $L_{\mathsf{Sign}}, L_{\mathsf{H}} := \emptyset$
3: $(N, T, \mathsf{corrupt}) \leftarrow \mathcal{A}(\mathsf{pp})$
4: $\mathsf{honest} := [N] \backslash \mathsf{corrupt}$
5: $\big(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}\big) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, T, N)$
6: $(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathsf{OSgn}_i(\cdot))_{i\in[3]}}(\mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{corrupt}})$
7: if $(\mathsf{msg} \in L_{\mathsf{Sign}})$ or $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) = 0$ then
8:     return $0$
9: return $1$

    OSgn_1(j, sid, contrib_1)
        ▷▷▷ Identical to Hybrid_1.OSgn_1 ◁◁◁

    OSgn_2(j, sid, act, msg)
    1: Fetch $\mathsf{sk}_{\mathsf{sig},j}$ from state.sk
    2: $\sigma_j \leftarrow \mathsf{Sign}_{\mathsf{sig}}(\mathsf{sk}_{\mathsf{sig},j}, \mathsf{sid} \,||\, \mathsf{act} \,||\, \mathsf{msg} \,||\, \mathsf{contrib}_1)$
    3: $\mathsf{Signed}[j] \leftarrow \mathsf{Signed}[j] \cup \{(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1)\}$
    4: Fetch $\mathbf{w}_j$ from state.sessions[sid].internal
    5: return $\mathsf{contrib}_1[j] := (\mathbf{w}_j, \sigma_j)$

    OSgn_3(j, sid, contrib_2)
    1: Fetch (sid, act, msg) from session
    2: Fetch $\mathsf{contrib}_1 = (\mathsf{cmt}_i, \mathbf{m}_i)_{i\in\mathsf{act}}$ from session.contrib_1
    3: for $i \in \mathsf{act}$ do
    4:     for $i \in \mathsf{honest}_{\mathsf{sid}}$ do
    5:         assert{ $(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1) \in \mathsf{Signed}[i]$ }
                                                        ▷ Check no signature was forged
        ▷▷▷ Otherwise, identical to Hybrid_1.OSgn_3 ◁◁◁
```

Figure 6: The second hybrid game used in the proof of Theorem 7.2. Differences from the previous hybrid are highlighted. Throughout the proof, $\mathsf{honest}_{\mathsf{sid}}$ and $\mathsf{corrupt}_{\mathsf{sid}}$ denote $\mathsf{honest} \cap \mathsf{act}$ and $\mathsf{corrupt} \cap \mathsf{act}$ for session id sid. For readability, we omit the consistency checks and do not explicitly discuss how the hybrids update their state.

```
Hybrid$_3$
  ▷▷▷ Identical to Hybrid$_2$ ◁◁◁

  OSgn$_1(j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$
  1: $L_{\mathsf{Sign}} \leftarrow L_{\mathsf{Sign}} \cup \{\mathsf{msg}\}$
  2: $(\mathbf{r}_j, \mathbf{e}'_j) \leftarrow \mathcal{D}^{\ell}_{\mathbf{w}} \times \mathcal{D}^{k}_{\mathbf{w}}$
  3: $\mathbf{w}_j := \mathbf{A} \cdot \mathbf{r}_j + \mathbf{e}'_j$
  4: $\mathsf{cmt}_j := \mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_j)$
  5: Fetch $(\mathsf{seed}_{j,i})_{i \in \mathsf{act}}$ from state.sk
  6: for $i \in \mathsf{corrupt}_{\mathsf{sid}}$ do
  7:     $\mathbf{m}_{j,i} := \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{sid})$
  8: for $i \in \mathsf{honest}_{\mathsf{sid}}$ do
  9:     $\mathbf{m}_{j,i} \leftarrow \mathcal{R}^k_q$        ▷ Sample individual mask uniformly when $j, i \in \mathsf{honest}_{\mathsf{sid}}$
  10: $\mathbf{m}_j := \sum_{i \in \mathsf{act}} \mathbf{m}_{j,i}$
  11: return $\mathsf{contrib}_1[j] := (\mathsf{cmt}_j, \mathbf{m}_j)$

    OSgn$_2(j, \mathsf{sid}, \mathsf{contrib}_1)$
      ▷▷▷ Identical to Hybrid$_2$.OSgn$_2$ ◁◁◁
```

Figure 7: The third hybrid game used in the proof of Theorem 7.2. Differences from the previous hybrid are highlighted. For readability, we omit OSgn$_3$ in all the remaining hybrids as we use the same description as those of Hybrid$_2$.

Hybrid$_3$. In this hybrid, for all $(i, j) \in \mathsf{honest}_{\mathsf{sid}} := \mathsf{honest} \cap \mathsf{act}$, the challenger samples $\mathbf{m}_{i,j} \leftarrow \mathcal{R}^k_q$ as opposed to as the output of a pseudorandom function. This is formally depicted in Fig. 7. Similarly to the previous hybrid, it is easy to check that assuming the pseudorandomness of the PRF, this hybrid is indistinguishable from Hybrid$_2$. In particular, we can construct an adversary $\mathcal{B}_{\mathsf{PRF}}$ against the pseudorandomness of PRF such that

$$\left| \mathsf{Adv}^{\mathsf{Hybrid}_3}_{\mathcal{A}}(\kappa) - \mathsf{Adv}^{\mathsf{Hybrid}_2}_{\mathcal{A}}(\kappa) \right| \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_{\mathsf{PRF}}}(\kappa).$$

Here, note that we do not incur the factor N reduction loss as we rely on the multi-instance version of pseudorandomness (see Definition A.6).

Hybrid$_4$. In this hybrid, the challenger samples the *row* and *column* masks $\mathbf{m}_i$ and $\mathbf{m}^*_i$, respectively, at random for all honest parties except h (the last honest party in the signing set) where it is uniquely defined by the other $\mathbf{m}$ values. (See Fig. 1 for a pictorial example of the masks.) This is formally depicted in Fig. 8, where the challenger uses two lists $\mathsf{Committed}[\cdot]$ and $\mathsf{LastSigner}[\cdot]$ to maintain how many honest signers opened a specific session sid. Here, note that by the consistency check (see Fig. 19), sid includes the active signer set act and the only honest users that can generate a valid first-round message are limited to $\mathsf{honest}_{\mathsf{sid}} \subseteq \mathsf{act}$. Moreover, we generate the column masks $\mathbf{m}_i$ in the first round but this is without loss of generality as they are fixed once sid is defined.

Let us analyze the distribution of the masks. First, the row masks $(\mathbf{m}_i)_{i \in \mathsf{honest}_{\mathsf{sid}}}$ were distributed uniformly at random in Hybrid$_3$ because these masks include the individual masks $\mathbf{m}_{i,i}$ used nowhere else. This is identical to Hybrid$_4$. Further, in Hybrid$_3$, for all $i \in \mathsf{honest}_{\mathsf{sid}}$ and $i \neq h := \mathsf{LastSigner}[\mathsf{sid}]$, the column mask $\mathbf{m}^*_i = \sum_{j \in \mathsf{act}} \mathbf{m}_{j,i}$ includes $\mathbf{m}_{h,i}$, used nowhere before user h is invoked on OSgn$_1$. Therefore, $\mathbf{m}^*_i$ is

```
Hybrid_4
1: $\mathsf{Committed}[\cdot], \mathsf{LastSigner}[\cdot] \coloneqq \bot$          ▷ Lists to maintain simulation state for sid
   ▷▷▷ Remaining lines are identical to Hybrid_3 ◁◁◁

   $\mathsf{OSgn}_1(j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$
   1: $\mathsf{Committed}[\mathsf{sid}] \leftarrow \mathsf{Committed}[\mathsf{sid}] \cup \{j\}$     ▷ Store honest users starting session sid
   2: $L_{\mathsf{Sign}} \leftarrow L_{\mathsf{Sign}} \cup \{\mathsf{msg}\}$
   3: $(\mathbf{r}_j, \mathbf{e}'_j) \leftarrow \mathcal{D}^{\ell}_{\mathbf{w}} \times \mathcal{D}^{k}_{\mathbf{w}}$
   4: $\mathbf{w}_j \coloneqq \mathbf{A} \cdot \mathbf{r}_j + \mathbf{e}'_j$
   5: $\mathsf{cmt}_j \coloneqq \mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_j)$
   6: $\mathbf{m}_j \leftarrow \mathcal{R}^k_q$                                  ▷ Sample row masks uniformly random
   7: if $\mathsf{Committed}[\mathsf{sid}] \neq \mathsf{honest}_{\mathsf{sid}}$ then          ▷ User j is not the last honest signer
   8:     $\mathbf{m}^*_j \leftarrow \mathcal{R}^k_q$                            ▷ Sample column masks uniformly random
   9: else                                                  ▷ User j is the last honest signer
   10:    $\mathsf{LastSigner}[\mathsf{sid}] \leftarrow j$
   11:    Fetch $(\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{i \in \mathsf{act}}$ from state.sk
   12:    for $i \in \mathsf{corrupt}_{\mathsf{sid}}$ do
   13:        $\mathbf{m}_{i,j} \coloneqq \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{sid})$
   14:        $\mathbf{m}_{j,i} \coloneqq \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{sid})$
   15:    $\mathbf{m}^*_j \coloneqq \sum_{i \in \mathsf{honest}_{\mathsf{sid}}} \mathbf{m}_i - \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i \neq j}} \mathbf{m}^*_i + \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{corrupt}_{\mathsf{sid}}}} (\mathbf{m}_{i',i} - \mathbf{m}_{i,i'})$
                                                        ▷ Set the last column mask consistently
   16: return $\mathsf{contrib}_1[j] \coloneqq (\mathsf{cmt}_j, \mathbf{m}_j)$

   $\mathsf{OSgn}_2(j, \mathsf{sid}, \mathsf{contrib}_1)$
       ▷▷▷ Identical to Hybrid_2.OSgn_2 ◁◁◁
```

Figure 8: The fourth security hybrid game used in the proof of Theorem 7.2. Differences from the previous hybrid are highlighted .

distributed uniformly at random as in Hybrid_4. It remains to argue that the $\mathbf{m}_h^*$ terms are sampled from the same distribution in both hybrids. In Hybrid_3 the column mask $\mathbf{m}_h^*$ is computed as

$$\mathbf{m}_h^* \coloneqq \sum_{i \in \mathsf{act}} \mathbf{m}_{i,h},$$

where $m_{i,j} \coloneqq \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{sid})$. On the other hand, in Hybrid_4 they are computed as

$$\mathbf{m}_h^* \coloneqq \sum_{i \in \mathsf{honest}_{\mathsf{sid}}} \mathbf{m}_i - \sum_{i \in \mathsf{honest}_{\mathsf{sid}}, i \neq h} \mathbf{m}_i^* + \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{corrupt}_{\mathsf{sid}}}} (\mathbf{m}_{i',i} - \mathbf{m}_{i,i'}).$$

We see that these distributions are the same because in Hybrid_3:

$$\begin{aligned}
&\sum_{i \in \mathsf{honest}_{\mathsf{sid}}} \mathbf{m}_i - \sum_{i \in \mathsf{honest}_{\mathsf{sid}}, i \neq h} \mathbf{m}_i^* + \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{corrupt}_{\mathsf{sid}}}} (\mathbf{m}_{i',i} - \mathbf{m}_{i,i'}) \\
&= \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{act}}} \mathbf{m}_{i,i'} - \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}}, i \neq h \\ i' \in \mathsf{act}}} \mathbf{m}_{i',i} + \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}}, \\ i' \in \mathsf{corrupt}_{\mathsf{sid}}}} (\mathbf{m}_{i',i} - \mathbf{m}_{i,i'}) \\
&= \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{honest}_{\mathsf{sid}}}} \mathbf{m}_{i,i'} - \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}}, i \neq h \\ i' \in \mathsf{honest}_{\mathsf{sid}}}} \mathbf{m}_{i',i} + \sum_{i' \in \mathsf{corrupt}_{\mathsf{sid}}} \mathbf{m}_{i',h} \\
&= \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{honest}_{\mathsf{sid}}}} \mathbf{m}_{i,i'} - \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{honest}_{\mathsf{sid}}, i' \neq h}} \mathbf{m}_{i,i'} + \sum_{i' \in \mathsf{corrupt}_{\mathsf{sid}}} \mathbf{m}_{i',h} \\
&= \sum_{i \in \mathsf{honest}_{\mathsf{sid}}} \mathbf{m}_{i,h} + \sum_{i' \in \mathsf{corrupt}_{\mathsf{sid}}} \mathbf{m}_{i',h} \\
&= \sum_{i \in \mathsf{act}} \mathbf{m}_{i,h}.
\end{aligned}$$

Hence the distributions are identical and

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}(\kappa) = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}(\kappa).$$

Hybrid_5. In this hybrid, the challenger reprograms the random oracle $\mathsf{H}_{\mathsf{com}}$ when a new session sid is queried to the second-round signing query. Effectively, it delays the computation of the commitment $\mathbf{w}_h$ for the last honest signer $h = \mathsf{LastSigner}[\mathsf{sid}]$ to the second-round. This is formally depicted in Fig. 9, where the challenger uses two lists $\mathsf{Commitments}[\cdot]$ and $\mathsf{Opened}[\cdot]$ to maintain the commitments generated by the honest users and to check if a second-round signing query was made for a specific session sid, respectively.

The signing responses in Hybrid_5 are identically distributed to Hybrid_4 unless the bad event occurs in which $\mathsf{OSgn}_1()$ is required to program a value that has already been queried by the adversary. Where $\mathbf{w}_h$ is sampled as an $\mathsf{MLWE}_{q,k,\ell,\mathcal{D}_{\mathsf{w}}}$ sample, this collision happens with maximum probability $Q_{\mathsf{H}}/2^{H_\infty(\mathcal{D}_{\mathsf{w}}^k)}$ in each signing query. Due to Lemma 3.8 and our parameter selection, since bit dropping can only decrease the min-entropy, we have

$$\begin{aligned}
&\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_5} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}\right| \\
\leq& 1 - \left(1 - Q_{\mathsf{H}} \cdot 2^{-H_\infty(\mathcal{D}_{\mathsf{w}}^k)}\right)^{Q_{\mathsf{Sign}}} \\
\leq& \left(1 - \left(1 - Q_{\mathsf{H}} \cdot 2^{-n+1}\right)^{Q_{\mathsf{Sign}}}\right) \cdot \left(1 - Q_{\mathsf{Sign}} \cdot 2^{-n+1}\right) + Q_{\mathsf{Sign}} \cdot 2^{-n+1} \\
\leq& \frac{(Q_{\mathsf{H}} + 1) \cdot Q_{\mathsf{Sign}}}{2^{n-1}}.
\end{aligned}$$

```
Hybrid_5
1: Committed[·], LastSigner[·], Commitments[·], Opened[·] := ⊥
                                    ▷ Lists to maintain simulation state for sid
   ▷▷▷ Remaining lines are identical to Hybrid_2 ◁◁◁

OSgn_1(j, sid, act, msg)
1:  Committed[sid] ← Committed[sid] ∪ {j}
2:  $L_{\mathsf{Sign}} \leftarrow L_{\mathsf{Sign}} \cup \{\mathsf{msg}\}$
3:  $\mathbf{m}_j \leftarrow \mathcal{R}_q^k$
4:  if Committed[sid] ≠ honest_sid then          ▷ User j is not last honest signer
5:    $(\mathbf{r}_j, \mathbf{e}'_j) \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell} \times \mathcal{D}_{\mathbf{w}}^{k}$
6:    $\mathbf{w}_j := \mathbf{A} \cdot \mathbf{r}_j + \mathbf{e}'_j$     ▷ Create commitment only if j is not last signer
7:    $\mathsf{cmt}_j := \mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_j)$
8:    $\mathsf{Commitments[sid]} \leftarrow \mathsf{Commitments[sid]} \cup \{\mathbf{w}_j\}$
9:    $\mathbf{m}^*_j \leftarrow \mathcal{R}_q^k$
10: else                                          ▷ User j is the last honest signer
11:   LastSigner[sid] ← j
12:   $\mathsf{cmt}_j \leftarrow \{0,1\}^{2\kappa}$      ▷ Set hash commitment to a random string
13:   Fetch $(\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{i \in \mathsf{act}}$ from state.sk
14:   for $i \in \mathsf{corrupt}_{\mathsf{sid}}$ do
15:     $\mathbf{m}_{i,j} := \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{sid})$
16:     $\mathbf{m}_{j,i} := \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{sid})$
17:   $\mathbf{m}^*_j := \sum_{i \in \mathsf{honest}_{\mathsf{sid}}} \mathbf{m}_i - \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i \neq j}} \mathbf{m}^*_i + \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{corrupt}_{\mathsf{sid}}}} (\mathbf{m}_{i',i} - \mathbf{m}_{i,i'})$
18: return $\mathsf{contrib}_1[j] := (\mathsf{cmt}_j, \mathbf{m}_j)$

OSgn_2(j, sid, contrib_1)
1:  Fetch $\mathsf{sk}_{\mathsf{sig},j}$ from state.sk
2:  $\sigma_j \leftarrow \mathsf{Sign}_{\mathsf{sig}}(\mathsf{sk}_{\mathsf{sig},j}, \mathsf{sid} \,||\, \mathsf{act} \,||\, \mathsf{msg} \,||\, \mathsf{contrib}_1)$
3:  $\mathsf{Signed}[j] \leftarrow \mathsf{Signed}[j] \cup \{(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1)\}$
4:  if $\mathsf{Opened[sid]} = \perp \wedge \mathsf{Committed[sid]} = \mathsf{honest}_{\mathsf{sid}}$ then
                                                   ▷ First second-round signing
                          ▷ query on sid and all honest users completed first round
5:    Opened[sid] ← ⊤
6:    h := LastSigner[sid]
7:    $(\mathbf{r}_h, \mathbf{e}'_h) \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell} \times \mathcal{D}_{\mathbf{w}}^{k}$    ▷ Create last honest signer's commitment
8:    $\mathbf{w}_h := \mathbf{A} \cdot \mathbf{r}_h + \mathbf{e}'_h$          ▷ Challenger stores $\mathbf{w}_h$ into user h's state
9:    $\mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_h) := \mathsf{cmt}_h$       ▷ Program random oracle
10: Fetch $\mathbf{w}_j$ from state.sessions[sid].internal
11: return $\mathsf{contrib}_2[j] := (\mathbf{w}_j, \sigma_j)$
```

Figure 9: The fifth security hybrid game used in the proof of Theorem 7.2. Differences from the previous hybrid are highlighted. We assume the game aborts and outputs 0 in case $\mathsf{H}_{\mathsf{com}}$ is already defined when executing OSgn_2. The list Commitments[·] is explicitly used in the next hybrid.

The last line follows from the Bernoulli's inequality ($(1+x)^r \geq 1+rx$ for any integer $r \geq 0$ and real number $x > -1$).

Hybrid_6. In this hybrid, whenever $\mathcal{A}$ has correctly committed to $\mathbf{w}_i$ for all corrupt parties $\mathsf{corrupt}_{\mathsf{sid}}$, the challenger replaces non-programmed random oracle outputs to H_c in the second-round signing oracle with programmed outputs. This is formally depicted in Fig. 10, where the challenger uses a list $\mathsf{Challenge}[\cdot]$ to maintain the challenges it sampled for a specific session sid. For each sid, the challenger samples an element c uniformly at random from the challenge space $\mathcal{C}$. When querying the second-round signing oracle on sid for the first time, the challenger checks if the first-round contribution $\mathsf{contrib}_1$ includes the contributions of all the honest users in $\mathsf{honest}_{\mathsf{sid}}$. It further checks if all the first-round contributions of the corrupt users in $\mathsf{corrupt}_{\mathsf{sid}}$ has an associating commitment $\mathbf{w}_i$ that uniquely explains the hash commitment cmt_i. If so, it collects all the commitments $(\mathbf{w}_i)_{i\in\mathsf{act}}$ that explains the hash commitments included in $\mathsf{contrib}_1$, computes the aggregated commitment $\mathbf{w}$, and programs $\mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$. If any of the above checks fails, then the challenger will not use the sampled c and remains identical to the previous hybrid.

Note that the signing responses in Hybrid_6 are identically distributed to Hybrid_5 unless: a) the bad event occurs in which $\mathsf{OSgn}_2()$ is required to program H_c on a value that has already been queried by the adversary; or b) the bad event occurs that $\mathcal{A}$ successfully queries $\mathsf{OSgn}_3()$ when $\mathcal{A}$ has incorrectly committed to $\mathbf{w}_i$ for some corrupt party. As $\mathbf{w}$ comes from the contribution of a party as a $\mathcal{D}^{\mathsf{bd\text{-}MLWE}}_{q,\ell,k,\sigma_{\mathbf{w}},\nu_{\mathbf{w}}}(\mathbf{A})$ sample, as in Definition 3.7, the first event happens with maximum probability $Q_{\mathsf{H}} \cdot 2^{-n+1}$ in each signing query by Lemma 3.8, with all but probability $Q_{\mathsf{Sign}} \cdot 2^{-n+1}$.

The second bad event happens if $\mathcal{A}$:

- Breaks the preimage resistance of the random oracle $\mathsf{H}_{\mathsf{com}}$. Specifically, a bad event occurs when $\mathsf{H}_{\mathsf{com}}$ has not output cmt_i in the second-round but $\mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_i) = \mathsf{cmt}_i$ in the third signing round. This happens with probability as most $\frac{Q_{\mathsf{H}}}{2^{2\kappa}}$.
- Breaks the collision resistance of the random oracle $\mathsf{H}_{\mathsf{com}}$. Namely there is a collision such that $\mathsf{H}_{\mathsf{com}}$ maps both $(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_i)$ and $(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}'_i)$ to the same cmt_i. This happens with probability as most $Q^2_{\mathsf{H}}/2^{2\kappa}$.

Following the same bound in the previous hybrid, we have

$$\left|\mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Hybrid}_5}_{\mathcal{A}}\right| \leq \frac{(Q_{\mathsf{H}}+1)\cdot Q_{\mathsf{Sign}}}{2^{n-1}} + \frac{Q_{\mathsf{H}} + Q^2_{\mathsf{H}}}{2^{2\kappa}}$$

Hybrid_7. In this hybrid, whenever $\mathcal{A}$ has correctly committed to $\mathbf{w}_i$ for all corrupt parties $\mathsf{corrupt}_{\mathsf{sid}}$, the challenger creates the commitment $\mathbf{w}_h$ and response $\mathbf{z}_h$ of the last honest user $h = \mathsf{LastSigner}[\mathsf{sid}]$ in reverser order. This is formally depicted in Fig. 11. When queried the third-round signing oracle on user h with sid, the challenger uses the already computed $\mathbf{z}_h$.

In Hybrid_6, $\mathbf{z}_h$ was set as $\mathbf{z}_h = c' \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{s}_h + \mathbf{r}_h + \mathbf{m}^*_h$, where $c' = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}')$, msg is included in sid, and $\mathbf{w}'$ is the aggregated commitment constructed with the individual commitments $(\mathbf{w}'_i)_{i\in\mathsf{act}}$ included in the first-round contribution $\mathsf{contrib}_1$. Notice that if we can guarantee $c = \mathsf{Challenge}[\mathsf{sid}] = c'$, then it is easy to check that $\mathbf{w}_h$ and $\mathbf{z}_h$ in both hybrids are distributed identically. To check this, we need to make sure that $\mathbf{w}'$ used in Hybrid_6 is identical to $\mathbf{w}$ used to program the random oracle H_c in Hybrid_7.

Notice in Hybrid_6, for user h to output a response, all honest users in $\mathsf{honest}_{\mathsf{sid}}$ must have signed the same first-round contribution $\mathsf{contrib}_1$ (see **assert** condition on Lines 6 in OSgn_3). Moreover, the hash commitments included in the individual contributions can be opened with a unique commitment $(\mathbf{w}'_i)_{i\in\mathsf{act}}$ (see **if** condition on Line 11 of OSgn_2 and the **assert** condition on Line 4 in OSgn_3). Thus, we are guaranteed that $(\mathbf{w}'_i)_{i\in\mathsf{act}} = (\mathbf{w}_i)_{i\in\mathsf{act}}$. Hence,

$$\mathsf{Adv}^{\mathsf{Hybrid}_7}_{\mathcal{A}}(\kappa) = \mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}}(\kappa).$$

```
Hybrid_6
1: Committed[·], LastSigner[·], Commitments[·], Opened[·], Challenge[·] := ⊥
                                        ▷ Lists to maintain simulation state for sid
   ▷▷▷ Remaining lines are identical to Hybrid_2 ◁◁◁

OSgn_1(j, sid, act, msg)
1: if Challenge[sid] = ⊥ then
2:     $c \leftarrow \mathcal{C}$                  ▷ Sample challenge for sid to be used later
3:     Challenge[sid] ← $c$
   ▷▷▷ Remaining lines are identical to Hybrid_5.OSgn_1 ◁◁◁

OSgn_2(j, sid, contrib_1)
1: Fetch $\mathsf{sk}_{\mathsf{sig},j}$ from state.sk
2: $\sigma_j \leftarrow \mathsf{Sign}_{\mathsf{sig}}(\mathsf{sk}_{\mathsf{sig},j}, \mathsf{sid} \,||\, \mathsf{act} \,||\, \mathsf{msg} \,||\, \mathsf{contrib}_1)$
3: $\mathsf{Signed}[j] \leftarrow \mathsf{Signed}[j] \cup \{(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1)\}$
4: if $\mathsf{Opened}[\mathsf{sid}] = \bot \wedge \mathsf{Committed}[\mathsf{sid}] = \mathsf{honest}_{\mathsf{sid}}$ then
5:     $\mathsf{Opened}[\mathsf{sid}] \leftarrow \top$
6:     $h := \mathsf{LastSigner}[\mathsf{sid}]$
7:     $(\mathbf{r}_h, \mathbf{e}'_h) \leftarrow \mathcal{D}^{\ell}_{\mathbf{w}} \times \mathcal{D}^{k}_{\mathbf{w}}$
8:     $\mathbf{w}_h := \mathbf{A} \cdot \mathbf{r}_h + \mathbf{e}'_h$
9:     $\mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_h) := \mathsf{cmt}_h$
10:    Parse $\mathsf{contrib}_1 = (\mathsf{cmt}_i, \mathbf{m}_i)_{i \in \mathsf{act}}$   ▷ Check if adversary committed honestly in first round
11:    if $\forall i \in \mathsf{corrupt}_{\mathsf{sid}}, \exists$ unique $(\mathsf{H}_{\mathsf{com}}, (\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_i), \mathsf{cmt}_i) \in L_{\mathsf{H}}$ then
12:        $c := \mathsf{Challenge}[\mathsf{sid}]$
13:        $(\mathbf{w}_i)_{i \in \mathsf{honest}_{\mathsf{sid}} \setminus \{h\}} \leftarrow \mathsf{Commitments}[\mathsf{sid}]$
14:        $\mathbf{w} := \left\lfloor \sum_{i \in \mathsf{act}} \mathbf{w}_i \right\rceil_{\nu_{\mathbf{w}}}$
15:        $\mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$                    ▷ Program random oracle
16: Fetch $\mathbf{w}_j$ from state.sessions[sid].internal
17: return $\mathsf{contrib}_2[j] := (\mathbf{w}_j, \sigma_j)$

OSgn_3(j, sid, contrib_2)
1: Fetch (sid, act, msg) from session
2: Fetch $\mathsf{contrib}_1 = (\mathsf{cmt}_i, \mathbf{m}_i)_{i \in \mathsf{act}}$ from session.contrib_1
3: for $i \in \mathsf{act}$ do
4:     assert{ $\exists$unique $(\mathsf{H}_{\mathsf{com}}, (\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_i), \mathsf{cmt}_i) \in L_{\mathsf{H}}$ }
5:     for $i \in \mathsf{honest}_{\mathsf{sid}}$ do
6:         assert{ $(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1) \in \mathsf{Signed}[i]$ }
   ▷▷▷ Remaining lines are identical to Hybrid_2.OSgn_3 ◁◁◁
```

Figure 10: The sixth security hybrid game used in the proof of Theorem 7.2. Differences from the previous hybrid are highlighted. We assume the game aborts and outputs 0 in case $\mathsf{H}_{\mathsf{com}}$ and H_c are already defined when executing OSgn_2.

```
Hybrid$_7$
▷▷▷ Identical to Hybrid$_2$ ◁◁◁

    OSgn$_1(j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$
        // Identical to Hybrid$_6$

    OSgn$_2(j, \mathsf{sid}, \mathsf{contrib}_1)$
     1: Fetch $\mathsf{sk}_{\mathsf{sig},j}$ from $\mathsf{state.sk}$
     2: $\sigma_j \leftarrow \mathsf{Sign}_{\mathsf{sig}}(\mathsf{sk}_{\mathsf{sig},j}, \mathsf{sid} \,||\, \mathsf{act} \,||\, \mathsf{msg} \,||\, \mathsf{contrib}_1)$
     3: $\mathsf{Signed}[j] \leftarrow \mathsf{Signed}[j] \cup \{(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1)\}$
     4: if $\mathsf{Opened}[\mathsf{sid}] = \bot \wedge \mathsf{Committed}[\mathsf{sid}] = \mathsf{honest}_{\mathsf{sid}}$ then
     5:     $\mathsf{Opened}[\mathsf{sid}] \leftarrow \top$
     6:     $h := \mathsf{LastSigner}[\mathsf{sid}]$
     7:     $(\mathbf{r}_h, \mathbf{e}'_h) \leftarrow \mathcal{D}^{\ell}_{\mathbf{w}} \times \mathcal{D}^{k}_{\mathbf{w}}$
     8:     Parse $\mathsf{contrib}_1 = (\mathsf{cmt}_i, \mathbf{m}_i)_{i \in \mathsf{act}}$
     9:     if $\forall i \in \mathsf{corrupt}_{\mathsf{sid}}, \exists$ unique $(\mathsf{H}_{\mathsf{com}}, (\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_i), \mathsf{cmt}_i) \in L_{\mathsf{H}}$ then
    10:         $c := \mathsf{Challenge}[\mathsf{sid}]$
    11:         $\mathbf{z}_h := c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{s}_h + \mathbf{r}_h + \mathbf{m}^*_h$
                                                  ▷ Create response before commitment
    12:         $\mathbf{w}_h := \mathbf{A} \cdot (\mathbf{z}_h - \mathbf{m}^*_h) - c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{A} \cdot \mathbf{s}_h + \mathbf{e}'_h$
    13:         $\mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_h) := \mathsf{cmt}_h$
    14:         $(\mathbf{w}_i)_{i \in \mathsf{honest}_{\mathsf{sid}} \setminus \{h\}} \leftarrow \mathsf{Commitments}[\mathsf{sid}]$
    15:         $\mathbf{w} := \left\lfloor \sum_{i \in \mathsf{act}} \mathbf{w}_i \right\rceil_{\nu_{\mathbf{w}}}$
    16:         $\mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$
    17:     else
    18:         $\mathbf{w}_h := \mathbf{A} \cdot \mathbf{r}_h + \mathbf{e}'_h$
    19:         $\mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_h) := \mathsf{cmt}_h$
    20: Fetch $\mathbf{w}_j$ from $\mathsf{state.sessions}[\mathsf{sid}].\mathsf{internal}$
    21: return $\mathsf{contrib}_2[j] := (\mathbf{w}_j, \sigma_j)$
```

Figure 11: The seventh security hybrid game used in the proof of Theorem 7.2. Differences from the previous hybrid are highlighted.

```
Hybrid_8
▷▷▷ Identical to Hybrid_2 ◁◁◁

OSgn_1(j, sid, act, msg)
1: if Challenge[sid] = ⊥ then
2:     $c \leftarrow \mathcal{C}$
3:     Challenge[sid] ← $c$
4: Committed[sid] ← Committed[sid] ∪ {$j$}
5: $L_{\mathsf{Sign}} \leftarrow L_{\mathsf{Sign}} \cup \{\mathsf{msg}\}$
6: $\mathbf{m}_j \leftarrow \mathcal{R}_q^k$
7: if Committed[sid] ≠ honest_sid then                    ▷ User $j$ is not the last honest signer
8:     $(\mathbf{r}_j, \mathbf{e}'_j) \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell} \times \mathcal{D}_{\mathbf{w}}^{k}$
9:     $\mathbf{w}_j := \mathbf{A} \cdot \mathbf{r}_j + \mathbf{e}'_j$
10:    $\mathsf{cmt}_j := \mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_j)$
11:    Commitments[sid] ← Commitments[sid] ∪ {$\mathbf{w}_j$}
12:    $\mathbf{z}_j \leftarrow \mathcal{R}_q^k$
                    ▷ Column mask implicitly set as $\mathbf{m}_j^* := \mathbf{z}_j - \mathbf{r}_j - c \cdot \lambda_{\mathsf{act},j} \cdot \mathbf{s}_j$
13: else                                                   ▷ User $j$ is the last honest signer
14:    LastSigner[sid] ← $j$
15:    $\mathsf{cmt}_j \leftarrow \{0,1\}^{2\kappa}$
16:    Fetch $(\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{i \in \mathsf{act}}$ from state.sk
17:    for $i \in \mathsf{corrupt}_{\mathsf{sid}}$ do
18:        $\mathbf{m}_{i,j} := \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{sid})$
19:        $\mathbf{m}_{j,i} := \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{sid})$
20:    $\mathbf{m}_j^* := \sum_{i \in \mathsf{honest}_{\mathsf{sid}}} \mathbf{m}_i - \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i \neq j}} (\mathbf{z}_j - \mathbf{r}_j - c \cdot \lambda_{\mathsf{act},j} \cdot \mathbf{s}_j) + \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{corrupt}_{\mathsf{sid}}}} (\mathbf{m}_{i',i} - \mathbf{m}_{i,i'})$
                                                           ▷ Replace column mask with response
21: return $\mathsf{contrib}_1[j] := (\mathsf{cmt}_j, \mathbf{m}_j)$

OSgn_2(j, sid, contrib_1)
    ▷▷▷ Identical to Hybrid_7.OSgn_2 ◁◁◁
```

Figure 12: The eighth security hybrid game used in the proof of Theorem 7.2. Differences from the previous hybrid are highlighted.

Hybrid_8. In this hybrid, the challenger samples the response $\mathbf{z}_i$ uniformly random for all honest users $\mathsf{honest}_{\mathsf{sid}}$, except for the last honest user. This is formally depicted in Fig. 12. More concretely, the challenger no longer samples the random column masks $\mathbf{m}_i^*$ for $i \in \mathsf{honest}_{\mathsf{sid}}$ and $i \neq \mathsf{LastSigner}[\mathsf{sid}]$, but directly prepares the response $\mathbf{z}_i$; put differently, the column mask is implicitly set as $\mathbf{m}_i^* := \mathbf{z}_i - \mathbf{r}_i - c \cdot \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i$.

Note that the distribution of the responses are identical if the challenge $c = \mathsf{Challenge}[\mathsf{sid}]$ is the challenge that user i uses in the third-round signature query. This can be established following the same argument we made in Hybrid_7. Hence, we have

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_8}(\kappa) = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_7}(\kappa).$$

Hybrid_9. In this hybrid, the challenger no longer generates the column mask $\mathbf{m}_h^*$ of the last honest signer $h = \mathsf{honest}_{\mathsf{sid}}$ and generates the commitment $\mathbf{w}_h$ and response $\mathbf{z}_h$ of the last honest signer $h = \mathsf{honest}_{\mathsf{sid}}$ without using its partial secret $\mathbf{s}_h$ or $\mathbf{m}_h^*$. This is formally depicted in Fig. 13.

Recall in Hybrid_8, the response was set as $\mathbf{z}_h := c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{s}_h + \mathbf{r}_h + \mathbf{m}_h^*$. Plugging in the definition of $\mathbf{m}_h^*$, we have

$$\begin{aligned}
\mathbf{z}_h &= c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{s}_h + \mathbf{r}_h + \mathbf{m}_h^* \\
&= c \cdot \Big(\mathbf{s} - \sum_{i \in \mathsf{act} \setminus \{h\}} \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i\Big) + \mathbf{r}_h + \sum_{i \in \mathsf{honest}_{\mathsf{sid}}} \mathbf{m}_i \\
&\quad - \sum_{i \in \mathsf{honest}_{\mathsf{sid}} \setminus \{h\}} (\mathbf{z}_i - \mathbf{r}_i - c \cdot \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i) + \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{corrupt}_{\mathsf{sid}}}} (\mathbf{m}_{i',i} - \mathbf{m}_{i,i'}) \\
&= \mathbf{z}_{\mathsf{sid}} - c \cdot \sum_{i \in \mathsf{corrupt}_{\mathsf{sid}}} \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i + \mathbf{m}_h \\
&\quad + \sum_{i \in \mathsf{honest}_{\mathsf{sid}} \setminus \{h\}} (\mathbf{m}_i - \mathbf{z}_i + \mathbf{r}_i) + \sum_{\substack{i \in \mathsf{honest}_{\mathsf{sid}} \\ i' \in \mathsf{corrupt}_{\mathsf{sid}}}} (\mathbf{m}_{i',i} - \mathbf{m}_{i,i'}),
\end{aligned}$$

where the second equality follows from the correctness of the Shamir secret sharing scheme, i.e., $\mathbf{s} = \sum_{i \in \mathsf{act}} \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i$. This is exactly how $\mathbf{z}_h$ is set in Hybrid_9.

Similarly, the commitment $\mathbf{w}_h$ in Hybrid_8 is set as $\mathbf{w}_h := \mathbf{A} \cdot (\mathbf{z}_h - \mathbf{m}_h^*) - c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{A} \cdot \mathbf{s}_h + \mathbf{e}'_h$. Plugging in $\mathbf{m}_h^*$ again, we have

$$\begin{aligned}
\mathbf{w}_h &= \mathbf{A} \cdot (\mathbf{z}_h - \mathbf{m}_h^*) - c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{A} \cdot \mathbf{s}_h + \mathbf{e}'_h \\
&= \mathbf{A} \cdot (c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{s}_h + \mathbf{r}_h) - c \cdot \lambda_{\mathsf{act},h} \cdot \mathbf{A} \cdot \mathbf{s}_h + \mathbf{e}'_h \\
&= \mathbf{A} \cdot (\mathbf{z}_{\mathsf{sid}} - c \cdot \mathbf{s}) + \mathbf{e}'_h \\
&= \mathbf{A} \cdot \mathbf{z}_{\mathsf{sid}} - c \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}) + c \cdot \mathbf{e} + \mathbf{e}'_h \\
&= \mathbf{A} \cdot \mathbf{z}_{\mathsf{sid}} - c \cdot \widehat{\mathbf{t}} + \mathbf{z}'_{\mathsf{sid}},
\end{aligned}$$

which is exactly how $\mathbf{w}_h$ is set in Hybrid_9. Hence, we have

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_9}(\kappa) = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_8}(\kappa).$$

Hybrid_{10}. Finally, in this hybrid, the challenger replaces the publi key $\mathsf{vk} = (\mathbf{A}, \lfloor \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \rceil_{\nu_{\mathbf{t}}})$ with $(\mathbf{A}, \left\lfloor \widehat{\mathbf{t}} \right\rceil_{\nu_{\mathbf{t}}})$, where $\widehat{\mathbf{t}}$ is sampled uniformly at random from $\mathcal{R}_q^k$. This is formally depicted in Fig. 14.

In Lemma 7.3, we show that we can construct an adversary $\mathcal{B}$ against the $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ problem such that

$$\left| \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_{10}}(\kappa) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_9}(\kappa) \right| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{Hint\text{-}MLWE}}(\kappa).$$

```
Hybrid$_9$
▷▷▷ Identical to Hybrid$_2$ ◁◁◁

OSgn$_1(j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$
1: if $\mathsf{Challenge}[\mathsf{sid}] = \bot$ then
2:     $c \leftarrow \mathcal{C}$; $\mathsf{Challenge}[\mathsf{sid}] \leftarrow c$
3: $\mathsf{Committed}[\mathsf{sid}] \leftarrow \mathsf{Committed}[\mathsf{sid}] \cup \{j\}$; $L_{\mathsf{Sign}} \leftarrow L_{\mathsf{Sign}} \cup \{\mathsf{msg}\}$
4: $\mathbf{m}_j \leftarrow \mathcal{R}_q^k$
5: if $\mathsf{Committed}[\mathsf{sid}] \neq \mathsf{honest}_{\mathsf{sid}}$ then        ▷ User $j$ is not the last honest signer
6:     $(\mathbf{r}_j, \mathbf{e}'_j) \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell} \times \mathcal{D}_{\mathbf{w}}^{k}$; $\mathbf{w}_j := \mathbf{A} \cdot \mathbf{r}_j + \mathbf{e}'_j$
7:     $\mathsf{cmt}_j := \mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_j)$
8:     $\mathsf{Commitments}[\mathsf{sid}] \leftarrow \mathsf{Commitments}[\mathsf{sid}] \cup \{\mathbf{w}_j\}$
9:     $\mathbf{z}_j \leftarrow \mathcal{R}_q^k$
10: else        ▷ User $j$ is the last honest signer
11:     $\mathsf{LastSigner}[\mathsf{sid}] \leftarrow j$; $\mathsf{cmt}_j \leftarrow \{0,1\}^{2\kappa}$
12: return $\mathsf{contrib}_1[j] := (\mathsf{cmt}_j, \mathbf{m}_j)$

OSgn$_2(j, \mathsf{sid}, \mathsf{contrib}_1)$
1: Fetch $\mathsf{sk}_{\mathsf{sig},j}$ from $\mathsf{state.sk}$
2: $\sigma_j \leftarrow \mathsf{Sign}_{\mathsf{sig}}(\mathsf{sk}_{\mathsf{sig},j}, \mathsf{sid} \,||\, \mathsf{act} \,||\, \mathsf{msg} \,||\, \mathsf{contrib}_1)$
3: $\mathsf{Signed}[j] \leftarrow \mathsf{Signed}[j] \cup \{(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1)\}$
4: if $\mathsf{Opened}[\mathsf{sid}] = \bot \wedge \mathsf{Committed}[\mathsf{sid}] = \mathsf{honest}_{\mathsf{sid}}$ then
5:     $\mathsf{Opened}[\mathsf{sid}] \leftarrow \top$
6:     $h := \mathsf{LastSigner}[\mathsf{sid}]$
7:     $(\mathbf{r}_h, \mathbf{e}'_h) \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell} \times \mathcal{D}_{\mathbf{w}}^{k}$
8:     Parse $\mathsf{contrib}_1 = (\mathsf{cmt}_i, \mathbf{m}_i)_{i \in \mathsf{act}}$
9:     if $\forall i \in \mathsf{corrupt}_{\mathsf{sid}}$, $\exists$unique $(\mathsf{H}_{\mathsf{com}}, (\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_i), \mathsf{cmt}_i) \in L_{\mathsf{H}}$ then
10:        $c := \mathsf{Challenge}[\mathsf{sid}]$
11:        $(\mathbf{z}_{\mathsf{sid}}, \mathbf{z}'_{\mathsf{sid}}) := (c \cdot \mathbf{s} + \mathbf{r}_h, c \cdot \mathbf{e} + \mathbf{e}'_h)$
       ▷▷▷ We remove user $h$'s partial secret $\mathbf{s}_h$ and column mask $\mathbf{m}^*_h$ from response and commitment ◁◁◁
12:        $\mathbf{z}_h := \mathbf{z}_{\mathsf{sid}} - c \cdot \sum_{i \in \mathsf{corrupt}_{\mathsf{sid}}} \lambda_{\mathsf{act},i} \cdot \mathbf{s}_i + \mathbf{m}_h + \sum_{i \in \mathsf{honest}_{\mathsf{sid}} \setminus \{h\}} (\mathbf{m}_i - \mathbf{z}_i + \mathbf{r}_i) + \sum_{i \in \mathsf{honest}_{\mathsf{sid}},\, i' \in \mathsf{corrupt}_{\mathsf{sid}}} (\mathbf{m}_{i',i} - \mathbf{m}_{i,i'})$
13:        $\mathbf{w}_h := \mathbf{A} \cdot \mathbf{z}_{\mathsf{sid}} - c \cdot \widehat{\mathbf{t}} + \mathbf{z}'_{\mathsf{sid}}$        ▷ $\widehat{\mathbf{t}} := \mathbf{A}\mathbf{s} + \mathbf{e}$, i.e., non-rounded $\mathbf{t}$
14:        $\mathsf{H}_{\mathsf{com}}(\mathsf{act}, \mathsf{msg}, \mathbf{w}_h) := \mathsf{cmt}_h$
15:        $(\mathbf{w}_i)_{i \in \mathsf{honest}_{\mathsf{sid}} \setminus \{h\}} \leftarrow \mathsf{Commitments}[\mathsf{sid}]$
16:        $\mathbf{w} := \lfloor \sum_{i \in \mathsf{act}} \mathbf{w}_i \rceil_{\nu_{\mathbf{w}}}$
17:        $\mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$
18:    else
19:        $\mathbf{w}_h := \mathbf{A} \cdot \mathbf{r}_h + \mathbf{e}'_h$
20:        $\mathsf{H}_{\mathsf{com}}(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathbf{w}_h) := \mathsf{cmt}_h$
21: return $\mathsf{contrib}_2[j] := (\mathbf{w}_j, \sigma_j)$
```

Figure 13: The ninth security hybrid game used in the proof of Theorem 7.2. Differences from the previous hybrid are highlighted.

```
Hybrid$_{10}$
1:  $\mathsf{Committed}[\cdot], \mathsf{LastSigner}[\cdot], \mathsf{Commitments}[\cdot], \mathsf{Opened}[\cdot], \mathsf{Challenge}[\cdot] := \perp$
2:  $\mathsf{Signed}[\cdot] := \perp$
3:  $L_{\mathsf{Sign}}, L_{\mathsf{H}} := \emptyset$
4:  $(N, T, \mathsf{corrupt}) \leftarrow \mathcal{A}(\mathsf{pp})$
5:  $\mathsf{honest} := [N] \backslash \mathsf{corrupt}$
    ▷ ▷ ▷ Rewriting $\mathsf{KeyGen}(\mathsf{pp}, T, N)$ ◁ ◁ ◁
6:  $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}$
7:  $(\mathbf{s}, \mathbf{e}) \leftarrow \mathcal{D}_{\mathbf{t}}^{\ell} \times \mathcal{D}_{\mathbf{t}}^{k}$
8:  $\widehat{\mathbf{t}} \leftarrow \mathcal{R}_q^{\ell}$                    ▷ In Hybrid$_9$, we set $\widehat{\mathbf{t}} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
9:  $\mathbf{t} := \lfloor \widehat{\mathbf{t}} \rceil_{\nu_{\mathbf{t}}}$
10: $\mathsf{vk} := (\mathbf{A}, \mathbf{t})$                    ▷ Set random public key
11: for $i \in \mathsf{corrupt}$ do
12:     $\mathbf{s}_i \leftarrow \mathcal{R}_q^{\ell}$                    ▷ Sample random secret key shares for corrupt users
13: for $i \in [N]$ do
14:     $(\mathsf{vk}_{\mathsf{sig},i}, \mathsf{sk}_{\mathsf{sig},i}) \leftarrow \mathsf{KeyGen}_{\mathsf{sig}}(1^{\kappa})$
15:     for $j \in [N]$ do
16:         $\mathsf{seed}_{i,j} \leftarrow \{0,1\}^{\kappa}$
17: for $i \in \mathsf{honest}$ do
18:     $\mathsf{sk}_i := (\perp, (\mathsf{vk}_{\mathsf{sig},i})_{i \in [N]}, \mathsf{sk}_{\mathsf{sig},i}, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j \in [N]})$
19: for $i \in \mathsf{corrupt}$ do
20:     $\mathsf{sk}_i := (\mathbf{s}_i, (\mathsf{vk}_{\mathsf{sig},i})_{i \in [N]}, \mathsf{sk}_{\mathsf{sig},i}, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j \in [N]})$
21: $(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathsf{OSgn}_i(\cdot))_{i \in [3]}}(\mathsf{vk}, (\mathsf{sk}_i)_{i \in \mathsf{corrupt}})$
22: if $(\mathsf{msg} \in L_{\mathsf{Sign}})$ or $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) = 0$ then
23:     return 0
24: return 1

    $\mathsf{OSgn}_1(j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$
        // Identical to Hybrid$_8$

    $\mathsf{OSgn}_2(j, \mathsf{sid}, \mathsf{contrib}_1)$
        ▷ ▷ ▷ Identical to Hybrid$_9$.OSgn$_2$ ◁ ◁ ◁
```

Figure 14: The final security hybrid game used in the proof of Theorem 7.2. Differences from the previous hybrid are highlighted .

So as not to interrupt the main proof, we postpone the proof of Lemma 7.3.

Now that the public key is a random vector and the challenger no longer requires the signing key for the unforgeability game, we are finally ready to invoke the $\mathsf{SelfTargetMSIS}$ problem. In Lemma 7.4, we show that we can construct an adversary $\mathcal{B}'$ against the $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},\nu_{\mathbf{w}},B_{\mathsf{stmsis}}}$ problem such that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_{10}}(\kappa) \leq \mathsf{Adv}_{\mathcal{B}'}^{\mathsf{SelfTargetMSIS}}(\kappa)$$

Collecting all the bounds, we obtain the following bound, which is the desired bound in the theorem statement.

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1}(\kappa) \leq N \cdot \mathsf{Adv}_{\mathcal{B}_{\mathsf{Sign}}}^{\mathsf{sig\text{-}uf}}(\kappa) + \mathsf{Adv}_{\mathcal{B}_{\mathsf{PRF}}}^{\mathsf{PRF}}(\kappa) + \frac{(Q_{\mathsf{H}}+1)\cdot Q_{\mathsf{Sign}}}{2^{n-1}} \\ + \frac{Q_{\mathsf{H}}+Q_{\mathsf{H}}^2}{2^{2\kappa}} + \mathsf{Adv}_{\mathcal{B}}^{\mathsf{Hint\text{-}MLWE}}(\kappa) + \mathsf{Adv}_{\mathcal{B}'}^{\mathsf{SelfTargetMSIS}}(\kappa)\end{aligned}$$

To complete the proof, it remains to prove the following Lemmas 7.3 and 7.4.

Lemma 7.3. *There exists an adversary $\mathcal{B}$ against the $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ problem such that*

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_{10}}(\kappa) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_9}(\kappa)\right| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{Hint\text{-}MLWE}}(\kappa)$$

with $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A})$.

Proof. Let us provide the description of $\mathcal{B}$. $\mathcal{B}$ is given $\left(\mathbf{A}, \mathbf{b}, (c_i, \mathbf{z}_i, \mathbf{z}'_i)_{i\in[Q_{\mathsf{Sign}}]}\right)$ as the $\mathsf{Hint\text{-}MLWE}$ problem, where $\mathbf{b}$ is either $\mathbf{As}+\mathbf{e}$ or random over $\mathcal{R}_q^k$. It simulates the view of the Hybrid_9 challenger to $\mathcal{A}$ by

- Setting $\widehat{\mathbf{t}} := \mathbf{b}$;
- Sampling secret shares $\mathbf{s}_i \leftarrow \mathcal{R}_q^\ell$ for $i \in \mathsf{corrupt}$;
- Sampling $(\mathsf{vk}_{\mathsf{sig},i}, \mathsf{sk}_{\mathsf{sig},i}) \leftarrow \mathsf{KeyGen}_{\mathsf{sig}}(1^\kappa)$ for $i \in [N]$;
- Sampling $\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i} \leftarrow \{0,1\}^\kappa$ for $i \in \mathsf{corrupt}$, $j \in [N]$;
- Setting $\mathsf{sk}_i = (\mathbf{s}_i, (\mathsf{vk}_{\mathsf{sig},i})_{i\in[N]}, \mathsf{sk}_{\mathsf{sig},i}, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[N]})$ for $i \in \mathsf{corrupt}$;

and running $\mathcal{A}^{\mathsf{H},(\mathsf{OSgn}_i)_{i\in[3]}}(\mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{corrupt}})$. All random oracle queries are simulated identically to the challenger.

When $\mathcal{A}$ queries the signing oracle OSgn_1 on $(j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$ for the n-th ($n \in [Q_{\mathsf{Sign}}]$) time, if $\mathsf{Challenge}[\mathsf{sid}] = \perp$ then $\mathcal{B}$ retrieves the n-th unused tuple $(c_n, \mathbf{z}_n, \mathbf{z}'_n)$ and sets $\mathsf{Challenge}[\mathsf{sid}] = c_n$. After $\mathcal{B}$ proceeds as in the real hybrid for lines 4 onwards.

When $\mathcal{A}$ queries the signing oracle OSgn_2 on $j, \mathsf{sid}, \mathsf{contrib}_1[j]$ then $\mathcal{B}$ computes lines 1-10 as in the real Hybrid. In line 11 then $\mathcal{B}$ looks up $\mathsf{Challenge}[\mathsf{sid}] = c_n$ and the corresponding $(c_n, \mathbf{z}_n, \mathbf{z}'_n)$, and sets $(\mathbf{z}_{\mathsf{sid}}, \mathbf{z}'_{\mathsf{sid}}) = (\mathbf{z}_n, \mathbf{z}'_n)$. After $\mathcal{B}$ proceeds as in the real hybrid for lines 12 onwards.

When $\mathcal{A}$ queries the signing oracle OSgn_3 then $\mathcal{B}$ proceeds as in the real hybrid.

Finally, when $\mathcal{A}$ outputs a valid signature that breaks unforgeability, $\mathcal{B}$ outputs 1.

It is clear that the signing oracle is perfectly simulated using the noise leakage provided by the $\mathsf{Hint\text{-}MLWE}$ problem. Thus, $\mathcal{B}$ perfectly simulates Hybrid_9 when $\mathbf{b}$ is a valid MLWE sample and Hybrid_{10} otherwise. $\square$

Lemma 7.4. *There exists an adversary $\mathcal{B}'$ against the $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ problem such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_{10}}(\kappa) \leq \mathsf{Adv}_{\mathcal{B}'}^{\mathsf{SelfTargetMSIS}}(\kappa)$$

with $\mathsf{Time}(\mathcal{B}') \approx \mathsf{Time}(\mathcal{A})$.

Proof. The proof is identical to those of the non-threshold signature since the forgery output by the adversary $\mathcal{A}$ must satisfy the conditions checked by the same verification algorithm. See Lemma C.4 for the proof. $\square$

This completes the proof of Theorem 7.2. $\square$

Remark 7.5 (Using MACs Instead of Signatures). Notice that since every users share a pair-wise independent PRF seed, they can use that to authenticate themselves, rather than using signatures. The benefit of using MACs is that they may be faster to implement compared to signatures and for small thresholds, it leads to better communication costs.

Here, we provide a sketch of how the above proof will be altered. The only modification is Hybrid_3, where we invoked the security of the signature scheme. Alternatively, if one is using MACs, then the challenger checks all the MACs observed in the third round were generated by the challenger. Then the asserted condition can be checked to hold by invoking the unforgeability of the MAC scheme.

In particular, we can construct an adversary $\mathcal{B}_{\mathsf{MAC}}$ against the unforgeability of the MAC scheme such that

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}(\kappa) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1}(\kappa)\right| \leq Q_{\mathsf{Sign}} \cdot N^2 \cdot \mathsf{Adv}_{\mathcal{B}_{\mathsf{MAC}}}^{\mathsf{MAC\text{-}uf}}(\kappa).$$

The reduction $\mathcal{B}_{\mathsf{MAC}}$ has access to an authenticating oracle and aims to output a forgery. Then $\mathcal{B}_{\mathsf{MAC}}$ simulates the view of the Hybrid_1 challenger to $\mathcal{A}$ by:

1. Computing $\mathsf{vk} = (\mathbf{A}, \mathbf{t})$ and $(\mathbf{s}_i)_{i\in[N]}$ the same as in Hybrid_1;
2. Choosing random $i^*, j^* \in \mathsf{honest}$ and random $k^* \in [Q_{\mathsf{Sign}}]$
3. Sampling $\mathsf{mackey}_{i,j} \leftarrow \mathsf{KeyGen}_{\mathsf{MAC}}(1^\kappa)$ for $i, j \neq i^*, j^*$;

and running $\mathcal{A}^{\mathsf{H},(\mathsf{OSgn}_i)_{i\in[3]}}(\mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{corrupt}})$. Here, note that we can simply assume all the users are given dedicated $\mathsf{mackeys}$ or derive them from the provided PRF seeds.

When $\mathcal{A}$ queries its signing oracles it responds the same as in Hybrid_1 except that MACs between i^*, j^* are generated by querying the MAC oracle. If $\mathcal{A}$ provides MACs on some message $(\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1) \notin \mathsf{Signed}[i^*]$ during the k^*th signing query then $\mathcal{B}_{\mathsf{MAC}}$ returns $((\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_1), \mathsf{MAC}_{j^*})$ as its forgery for MAC_{j^*} the j^*th MAC.

Since i^*, j^*, k^* is information theoretically hidden from $\mathcal{A}$, the probability that $\mathcal{B}$ correctly guesses which public key and signing query $\mathcal{A}$ provides a forgery for is $\frac{1}{Q_{\mathsf{Sign}} N^2}$. Note here we additionally guess the signing query because $\mathcal{B}$ cannot verify the MACs correctness.

8 Concrete Instantiation

The goal of this section is to translate our main theorem (Theorem 7.2) in concrete parameters sets for our threshold signature scheme. We recall the main equation of Theorem 7.2, with added annotations:

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1}(\kappa) \quad &\leq \overbrace{N \cdot \mathsf{Adv}_{\mathcal{B}_{\mathsf{Sign}}}^{\mathsf{sig\text{-}uf}}(\kappa)}^{(A1)} + \overbrace{\mathsf{Adv}_{\mathcal{B}_{\mathsf{PRF}}}^{\mathsf{PRF}}(\kappa)}^{(A2)} + \overbrace{\frac{(Q_{\mathsf{H}}+1)\cdot Q_{\mathsf{Sign}}}{2^{n-1}}}^{(A3)} \\ &+ \underbrace{\frac{Q_{\mathsf{H}} + Q_{\mathsf{H}}^2}{2^{2\kappa}}}_{(A4)} + \underbrace{\mathsf{Adv}_{\mathcal{B}}^{\mathsf{Hint\text{-}MLWE}}(\kappa)}_{(A5)} + \underbrace{\mathsf{Adv}_{\mathcal{B}'}^{\mathsf{SelfTargetMSIS}}(\kappa)}_{(A6)}\end{aligned} \tag{11}$$

Recall that we allow the adversary Q_{Sign} signing queries and Q_{H} hash queries. As is standard, we must divide the probability of success of an adversary by its running time $T_{\mathcal{A}} \geq Q_{\mathsf{Sign}} + Q_{\mathsf{H}}$ when computing the bit-security.

First, it is clear from the main equation that the terms in $(A1)$, $(A2)$ and $(A4)$, once normalised by $T_{\mathcal{A}}$, are smaller than $2^{-\kappa}$. The term in $(A3)$ is conditioned on the initial conditions of Lemma 3.8 being satisfied; we ensure throughout parameter selection that this is the case. Since $n \geq 512$, $(A3)$ is smaller than $2^{-\kappa}$ for all parameter sets we consider.

It only remains to bound the terms $(A5)$ and $(A6)$. These are studied in Section 8.2 and Section 8.1, respectively. Both can be translated naturally in terms of problems over module lattices, which are structured. This additional structure does not seem to provide meaningful improvements to the state-of-the-art attacks. Therefore, we ignore this structure in Sections 8.1 and 8.2 and treat these problems as unstructured lattice problems of equivalent dimensions.

8.1 Direct Forgery and $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$

One of the two main ways an adversary can break our scheme is by breaking the $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ assumption, corresponding to the term $(A6)$ in Eq. (11). Concretely, this corresponds to breaking unforgeability directly, given only a valid verification key. The adversary needs to find $\mathbf{z}_{\mathsf{sol}}$ such that:

$$\left(\mathbf{z}_{\mathsf{sol}} = \begin{bmatrix} c \\ \mathbf{z}' \end{bmatrix}\right) \wedge \left(\|\mathbf{z}_{\mathsf{sol}}\|_2 \leq B_{\mathsf{stmsis}}\right) \wedge \ \mathsf{H}_c\left(\left[-\widehat{\mathbf{t}} \mid \mathbf{A} \mid \mathbf{I}\right] \cdot \mathbf{z}_{\mathsf{sol}},\ \mathsf{msg}\right) = c\right]. \tag{12}$$

B_{stmsis} is set according to the analysis in Section 7.3:

$$B_{\mathsf{stmsis}} = B_{2,T} + \sqrt{\omega} + \left(\omega \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1}\right) \cdot \sqrt{nk},$$
$$\text{where } B_{2,T} = e^{1/4} \cdot \left(\omega\, \sigma_{\mathbf{t}} + \sqrt{T}\, \sigma_{\mathbf{w}}\right)\sqrt{n(k+\ell)} + \left(\omega \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1}\right) \cdot \sqrt{nk}.$$

Following [LDK+22, Section C.3], we assume that the best way to solve (12) is either by (i) breaking the second preimage resistance of H_c or by (ii) generating $\mathbf{w}$ at random, computing $c = \mathsf{H}_c(\mathbf{w},\ \mathsf{msg})$, and finally solving the inhomogenous SIS instance:

$$\left(\left[\mathbf{A} \mid \mathbf{I}\right] \cdot \mathbf{z}' = \mathbf{w} - c \cdot \widehat{\mathbf{t}}\right) \wedge \left(\|\mathbf{z}'\| \leq B_{\mathsf{stmsis}} - \omega\right). \tag{13}$$

We highlight that we do not consider the reduction loss appearing in Appendix B.1 when setting the parameters as it does not seem to reflect any concrete attacks. This is the same approach taken by Dilithium [LDK+22, Section C.3] and Raccoon [dPEK+23, Section 4.3.5].

Below, we study both items (i) and (ii) in two separate paragraphs.

8.1.1 Solving Inhomogeneous MSIS.

Eq. (13) is an inhomogeneous MSIS problem. The state-of-the-art for solving this problem is an optimised analysis by Chuengsatiansup et al. [CPS+20]. Under the geometric series assumption (GSA), [CPS+20] states that we need to enforce the condition:

$$B_{\mathsf{stmsis}} \leq \min_{\ell n \leq m \leq (k+\ell)n} \left(\delta^m q^{\frac{k\,n}{m}}\right), \quad \text{for} \quad \delta = \left(\frac{(\pi \cdot \beta_{\mathsf{bkz}})^{1/\beta_{\mathsf{bkz}}} \cdot \beta_{\mathsf{bkz}}}{2\pi e}\right)^{1/(2(\beta_{\mathsf{bkz}}-1))}$$

Note that B^2_{stmsis} is affine in the number of signers T. Since the hardness of $\mathsf{SelfTargetMSIS}$ is a decreasing function of B_{stmsis}, this needs to be compensated by increasing other parameters (such as the dimensions). Therefore, the signature size is an increasing function of T. However, both the communication cost and the signature size are completely independent of the total number of parties N.

8.1.2 Challenge Space.

We need the hash function H to be second preimage resistant. To guarantee this we ensure that $|\mathcal{C}| > 2^{\kappa}$. Considering how $\mathcal{C}$ is defined in Eq. (9) it is enough to set ω such that: $\binom{n}{\omega} \cdot 2^{\omega} \geq 2^{\kappa}$.

8.2 Pseudorandomness of the Verification Key and Hint-MLWE

The second main way an adversary can break our scheme is by breaking the $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ assumption, which corresponds to the term $(A5)$ in Eq. (11). Concretely, this means distinguishing the verification key vk from uniform, given vk *and* a number Q_{Sign} of valid signatures. Lemmas A.5 and B.2 state that this assumption is at least as hard as $\mathsf{MLWE}_{q,\ell,k,\sigma}$, where $\frac{1}{\sigma^2} = 2 \cdot \left(\frac{1}{\sigma_{\mathbf{t}}^2} + \frac{B_{\mathsf{hint}}}{\sigma_{\mathbf{w}}^2}\right)$ and B_{hint} is set as in Lemma B.2.

Going forward, we can now rely on the large body of existing litterature on the cryptanalysis of MLWE. To this day, the state-of-the-art for estimating the concrete hardness of MLWE remains the lattice estimator (`https://github.com/malb/lattice-estimator`), first developed in [APS15]. According to it, the best known attacks are the primal uSVP attack by Alkim et al. [ADPS16] and the dual/hybrid attack by Espitau et al. [EJK20]. Finally, we can apply the *dimensions for free* optimisation by Ducas [Duc18] to gain a few additional bits when using a sieve-based BKZ.

The hardness of $\mathsf{MLWE}_{q,\ell,k,\sigma}$ is an increasing function in σ, which means it is a decreasing function of B_{hint} and therefore of the total number of signing queries Q_{Sign}. Concretely, this means that the security of our parameter sets are conditioned to a strict limit on Q_{Sign}.

8.3 Parameter Sets

Despite the many variables, parameters are easy to set in a systematic way. The crucial variables are $(k, \ell, n, q, \sigma_{\mathbf{w}})$. We then set ω from Section 8.1, and set $(\nu_{\mathbf{t}}, \nu_{\mathbf{w}}, \sigma_{\mathbf{t}})$ large but such that $\omega \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1} = O\left(\sigma_{\mathbf{w}}\sqrt{T(1+\ell/k)}\right)$ and $\sigma_{\mathbf{t}} = o(\sigma_{\mathbf{w}}/\omega)$. The resulting parameters are in Table 2.

Table 2: Parameter sets. The sizes |vk| and |sig| are provided in kilobytes. All parameter sets satisfy $(\lfloor \log q \rceil, n, \sigma_{\mathbf{t}}, \max T) = (49, 512, 2^{20}, 1024)$.

κ	Q_{Sign}	$\sigma_{\mathbf{w}}\sqrt{T}$	$\nu_{\mathbf{t}}$	$\nu_{\mathbf{w}}$	ℓ	k	ω	|vk|	|sig|	|trans|$/T$
128	2^{60}	2^{42}	37	40	4	5	19	3.9	12.7	40.8
192	2^{64}	2^{42}	36	40	6	7	31	5.8	18.9	59.6
256	2^{60}	2^{42}	35	41	7	8	44	7.2	21.6	69.1

9 Implementation and Experiments

We have developed a high-performance implementation of TRaccoon which can easily accomodate $T = 1024$ simulated signers (with the parameters in Table 2.) If we ignore possible communication latencies and enable 4.5 gHz turbo on an i7-12700, creating a signature (the three steps of ShareSign, $\kappa = 128$) requires from 11.1 ms ($T = 4$) to 116 ms ($T = 1024$) of single-core computation from each signer. The verification function is independent of T and N, and requires approximately 0.230 ms. Table 3 contains more detailed benchmarking results.

This implementation recycles components such as NTT and signature serialisation from the Raccoon NIST submission [dPEK+23]. It uses κ-bit MACs keyed with pairwise $\mathsf{seed}_{i,j}$ (and sid) to authenticate contributions, as discussed in Section 6.1. The Uniform and Gaussian random samplers, MACs and PRFs are built from the SHAKE128 [NIS15] extensible output function. This function (or, more precisely, its Keccak permutation component) dominates the overall running time, requiring up to 80% of cycles. This is despite the code utilizing an AVX2 SIMD Keccak that computes four permutations at the same time.

The distributions $\mathcal{D}_{\mathbf{t}}$ and $\mathcal{D}_{\mathbf{w}}$ are in a region $\sigma \geq 2^{20}$ where table-based Discrete Gaussian samplers perform poorly. Furthermore, $\sigma_{\mathbf{w}}$ may change between signatures as it depends on $\sqrt{T}$. Hence a sampler based on rounded Gaussians [*HLS*18] is used in this implementation. The sampler is based on Marsaglia-Bray polar method [MB64], and can be made side-channel secure if needed. A keen reader may be interested

Table 3: $\mathsf{TRaccoon}$ ($\kappa = 128$) Cycle counts on a single core of an Intel i7-12700 CPU with "turbo boost" disabled. The units are millions of cycles; divide by 2.1 (fixed clock frequency in gHz) to obtain millisecond numbers. Measurements for KeyGen, $\mathsf{Combine}$, and Verify are for the entire process, while $\mathsf{ShareSign}_i$ is per signer (when signing is a parallel process, this is equivalent to the elapsed time).

T	KeyGen	$\mathsf{ShareSign}_1$	$\mathsf{ShareSign}_2$	$\mathsf{ShareSign}_3$	Combine	Verify
4	0.592	20.092	0.539	1.588	1.128	1.094
16	0.417	20.076	2.102	5.559	1.209	1.093
64	0.817	21.830	8.216	21.350	1.579	1.100
256	2.838	33.549	32.788	84.333	3.186	1.095
1024	11.491	67.213	131.887	338.614	11.571	1.106

in the discrepancy between the theory and the implementation; the former using a discrete Gaussian, while the latter using the rounded Gaussian. For completeness, we provide in Appendix E that this discrepancy only incurs a negligible difference for our choices of parameters.

At $\kappa = 128$, public key is $|\mathsf{vk}| = 3856$ bytes. Due to non-uniform distributions, the actual signature encoding size is variable, but can be (with high probability) upper bounded at $|\mathsf{sig}| \leq 12736$ bytes. Communicating the secret key shares and PRF/MAC seed pairs to each of the N potential signers requires $12556 + 32N$ bytes with this implementation. The signing bandwidth requirements (in bytes) are $\frac{1}{T}|\mathsf{contrib}_1| = 12576$, $\frac{1}{T}|\mathsf{contrib}_2| = 15680+16T$, and $\frac{1}{T}|\mathsf{contrib}_3| = 12544$, bringing the total per-signer contribution to $40800+16T$ bytes. If asymmetric signatures rather than pairwise MACs were used, each signer contribution would have a size asymptotically independent of T.

Acknowledgement. This work has been supported in part by JSPS KAKENHI Grant Numbers JP23KJ0548 and JPMJCR22M1.

References

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. Nistir 8413 – status report on the third round of the nist post-quantum cryptography standardization process, 2022. https://doi.org/10.6028/NIST.IR.8413.

[ABV+12] Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 280–297. Springer, Heidelberg, May 2012.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 327–343. USENIX Association, August 2016.

[AL21] Martin R. Albrecht and Russell W. F. Lai. Subtractive sets over cyclotomic rings - limits of Schnorr-like arguments over lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 519–548, Virtual Event, August 2021. Springer, Heidelberg.

[ANO+22] Damiano Abram, Ariel Nof, Claudio Orlandi, Peter Scholl, and Omer Shlomovits. Low-bandwidth threshold ECDSA via pseudorandom correlation generators. In *2022 IEEE Symposium on Security and Privacy*, pages 2554–2572. IEEE Computer Society Press, May 2022.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl, July 2022.

[BBD$^+$23] Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Andreas Hülsing, Yi Lee, and Xiaodi Wu. Fixing and mechanizing the security proof of fiat-shamir with aborts and dilithium. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 358–389. Springer, Heidelberg, August 2023.

[BCK$^+$22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550. Springer, Heidelberg, August 2022.

[BGG$^+$18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Heidelberg, August 2018.

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, December 2019.

[BLL$^+$15] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 3–24. Springer, Heidelberg, November / December 2015.

[BLMP19] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the CSIDH: Optimizing quantum evaluation of isogenies. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 409–441. Springer, Heidelberg, May 2019.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013.

[BLR$^+$18] Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, April 2018.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003.

[BS20] Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 493–522. Springer, Heidelberg, May 2020.

[BTT22] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 276–305. Springer, Heidelberg, August 2022.

[CAD$^+$20] David Cooper, Daniel Apon, Quynh Dang, Michael Davidson, Morris Dworkin, and Carl Miller. Recommendation for stateful hash-based signature schemes. National Institute of Standards and Technology, 2020. https://doi.org/10.6028/NIST.SP.800-208.

[CCK23] Jung Hee Cheon, Wonhee Cho, and Jiseung Kim. Improved universal thresholdizer from threshold fully homomorphic encryption. Cryptology ePrint Archive, Paper 2023/545, 2023. https://eprint.iacr.org/2023/545.

[Che23a] Yanbo Chen. DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. Cryptology ePrint Archive, Report 2023/263, 2023. https://eprint.iacr.org/2023/263.

[Che23b] Yanbo Chen. Dualms: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. Cryptology ePrint Archive, Paper 2023/263, 2023. To Appear at CRYPTO.

[CKM21] Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. Cryptology ePrint Archive, Report 2021/1375, 2021. https://eprint.iacr.org/2021/1375.

[CPS$^+$20] Chitchanok Chuengsatiansup, Thomas Prest, Damien Stehlé, Alexandre Wallet, and Keita Xagawa. ModFalcon: Compact signatures based on module-NTRU lattices. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIACCS 20*, pages 853–866. ACM Press, October 2020.

[CS20] Daniele Cozzo and Nigel P. Smart. Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 169–186. Springer, Heidelberg, 2020.

[CSCJR22] Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. The SQALE of CSIDH: sublinear Vélu quantum-resistant isogeny action with low exponents. *Journal of Cryptographic Engineering*, 12(3):349–368, September 2022.

[Csi63] Imre Csiszár. Eine informationstheoretische Ungleichung und ihre Anwendung auf den Beweis der Ergodizitat von Markoffschen Ketten. *Magyar. Tud. Akad. Mat. Kutató Int. Közl*, 8:85–108, 1963.

[CSS$^+$22] Siddhartha Chowdhury, Sayani Sinha, Animesh Singh, Shubham Mishra, Chandan Chaudhary, Sikhar Patranabis, Pratyay Mukherjee, Ayantika Chatterjee, and Debdeep Mukhopadhyay. Efficient threshold FHE with application to real-time systems. Cryptology ePrint Archive, Report 2022/1625, 2022. https://eprint.iacr.org/2022/1625.

[Des90] Yvo Desmedt. Abuses in cryptography and how to fight them. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 375–389. Springer, Heidelberg, August 1990.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, August 1990.

[DFPS23] Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of fiat-shamir with aborts. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 327–357. Springer, Heidelberg, August 2023.

[DKL$^+$18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR TCHES*, 2018(1):238–268, 2018. https://tches.iacr.org/index.php/TCHES/article/view/839.

[DLN$^+$21] Julien Devevey, Benoît Libert, Khoa Nguyen, Thomas Peters, and Moti Yung. Non-interactive CCA2-secure threshold cryptosystems: Achieving adaptive security in the standard model without pairings. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 659–690. Springer, Heidelberg, May 2021.

[DM20] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 187–212. Springer, Heidelberg, May 2020.

[DOTT21] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 99–130. Springer, Heidelberg, May 2021.

[DOTT22] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):14, April 2022.

[dPEK$^+$23] Rafaël del Pino, Thomas Espitau, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, Mélissa Rossi, and Markku-Juhani Saarinen. Raccoon. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[dPPRS23] Rafaël del Pino, Thomas Prest, Mélissa Rossi, and Markku-Juhani O. Saarinen. High-order masking of lattice signatures in quasilinear time. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, 22-25 May 2023*, pages 1168–1185. IEEE, May 2023.

[Duc18] Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 125–145. Springer, Heidelberg, April / May 2018.

[EJK20] Thomas Espitau, Antoine Joux, and Natalia Kharchenko. On a dual/hybrid approach to small secret LWE - A dual/enumeration technique for learning with errors and application to security estimates of FHE schemes. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 440–462. Springer, Heidelberg, December 2020.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[FSZ22] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1109–1123. ACM Press, November 2022.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 230–240. Tsinghua University Press, 2010.

[GKS23] Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice signatures from threshold homomorphic encryption. Cryptology ePrint Archive, Paper 2023/1318, 2023. https://eprint.iacr.org/2023/1318.

[HLS18] Andreas Hülsing, Tanja Lange, and Kit Smeets. Rounded Gaussians - fast and secure constant-time sampling for lattice-based crypto. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 728–757. Springer, Heidelberg, March 2018.

[Jan06] Svante Janson. Rounding of continuous random variables and oscillatory asymptotics. *The Annals of Probability*, 34(5):1807 – 1826, 2006.

[KCLM22] Irakliy Khaburzaniya, Konstantinos Chalkias, Kevin Lewi, and Harjasleen Malvai. Aggregating and thresholdizing hash-based signatures using STARKs. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIACCS 22*, pages 393–407. ACM Press, May / June 2022.

[KG20] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Heidelberg, October 2020.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Heidelberg, April / May 2018.

[KLSS23] Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-mlwe. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 549–580, Cham, 2023. Springer Nature Switzerland.

[KY02] Jonathan Katz and Moti Yung. Threshold cryptosystems based on factoring. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 192–205. Springer, Heidelberg, December 2002.

[LDK$^+$22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[Lin22] Yehuda Lindell. Simple three-round multiparty schnorr signing with full simulatability. Cryptology ePrint Archive, Report 2022/374, 2022. https://eprint.iacr.org/2022/374.

[LJY14] Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM PODC*, pages 303–312. ACM, July 2014.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Heidelberg, May 2013.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[LST18] Benoît Libert, Damien Stehlé, and Radu Titiu. Adaptively secure distributed PRFs from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 391–421. Springer, Heidelberg, November 2018.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, April 2012.

[MB64] George Marsaglia and Thomas A. Bray. A convenient method for generating normal variables. *SIAM Review*, 6(3):260–264, 1964.

[MR02] Silvio Micali and Leonid Reyzin. Improving the exact security of digital signature schemes. *Journal of Cryptology*, 15(1):1–18, January 2002.

[NIS15] NIST. SHA-3 standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards Publication FIPS 202, August 2015.

[NIS22] NIST. Call for additional digital signature schemes for the post-quantum cryptography standardization process. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf, 2022.

[NIS23] NIST. Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standards Publication FIPS 204 (Draft), August 2023.

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 354–369. Springer, Heidelberg, August 1998.

[PB23] René Peralta and Luís T.A.N. Brandão. Nist first call for multi-party threshold schemes. National Institute of Standards and Technology, 2023. https://doi.org/10.6028/NIST.IR.8214C.ipd.

[Pei20] Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 463–492. Springer, Heidelberg, May 2020.

[Pre17] Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 347–374. Springer, Heidelberg, December 2017.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[Rén61] Alfréd Rényi. On Measures of Entropy and Information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 547–561, Berkeley, Calif., 1961. University of California Press.

[RRJ$^+$22] Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. ROAST: Robust asynchronous schnorr threshold signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2551–2564. ACM Press, November 2022.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Sha79a] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sha79b] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[Sho00] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, Heidelberg, May 2000.

[Val84] Leslie G. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5(3):363–366, 1984.

[vEH14] Tim van Erven and Peter Harremoës. Rényi divergence and kullback-leibler divergence. *IEEE Trans. Information Theory*, 60(7):3797–3820, 2014.

A Deferred Definitions

In this section, we provide the deferred definitions from Section 3.

A.1 Hardness of Lattice-Related Problems

Here we provide all the omitted details on the lattice-related hardness problems. We first introduce the MLWE and MSIS problems below.

Definition A.1 (MLWE)**.** *Let ℓ, k, q be integers and $\mathcal{D}$ be a probability distribution over $\mathcal{R}_q$. The advantage of an adversary $\mathcal{A}$ against the* Module Learning with Errors $\mathsf{MLWE}_{q,\ell,k,\mathcal{D}}$ *problem is defined as:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MLWE}}(\kappa) = |\Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{A}\cdot\mathbf{s}+\mathbf{e})] - \Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b})]|$$

where $(\mathbf{A}, \mathbf{b}, \mathbf{s}, \mathbf{e}) \leftarrow \mathcal{R}_q^{k\times\ell} \times \mathcal{R}_q^k \times \mathcal{D}^\ell \times \mathcal{D}^k$. The $\mathsf{MLWE}_{q,\ell,k,\mathcal{D}}$ assumption states that any efficient adversary $\mathcal{A}$ has negligible advantage. We may write $\mathsf{MLWE}_{q,\ell,k,\sigma}$ as a shorthand for $\mathsf{MLWE}_{q,\ell,k,\mathcal{D}}$ when $\mathcal{D}$ is the Gaussian distribution of standard deviation σ.

Definition A.2 (MSIS)**.** *Let ℓ, k, q be integers and $B_{\mathsf{msis}} > 0$ a real number. The advantage of an adversary $\mathcal{A}$ against the* Module Short Integer Solution $\mathsf{MSIS}_{q,\ell,k,B_{\mathsf{msis}}}$ *problem is defined as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MSIS}}(\kappa) = \Pr\left[\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}, \mathbf{s} \leftarrow \mathcal{A}(\mathbf{A}) : 0 < \|\mathbf{s}\|_2 \le B_{\mathsf{msis}} \wedge \left[\mathbf{A} \mid \mathbf{I}\right]\cdot\mathbf{s} = \mathbf{0} \bmod q\right].$$

The $\mathsf{MSIS}_{q,\ell,k,B_{\mathsf{msis}}}$ assumption states that any efficient adversary $\mathcal{A}$ has negligible advantage.

The following two results relate the MLWE and MSIS problems to certain worst-case lattice problems, where recall n is the dimension of $\mathcal{R}_q$ and all the parameters are implicitly a function of the security parameter.

Lemma A.3 (Hardness of MLWE ([LS15]))**.** *For any integers ℓ, k, q, n and real σ such that $q \le \mathsf{poly}(n\ell)$, $k \le \mathsf{poly}(\ell)$, and $\sigma \ge \sqrt{\ell}\cdot\omega_{\mathsf{asymp}}(\sqrt{\log n})$, the $\mathsf{MLWE}_{q,\ell,k,\sigma}$ problem is as hard as the worst-case lattice Generalised-Independent-Vector-Problem (GIVP) in dimension $N = n\ell$ with approximation factor $\sqrt{8N\ell}\cdot\omega_{\mathsf{asymp}}(\sqrt{\log n})\cdot q/\sigma$.*

Lemma A.4 (Hardness of MSIS ([LS15])). *For any integers ℓ, k, q, n and positive real B_{msis} such that $q > B_{\mathsf{msis}}\sqrt{nk} \cdot \omega_{\mathsf{asymp}}(\log(nk))$, $\ell, \log q \leq \mathsf{poly}(nk)$, the $\mathsf{MSIS}_{q,\ell,k,B_{\mathsf{msis}}}$ problem is as hard as the worst-case lattice Generalised-Independent-Vector-Problem (GIVP) in dimension $N = nk$ with approximation factor $B_{\mathsf{msis}}\sqrt{N} \cdot \omega_{\mathsf{asymp}}(\sqrt{\log N})$.*

Lastly, we recall the following result establishing the hardness of the Hint-MLWE problem based on the MLWE problem. Below, $s_1(c)$ denotes the spectral norm of $c \in \mathcal{R}_q$ and c^* denotes the Hermitian adjoint of c (see Appendix B.2 for more details on the spectral norm of ring elements).

Lemma A.5 (Hardness of Hint-MLWE [KLSS23]). *For any integers ℓ, k, q, n, Q, set $\mathcal{C} \subset \mathcal{R}_q$, and positive reals $B_{\mathsf{hint}}, \sigma, \sigma_{\mathcal{D}}, \sigma_{\mathcal{G}}$ such that $\Pr[s_1(\sum_{i\in[Q]} c_i \cdot (c_i)^*) < B_{\mathsf{hint}} : c_i \leftarrow \mathcal{C}] \geq 1 - \mathsf{negl}(\kappa)$, $\sigma = \omega_{\mathsf{asymp}}(\sqrt{\log n})$, and $\frac{1}{\sigma^2} = 2 \cdot \left(\frac{1}{\sigma_{\mathcal{D}}^2} + \frac{B_{\mathsf{hint}}}{\sigma_{\mathcal{G}}^2}\right)$, the $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q,\sigma_{\mathcal{D}},\sigma_{\mathcal{G}},\mathcal{C}}$ problem is as hard as the $\mathsf{MLWE}_{q,\ell,k,\sigma}$ problem.*

A.2 Pseudorandom Function

Here, we define pseudorandom functions PRF. Below, we define a multi-instance variant where the adversary can query many PRF seeds.

Definition A.6 (Pseudorandom function (PRF)). *Let n, ℓ be positive integers, implicitly a function of the security parameter κ. We define $\mathsf{Game}_{\mathcal{A}}^{\mathsf{PRF}}$ in Fig. 15 for an adversary $\mathcal{A}$. We say a deterministic PPT algorithm $\mathsf{PRF} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^\ell$ is a* pseudorandom function *if for any efficient adversary $\mathcal{A}$, the following advantage of $\mathcal{A}$ is in $\mathsf{negl}(\kappa)$:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRF}}(\kappa) = \left| \Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{PRF}}(\kappa) = 1] - \frac{1}{2} \right|.$$

```
Alg. 35: $\mathsf{Game}_{\mathcal{A}}^{\mathsf{PRF}}(\kappa)$
1: $L[\cdot] := \perp$
2: $b \leftarrow \{0,1\}$
3: $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{PRF}}(\cdot)}(\kappa)$
4: if $b = b'$ then
5:     return 1
6: return 0
```

```
Alg. 36: $\mathcal{O}_{\mathsf{PRF}}(i, x \in \{0,1\}^n)$
1: if $L[i] = \perp$ then
2:     $\mathsf{seed} \leftarrow \{0,1\}^\kappa$
3:     $L[i] \leftarrow \mathsf{seed}$
4: $\mathsf{seed} := L[i]$
5: $y_0 \leftarrow \{0,1\}^\ell$
6: $y_1 := \mathsf{PRF}(\mathsf{seed}, x)$
7: return $y_b$
```

Figure 15: PRF security game. For simplicity, we restrict the adversary to only query an input string x of length ℓ.

A.3 Signature Scheme

We provide the formal definition of a standard single signer signature scheme. A *signature* scheme is a triple of algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ such that:

$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$:

The key generation algorithm takes as input the security parameter κ and outputs a public key vk and a signing key sk.

```
Alg. 37: $\mathsf{Game}^{\mathsf{sig\text{-}uf}}_{\mathcal{A}}(\kappa)\}$
1: $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$
2: $L_{\mathsf{Sign}} := \emptyset$
3: $(\mathsf{msg}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{OSgn}(\cdot)}(\mathsf{vk})$
4: if $\exists \sigma'$ s.t. $(\mathsf{msg}^*, \sigma') \in L_{\mathsf{Sign}}$ then
5:     return 0
6: return $\mathsf{Verify}(\sigma^*, \mathsf{msg}^*, \mathsf{vk})$
```

```
Alg. 38: $\mathsf{OSgn}(\mathsf{msg})$
1: $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$
2: $L_{\mathsf{Sign}} := L_{\mathsf{Sign}} \cup \{(\mathsf{msg}, \sigma)\}$
3: return $\sigma$
```

Figure 16: Unforgeability game for signatures.

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$:

The signing algorithm takes as input a signing key sk and a message msg, and outputs a signature σ.

$\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \sigma)$:

The verification algorithm takes as input a public key vk, a message msg and a signature σ, and outputs 1 if the signature is accepted and 0 otherwise.

Definition A.7 (Correctness)**.** *A digital signature is* correct *if for any valid message* $\mathsf{msg} \in \mathcal{M}$*, it holds that*

$$\Pr[\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})) = 1 : (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)] \geq 1 - \mathsf{negl}(\kappa).$$

Definition A.8 (Unforgeability)**.** *We define* $\mathsf{Game}^{\mathsf{sig\text{-}uf}}_{\mathcal{A}}$ *in Fig. 16 for an adversary* $\mathcal{A}$*. We say a digital signature is* unforgeable *if for any efficient* $\mathcal{A}$*, the following advantage of* $\mathcal{A}$ *is* $\mathsf{negl}(\kappa)$*:*

$$\mathsf{Adv}^{\mathsf{sig\text{-}uf}}_{\mathcal{A}}(\kappa) := \Pr[\mathsf{Game}^{\mathsf{sig\text{-}uf}}_{\mathcal{A}}(\kappa) = 1].$$

We also define a Q_{Sign}-bounded scheme where any adversary is limited to make at most $Q_{\mathsf{Sign}} = \mathsf{poly}(\kappa)$ signing queries. See Remark 4.3 for more discussion.

A.4 Correctness of Threshold Signatures

We define the correctness of a threshold signature scheme in this section.

Definition A.9 (Correctness)**.** *Let* $\mathsf{pp}(\kappa)$ *be a parameter generating algorithm that takes as input the security parameter* 1^κ*. We define* $\mathsf{Game}^{\mathsf{ts\text{-}corr}}$ *in Fig. 17. We say a threshold signature is* correct *if for any threshold* T*, number of parties* N*,* $\mathsf{act} \subseteq [N]$ *with* $|\mathsf{act}| \geq T$*, and message* $\mathsf{msg} \in \{0,1\}^*$*, it holds that:*

$$\Pr[\mathsf{Game}^{\mathsf{ts\text{-}corr}}(1^\kappa, T, N, \mathsf{act}, \mathsf{msg}) = 1] > 1 - \mathsf{negl}(\kappa).$$

A.5 Communication Channel

As with all threshold signatures, $\mathsf{TRaccoon}$ requires a communication network such that parties can send messages to each other. In this work we abstract away the mechanics about the network. We do not specify how parties receive the inputs $\mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{contrib}_i$ for the algorithms Sign_i, which ultimately is a consensus problem.

There are two common solutions for agreeing on the parties inputs for schemes such as $\mathsf{TRaccoon}$ that require at least T honest parties in order to terminate with a valid signature. Both solutions ensure that if all parties are honest then a valid signature will eventually be produced.

```
$\mathsf{Game}^{\mathsf{ts\text{-}corr}}(1^\kappa, T, N, \mathsf{act}, \mathsf{msg})$
1: assert{ $\mathsf{act} \subseteq [N]$ and $|\mathsf{act}| \geq T$ }
2: $(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}(1^\kappa), T, N)$
3: for $j \in \mathsf{act}$ do
4:     $(\mathsf{state}_j, \mathsf{session}_{\mathsf{sid}}, \mathsf{contrib}_1[j]) \leftarrow \mathsf{ShareSign}_1(\mathsf{state}_j, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$
5: for $i \in \{2, \ldots, \mathsf{rnd}\}$ do
6:     for $j \in \mathsf{act}$ do
7:         $(\mathsf{state}_j, \mathsf{contrib}_i[j]) \leftarrow \mathsf{ShareSign}_i(\mathsf{state}_j, \mathsf{contrib}_{i-1})$
8: $\mathsf{sig} := \mathsf{Combine}(\mathsf{vk}, \mathsf{sid}, \mathsf{msg}, (\mathsf{contrib}_i)_{i\in[\mathsf{rnd}]})$
9: return $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$
```

Figure 17: Correctness game for threshold signatures.

Synchronous broadcast channels.

The first solution is to assume a synchronous broadcast channel where all parties receive the outputs of all other parties in each round. The adversary cannot modify messages sent over the broadcast channel, nor prevent their delivery. Here we can assume that the protocol will terminate but not that the output will verify. Synchronous broadcast channels are difficult to realise in practice and typically require a peer-to-peer network and large time bounds.

Coordinator.

The second method for agreeing on inputs is to assume that the network is asynchronous with no time bounds on when signers should respond, but that there is a *coordinator i.e.* an authority that routes communications toward the correct receiver. The coordinator waits until they have received messages from all parties and only then forward the relevant $\mathsf{contrib}_i$ back. Unforgeability holds even when the coordinator is malicious. If a signer never responds then the session will not terminate. Similarly if a malicious coordinator aborts then the session will not terminate and no signature will be produced.

B Deferred Proofs from Section 3

In this section, we provide the deferred proofs from Section 3.

B.1 Proof of Hardness of SelfTargetMSIS

As discussed in Section 3.4, SelfTargetMSIS is known to be as difficult as MSIS. For completeness, we provide a reduction from SelfTargetMSIS to MSIS and display the asymptotic relations of the parameters.

Lemma B.1 (Hardness of SelfTargetMSIS)**.** *Then, for any adversary $\mathcal{A}$ against the* $\mathsf{SelfTargetMSIS}_{q,\ell,k,\mathcal{C},B_{\mathsf{stmsis}}}$ *problem making at most Q_{H} random oracle queries, there exists an adversary $\mathcal{B}$ against the* $\mathsf{MSIS}_{q,\ell,k,B_{\mathsf{msis}}}$ *problem with $B_{\mathsf{msis}} = 2B_{\mathsf{stmsis}}$ such that*

$$\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{A}}(\kappa) \leq \sqrt{Q_{\mathsf{H}} \cdot \mathsf{Adv}^{\mathsf{MSIS}}_{\mathcal{B}}(\kappa)} + \frac{Q_{\mathsf{H}}}{|\mathcal{C}|},$$

where $\mathsf{Time}(\mathcal{B}) \approx 2 \cdot \mathsf{Time}(\mathcal{A})$.

Proof. To construct $\mathcal{B}$, we simply invoke the standard forking lemma [BN06] to run $\mathcal{A}$ twice. In particular, when $\mathcal{B}$ receives $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}$ as input, it invokes $\mathcal{A}$ on input $\mathbf{A}$. $\mathcal{B}$ simulates the random oracle H on the fly by sampling a random $c \leftarrow \mathcal{C}$. Eventually, $\mathcal{A}$ outputs $(\mathbf{z}, \mathsf{msg}) \in \mathcal{R}_q^{\ell+k} \times \{0,1\}^{2\kappa}$ that breaks $\mathsf{SelfTargetMSIS}$. That is

$$\left(\mathbf{z} = \begin{bmatrix} c \\ \mathbf{z}' \end{bmatrix}\right) \wedge \|\mathbf{z}\|_2 \leq B_{\mathsf{stmsis}} \;\wedge\; \mathsf{G}\left(\left[\mathbf{A} \mid \mathbf{I}\right] \cdot \mathbf{z},\; \mathsf{msg}\right) = c.$$

Using the forking lemma [BN06], we can argue that when $\mathcal{B}$ rewinds $\mathcal{A}$, $\mathcal{A}$ outputs $(\widehat{\mathbf{z}}, \widehat{\mathsf{msg}})$ with a different $\widehat{c} \neq c$ such that

$$\left(\widehat{\mathbf{z}} = \begin{bmatrix} \widehat{c} \\ \widehat{\mathbf{z}'} \end{bmatrix}\right) \wedge \|\widehat{\mathbf{z}}\|_2 \leq B_{\mathsf{stmsis}} \;\wedge\; \mathsf{G}\left(\left[\mathbf{A} \mid \mathbf{I}\right] \cdot \widehat{\mathbf{z}},\; \widehat{\mathsf{msg}}\right) = \widehat{c}.$$

with the specified probability in the statement. Moreover, the forking lemma allows us to argue that the input are the same on the challenge output. That is, we have $\left[\mathbf{A} \mid \mathbf{I}\right] \cdot \mathbf{z} = \left[\mathbf{A} \mid \mathbf{I}\right] \cdot \widehat{\mathbf{z}}$.

To conclude, $\mathcal{B}$ simply sets $\mathbf{z}^* = \mathbf{z} - \widehat{\mathbf{z}} \in \mathcal{R}_q^{\ell+k}$ as its MSIS solution. When $c \neq \widehat{c}$, which happens with probability at least $1 - \frac{Q_{\mathsf{H}}}{|\mathcal{C}|}$, $\mathbf{z}^* \neq \mathbf{0}_{\ell+k}$ as desired. Moreover, $\|\mathbf{z}^*\|_2 \leq \|\mathbf{z}\|_2 + \|\widehat{\mathbf{z}}\|_2 \leq 2B_{\mathsf{stmsis}} = B_{\mathsf{msis}}$ as desired. Thus $\mathbf{z}^*$ is a valid MSIS solution and this completes the proof. □

B.2 Bounding the Spectral Norm

The spectral norm $s_1(\mathbf{M})$ of a matrix $\mathbf{M}$ is defined as the value $\max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{M}\,\mathbf{x}\|}{\|\mathbf{x}\|}$. We recall that if a matrix is symmetric, then its spectral norm is also its largest eigenvalue. Given a polynomial $c \in \mathcal{R}$, we may abusively use the term "spectral norm $s_1(c)$ of c" when referring to the spectral norm of the anti-circulant matrix $\mathcal{M}(c)$ associated to c. Finally, if $c(x) = \sum_{0\leq i<n} c_i\, x^i$, then the Hermitian adjoint of c, which we denote by c^*, is defined as $c^*(x) = c_0 - \sum_{0<i<n} c_{n-i}\, x^i$. Note that $\mathcal{M}(c)^t = \mathcal{M}(c^*)$.

Lemma B.2. *For $j \in [Q_{\mathsf{Sign}}]$, let $c^{[j]} \leftarrow \mathcal{C}$, where $\mathcal{C}$ is defined as in Eq. (9). Let $D = \sum_{j\in[Q_{\mathsf{Sign}}]} c^{[j]}\,(c^{[j]})^*$. We then have $\Pr\left[s_1(D) \geq B_{\mathsf{hint}}\right] \leq 2^{-\kappa}$, where:*

$$B_{\mathsf{hint}} = Q_{\mathsf{Sign}} \cdot \omega \cdot \left(1 + n\frac{1}{\sqrt{Q_{\mathsf{Sign}}}}\left(\kappa + 1 + 2\,\log(n)\right)\right)$$

Specifically, when $(\kappa\, n)^2 = o(Q_{\mathsf{Sign}})$, then $s_1(D)$ is equivalent to $Q_{\mathsf{Sign}} \cdot \omega$.

Proof. First, let us consider a single $c \leftarrow \mathcal{C}$. Let $d(x) = c\,c^* = \sum_{0\leq i<n} d_i\, x^i$. For each $k \in \{0,\ldots,n-1\}$, d_k can be expressed explicitly as:

$$d_k = \sum_{0\leq i<n-k} c_i\, c_{i+k} - \sum_{n-k\leq i<n} c_i\, c_{i+k-n}. \tag{14}$$

From Eq. (14), it is clear that:

1. $d_0 = \|c\|_2^2 = \omega$
2. If $k \neq 0$, d_k is a random variable satisfying:

$$\|d_k\| \leq \|c\|_2^2 = \omega \tag{15}$$
$$\mathbb{E}[d_k] = 0. \tag{16}$$

While Eq. (15) is immediate from Eq. (14), Eq. (16) is a bit more subtle. The mapping $(k,i) \in \{0,\ldots,n-1\}^2 \mapsto (i+k \bmod n)$ is a group action. As such, its orbits form a partition of $\{0,\ldots,n-1\}$, and we note that each orbit has an even number of elements. Each orbit can be written as

$\{i_0 + k\,x \bmod n | x \in n/\gcd(k,n)\}$, where i_0 is (say) the smallest element in this orbit. We define the function $\delta_k : \{0, \ldots, n-1\} \to \{-1, 1\}$ as follows: for each $0 \leq i < n$, we determine its orbit, write $i = i_0 + k\,x$ in this orbit, and set $\delta_k(i) = (-1)^x$. Now consider the mapping $\varphi_k : \mathcal{C} \to \mathcal{C}$ defined as

$$\varphi_k : \left(c = \sum_{0 \leq i < n} c_i\, x^i \right) \longrightarrow \left(c' = \sum_{0 \leq i < n} \delta_k(i)\, c_i\, x^i \right). \tag{17}$$

φ_k is an involution of $\mathcal{C}$, and one can check from Eqs. (14) and (17) that $(c\,c^*)_k = -(c'\,c'^*)_k$. Since φ_k is an involution, this implies that $\mathbb{E}[d_k] = 0$.

Let us note $d^{[j]} = c^{[j]}\,(c^{[j]})^*$. For $d \neq 0$, we can bound the sum $D_k = \sum_{j \in [Q_{\mathsf{Sign}}]} d_k^{[j]}$ by combining Hoeffding's inequality with Eqs. (15) and (16):

$$|D_k| \leq \omega \sqrt{2 Q_{\mathsf{Sign}}}\,((\kappa+1)\log(2) + \log(n)) \tag{18}$$

except with probability at most $2^{-\kappa}/n$. From the union bound, the above inequality is true for all $k \neq 0$, except with probability $\leq 2^{-\kappa}$. We can now bound the spectral norm of D. Since D is self-adjoint, $s_1(D)$ is the largest eigenvalue of D, that is $D(\zeta)$ for some primitive root of unity $\zeta, |\zeta| = 1$. Therefore with probability $\geq 1 - 2^{-\kappa}$:

$$\begin{aligned} s_1(D) = D(\zeta) &\leq D_0 + \sum_{k \neq 0} |D_k| \\ &\leq Q_{\mathsf{Sign}}\,\omega + (n-1)\,\omega\,\sqrt{2\,Q_{\mathsf{Sign}}}\,((\kappa+1)\log(2) + \log(n)) \end{aligned}$$

This concludes the proof after a few simplifications. □

C Correctness and Security of Raccoon: Theorem 5.1

In this section, we provide the correctness and security proofs of our slight variant of Raccoon [dPPRS23].

C.1 Asymptotic Parameters

We first give a asymptotic parameter for which the scheme can be proven correct and secure. For reference, we recall the set of parameters used by the scheme in Table 1. Note the restriction on $(q_{\mathbf{t}}, q_{\mathbf{w}})$ allows us to perform rounding operations nicely (see Lemma 3.2).

For unforgeability, we require the $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ and $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ problems to be hard. More specifically, we require the following conditions.

- $\sigma \geq \sqrt{\ell} \cdot \omega_{\mathsf{asymp}}(\sqrt{\log n})$ for Lemma A.3 (hardness of MLWE).
- $B_{\mathsf{hint}} = Q_{\mathsf{Sign}} \cdot \omega \cdot \left(1 + n \frac{1}{\sqrt{Q_{\mathsf{Sign}}}} (\kappa + 1 + 2\log(n))\right)$ and $\frac{1}{\sigma^2} = 2 \cdot \left(\frac{1}{\sigma_{\mathbf{t}}^2} + \frac{B_{\mathsf{hint}}}{\sigma_{\mathbf{w}}^2}\right)$ for Lemmas A.5 and B.2 (reduction from $\mathsf{Hint\text{-}MLWE}$ to MLWE).
- $q > B_{\mathsf{msis}} \sqrt{nk} \cdot \omega_{\mathsf{asymp}}(\log(nk))$ for Lemma A.4 (hardness of MSIS).
- $B_{\mathsf{msis}} = 2\,B_{\mathsf{stmsis}}$ for Lemma B.1 (reduction from $\mathsf{SelfTargetMSIS}$ to MSIS).

In the above, note that Q_{Sign} denotes the maximum signature query an adversary can perform.

Candidate Asymptotic Parameters. We give a set of asymptotic parameters which fit the above constraints.

- $n, \ell, k = \mathsf{poly}(\kappa)$ such that $n \geq \kappa$.

- $\omega = \omega_{\mathsf{asymp}}(1)$ for $|\mathcal{C}| \geq 2^\kappa$ for Lemma B.1 (hardness of $\mathsf{SelfTargetMSIS}$).
- $(\sigma_{\mathbf{t}}, \sigma_{\mathbf{w}}) = \left(2\sqrt{\ell} \cdot \log n, 2\sqrt{B_{\mathsf{hint}} \cdot \ell} \cdot \log n\right)$.
- $\nu_{\mathbf{t}}, \nu_{\mathbf{w}} = O(\log \lambda)$, where $\nu_{\mathbf{w}} \geq 4$ for correctness (see Appendix C.2).
- $B_2 = e^{1/4} \cdot (\omega\sigma_{\mathbf{t}} + \sigma_{\mathbf{w}})\sqrt{n(k+\ell)} + (\omega \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1}) \cdot \sqrt{nk}$ for correctness (see Appendix C.2).
- $B_{\mathsf{stmsis}} = B_2 + \sqrt{\omega} + (\omega \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1}) \cdot \sqrt{nk}$ for Lemma C.4.
- q is the smallest prime larger than $2\, B_{\mathsf{stmsis}}\sqrt{nk}\, \log(nk)^2$ such that $(q, \nu_{\mathbf{t}}, \nu_{\mathbf{w}})$ satisfy the condition in Table 1.

C.2 Correctness

The following establishes the correctness of the Raccoon signature scheme in Fig. 3.

Lemma C.1 (Correctness)**.** *The Raccoon signature scheme in Fig. 3 is correct if $\nu_{\mathbf{w}} \geq 4$ and:*

$$B_2 = e^{1/4} \cdot (\omega\sigma_{\mathbf{t}} + \sigma_{\mathbf{w}})\sqrt{n(k+\ell)} + (\omega \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1}) \cdot \sqrt{nk}.$$

Proof. It is clear that the check $\{c = c'\}$ holds. In the proof, we focus on the check of the L_2-norm of the signature. Let us denote $\hat{\mathbf{t}} = \mathbf{A}\mathbf{s} + \mathbf{e}$ and $\hat{\mathbf{w}} = \mathbf{A}\mathbf{r} + \mathbf{e}'$, where $\mathbf{t} = \lfloor \hat{\mathbf{t}} \rceil_{\nu_{\mathbf{t}}}$ and $\mathbf{w} = \lfloor \hat{\mathbf{w}} \rceil_{\nu_{\mathbf{w}}}$. Below, we will be precise on where each values live and explicit with the lift. Let us define $\mathbf{y}$ as follows:

$$\begin{aligned}\mathbf{y} &\coloneqq \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathbf{t}}} \cdot c \cdot \overline{\mathbf{t}} \rceil_{\nu_{\mathbf{w}}} = \lfloor c \cdot \mathbf{A}\mathbf{s} + \mathbf{A}\mathbf{r} - 2^{\nu_{\mathbf{t}}} \cdot c \cdot \overline{\mathbf{t}} \rceil_{\nu_{\mathbf{w}}} \\ &= \left\lfloor \hat{\mathbf{w}} + c \cdot \underbrace{\left(\hat{\mathbf{t}} - 2^{\nu_{\mathbf{t}}} \cdot \overline{\lfloor \hat{\mathbf{t}} \rceil_{\nu_{\mathbf{t}}}}\right)}_{=:\mathbf{a}_{\mathbf{t}} \in \mathcal{R}_q^k} - \underbrace{c \cdot \mathbf{e} - \mathbf{e}'}_{=:\mathbf{a} \in \mathcal{R}_q^k} \right\rceil_{\nu_{\mathbf{w}}} .\end{aligned}$$

Plugging in the above $\mathbf{y}$ into $\mathbf{h}$, we have

$$\begin{aligned}\left\| 2^{\nu_{\mathbf{w}}} \cdot \bar{\mathbf{h}} \mod q \right\|_2 &= \left\| 2^{\nu_{\mathbf{w}}} \cdot \overline{(\mathbf{w} - \mathbf{y})} \mod q \right\|_2 \\ &= \left\| 2^{\nu_{\mathbf{w}}} \cdot \overline{(\lfloor \hat{\mathbf{w}} \rceil_{\nu_{\mathbf{w}}} - \lfloor \hat{\mathbf{w}} + c \cdot \mathbf{a}_{\mathbf{t}} - \mathbf{a} \rceil_{\nu_{\mathbf{w}}})} \mod q \right\|_2 \\ &\leq \left\| 2^{\nu_{\mathbf{w}}} \cdot \overline{\lfloor c \cdot \mathbf{a}_{\mathbf{t}} - \mathbf{a} \rceil_{\nu_{\mathbf{w}}}} \mod q \right\|_2 + \sqrt{nk} \cdot 2^{\nu_{\mathbf{w}}} \\ &\leq \| c \cdot \mathbf{a}_{\mathbf{t}} - \mathbf{a} \|_2 + \sqrt{nk} \cdot 2^{\nu_{\mathbf{w}}+1} \\ &\leq \| c \cdot \mathbf{a}_{\mathbf{t}} \|_2 + \| \mathbf{a} \|_2 + \sqrt{nk} \cdot 2^{\nu_{\mathbf{w}}+1},\end{aligned}$$

where we used Lemma 3.2, Eq. (6) (resp. Eq. (5)) in the first (resp. second) inequality and Lemma 3.1 for the last. Using the Minkowski inequality and Lemma 3.2, Eq. (5), we have:

$$\| c \cdot \mathbf{a}_{\mathbf{t}} \|_2 \leq \| c \|_1 \cdot \sqrt{nk} \cdot 2^{\nu_{\mathbf{t}}}.$$

Moreover, using Lemma 3.4, the following holds with overwhelming probability:

$$\begin{aligned}\| (\mathbf{z}, \mathbf{a}) \|_2 &\leq \| c \cdot (\mathbf{s}, \mathbf{e}) \|_2 + \| (\mathbf{r}, \mathbf{e}') \|_2 \\ &\leq e^{1/4} \cdot (\| c \|_1 \sigma_{\mathbf{t}} + \sigma_{\mathbf{w}}) \cdot \sqrt{n(k+\ell)}\end{aligned}$$

Collecting all the bounds and plugging in $\| c \|_1 = \omega$, we can check that $\left\| (\mathbf{z}, 2^{\nu_{\mathbf{w}}} \cdot \bar{\mathbf{h}} \mod q) \right\|_2 \leq B_2$ as desired. □

C.3 Unforgeability

The following is the main theorem, stated more explicitly than Theorem 5.1, establishing the unforgeability of the variant of the Raccoon signature scheme. The statement assumes the asymptotic parameter selections in Appendix C.1.

Theorem C.2. *The Raccoon signature scheme in Fig. 3 is unforgeable under the* $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ *and* $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ *assumptions.*

Formally, for any adversary $\mathcal{A}$ *against the unforgeability game making at most* Q_{H} *and* Q_{Sign} *queries to the random oracle* H_c *and the signing oracle, respectively, there exists adversaries* $\mathcal{B}$ *and* $\mathcal{B}'$ *against the* $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ *and* $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ *problems, respectively, such that*

$$\mathsf{Adv}^{\mathsf{sig\text{-}uf}}_{\mathcal{A}}(\kappa) \leq \mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}}_{\mathcal{B}}(\kappa) + \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{B}'}(\kappa) + \frac{(Q_{\mathsf{H}}+1)\cdot Q_{\mathsf{Sign}}}{2^{n-1}},$$

where $\mathsf{Time}(\mathcal{B}), \mathsf{Time}(\mathcal{B}') \approx \mathsf{Time}(\mathcal{A})$.

Proof. To prove Theorem C.2, we use a series of hybrid games as defined in Fig. 18, where Hybrid_1 corresponds to the original unforgeability game. The final Hybrid_4 is designed such that a reduction $\mathcal{B}'$ can simulate the hybrid game given a $\mathsf{SelfTargetMSIS}$ instance, and extract a solution of this problem from a successful adversary. Our aim is thus to ensure that $\mathcal{B}'$ does not need to know the secrets $\mathbf{s}$ or $\mathbf{e}$ in order to respond to signature queries. From Hybrid_1 to Hybrid_3 we edit how the oracle signer $\mathsf{OSgn}(\mathsf{msg})$ will respond when queried on a message msg, such that the response $\mathbf{z}$ is chosen at random independently from $\mathbf{s}$. To ensure correctness we now choose the commitment $\mathbf{w}$ to depend on the response $\mathbf{z}$ and the challenge c, and we program the oracle to ensure the signature verifies.

In Hybrid_4, we swap the public key $\mathbf{t}$ to be a uniformly random vector. However, this step cannot be completed by using the standard MLWE problem since the commitment $\mathbf{w}$ and response $\mathbf{z}$ depend on the secret key. To this end, we use the *hint* MLWE problem to simulate these components. Prior works had to add a statistical step (i.e., Rényi divergence, or noise flooding) to rely on the MLWE problem, resorting to sub-optimal parameter selections. Finally, in Hybrid_5, we require that the public key $\mathbf{t}$ can be computed from a $\mathsf{SelfTargetMSIS}$ instance. Below, let $\mathcal{A}$ be an adversary against the unforgeability of the signature scheme.

Hybrid_1**:** This is the unforgeability security game.

Hybrid_2**:** In this hybrid, the challenger replaces non-programmed random oracle outputs in the signing oracle with programmed outputs. First the challenger samples an element c uniformly at random from the challenge space $\mathcal{C}$. Then the hash function is programmed to consistently return this value c on input $(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$ during further interaction with the adversary.

Note that the signing responses in Hybrid_2 are identically distributed to Hybrid_1 unless the bad event occurs in which $\mathsf{OSgn}(\cdot)$ is required to program a value that has already been queried by the adversary. As $\mathbf{w}$ is sampled randomly following the $\mathcal{D}^{\mathsf{bd\text{-}MLWE}}_{q,\ell,k,\sigma_{\mathbf{w}},\nu_{\mathbf{w}}}(\mathbf{A})$ distribution as in Definition 3.7, this happens with probability at most $Q_{\mathsf{H}} \cdot 2^{-H_\infty(\mathcal{D}^{\mathsf{bd\text{-}MLWE}}_{q,\ell,k,\sigma_{\mathbf{w}},\nu_{\mathbf{w}}}(\mathbf{A}))}$ in each signing query, except with probability $Q_{\mathsf{Sign}} \cdot 2^{-n+1}$ using the union bound. Thus, using Lemma 3.8 and our parameter selection, we have the following:

$$\begin{aligned}
&\left|\mathsf{Adv}^{\mathsf{Hybrid}_2}_{\mathcal{A}}(\kappa) - \mathsf{Adv}^{\mathsf{Hybrid}_1}_{\mathcal{A}}(\kappa)\right| \\
\leq &\left(1 - \left(1 - Q_{\mathsf{H}} \cdot 2^{-n+1}\right)^{Q_{\mathsf{Sign}}}\right) \cdot (1 - Q_{\mathsf{Sign}} \cdot 2^{-n+1}) + Q_{\mathsf{Sign}} \cdot 2^{-n+1} \\
\leq &(Q_{\mathsf{H}}+1) \cdot Q_{\mathsf{Sign}} \cdot 2^{-n+1},
\end{aligned}$$

where we have used Bernoulli's inequality, that is, $(1+x)^r \geq 1 + rx$ for every integer $r \geq 0$ and real number $x > -1$.

```
Hybrid_1
1: $\widehat{\mathbf{t}} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e}, \mathbf{t} := \left\lceil \widehat{\mathbf{t}} \right\rfloor_{\nu_{\mathbf{t}}}$
2: $L_{\mathsf{Sign}} := \emptyset$
3: $(\mathsf{msg}, \sigma) \leftarrow \mathcal{A}^{\mathsf{OSgn}(\cdot)}(\mathbf{A}, \mathbf{t})$
4: if $(\mathsf{msg}, \sigma) \in L_{\mathsf{Sign}}$ return 0
5: return $\mathsf{Verify}((\mathbf{A}, \mathbf{t}), \mathsf{msg}, \sigma)$

    OSgn(msg)
    1: $\mathbf{r} \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell}$
    2: $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbf{w}}^{k}$
    3: $\mathbf{w} := \lfloor \mathbf{A} \cdot \mathbf{r} + \mathbf{e}' \rceil_{\nu_{\mathbf{w}}}$
    4: $c := \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$
    5: $\mathbf{z} := c \cdot \mathbf{s} + \mathbf{r}$
    6: $\mathbf{h} := \mathbf{w} - \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathbf{t}}} \cdot c \cdot \mathbf{t} \rceil_{\nu_{\mathbf{w}}}$
    7: $L_{\mathsf{Sign}} := L_{\mathsf{Sign}} \cup \{\mathsf{msg}, (c, \mathbf{z}, \mathbf{h})\}$
    8: return $\sigma := (c, \mathbf{z}, \mathbf{h})$
```

```
Hybrid_2
1: $\widehat{\mathbf{t}} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e}, \mathbf{t} := \left\lceil \widehat{\mathbf{t}} \right\rfloor_{\nu_{\mathbf{t}}}$
2: $L_{\mathsf{Sign}} := \emptyset$
3: $(\mathsf{msg}, \sigma) \leftarrow \mathcal{A}^{\mathsf{OSgn}(\cdot)}(\mathbf{A}, \mathbf{t})$
4: if $(\mathsf{msg}, \sigma) \in L_{\mathsf{Sign}}$ return 0
5: return $\mathsf{Verify}((\mathbf{A}, \mathbf{t}), \mathsf{msg}, \sigma)$

    OSgn(msg)
    1: $\mathbf{r} \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell}$
    2: $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbf{w}}^{k}$
    3: $\mathbf{w} := \lfloor \mathbf{A} \cdot \mathbf{r} + \mathbf{e}' \rceil_{\nu_{\mathbf{w}}}$
    4: $c \leftarrow \mathcal{C}$
    5: $\mathbf{z} := c \cdot \mathbf{s} + \mathbf{r}$
    6: $\mathbf{h} := \mathbf{w} - \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathbf{t}}} \cdot c \cdot \mathbf{t} \rceil_{\nu_{\mathbf{w}}}$
    7: $\mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$
    8: $L_{\mathsf{Sign}} := L_{\mathsf{Sign}} \cup \{\mathsf{msg}, (c, \mathbf{z}, \mathbf{h})\}$
    9: return $\sigma := (c, \mathbf{z}, \mathbf{h})$
```

```
Hybrid_3
1: $\widehat{\mathbf{t}} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e}, \mathbf{t} := \left\lceil \widehat{\mathbf{t}} \right\rfloor_{\nu_{\mathbf{t}}}$
2: $L_{\mathsf{Sign}} := \emptyset$
3: $(\mathsf{msg}, \sigma) \leftarrow \mathcal{A}^{\mathsf{OSgn}(\cdot)}(\mathbf{A}, \mathbf{t})$
4: if $(\mathsf{msg}, \sigma) \in L_{\mathsf{Sign}}$ return 0
5: return $\mathsf{Verify}((\mathbf{A}, \mathbf{t}), \mathsf{msg}, \sigma)$

    OSgn(msg)
    1: $\mathbf{r} \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell}$
    2: $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbf{w}}^{k}$
    3: $c \leftarrow \mathcal{C}$
    4: $\mathbf{z} := c \cdot \mathbf{s} + \mathbf{r}$
    5: $\mathbf{z}' := c \cdot \mathbf{e} + \mathbf{e}'$
    6: $\mathbf{w} := \left\lfloor \mathbf{A} \cdot \mathbf{z} - c \cdot \widehat{\mathbf{t}} + \mathbf{z}' \right\rceil_{\nu_{\mathbf{w}}}$
    7: $\mathbf{h} := \mathbf{w} - \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathbf{t}}} \cdot c \cdot \mathbf{t} \rceil_{\nu_{\mathbf{w}}}$
    8: $\mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$
    9: $L_{\mathsf{Sign}} := L_{\mathsf{Sign}} \cup \{\mathsf{msg}, (c, \mathbf{z}, \mathbf{h})\}$
    10: return $\sigma := (c, \mathbf{z}, \mathbf{h})$
```

```
Hybrid_4
1: $\widehat{\mathbf{t}} \leftarrow \mathcal{R}_q^k$, $\mathbf{t} := \left\lceil \widehat{\mathbf{t}} \right\rfloor_{\nu_{\mathbf{t}}}$
2: $L_{\mathsf{Sign}} := \emptyset$
3: $(\mathsf{msg}, \sigma) \leftarrow \mathcal{A}^{\mathsf{OSgn}(\cdot)}(\mathbf{A}, \mathbf{t})$
4: if $(\mathsf{msg}, \sigma) \in L_{\mathsf{Sign}}$ return 0
5: return $\mathsf{Verify}((\mathbf{A}, \mathbf{t}), \mathsf{msg}, \sigma)$

    OSgn(msg)
    1: $\mathbf{r} \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell}$
    2: $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbf{w}}^{k}$
    3: $c \leftarrow \mathcal{C}$
    4: $\mathbf{z} := c \cdot \mathbf{s} + \mathbf{r}$
    5: $\mathbf{z}' := c \cdot \mathbf{e} + \mathbf{e}'$
    6: $\mathbf{w} := \left\lfloor \mathbf{A} \cdot \mathbf{z} - c \cdot \widehat{\mathbf{t}} + \mathbf{z}' \right\rceil_{\nu_{\mathbf{w}}}$
    7: $\mathbf{h} := \mathbf{w} - \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathbf{t}}} \cdot c \cdot \mathbf{t} \rceil_{\nu_{\mathbf{w}}}$
    8: $\mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$
    9: $L_{\mathsf{Sign}} := L_{\mathsf{Sign}} \cup \{\mathsf{msg}, (c, \mathbf{z}, \mathbf{h})\}$
    10: return $\sigma := (c, \mathbf{z}, \mathbf{h})$
```

Figure 18: The security hybrid games used in the proof of Theorem C.2. Differences from Hybrid_i to Hybrid_{i+1} are highlighted. We assume the game aborts and outputs 0 in case the random oracle H_c is already defined when executing $\mathsf{OSgn}(\cdot)$.

Hybrid_3**:** This game is identical to Hybrid_2 with the exception that the way $\mathbf{w}$ is computed is modified using the public key $\mathbf{t}$ instead of an MLWE sample $\mathbf{A} \cdot \mathbf{r} + \mathbf{e}'$. As the challenger computes $\mathbf{w} = \lfloor \mathbf{Ar} + \mathbf{e}' \rceil_{\nu_{\mathbf{w}}} = \lfloor \mathbf{Az} - c \cdot \mathbf{A} \cdot \mathbf{s} + \mathbf{e}' \rceil_{\nu_{\mathbf{w}}}$ in the previous game, one can verify that $\mathbf{Az} - c \cdot \mathbf{As} + \mathbf{e}' = \mathbf{Az} - c \cdot \widehat{\mathbf{t}} + c \cdot \mathbf{e} + \mathbf{e}'$, which yields the equation in Hybrid_3.

As it is simply a rewriting of $\mathbf{w}$, it remains indistinguishable from Hybrid_2:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}(\kappa) = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}(\kappa).$$

Hybrid_4**:** Finally, this game is the same as Hybrid_3 with the exception that the verification key $\mathsf{vk} = (\mathbf{A}, \lfloor \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \rceil_{\nu_{\mathbf{t}}})$ is replaced by $(\mathbf{A}, \left\lfloor \widehat{\mathbf{t}} \right\rceil_{\nu_{\mathbf{t}}})$ where $\widehat{\mathbf{t}}$ is sampled uniformly at random from $\mathcal{R}_q^k$.

In Lemma C.3, we show that we can construct an adversary $\mathcal{B}$ against the $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ problem such that

$$\left| \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}(\kappa) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}(\kappa) \right| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{Hint\text{-}MLWE}}(\kappa).$$

So as not to interrupt the main proof, we postpone the proof of Lemma C.3.

Now that the public key is a random vector and the challenger no longer requires the signing key for the unforgeability game, we are finally ready to invoke the $\mathsf{SelfTargetMSIS}$ problem. In Lemma C.4, we show that we can construct an adversary $\mathcal{B}'$ against the $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ problem such that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}(\kappa) \leq \mathsf{Adv}_{\mathcal{B}'}^{\mathsf{SelfTargetMSIS}}(\kappa).$$

Collecting all the bounds, we obtain the following bound, which is the desired bound in the theorem statement.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{sig\text{-}uf}}(\kappa) \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{Hint\text{-}MLWE}}(\kappa) + \mathsf{Adv}_{\mathcal{B}'}^{\mathsf{SelfTargetMSIS}}(\kappa) + \frac{(Q_{\mathsf{H}} + 1) \cdot Q_{\mathsf{Sign}}}{2^{n+1}}.$$

To complete the proof, it remains to prove the following Lemmas C.3 and C.4.

Lemma C.3. *There exists an adversary $\mathcal{B}$ against the $\mathsf{Hint\text{-}MLWE}_{q,\ell,k,Q_{\mathsf{Sign}},\sigma_{\mathbf{t}},\sigma_{\mathbf{w}},\mathcal{C}}$ problem such that*

$$\left| \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}(\kappa) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}(\kappa) \right| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{Hint\text{-}MLWE}}(\kappa)$$

with $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A})$.

Proof. Let us provide the description of $\mathcal{B}$. $\mathcal{B}$ is given $\left(\mathbf{A}, \mathbf{b}, (c_i, \mathbf{z}_i, \mathbf{z}'_i)_{i \in [Q_{\mathsf{Sign}}]}\right)$ as the $\mathsf{Hint\text{-}MLWE}$ problem, where $\mathbf{b}$ is either $\mathbf{As} + \mathbf{e}$ or random over $\mathcal{R}_q^k$. It simulates the view of the Hybrid_3 challenger to $\mathcal{A}$ by setting $\widehat{\mathbf{t}} := \mathbf{b}$ and giving the public key $\mathsf{vk} = (\mathbf{A}, \left\lfloor \widehat{\mathbf{t}} \right\rceil_{\nu_{\mathbf{t}}})$ to $\mathcal{A}$. All random oracle queries are simulated identically to the challenger. When $\mathcal{A}$ queries the signing oracle on msg for the i-th ($i \in [Q_{\mathsf{Sign}}]$) time, $\mathcal{B}$ retrieves the i-th unused tuple $(c_i, \mathbf{z}_i, \mathbf{z}'_i)$. It then uses this to compute the commitment $\mathbf{w}$ and hint $\mathbf{h}$, and programs the random oracle H_c. Finally, when $\mathcal{A}$ outputs a valid signature that breaks unforgeability, $\mathcal{B}$ outputs 1.

It is clear that the signing oracle is perfectly simulated using the noise leakage provided by the $\mathsf{Hint\text{-}MLWE}$ problem. Thus, $\mathcal{B}$ perfectly simulates Hybrid_3 when $\mathbf{b}$ is a valid MLWE sample and Hybrid_4 otherwise. This completes the proof. □

Lemma C.4. *There exists an adversary $\mathcal{B}'$ against the $\mathsf{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\mathsf{stmsis}}}$ problem such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}(\kappa) \leq \mathsf{Adv}_{\mathcal{B}'}^{\mathsf{SelfTargetMSIS}}(\kappa)$$

with $\mathsf{Time}(\mathcal{B}') \approx \mathsf{Time}(\mathcal{A})$.

Proof. Let us provide the description of $\mathcal{B}'$. $\mathcal{B}'$ is given $\mathbf{M} \in \mathcal{R}_q^{k\times(\ell+1)}$ as the $\mathsf{SelfTargetMSIS}$ problem. It sets $-\widehat{\mathbf{t}}$ as the first column and $\mathbf{A}$ to be the remaining ℓ columns of $\mathbf{M}$, and sets the public key vk to be $(\mathbf{A}, \lfloor \widehat{\mathbf{t}} \rceil_{\nu_{\mathsf{t}}})$. $\mathcal{B}'$ simulates the Hybrid_4 challenger except for one difference: whenever the random oracle H_c is invoked on an input input, $\mathcal{B}'$ tries to parse it as $(\mathsf{vk}, \mathsf{msg}, \mathbf{y})$ for some $\mathbf{y} \in \mathcal{R}^k_{q_{\mathsf{w}}}$. If it does not, it samples a random output and simulates H_c as the Hybrid_4 challenger. Otherwise, it queries $(2^{\nu_{\mathsf{w}}} \cdot \bar{\mathbf{y}} \mod q, \mathsf{msg})$ to the oracle $\mathsf{G} : \mathcal{R}^k_q \times \{0,1\}^{2\kappa} \to \mathcal{C}$ provided by the $\mathsf{SelfTargetMSIS}$ problem and uses that as the output instead. Here, recall that $\bar{\mathbf{y}}$ is the unique lift of $\mathbf{y} \in \mathbb{Z}^k_{q_{\mathsf{w}}}$ to $\{0, 1, \cdots, q_{\mathsf{w}} - 1\}^k$. Importantly, by definition of q_{w}, we have $2^{\nu_{\mathsf{w}}} \cdot \bar{\mathbf{y}} \in \{0, 1, \cdots, q-1\}^k$. This means that the map $\mathbf{y} \mapsto (2^{\nu_{\mathsf{w}}} \cdot \bar{\mathbf{y}} \mod q)$ is injective, meaning that changing how we answer the random oracle does not alter the view of $\mathcal{A}$. Finally, since the Hybrid_4 challenger no longer uses the signing key, $\mathcal{B}'$ can perfectly simulate the signing oracle where it uses G instead of H_c when programming the challenge it samples. In summary, $\mathcal{B}'$ perfectly simulates the view of Hybrid_4 to $\mathcal{A}$.

At the end of the game, the adversary $\mathcal{A}$ outputs a forgery $(c^*, \mathbf{z}^*, \mathbf{h}^*)$ for a message msg^*. Since this is a valid forgery, we have $\left\|(\mathbf{z}^*, 2^{\nu_{\mathsf{w}}} \cdot \overline{\mathbf{h}^*} \mod q)\right\|_2 \le B_2$ and $\lfloor \mathbf{A} \cdot \mathbf{z}^* - 2^{\nu_{\mathsf{t}}} \cdot c^* \cdot \bar{\mathbf{t}} \rceil_{\nu_{\mathsf{w}}} + \mathbf{h}^* \in \mathcal{R}^k_{q_{\mathsf{w}}}$, where note that from this point on we will be explicit with the lifting notation and where the norm is taken. Due to how $\mathcal{B}'$ simulates the random oracle, we have

$$c^* = \mathsf{G}\left(2^{\nu_{\mathsf{w}}} \cdot \left(\overline{\lfloor \mathbf{A} \cdot \mathbf{z}^* - 2^{\nu_{\mathsf{t}}} \cdot c^* \cdot \bar{\mathbf{t}} \rceil_{\nu_{\mathsf{w}}} + \mathbf{h}^*}\right) \mod q, \mathsf{msg}^*\right). \tag{19}$$

It remains to show how $\mathcal{B}'$ turns this into a $\mathsf{SelfTargetMSIS}$ solution. First, using Lemma 3.1 and Lemma 3.2, Eq. (5), we have

$$2^{\nu_{\mathsf{t}}} \cdot \bar{\mathbf{t}} = 2^{\nu_{\mathsf{t}}} \cdot \overline{\lfloor \widehat{\mathbf{t}} \rceil_{\nu_{\mathsf{t}}}} = \widehat{\mathbf{t}} + \boldsymbol{\delta}_{\mathbf{t}}$$

over modulo q for some $\boldsymbol{\delta}_{\mathbf{t}} \in \mathcal{R}^k_q$ with $\|\boldsymbol{\delta}_{\mathbf{t}}\|_\infty \le 2^{\nu_{\mathsf{t}}} - 1$. We also have

$$\begin{aligned}\overline{\lfloor \mathbf{A} \cdot \mathbf{z}^* - 2^{\nu_{\mathsf{t}}} \cdot c^* \cdot \bar{\mathbf{t}} \rceil_{\nu_{\mathsf{w}}} + \mathbf{h}^*} &= \overline{\lfloor \mathbf{A} \cdot \mathbf{z}^* - 2^{\nu_{\mathsf{t}}} \cdot c^* \cdot \bar{\mathbf{t}} \rceil_{\nu_{\mathsf{w}}}} + \overline{\mathbf{h}^*} - \boldsymbol{\delta}_1 \\ &= \overline{\lfloor \mathbf{A} \cdot \mathbf{z}^* - c^* \cdot (\widehat{\mathbf{t}} + \boldsymbol{\delta}_{\mathbf{t}}) \rceil_{\nu_{\mathsf{w}}}} + \overline{\mathbf{h}^*} - q_{\mathsf{w}} \cdot \boldsymbol{\delta}_1\end{aligned}$$

over the integers for some $\boldsymbol{\delta}_1$ satisfying $\|\boldsymbol{\delta}_1\|_\infty \le 1$.

Plugging this into the first input of G in Eq. (19), we have

$$\begin{aligned}2^{\nu_{\mathsf{w}}} \cdot \left(\overline{\lfloor \mathbf{A} \cdot \mathbf{z}^* - 2^{\nu_{\mathsf{t}}} \cdot c^* \cdot \bar{\mathbf{t}} \rceil_{\nu_{\mathsf{w}}} + \mathbf{h}^*}\right) \mod q &= 2^{\nu_{\mathsf{w}}} \cdot \left(\overline{\lfloor \mathbf{A} \cdot \mathbf{z}^* - c^* \cdot (\widehat{\mathbf{t}} + \boldsymbol{\delta}_{\mathbf{t}}) \rceil_{\nu_{\mathsf{w}}}} + \overline{\mathbf{h}^*} - q_{\mathsf{w}} \cdot \boldsymbol{\delta}_1\right) \mod q \\ &= \mathbf{A} \cdot \mathbf{z}^* - c^* \cdot (\widehat{\mathbf{t}} + \boldsymbol{\delta}_{\mathbf{t}}) + 2^{\nu_{\mathsf{w}}} \cdot (\overline{\mathbf{h}^*} - q_{\mathsf{w}} \cdot \boldsymbol{\delta}_1) + \boldsymbol{\delta}_2 \mod q \\ &= \mathbf{M} \begin{bmatrix} c^* \\ \mathbf{z}^* \end{bmatrix} - c^* \cdot \boldsymbol{\delta}_{\mathbf{t}} + 2^{\nu_{\mathsf{w}}} \cdot (\overline{\mathbf{h}^*} - q_{\mathsf{w}} \cdot \boldsymbol{\delta}_1) + \boldsymbol{\delta}_2 \mod q, \quad (20)\end{aligned}$$

where the second equality holds for some $\boldsymbol{\delta}_2$ with $\|\boldsymbol{\delta}_2\|_\infty \le 2^{\nu_{\mathsf{w}}} - 1$ using Lemma 3.1 and Lemma 3.2, Eq. (5) and the last equality follows by plugging in $\mathbf{M} = [-\widehat{\mathbf{t}} \mid \mathbf{A}]$. Finally, Eq. (20) is equivalent to $[\mathbf{M} \mid \mathbf{I}] \cdot \mathbf{z}_{\mathsf{sol}} \mod q$, where

$$\mathbf{z}_{\mathsf{sol}} = \begin{bmatrix} c^* \\ \mathbf{z}^* \\ -c^* \cdot \boldsymbol{\delta}_{\mathbf{t}} + 2^{\nu_{\mathsf{w}}} \cdot (\overline{\mathbf{h}^*} - q_{\mathsf{w}} \cdot \boldsymbol{\delta}_1) + \boldsymbol{\delta}_2 \end{bmatrix} \in \mathcal{R}_q^{k+\ell+1}$$

Plugging this back into Eq. (19), we have

$$c^* = \mathsf{G}\left([\mathbf{M} \mid \mathbf{I}] \cdot \mathbf{z}_{\mathsf{sol}} \mod q, \mathsf{msg}^*\right).$$

To conclude, $\mathcal{B}'$ efficiently computes $\mathbf{z}_{\mathsf{sol}}$ from the forgery and outputs $(\mathbf{z}_{\mathsf{sol}}, \mathsf{msg}^*)$ as the $\mathsf{SelfTargetMSIS}$ solution, where note that we can efficiently compute $(\boldsymbol{\delta}_{\mathbf{t}}, \boldsymbol{\delta}_1, \boldsymbol{\delta}_2)$. Here, since the first entry of $\mathbf{z}_{\mathsf{sol}}$ is c^*, it is in a desired format. Moreover,

$$\begin{aligned}
\|\mathbf{z}_{\mathsf{sol}}\|_2 &\le \|c^*\|_2 + \left\|(\mathbf{z}^*, 2^{\nu_{\mathsf{w}}} \cdot \overline{\mathbf{h}^*} \mod q)\right\|_2 + \|c^* \cdot \boldsymbol{\delta}_{\mathbf{t}}\|_2 + \|2^{\nu_{\mathsf{w}}} \cdot q_{\mathbf{w}} \cdot \boldsymbol{\delta}_1 \mod q\|_2 + \|\boldsymbol{\delta}_2\|_2 \\
&\le (\sqrt{\omega}) + (B_2) + (\omega \cdot 2^{\nu_{\mathsf{t}}}\sqrt{nk}) + (2^{\nu_{\mathsf{w}}} \cdot \sqrt{nk}) + (2^{\nu_{\mathsf{w}}} \cdot \sqrt{nk}) \\
&= B_{\mathsf{stmsis}}
\end{aligned}$$

Hence, $\mathbf{z}_{\mathsf{sol}}$ is indeed a valid $\mathsf{SelfTargetMSIS}$ solution, This completes the proof of Lemma C.4. □

This completes the proof of Theorem C.2. □

D Omitted Consistency Check Algorithms in TRaccoon

Here we include the deferred consistency checks from the signing protocol in $\mathsf{TRaccoon}$ (see Fig. 5). Note that by $\mathsf{ConsistCheck}_1$, we always have $j \in \mathsf{act}$ and $\mathsf{act} \subseteq [N]$ for any user index j in state and act in $\mathsf{state.session[sid]}$, if a $\mathsf{session}$ for sid exists. In particular, this check will be omitted from $\mathsf{ConsistCheck}_2$ and $\mathsf{ConsistCheck}_3$.

E Using Rounded Instead of Discrete Gaussians

In this section, we explain the effect of using rounded Gaussians instead of discrete Gaussians. Concretely, we bound the Rényi divergence between the rounded and discrete Gaussians. At a high level, we can instead start our unforgeability proof using rounded Gaussians (i.e., our implementation parameters) and then swap to a hybrid using discrete Gaussians, at which point, we can rely on the proofs we already have in Theorem 7.2.

We note that in our implementation we use a trick from [Jan06] in which the continuous Gaussian distribution is compensated for the 1/12 additional variance caused by integer rounding by setting $\sigma' = \sqrt{\sigma^2 - 1/12}$. However, for the sake of simplicity, we prove a (slightly worse) bound where this trick is not taken into account, i.e., we compute the Rényi divergence between the rounded and discrete Gaussians for the same standard deviation.

E.1 Background

We recall some useful definitions for this section.

Gaussians. We first prepare the definition of continuous and rounded Gaussians.

Definition E.1. *Let ρ_σ be the Gaussian function:*

$$\forall x \in \mathbb{R}, \quad \rho_\sigma(x) = \frac{1}{\sigma\sqrt{2\,\pi}} \exp\left(-\frac{x^2}{2\,\sigma^2}\right).$$

Definition E.2. *We note RG_σ the rounded Gaussian distribution:*

$$\begin{aligned}
\forall k \in \mathbb{Z}, \quad RG_\sigma(k) &= \int_{k-1/2}^{k+1/2} \rho_\sigma(x)dx \\
&= \frac{1}{\sigma\sqrt{2\,\pi}} \int_{k-1/2}^{k+1/2} \exp\left(-\frac{x^2}{2\,\sigma^2}\right) dx.
\end{aligned}$$

We also define a B-bounded variant of the discrete and rounded Gaussians.

Alg. 40: $\mathsf{ConsistCheck}_1(\mathsf{state}, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$

```
1: assert{ sid contains (act, msg) }        ▷ Check that session id sid is in correct form
2: assert{ $\mathsf{state.session}[\mathsf{sid}] = \bot$ }        ▷ Check that user never signed using sid
3: Fetch user index $j$ from state
4: assert{ $j \in \mathsf{act} \ \wedge\ \mathsf{act} \subseteq [N]$ }
```

Alg. 41: $\mathsf{ConsistCheck}_2(\mathsf{state}, \mathsf{sid}, \mathsf{contrib}_1)$

```
1: assert{ $\mathsf{state.session}[\mathsf{sid}] \neq \bot$ }        ▷ The user already created a session with sid
2: Let $\mathsf{session} = \mathsf{state.session}[\mathsf{sid}]$
3: Fetch user index $j$ from state
4: Fetch $(\mathsf{sid}', \mathsf{act}, \mathsf{cmt}_j, \mathbf{m}_j)$ from session
5: assert{ The set of keys (i.e. indices) in $\mathsf{contrib}_1$ is exactly act }
        ▷ Check $\mathsf{contrib}_1$ includes $|\mathsf{act}|$ number of first round messages
6: for $i \in \mathsf{act}$ do
7:     assert{ $\mathsf{contrib}_1[i]$ is of the form $\mathsf{contrib}_1[i] = (\mathsf{cmt}_i, \mathbf{m}_i)$ }
        ▷ Check $\mathsf{contrib}_1$ is defined over the indices in act and of a valid form
8: assert{ session is of the form $\mathsf{session} = \{\mathsf{sid}', \mathsf{act}, \mathsf{msg}, 1, \{\mathbf{r}_j, \mathbf{w}_j, \mathsf{contrib}_1[j]\}, \emptyset\}$ }    ▷ Check sid and $\mathsf{contrib}_1[j]$ is consistent with internal state
```

Alg. 42: $\mathsf{ConsistCheck}_3(\mathsf{state}, \mathsf{sid}, \mathsf{contrib}_2)$

```
1: assert{ $\mathsf{state.session}[\mathsf{sid}] \neq \bot$ }        ▷ The user already opened a session with sid
2: Let $\mathsf{session} = \mathsf{state.session}[\mathsf{sid}]$
3: Fetch user index $j$ from state
4: Fetch $(\mathsf{sid}', \mathsf{act})$ from session
5: assert{ The set of keys in $\mathsf{contrib}_2$ is exactly act }
        ▷ Check $\mathsf{contrib}_2$ includes $|\mathsf{act}|$ number of second round messages
6: for $i \in \mathsf{act}$ do
7:     assert{ $\mathsf{contrib}_2[i]$ is of the form $\mathsf{contrib}_2[i] = (\mathbf{w}_i, \sigma_i)$ }
        ▷ Check $\mathsf{contrib}_2$ is defined over the indices in act and of a valid form
8: assert{ session is of the form $\mathsf{session} = \{\mathsf{sid}', \mathsf{act}, \mathsf{msg}, 2, \{\mathbf{r}_j, \mathbf{w}_j, \mathsf{contrib}_1[j]\}, \mathsf{contrib}_1\}$ }    ▷ Check sid is consistent with internal state
```

Figure 19: Consistency checks for Threshold Raccoon

Definition E.3. *We note $D_\sigma[B]$ the tail cut discrete Gaussian distribution:*

$$\begin{aligned}\forall k \in \mathbb{Z}, \quad &\textit{if } |k| < B, D_\sigma[B](k) = \frac{\rho_\sigma(k)}{\sum_{z\in\mathbb{Z}, |z|<B} \rho_\sigma(z)}. \\ &\textit{if } |k| > B, D_\sigma[B](k) = 0\end{aligned}$$

Definition E.4. *For any $B > 1$, let $M(B) = \int_{-\lfloor B\rfloor-1/2}^{\lfloor B\rfloor+1/2} \rho_\sigma(x)dx$. We note $RG_\sigma[B]$ the tail-cut rounded Gaussian distribution:*

$$\begin{aligned}\forall k \in \mathbb{Z}, \quad &\textit{if } |k| < B, RG_\sigma[B](k) = \int_{k-1/2}^{k+1/2} \rho_\sigma(x)dx/M(B) \\ &\textit{if } |k| > B, RG_\sigma[B](k) = 0\end{aligned}$$

Lemma E.5. *[[Lyu12] Lemma 4.4] Let $x \leftarrow D_\sigma$ (as well as $x \leftarrow \rho_\sigma$). Then, we have*

$$\Pr\left[|x| > a\cdot\sigma\right] \le 2e^{-\frac{a^2}{2}}$$

Fact 1. We have the following for Gaussian functions:

$$\frac{d}{dx}\rho_\sigma(x) = -\frac{x}{\sigma^2}\rho_\sigma(x) \tag{21}$$

$$\frac{d^2}{dx^2}\rho_\sigma(x) = \left(\frac{x^2}{\sigma^2} - 1\right)\frac{1}{\sigma^2}\rho_\sigma(x), \tag{22}$$

$$\frac{d^4}{dx^4}\rho_\sigma(x) = \left(3 - 6\frac{x^2}{\sigma^2} + \frac{x^4}{\sigma^4}\right)\frac{1}{\sigma^4}\rho_\sigma(x). \tag{23}$$

Rényi Divergence Properties. The Rényi divergence [Rén61] is a tool from information theory which has recently found many applications in lattice-based cryptography, see for instance [BLL+15, Pre17]. We use the "exponential form" of the Rényi divergence, as it is common in lattice-based cryptography.

Definition E.6 (Rényi divergence)**.** *Let $\mathcal{P}, \mathcal{Q}$ be two discrete distributions such that* $\operatorname{Supp}(\mathcal{P}) \subseteq \operatorname{Supp}(\mathcal{Q})$*, and $\alpha \in (1; +\infty)$. The Rényi divergence of order α is:*

$$R_\alpha(\mathcal{P}; \mathcal{Q}) = \left(\sum_{x\in X} \frac{\mathcal{P}(x)^\alpha}{\mathcal{Q}(x)^{\alpha-1}}\right)^{\frac{1}{\alpha-1}}$$

Following Csiszár's f-divergence framework [Csi63], $(R_\alpha^{\alpha-1} - 1)$ is an f-divergence for $f : x \mapsto x^\alpha - 1$. Lemma E.7 presents some properties of the Rényi divergence; proofs can be found in van Erven and Harremoës [vEH14] or Bai et al. [BLR+18].

Lemma E.7. *For distributions $\mathcal{P}, \mathcal{Q}$ and finite families of distributions $(\mathcal{P}_i)_{i\in[n]}$, $(\mathcal{Q}_i)_{i\in[n]}$, the Rényi divergence satisfies the following properties:*

1. ***Data processing inequality.*** *For a (randomized) function f,*

$$R_\alpha(f(\mathcal{P}); f(\mathcal{Q})) \le R_\alpha(\mathcal{P}; \mathcal{Q}).$$

2. ***Probability preservation.*** *For any event $E \subseteq \operatorname{Supp}(\mathcal{Q})$:*

$$\mathcal{P}(E) \le \mathcal{Q}(E)^{\frac{\alpha-1}{\alpha}} \cdot R_\alpha(\mathcal{P}; \mathcal{Q}),$$

3. ***Multiplicativity.*** $R_\alpha(\prod_i \mathcal{P}_i; \prod_i \mathcal{Q}_i) = \prod_i R_\alpha(\mathcal{P}_i; \mathcal{Q}_i)$.

E.2 Bounding the Divergence Between Rounded and Discrete Gaussians

The following lemma establishes that the B-bounded variant of the rounded and discrete Gaussians are close in respect to the Rényi divergence.

Lemma E.8. *Let $\beta > \ln(2) + 3$, $\sigma \geq \sqrt{\kappa + \beta}$, $B \geq \sqrt{\kappa+\beta} \cdot \sigma$. Then:*

$$R_\alpha(RG_\sigma[B]; D_\sigma[B]) \leq \exp\left(\frac{\alpha}{2}\left(\frac{\kappa+\beta}{6\sigma^2} + 2^{-\kappa}\right)^2\right) \tag{24}$$

Proof. We first prove that for any $a > 0$, $k < a \cdot \sigma - 1/2$:

$$\mathrm{RG}_\sigma[B](k) \cdot M(B) \leq \rho_\sigma(k)\left(1 - \frac{1}{12\,\sigma^2} + \frac{k^2}{12\,\sigma^4} + \frac{3 + a^4}{30\,\sigma^4}\right)$$

Under Taylor's remainder theorem, for any function f that is C_∞ over $\mathbb{R}$ and any $b, x \in \mathbb{R}$, there exists c between a and x such that:

$$f(x) = \sum_{j \leq n} \frac{f^{(j)}(b)}{j!}(x-b)^j + \frac{f^{(n+1)}(c)}{(n+1)!}(x-b)^{n+1} \tag{25}$$

For such a c, let us note $R_4(k) = \int_{k-1/2}^{k+1/2} \frac{f^{(4)}(c)}{4!}(x-k)^4 dx$. We have:

$$\begin{aligned}
0 \leq R_4(k) &\leq \max_{k-1/2 \leq x \leq k+1/2} \frac{f^{(4)}(x)}{4!} \int_{k-1/2}^{k+1/2} (x-k)^n \\
&= \max_{k-1/2 \leq x \leq k+1/2} \frac{f^{(4)}(x)}{4! \cdot 80} \\
&\leq \frac{1}{1920} \frac{3+a^4}{\sigma^4}\; \rho_\sigma(k - 1/2) \\
&\leq \frac{e}{1920} \frac{3+a^4}{\sigma^4}\; \rho_\sigma(k)
\end{aligned} \tag{26}$$

Setting $f = \rho_\sigma$ and $n = 3$ in Eq. (25) gives:

$$\begin{aligned}
\int_{k-1/2}^{k+1/2} \rho_\sigma(x)dx &= \int_{k-1/2}^{k+1/2} \left(f(k) + f'(k)(x-k) + \frac{f''(k)}{2}(x-k)^2 + \frac{f^{(3)}(k)}{6}(x-k)^3 + \frac{f^{(4)}(c)}{24}(x-k)^4\right) dx \\
&\leq \int_{k-1/2}^{k+1/2} \left(f(k) + \frac{f''(k)}{2}(x-k)^2\right) dx + R_4(k) && (27) \\
&= \rho_\sigma(k) \int_{k-1/2}^{k+1/2} \left(1 + \left(\frac{k^2}{\sigma^2} - 1\right) \frac{1}{\sigma^2}(x-k)^2\right) dx + R_4(k) && (28) \\
&\leq \rho_\sigma(k)\left(1 - \frac{1}{12\,\sigma^2} + \frac{k^2}{12\,\sigma^4} + \frac{e}{1920}\frac{3+a^4}{\sigma^4}\right) && (29)
\end{aligned}$$

Since we integrate in the interval $(k-1/2, k+1/2)$, the terms $\frac{f^{(n)}(k)}{2}(x-k)^n$ for odd n disappear upon integration in Eq. (27). Then Eq. (28) follows from Eq. (22). Finally, Eq. (29) follows from Eq. (26). Since $D_\sigma[B](k) = D_\sigma(k) \cdot M'(B)$, with $M'(B) = \sum_{z \in \mathbb{Z}, |z| < B} \rho_\sigma(z)$, we have:

$$\left|\frac{\mathrm{RG}_\sigma[B](k)}{D_\sigma[B](k)} - 1\right| \leq \left|\frac{M'(B)}{M(B)} \cdot \left(1 - \frac{1}{12\,\sigma^2} + \frac{k^2}{12\,\sigma^4} + \frac{e}{1920}\frac{3+a^4}{\sigma^4}\right) - 1\right| \tag{30}$$

Since $k < a \cdot \sigma - 1/2$, for $a = \sqrt{\kappa + \beta}$ and $\sigma \geq a$, we have :

$$
\begin{aligned}
-\frac{1}{12\,\sigma^2} + \frac{k^2}{12\,\sigma^4} + \frac{e}{1920}\frac{3+a^4}{\sigma^4} &= \frac{1}{12\sigma^2}\left(a^2 + \frac{e \cdot a^4}{160\sigma^2} + \frac{e \cdot 3}{160\sigma^2} - 1\right) \\
&\leq \frac{a^2}{12\sigma^2}(1 + \frac{e}{160}) \\
&\leq \frac{\kappa+\beta}{6\sigma^2}
\end{aligned}
$$

We now compute $M'(B)/M(B)$, first observe that $M'(B) = D_\sigma(\mathbb{Z}) - \sum_{|z| \geq B} D_\sigma(z)$ and $M(B) = 1 - 2 \cdot \int_{\lfloor B \rfloor - 1/2}^{\infty} D_\sigma(x)dx$. Hence for $B \geq \sqrt{\kappa+\beta}\sigma \geq \sqrt{\kappa + \ln(2) + 3}\sigma$, and ϵ the smoothing parameter of $\mathbb{Z}$:

$$
\begin{aligned}
\left|\frac{M'(B)}{M(B)} - 1\right| &= \left|\frac{M'(B) - M(B)}{M(B)}\right| \\
&\leq \frac{1 + \epsilon + 2^{-\kappa-3} - 1 + 2^{-\kappa-3}}{1/2} \\
&\leq 2^{-\kappa}
\end{aligned}
$$

Finally we simplify Eq. (30) by considering that $\epsilon \ll 2^{-\kappa}$, and $2^{-\kappa} \cdot \left(\frac{1}{12\,\sigma^2} + \frac{k^2}{12\,\sigma^4} + \frac{e}{1920}\frac{3+a^4}{\sigma^4}\right) \ll \min(| - \frac{1}{12\,\sigma^2} + \frac{k^2}{12\,\sigma^4} + \frac{e}{1920}\frac{3+a^4}{\sigma^4}|, 2^{-\kappa})$ We get,

$$
\left|\frac{\mathrm{RG}_\sigma[B](k)}{D_\sigma[B](k)} - 1\right| \leq \frac{\kappa+\beta}{6\sigma^2} + 2^{-\kappa}
$$

We can conclude by combining Eq. (29) with [Pre17, Lemma 3]. □

E.3 Theorem 7.2 with Rounded Gaussians

We now sketch a reduction where our three-round threshold signature is using the rounded Gaussian instead of the discrete Gaussian. As explained in the beginning of this section, we merely include a hybrid where we modify the rounded Gaussian to a discrete Gaussian using an argument with Rényi divergence.

Proof sketch of Theorem 7.2 using rounded Gaussians. We only highlight the main modifications required.

Hybrid_0. In this first Hybrid we consider the scheme of Fig. 5, where all Gaussians are sampled according to rounded Gaussian distributions.

Hybrid_1. In this hybrid, the Gaussians are sampled according to the tail-cut rounded distributions. Setting $\beta = \ln(2Q_G)$ where $Q_G = n(\ell + k)(Q_S + 1)$ is the total number of queries made to the one-dimensional Gaussian sampler, we get:

$$
|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1}(\kappa) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_0}(\kappa)| \leq Q_G \cdot \Delta(\mathrm{RG}_\sigma[B], \mathrm{RG}_\sigma) \leq 2^{-\kappa}.
$$

Hybrid_2. In this hybrid, we switch all the samples to the tail-cut discrete Gaussians, using Lemma E.8 we get:

$$
\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}(\kappa) \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1}(\kappa) \cdot \exp\left(\frac{Q_G \alpha}{2}\left(\frac{\kappa+\beta}{6\sigma^2} + 2^{-\kappa}\right)^2\right).
$$

By setting α accordingly, the multiplicative factor is bounded by a polynomial factor.

Hybrid_3. In this hybrid we replace all the tail-cut samples with regular discrete Gaussians.

$$
|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}(\kappa) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}(\kappa)| \leq Q_G \cdot \Delta(D_\sigma[B], D_\sigma) \leq 2^{-\kappa}.
$$

At this point, we arrive at the protocol described in our TRaccoon described in Section 5, relying on the discrete Gaussian. □