Advanced Post-Quantum Signatures

Alexandros Themelis[1], Geoffroy Couteau[1], and Sihang Pu[1]

Université Paris Cité, CNRS, IRIF
`{themelis,couteau,sihang.pu}@irif.fr`

General context

Despite over a decade of development, current proposals for post-quantum cryptographic primitives are typically less efficient or versatile than their non-quantum resistant counterparts. Especially in the course of the NIST post-quantum standardization effort [CCJ+16], the construction of standard cryptographic primitives, such as encryption or signatures, from post-quantum problems has already been well-studied. However, KEMs[1] and digital signatures represent only a small fraction of the rich toolbox of cryptographic primitives and protocols that cryptographers have developed over the years, and securing online communications is only one of the many goals of cryptography. Yet, more advanced cryptographic primitives still heavily rely on classical assumptions, such as the discrete logarithm problem.

Research problem

In a threshold signature scheme [Des90, DF90], or more precisely, in a T-out-of-N threshold signature scheme, the signing key is split and distributed among N different parties. The goal is for any subset of parties with size T to be able to jointly create a valid signature on any message μ. A secure T-out-of-N threshold signature scheme should prevent an adversary from producing a valid signature, even if they have corrupted up to $T-1$ users of the original set.

The need for efficient post-quantum threshold signature schemes is evident. Threshold schemes, beyond their numerous applications in the blockchain ecosystem, enable the decentralization of trust regarding the creation, storage, and use of the secret key. Throughout the protocol—before, during, and after its execution— the secret key is not combined in a single place. This is one of the many reasons threshold signatures have attracted interest from significant entities, including the US agency NIST. In January 2023, NIST released a call for threshold schemes, not necessarily post-quantum, highlighting their importance [BP23].

The aim of the internship was to design a novel threshold signature scheme, with the goal of improving the state-of-the-art schemes based on lattice assumptions. Lattice assumptions are widely regarded as the most promising choice in the transition to post-quantum cryptography, as evidenced by the PQC Standardization Process, where three out of the four candidates for standardization are based on lattice assumptions. Thus, it remains logical to continue investigating and enhancing advanced primitives that rely on these same computational assumptions.

Your contribution

In joint work with Sihang Pu, we propose a novel lattice-based threshold signature scheme designed to improve the state-of-the-art lattice-based threshold signature schemes. More concretely, our protocol requires only two rounds of interaction, compared to the three rounds needed by the current state-of-the-art Threshold Raccoon [DPKM+24], and does not rely on any new, non-standard lattice assumptions as proposed in [EKT24]. Additionally, the first round of our protocol is not message-dependant and therefore can be precomputed "offline".

Arguments supporting its validity

In our work, we successfully integrate improvements in both efficiency and security from previous works into a single signature scheme. In a bird's-eye view, our protocol builds on [EKT24] by reducing

[1] Key Encapsulation Mechanism: A public-key cryptosystem that allows a sender to generate a short secret key and transmit it to a receiver securely, in spite of eavesdropping and intercepting adversaries.

the interaction to two rounds, while using some technical tools from DualMS [Che23] to avoid relying on a non-standard computational assumption.

Given that lattice assumptions are the most promising for replacing classical computational assumptions such as integer factoring and the discrete logarithm problem, it seems that our work is appears to be headed in the correct direction. By combining existing works, we aim to produce a state-of-the-art result, addressing key research questions and needs in both academia and industry.

To prove the security of a protocol, cryptographic methods typically involve presenting a mathematical proof in the Random Oracle Model [BR93]. In this report, we describe our proposed protocol and provide the corresponding proof to validate its security.

Summary and future work

During the internship, we tried to extend the work of [DPKM+24,EKT24] by proposing a novel lattice-based threshold signature scheme that addresses limitations observed in the aforementioned works. This contribution leads the way for improving such signature schemes while staying aligned with the established state of the art. This is crucial since prior research, e.g. [BKP13,CS19,BGG+18,ASY22], has demonstrated that general methods, such as general multi-party computation (MPC) techniques and fully homomorphic encryption (FHE) cannot yet produce efficient schemes.

Due to time-constraints, we have not yet completed a finished article. However, the final goal of the internship is to publish our work in a peer-reviewed conference. Apart from filling in some technical details in our protocol and its security proof, another missing aspect is a concrete parameter selection to determine byte sizes and accurately measure the protocol's computational complexity.

We note that towards the end of internship, a preprint of a new lattice-based threshold scheme called Ringtail [BKL+24] was published. We observe that it achieves the same main goals, namely two-round and standard assumptions. However, a further comparison between the two schemes is necessary. Their approach appears similar to ours, as it relies on [DPKM+24, EKT24] using some technical workarounds from MuSig−L [BTT22]. Moreover, because lattice-based cryptosystems are highly parameter-dependent, implementing our scheme would make more efficient benchmarking and comparison between different schemes.

Lastly, an obvious but interesting follow-up question is where the next breakthrough of lattice-based threshold signatures might emerge. Achieving a single round of interaction (without the offline computation) while maintaining efficiency seems distant and MLWE/MSIS (the computational assumptions that we prove security against) are the only standard lattice assumptions. However, there should be room for improvements in parameters. Ultimately, reducing key and signature sizes as well as per-party computational latency, is essential to making cryptographic schemes practical and efficient for real-world use. Integrating different lattice-specific technical techniques into the protocols and their security proofs should lead to improved parameters.

Acknowledgments

I would like to express my gratitude to my supervisor, Geoffroy Couteau, for all the helpful advice (both within and outside of cryptography), his approach to science, and his sincere interest in his students. I am also deeply thankful to Sihang Pu for his constant help and availability throughout this internship, which led to a highly enjoyable and productive collaboration.

I wish to thank my office mate, Owen, for the necessary procrastination we shared in office 3010; Carmen for consistently filling the office blackboard with interesting graphs; and all the (non-)cryptographers in the "standard" afternoon beer, always engaging to interesting (non-)cryptographic discussions.

I owe much to Katerina Mandilara; without her guidance, I would probably not have had the opportunity to experience this memorable time in Paris. I also want to thank Alessandro with his help from day one at MPRI, and for being a true "crypto-bro", as he once said.

I am grateful to Spyros for being here over the years and look forward to many more. I want to thank my family for their unconditional love and for supporting my decisions throughout the years.

Lastly, I must thank Mania for her constant support and genuine interest over the past year. This year would be distinguishable with non-negligible probability without you.

1 Preliminaries

1.1 Notation

Define $\lambda \in \mathbb{N}$ be the security parameter. For $m \in \mathbb{N}$, we define $[m] := \{1, 2, ..., m\}$. Let $n, q \in \mathbb{N}$ 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 $\mathcal{R}/q\mathcal{R}$. A vector will be written with bold small letters and matrices with bold capital letters. We view vectors in column form. For any vector $\mathbf{v}$, $\mathbf{v}_i$ denotes its i-th element. Let S be any set, then $|S|$ represents the cardinality of S. For a vector $\mathbf{v} \in \mathbb{R}^m$, we write $\|\mathbf{v}\|_p$ for the ℓ_p-norm. Assign the value c to the x variable is denoted as $x \leftarrow c$. Sampling x from the distribution $\mathcal{D}$ is denoted as $x \leftarrow\!\!\$\ \mathcal{D}$, while $x \leftarrow\!\!\$\ \mathcal{R}$ denotes that we sample with the uniform distribution over $\mathcal{R}$.

1.2 Statistical Indistinguishability

Definition 1 (Statistical Distance). *Let X and Y be random variables distributed according to $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively and let $\mathcal{V} = X([\mathcal{D}_1]) \cup Y([\mathcal{D}_2])$. We define the statistical distance Δ as:*

$$\Delta[X, Y] = \frac{1}{2} \sum_{u \in \mathcal{V}} \left| \Pr\left[X = u \mid X \leftarrow\!\!\$\ \mathcal{D}_1 \right] - \Pr\left[Y = u \mid Y \leftarrow\!\!\$\ \mathcal{D}_2 \right] \right|$$

We define the security parameter 1^λ with $\lambda \in \mathbb{N}$.

Definition 2 (Negligible Function). *A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is negligible if for all $c \in \mathbb{R}$, there exists $n_0 \in \mathbb{N}$ such that*

$$\mathsf{negl}(n) \le 1/n^c$$

for all $n \ge n_0$.

Definition 3 (Probability Ensemble). *A* (probability) ensemble *is a collection of distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$.*

Definition 4 (Statistical Indistinguishability). *Let X and Y be random variables over ensembles $\mathcal{D}$ and $\mathcal{D}'$. We say $\mathcal{D}$ and $\mathcal{D}'$ are statistically indistinguishable if $\Delta[X, Y]$ is a negligible function in n.*

1.3 Gaussians

Definition 5. *(Discrete Gaussian Distribution over $\mathcal{R}^m$) For $\mathbf{x} \in \mathcal{R}^m$, the Gaussian function of parameter $\mathbf{v} \in \mathcal{R}^m$ and $s \in \mathbb{R}$ is defined as $\rho_{\mathbf{v},s}(\mathbf{x}) = \exp\left(-\pi\|\mathbf{x} - \mathbf{v}\|_2^2/s^2\right)$. The discrete Gaussian distribution $\mathcal{D}^m_{\mathbf{v},s}$ centered at $\mathbf{v}$ is defined as*

$$\mathcal{D}^m_{\mathbf{v},s}(\mathbf{x}) = \frac{\rho_{\mathbf{v},s}(\mathbf{x})}{\rho_{\mathbf{v},s}(\mathcal{R}^m)}$$

We omit the subscript $\mathbf{v}$ when $\mathbf{v} = 0$.

Lemma 6. *[LPR13] For positive integers k and ℓ, suppose $m = \ell + k \le \mathsf{poly}(n)$. Let $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in \mathcal{R}_q^{k \times m}$, where $\mathbf{A}$ is uniformly distributed over $\mathcal{R}_q^{k \times \ell}$. Then, with probability $1 - 2^{-\Omega(n)}$ over the choice of $\mathbf{A}$, the distribution of $\bar{\mathbf{A}}\mathbf{x} \in \mathcal{R}_q^k$, where $\mathbf{x} \leftarrow\!\!\$\ \mathcal{D}_s^m$ with parameters $s > 2nq^{k/m + 2/(Nm)}$, satisfies that the probability of each of the q^{nk} possible outcomes is in the interval $(1 \pm 2^{-\Omega(n)})q^{-nk}$. In particular, it is with statistical distance $2^{-\Omega(n)}$ of the uniform distribution over $\mathcal{R}_q^k$.*

Lemma 7. *[DPKM+24] For $\mathbf{s} \leftarrow\!\!\$\ \mathcal{D}_\sigma^k$ and $v \in \mathcal{R}$, we have:*

$$\Pr\left[\|v \cdot \mathbf{s}\|_2 \ge e^{1/4}\|v\|_1 \sigma \cdot \sqrt{nk} \right] \le 2^{-nk/10}$$

Lemma 8. *[MR04, GMPW20] Let T be a positive integer and $\sigma > \sqrt{\frac{\log(2n)+\lambda}{\pi}}$. Then, the distribution of $x := \sum_{i \in T} x_i$ for $x \leftarrow\!\!\$\ \mathcal{D}_\sigma$ is within statistical distance $2^{-\lambda}$ of the distribution $x \leftarrow\!\!\$\ \mathcal{D}_{\sqrt{T} \cdot \sigma}$.*

Lemma 9. *[Lyu12] For $\mathbf{z} \leftarrow\!\!\$\ \mathcal{D}_\sigma^m$ and for any $k > 1$, it holds:*

$$\Pr\left[\|\mathbf{z}\|_2 > k\sigma\sqrt{m} \right] < k^m e^{\frac{m}{2}(1-k^2)}$$

1.4 Linear Secret Sharing

As the basis of our T-out-of-N threshold scheme, in the beginning of the protocol, we have to distribute the secret signing key to the participating users. We will achieve such using the *linear Shamir secret sharing scheme* [Sha79]. The secret will be ultimately distributed as the constant term of a $T-1$ degree polynomial over $\mathcal{R}_q$. Afterwards, the reconstruction of the signing key will be later done via polynomial interpolation using Lagrange polynomials evaluated at zero.

Let $N < q$ be an integer such that for distinct $i, j \in [N]$, $(i-j)$ is invertible over $\mathbb{Z}_q$. Let $\mathsf{SS} \subseteq [N]$ be at least of size T. Given $i \in \mathsf{SS}$, we define the *Lagrange coefficient* $L_{\mathsf{SS},i}$ as

$$L_{\mathsf{SS},i} := \prod_{j \in \mathsf{SS} \setminus \{i\}} \frac{-j}{i-j}$$

Let $s \in \mathcal{R}_q$ be the 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 $E = \{(i, y_i)\}_{i \in \mathsf{SS}}$ such that $y_i = P(i)$ for all $i \in \mathsf{SS}$, we note that

$$s = \sum_{i \in \mathsf{SS}} L_{\mathsf{SS},i} \cdot y_i$$

The notations naturally extend to secret that are in vector form. With a slight abuse of notation, we denote $\overrightarrow{P} \in \mathcal{R}_q^{\ell}[X]$ is of degree $T-1$ if each entry of $\overrightarrow{P}$ is a degree $T-1$ polynomial. Moreover, $\overrightarrow{P}(x)$ denotes the evaluation of each entry of $\overrightarrow{P}$ on the point x.

We note that lattice-based threshold signatures "suffer" from Shamir's secret sharing due to the Lagrange coefficients "ruining" the parameters. Ideally, a novel secret sharing scheme would immediately lead to more efficient protocols.

1.5 Pseudorandom Function

Definition 10. *Let* $\mathsf{PRF} := \{\mathsf{PRF}_\lambda : \{0,1\}^\lambda \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}\}_{\lambda \in \mathbb{N}}$ *be a function family. We say* PRF *is a pseudorandom function if for any efficient adversary* $\mathcal{A}$*, the advantage* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRF}}(1^\lambda)$ *defined below is negligible:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRF}}(1^\lambda) := \left| \Pr\left[\mathcal{A}(1^\lambda)^{\mathsf{PRF}_\lambda(\mathsf{seed},\cdot)} : \mathsf{seed} \leftarrow\!\!\$ \{0,1\}^\lambda\right] - \Pr\left[\mathcal{A}(1^\lambda)^{\mathsf{Rand}(\cdot)} : \mathsf{Rand} \leftarrow\!\!\$ \mathsf{Func}(\{0,1\}^\ell, \{0,1\}^n)\right] \right|$$

, where $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ *denotes the set of all functions from* $\mathcal{X}$ *to* $\mathcal{Y}$*.*

For simplicity, we may omit the subscript λ.

1.6 Lattices

In this section, we present a brief introduction to lattices and their applications to cryptography. We refer the reader to [Pei16] for a more extensive read in the subject.

Definition 11 (Lattice). *An n-dimensional lattice $\mathcal{L}$ is any subset of $\mathbb{R}^n$ that is both:*

1. *an* additive subgroup*:* $\mathbf{0} \in \mathcal{L}$*, and* $-\mathbf{x}, \mathbf{x}+\mathbf{y} \in \mathcal{L}$ *for every* $\mathbf{x}, \mathbf{y} \in \mathcal{L}$*; and*
2. discrete*: every* $\mathbf{x} \in \mathcal{L}$ *has a neighborhood in* $\mathbb{R}^n$ *in which* $\mathbf{x}$ *is the only lattice point.*

Examples of lattice include the integer lattice $\mathbb{Z}^n$, the scaled lattice $c\mathcal{L}$ for any $c \in \mathbb{R}$ and lattice $\mathcal{L}$, and the "checkerboard" lattice $\{\mathbf{x} \in \mathbb{Z}^n \mid \sum_i x_i \text{ is even}\}$.

Definition 12 (Minimum distance). *The minimum distance of a lattice is defined as the length of a shortest non-zero lattice vector:*

$$\lambda_1(\mathcal{L}) := \min_{\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{v}\|$$

Lemma 13. *Although every (non-trivial) lattice $\mathcal{L}$ is infinite, it is always finitely generated as the integer linear combinations of some linearly independent* basis *vectors* $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_k\}$ *such that*

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) := \mathbf{B} \cdot \mathbb{Z}^k = \left\{ \sum_{i=1}^{k} z_i \mathbf{b}_i : z_i \in \mathbb{Z} \right\}$$

We denote the integer k as the *rank* of the basis, and it is an invariant of the lattice. In this report, we restrict ourselves to *full-rank* lattices, i.e., $k = n$. We note that a lattice basis $\mathbf{B}$ is *not* unique (see Fig. 1). We can observe that for any matrix $\mathbf{U} \in \mathbb{Z}^{n \times n}$ with determinant ± 1 (called unimodular), $\mathbf{B} \cdot \mathbf{U}$ is also a basis of $\mathcal{L}(\mathbf{B})$, since $\mathbf{U} \cdot \mathbb{Z}^n = \mathbb{Z}^n$.

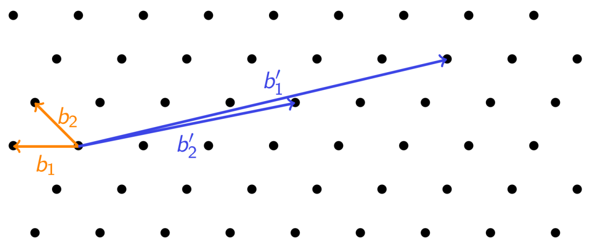


Fig. 1. Example of two non-unique basis, generating the same lattice[2]

1.7 Hardness Assumptions

While many lattice problems have been extensively studied, perhaps the most fruitful computational problem on lattices for cryptography is the *Shortest Vector Problem*(SVP)[3].

Definition 14 (Shortest Vector Problem (SVP)). *Given an arbitrary basis* $\mathbf{B}$ *of some lattice* $\mathcal{L} = \mathcal{L}(\mathbf{B})$, *find a shortest nonzero lattice vector, i.e., a* $\mathbf{v} \in \mathcal{L}$ *for which* $\|v\| = \lambda_1(\mathcal{L})$.

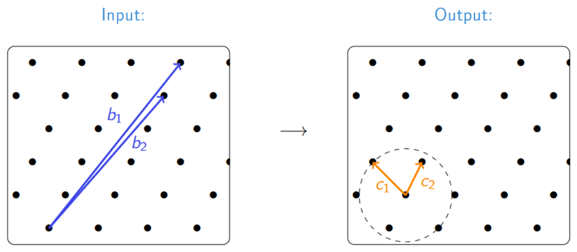


Fig. 2. Example of an SVP instance[4]

While SVP does not directly provide an assumption that can be used to construct concrete lattice-based protocols, a highly useful variation of the SVP problem does allow us to do so. The *Approximate* Shortest Vector Problem is basically the SVP parameterized by an approximation factor $\gamma \geq 1$ that is typically taken to be a function of the lattice dimension n, i.e., $\gamma = \gamma(n)$. Via the approximate-SVP, we can prove secure useful assumptions, which in turn enable us to construct efficient cryptographic schemes.

Definition 15 (Approximate Shortest Vector Problem (SVP$_\gamma$)). *Given a basis* $\mathbf{B}$ *of an n-dimensional lattice* $\mathcal{L} = \mathcal{L}(\mathbf{B}$, *find a non-zero vector* $v \in \mathcal{L}$ *for which* $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$.

[2] Figure from Alice Pellet-Mary's slides
[3] In the bibliography, it can also be found as *Shortest Independent Vectors Problem*(SIVP).
[4] Figure from Alice Pellet-Mary's slides

We can observe that by setting $\gamma = 1$, we reduce to the original SVP problem.

The complexity of approximate-SVP is highly dependant of the approximation factor. For example, for any approximation factor $\gamma \leq O(1)$, SVP_γ is known to be NP-hard [BS99]. However, for an an exponential approximation factor of $\gamma = 2^{\Theta(n \log\log n/\log n)}$ (n being the dimension of the lattice), there do exist polynomial-time algorithms like the Lenstra–Lenstra–Lovász (LLL) lattice basis reduction algorithm [LLL82]. Of course, obtaining polynomial or better approximation factors, either requires superexponential $2^{\Theta(n\log n)}$ time, or exponential $2^{\Theta(n)}$ time and space, obtained by using known algorithms like [Kan83, AKS01, MV10, ADRS15]. Fortunately for cryptographic applications, it is widely accepted to assume that there is no polynomial-time algorithm that achieves a polynomial in n approximation factor, even using a quantum computer [MR09].

In reality however, there are two foundational problems that lattice-based cryptographic protocols rely on, obtaining their security by SVP_γ.

The first one is the *Short Integer Solution problem* (SIS), introduced in 1996 by Ajtai [Ajt96], presenting a worst-case to average-case reduction from SVP_γ to SIS. It is worth noting that this was the first such reduction for a lattice problem, marking the beginning of lattice-based cryptography. The problem can be stated as such: For parameters n, m and q positive integers, find a short nonzero solution $\mathbf{x} \in \mathbb{Z}^m$ to the homogeneous linear system $\mathbf{A}\mathbf{x} = \mathbf{0} \mod q$ for a uniformly random $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.

The second one, introduced in 2005 by Regev [Reg05] is the *Learning With Errors Problem* (LWE), having a quantum reduction from SVP_γ to LWE [Reg05, Reg09]. In terms of linear algebra, for q a positive integer, if m independent samples $(\mathbf{a}_i, \frac{1}{q}\langle \mathbf{a}_i, \mathbf{s}\rangle + e_i)$ are considered, the goal of LWE is to find $\mathbf{s}$ from $(\mathbf{A}, \frac{1}{q}\mathbf{A}\mathbf{s} + \mathbf{e})$, where the rows of $\mathbf{A}$ correspond to the $\mathbf{a}_i$'s and $\mathbf{e} = (e_1, ..., e_m)^\top$. The corresponding decision problem of LWE (the version we shall use in our work), consists of distinguishing between arbitrarily many independent pairs $(\mathbf{a}, \frac{1}{q}\langle \mathbf{a}, \mathbf{s}\rangle + e)$ sampled as in the search version and the same number of uniformly random and independent pairs.

Module Lattices Unfortunately, cryptographic protocols relying on the hardness of SIS or LWE are fundamentally inefficient since the matrix $\mathbf{A}$ that is included in the public data, is large. In order to tackle this problem, Micciancio proposed an efficient variant of the SIS and LWE problems [Mic02, Mic07]. Now, the matrix $\mathbf{A}$ has a certain structure thus admitting to smaller keys and more efficient algorithms. That was the beginning of improved variants like the Short Integer Solution *over Rings* (RSIS) [LM06, SSTX09] and respectively, Learning with Errors *over Rings* (RLWE) [LPR10]. Of course, now, the corresponding worst-case problem is SPV_γ when restricted to *ideal lattices* called Id-SVP_γ.

In this work, we rely on the Module-SIS (MSIS) and Module-LWE (MLWE) problems [BGV11, LS15]. A module is an algebraic structure that generalizes rings and vector spaces. In this way, module lattices (that correspond to finitely generated modules over the ring of integers of a number field) generalize both arbitrary lattices and ideal lattices.

A great difference between Module and Ring assumptions is that for the worst-case to average case reductions for the module problems, there exist converse reductions, just like in the SVP to SIS/LWE reductions. That does not hold in the Ring setting. For example, Id-SVP could be easier to solve than RSIS/RLWE. With an efficiency slowdown (in terms of memory requirements, communication costs and algorithm run-times) bounded just by a constant factor, one can enjoy the security of a possibly harder problem, Module-SVP [LS15].

1.8 Background

Below, we provide a (brief) introduction to module lattices.

Let $\zeta = \zeta_\mathfrak{f} \in \mathbb{C}$ denote any fixed primitive $\mathfrak{f}$-th root of unity where $\mathfrak{f}$ is a power of 2, $\mathcal{K} = \mathbb{Q}(\zeta)$ the cyclotomic field of conductor[5] $\mathfrak{f}$ and degree $\varphi = \varphi(\mathfrak{f}) = \mathfrak{f}/2$, and $\mathcal{R} = \mathbb{Z}[\zeta] \cong \mathbb{Z}[X]/\langle \Phi_\mathfrak{f}(X)\rangle$ its ring of integers, also called a cyclotomic ring, where $\Phi_\mathfrak{f}(X) = X^\varphi + 1$ is the $\mathfrak{f}$-th cyclotomic polynomial.

We can represent an element $x \in \mathcal{K}$ (resp. $\mathcal{R}$) as a linear combination of the power basis, i.e., $x = \sum_{i=0}^{\varphi-1} x_i\zeta^i$ where $x_i \in \mathbb{Q}$ (resp. $\mathbb{Z}$). We define the *coefficient embedding* as the vector $\mathsf{coeff}(x) := (x_i)_i^{\varphi-1}$.

[5] If $E/\mathbb{Q}$ is a subextension of a cyclotomic field, the conductor is the smallest $\mathfrak{f}$ such that $E \subseteq \mathbb{Q}(\zeta_\mathfrak{f})$.

Theorem 16. *When $\mathfrak{f}$ is a power of 2, the spaces $\mathcal{K}^m$ and $\mathbb{R}^{\varphi m}$ are isomorphic as inner-product spaces via the coefficient embedding* $\mathsf{coeff}(\cdot)$. *The module $\mathcal{R}^m$ can thus be viewed as a lattice.*

For the needs of our report, we can just focus that most operations instead of working over $\mathbb{Z}$ and $\mathbb{Z}_q$, work over polynomial rings $\mathcal{R} = \mathbb{Z}[X]/(f(X))$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/(f(X))$, where $f(X) = X^N + 1$ with n a power-of-two, is the $2n$-th cyclotomic polynomial, and q is a prime that satisfies $q \equiv 5 \mod 8$. Elements over the latter ring have coefficients between $-(q-1)/2$ and $(q-1)/2$. The L_p-norm for a vector of ring elements $\mathbf{v} = \left[\sum_{i=0}^{n-1} v_{1,i}X^i, ..., \sum_{i=0}^{n-1} v_{m,i}X^i\right]^\top \in \mathcal{R}^m$ is defined as

$$\|\mathbf{v}\|_p = \|[v_{1,0}, ..., v_{1,n-1}, ..., v_{m,0}, ..., v_{m,n-1}]\|_p$$

(Module) Lattice assumptions Finally, we review the standard (module) lattice-based hardness assumptions that the security of our threshold signature scheme is based on.

Definition 17 (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:*

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

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

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MSIS}}(1^\lambda) = \Pr\left[\mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k\times\ell}, \mathbf{s} \leftarrow\!\!\$\ \mathcal{A}(\mathbf{A}) : (0 < \|\mathbf{s}\|_2 \leq \beta) \wedge [\mathbf{A}|\mathbf{I}]\mathbf{s} = \mathbf{0} \mod q\right]$$

Lemma 19 (Hardness of MLWE [LS15]). *Let $k(\lambda), \ell(\lambda), q(\lambda), n(\lambda), \sigma(\lambda)$ such that $q \leq \mathsf{poly}(n\ell)$, $k \leq \mathsf{poly}(\ell)$, and $\sigma \geq \sqrt{\ell}\cdot\omega(\sqrt{\log n})$. If $\mathcal{D}$ is a discrete Gaussian distribution with standard deviation σ, then the $\mathsf{MLWE}_{q,\ell,k,\mathcal{D}}$ problem is as hard as the worst-case lattice Generalized-Independent-Vector-Problem (*GIVP*) in dimension $N = n\ell$ with approximation factor $\sqrt{8\cdot N\ell}\cdot\omega(\sqrt{\log n})\cdot q/\sigma$.*

Lemma 20 (Hardness of MSIS [LS15]). *For any $k(\lambda), \ell(\lambda), q(\lambda), n(\lambda), \beta(\lambda)$ such that $q > \beta\sqrt{nk}\cdot\omega(\log(nk))$, and ℓ, $\log q \leq \mathsf{poly}(nk)$. The $\mathsf{MSIS}_{q,\ell,k,\beta}$ problem is as hard as the worst-case lattice Generalized-Independent-Vector-Problem (*GIVP*) in dimension $N = nk$ with approximation factor $\beta\sqrt{N}\cdot\omega(\sqrt{\log N})$.*

1.9 Norms and Modulus Rounding

Let q and n positive integers. We use the *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 $x \equiv t \mod q$. Given a class $x + q\mathbb{Z}^n \in \mathbb{Z}_q$, we define the corresponding lift $\bar{x}$ to the unique integer in $x + q\mathbb{Z} \cap [0, \ldots, q-1]$.

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 $\|x\|$. It is shown in [DPKM+24] that $\|\cdot\|$ is indeed a norm. We state the triangular inequality, a norm property we will use later on.

Lemma 21. *[DPKM+24] For any $q, n \in \mathbb{N}\backslash\{0,\}$, and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, we have:*

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

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

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

which consists essentially in separating the lower-order bits from the higher-order ones, i.e., dropping ν lower bits. We will use bit dropping in order to compress the public key and improve the efficiency of our scheme, following [DPKM+24]. We define the function

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

where $\|\cdot\| : \mathbb{R} \to \mathbb{Z}$ denotes the *rounding* operator. More precisely, the "rounding half-up" method $\lfloor x \rceil = \lfloor x + 1/2 \rceil$ 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 $\|\cdot\|_\nu$ 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

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

where $\bar{x} \in \mathbb{Z}_q$ is assumed to have the unsigned representative, i.e., $x \in \{0, 1, ..., q-1\}$.

The function $\lfloor \cdot \rceil_\nu$ naturally extends to vectors coefficient-wise. The following lemma is useful when we need to bound the norm difference of two vectors caused by performing modular rounding for efficiency.

Lemma 22. *[DPKM+24, EKT24] Let $\nu, q \in \mathbb{N}$ such that $q > 2^\nu$, $\nu \geq 4$, 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 $q_\perp \in [0, 2^{\nu-1} - 1]$. Them. for any $x \in \mathbb{Z}_q$, we have*

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

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

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

In order to aid readability, in the rest of the report we will not be as precise as above when the context is clear, e.g. x instead of $\bar{x}$ or $|2^\nu \cdot x|$ instead of $|2^\nu \cdot \bar{x} \mod q|$.

1.10 Random Oracle Model

While random oracles were already a tool in computational complexity theory, [BR93] first introduced the Random Oracle Model (ROM) and showed that it is indeed practical.

We present an "informal" definition of the Random Oracle.

Definition 23 (Random Oracle [Kia]). *A random oracle is a function that produces a random looking output for each query it receives. It must be consistent with its replies; if a question is repeated, the random oracle must return the same answer*

The figure below illustrates how a hash function $H : \{0,1\}^* \to \{0,1\}^{256}$ is modeled as a random oracle.

```
$H(M)$ as a random oracle
1 : if $M \notin$ History
2 :     $t \leftarrow_\$ \{0,1\}^{256}$
3 :     Add $(M, t)$ to History
4 : return $t$, such that $(M, t) \in$ History
```

In cryptography, when proving security in the Random Oracle Model, every hash function is replaced by a random oracle. Proving digital signature schemes under the ROM is the standard way of proving security, and it is regarded being a proper model to prove security under.

For bibliographical reasons, we note that it has been shown that there exist signature and encryption schemes that are secure in the Random Oracle Model, but for which any implementation of the random oracle results in insecure schemes [CGH98].

1.11 Forking Lemma

Pointcheval and Stern [PS00] introduced the *forking lemma* in the context of signature schemes, using it as the main ingredient for providing security arguments for many schemes like the security of the Schnorr Digital Signature Scheme [Sch90, Sch91]. Bellare and Neven [BN06] reformulated the forking lemma to extract the purely probabilistic nature of the lemma by making no mention of signature schemes or random oracles, known as *General forking lemma.* Here we shall state the (General) forking lemma that we will use later to finish the security proof of our threshold signature scheme.

Definition 24 (General forking lemma [BN06]). *Fix an integer $Q \geq 1$, a set $\mathcal{H}$ of size $|\mathcal{H}| \geq 2$. Let $\mathcal{A}$ be a randomized algorithm that on input $X, h_1, ..., h_Q$, takes as random coin tosses from set $\mathcal{R}$, and outputs tuple (I, Y) where $I \in \{0, ..., Q\}$ and Y is what we call "side output". Let $\mathcal{D}_X$ be an unspecified distribution. The accepting probability of $\mathcal{A}$, denoted* acc, *is defined as*

$$\mathsf{acc} = \Pr\left[\, I \geq 1 \mid X \leftarrow\!\!\$\, \mathcal{D}_X; h_1, ..., h_Q \leftarrow\!\!\$\, \mathcal{H}; (I, Y) \leftarrow\!\!\$\, \mathcal{A}(X, h_1, ..., h_Q) \,\right]$$

Define the forking algorithm $\mathsf{Fork}_{\mathcal{A}}$ *with respect to* $\mathcal{A}$ *as follows:*

```
Fork_A(X) :
1: $\rho \leftarrow\!\!\$\, \mathcal{R}$
2: $h_1, ..., h_Q \leftarrow\!\!\$\, \mathcal{H}$
3: $(I, Y) \leftarrow \mathcal{A}(X, h_1, ..., h_Q; \rho)$
4: if $I = 0$ then return $\bot$
5: $h'_I, ..., h'_Q \leftarrow\!\!\$\, \mathcal{H}$
6: $(I', Y') \leftarrow \mathcal{A}(X, h_1, ..., h_{I-1}, h'_I, ..., h'_Q; \rho)$
7: if $I \neq I' \vee h_I = h'_I$ then return
8: return $(I, Y, Y')$
```

Let

$$\mathsf{frk} = \Pr\left[\, \mathsf{Fork}_{\mathcal{A}}(X) \neq \bot \mid X \leftarrow\!\!\$\, \mathcal{D}_X \,\right]$$

Then

$$\mathsf{frk} \geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}}{Q} - \frac{1}{|\mathcal{H}|} \right)$$

Alternatively,

$$\mathsf{acc} \leq \frac{Q}{|\mathcal{H}|} + \sqrt{Q \cdot \mathsf{frk}}$$

1.12 Threshold Signatures

We present the syntax for our two-round T-out-of-N threshold signature scheme, where the first round is message independent (offline round). Let N denote the total number of parties, T the minimum number of parties required to sign, $\mathsf{SS} \subseteq [N]$ the subset of parties that are actively participating in the signature. We assume the adversary can corrupt up to $T - 1$ parties.

Definition 25 ((T, N)-Threshold Signature Scheme). *A (T, N)-threshold signature scheme is a tuple of interactive PPT*[6] *algorithms* $\mathsf{TS} = (\mathsf{TS.Setup}, \mathsf{TS.KeyGen}, \mathsf{TS.PP}, \mathsf{TS.Sign}, \mathsf{TS.Agg}, \mathsf{TS.Verify})$ *such that:*

- $\mathsf{TS.Setup}(1^\lambda, N, T) \to \mathsf{tspar}$: The setup algorithm takes as input public parameters such as the security parameter, the total number of parties and signing threshold, and outputs the "threshold parameters", denoted as tspar.

- $\mathsf{TS.KeyGen}(\mathsf{tspar}) \to \big(\mathsf{vk}, (\mathsf{sk}_i)_{i \in [N]}\big)$: In the key generation algorithm, the verification key vk is generated, and the signing (secret) keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ are produced accordingly for distribution to each party in the set. We note that from this point forward, signatures under vk can only be generated by parties controlling at least T secret shares sk_i.

- $\mathsf{TS.PP}(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i) \to \mathsf{pp}_i$: In the TS.PP algorithm, each party generates its *commitment.* During this phase, the first round of interaction occurs, where each party broadcasts its commitment.

  We emphasize that, up to this point, our signature scheme is message-independent allowing every party to compute and broadcast its commitment "offline", without prior knowledge of the message to be signed.

[6] PPT: Probabilistic Polynomial Time

- $\mathsf{TS.Sign}(\mathsf{vk}, \mathsf{SS}, M, i, (\mathsf{pp}_j)_{j\in\mathsf{SS}}, \mathsf{sk}_i, \mathsf{st}_i) \to (\widehat{\mathsf{sig}}_i, \mathsf{st}_i)$: In the signing algorithm, each signer provides the verification key vk, the set SS consisting of parties actively generating a signature for the message M, the commitments for all individual parties, as well as its own secret key and state, and inputs them to the algorithm. In the end, the partial signature $\widehat{\mathsf{sig}}_i$ is generated and broadcast to the other members. This marks the second and final interaction required by our protocol.

- $\mathsf{TS.Agg}(\mathsf{vk}, \mathsf{SS}, M, (\widehat{\mathsf{sig}}_j)_{j\in\mathsf{SS}}) \to \mathsf{sig} := (c, \widetilde{\mathbf{z}}, \mathbf{h}, \widetilde{\mathbf{p}})$: The aggregation algorithm combines all the broadcast information to generate a final signature sig for the message M under the verification key vk. We emphasize that, since the inputs of the algorithm are all public, the role of the aggregator can be fulfilled even by parties outside of the signing set SS. In fact, the aggregator does not need to be part of the original set $[N]$.

- $\mathsf{TS.Verify}(\mathsf{vk}, M, \mathsf{sig}) \to 1/0$: The deterministic verification algorithm takes as input the verification key, the signed message and the signature, and outputs 1 if the signature is valid and 0 otherwise.

We note that the signers have to maintain just a temporary state with respect to a particular session. This is a minimal requirement for any interactive protocol and is not the same as with Threshold Raccoon [DPKM+24], where the signer has to keep a general state of all the previous signed messages, even between different sessions.

A threshold signature scheme should satisfy the correctness and unforgeability properties as defined below.

Definition 26 (Correctness). *Correctness holds if an honest set of signers* SS *and aggregator always create a valid signature in the end of the protocol. More formally, for all* $\lambda \in \mathbb{N}$*, there exists a negligible function* negl *such that for all* $T, N \in \mathsf{poly}(\lambda)$ *such that* $1 \le T \le N$*, for all* $\mathsf{SS} \subseteq [N]$ *with* $|\mathsf{SS}| \ge T$*, and for all messages* $M \in \{0,1\}^*$*, the following inequality holds*

$$\Pr\left[\mathsf{TS.Verify}(\mathsf{vk}, M, \mathsf{sig}) = 1 \;\middle|\; \begin{array}{c} \mathsf{tspar} \leftarrow \mathsf{TS.Setup}(1^\lambda, N, T) \\ (\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{TS.KeyGen}(\mathsf{tspar}) \\ \forall\ i \in \mathsf{SS},\ \mathsf{pp}_i \leftarrow \mathsf{TS.PP}(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i) \\ \forall\ i \in \mathsf{SS},\ (\widehat{\mathsf{sig}}_i, \mathsf{st}_i) \leftarrow \mathsf{TS.Sign}(\mathsf{vk}, \mathsf{SS}, M, i, (\mathsf{pp}_j)_{j\in\mathsf{SS}}, \mathsf{sk}_i, \mathsf{st}_i) \\ \mathsf{sig} \leftarrow \mathsf{TS.Agg}(\mathsf{vk}, \mathsf{SS}, M, (\widehat{\mathsf{sig}}_j)_{j\in\mathsf{SS}}) \end{array}\right] \ge 1 - \mathsf{negl}(\lambda)$$

In our unforgeability model, aside from allowing the adversary to corrupt up to $T-1$ parties, we allow the adversary to trigger *concurrent signing* in the trusted key generation model. Moreover, we are allowing *public aggregation*, i.e., the adversary is able to observe the partial signatures of honest parties even when the set of signers SS only includes honest parties.

Definition 27 (Unforgeability). *For a two round threshold signature scheme* TS*, the advantage of an adversary* $\mathcal{A}$ *against the unforgeability of* TS *in the random oracle is defined as*

$$\mathcal{A}^{\mathsf{ts-uf}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T) = \Pr\left[\mathsf{Game}^{\mathsf{ts-uf}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T) = 1\right]$$

, where $\mathsf{Game}^{\mathsf{ts-uf}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T)$ *is described in Fig. 5. Unforgeability in the ROM holds if, for all* $N, T \in \mathsf{poly}(\lambda)$ *such that* $1 \le T \le N$*, and an efficient adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{ts-uf}}_{\mathsf{TS},\mathcal{A}}(1^\lambda) = \mathsf{negl}(\lambda)$ *holds.*

A common practice between practical post-quantum signature schemes [FHK+18,dPEK+23,DPKM+24, BKL+24] is to require unforgeability to hold against any adversary making at most $Q_S = \mathsf{poly}(\lambda)$ signing queries to $\mathcal{O}_{\mathsf{TS.Sign}}$, e.g., $Q_S \approx 2^{60}$ as recommended by NIST [BP23], a practically infinite number of signatures. We follow the same approach since allowing unbounded polynomially many signing queries, would make the scheme impractical requiring the modulus q to be of super-polynomial size.

2 Overview

In this section we provide an overview of our offline-online two-round threshold signature scheme based on the MLWE and MSIS assumptions.

2.1 Intuition

We will firstly describe the intuition of how can someone arrive to our signature scheme.

The foundation of the aforementioned related works and subsequently our work is Lyubashevsky's lattice-based signature scheme [Lyu09, Lyu12]. The protocol goes as follows. Let $\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$ and $\mathbf{T} = \mathbf{A}\cdot\mathbf{s}+\mathbf{e} \in \mathcal{R}_q^k$ for "short" vectors $(\mathbf{s},\mathbf{e})$. The tuple $(\mathbf{A},\mathbf{T})$ will be the verification key (*public* key) denoted as vk while the tuple $(\mathbf{s},\mathbf{e})$ will be the signing key (*secret* key) denoted as sk. In order for the signer, in possession of the signing key $\mathsf{sk} = (\mathbf{s},\mathbf{e})$, to sign a message μ, first constructs a *commitment* $\mathbf{w} = \mathbf{A}\cdot\mathbf{y}+\mathbf{e}'$, where $(\mathbf{y},\mathbf{e}')$ are "short" vectors sampled by some specific distribution. A *challenge* $c \leftarrow H_{\mathsf{sig}}(\mathsf{vk}, M, \mathbf{w})$, followed by a "short" *response* $(\mathbf{z},\mathbf{z}') := (c\cdot\mathbf{s}+\mathbf{y}, c\cdot\mathbf{e}+\mathbf{e}')$ is then computed. Finally, $(c,\mathbf{z},\mathbf{z}')$ is the signature. To verify, we check if $(\mathbf{z},\mathbf{z}')$ are "short" and that $c = H_{\mathsf{sig}}(\mathsf{vk}, M, \mathbf{A}\cdot\mathbf{z}+\mathbf{z}'-\mathbf{T}c)$.

Afterwards, one can perform rejection sampling [Lyu09, Lyu12] to make the distribution of the responses independent of the signing key. However, with rejection sampling one has to "abort" the protocol some times, leading to performance issues. One can perform noise "flooding" [GKPV10], a technique used both in [EKT24, DPKM$^+$24] that allows participating signers not to abort and thus reducing the "actual" rounds of interaction. Of course, we note that noise "flooding" comes the with the drawback of ruining the protocol parameters.

Schnorr signatures To make more understandable the base of our scheme, it is important to explain the origins of the keywords *commitment, challenge* and *response.* Lyubashevsky's signature scheme is based on the Schnorr identification protocol [Sch91]. The purpose of this protocol is to allow one party, Alice, to prove to another party, Bob, that she possesses knowledge of a discrete logarithm, without revealing the discrete logarithm itself.

The protocol will be interactive. Initially, Alice generates a *commitment*[7], e.g. an integer. Following this, Bob generates a *challenge*, e.g. another integer, that depends on Alice's commitment. After receiving the challenge, Alice must create a *response* and send it back to Bob. Bob can then verify that Alice indeed possesses the knowledge of the discrete logarithm if she successfully provides a valid response. The core idea here is that Alice is bound by her commitment before she knows the challenge. The security of the protocol relies on the fact that Alice cannot forge a correct response to an arbitrary challenge.

The Fiat-Shamir heuristic [FS87] provides a method to convert an interactive proof of knowledge, such as the one described above, into a digital signature, effectively transforming it into a non-interactive protocol. The idea goes as follows: After Alice generates a commitment, she creates *herself* a challenge by hashing the commitment. Due to the collision resistance of cryptographically secure hash functions, Alice cannot deliberately craft a specific challenge. Consequently, Alice "simulates" a valid challenge in the same way that Bob would generate one. She then proceeds to create the appropriate response. Bob's task is now simply to verify that the challenge was derived correctly—that is, that the hash of the commitment indeed matches the challenge—and to confirm that Alice's response is valid.

That is the reason we use the terms *commitment, challenge* and *response* even though all of them are generated by the same user.

Thresholdizing Lattices The intuitive approach for building a threshold signature scheme over Lyubashevsky's work is to use Shamir's secret sharing scheme [Sha79] in order to split and share the signing key. Now, each partial signing key $(\mathbf{s}_i)_i \in [N]$ satisfies $\mathbf{s} = \sum_{i\in[N]} L_{\mathsf{SS},i}\cdot\mathbf{s}_i$ for any set $\mathsf{SS} \subset [N]$ with $|\mathsf{SS}| = T$, where $L_{\mathsf{SS},i}$ is the Lagrange coefficient. Correctness holds since $\mathbf{z} = \sum_{i\in\mathsf{SS}} \mathbf{z}_i = c\cdot\mathbf{s}+\mathbf{y}$, where $\mathbf{y} := \sum_{i\in\mathsf{SS}} \mathbf{y}_i$. Thus, the protocol would proceed as follows. Each user $i \in \mathsf{SS}$ computes $\mathbf{w}_i = \mathbf{A}\cdot\mathbf{y}_i+\mathbf{e}'_i$. In a standard procedure for threshold signature schemes [BN06, CKM23], to counter-act from rushing adversaries, i.e., maliciously generating its commitment *after* learning all the honest users commitments, user i will initially output only a hash commitment $H_{\mathsf{com}}(\mathbf{w}_i)$. Only after obtaining the hash commitments from all users in SS, user i will reveal $\mathbf{w}_i$ and check the correctness of all the other reveals. Afterwards, he computes a *challenge* $c \leftarrow H_{\mathsf{sig}}(\mathsf{vk}, M, \mathbf{w})$ where $\mathbf{w} := \sum_{j\in\mathsf{SS}} \mathbf{w}_j$. His

[7] It is important to note that this term should not be confused with the cryptographic primitive called commitment scheme. In this context, "commitment" is an informal restriction on Alice for her future behavior. Once she declares a certain value, she is bound to it throughout the protocol.

partial signature is defined as $(\mathbf{z}_i, \mathbf{z}'_i) := (c \cdot L_{\mathsf{SS},i} \cdot \mathbf{s}_i + \mathbf{y}_i, c \cdot \mathbf{e}_i + \mathbf{e}'_i)$. The *aggregated signature* will simply be $(c, \mathbf{z}, \mathbf{z}') := (c, \sum_{j \in \mathsf{SS}} \mathbf{z}_j, \sum_{j \in \mathsf{SS}} \mathbf{z}'_j)$. To verify, as in the original scheme, we check if $(\mathbf{z}, \mathbf{z}')$ are "short" and that $c = H_{\mathsf{sig}}(\mathsf{vk}, M, \mathbf{A}\mathbf{z} + \mathbf{z}' - \mathbf{T}c)$.

However, there is a security liability due to the nature of Lagrange coefficients. Due to the fact that they can be arbitrarily large over modulo q, an adversary can choose them adaptively in order to forge the "intuitive" threshold version of Lyubashevski's signature scheme. Examining the partial signature of user i, the adversary can observe that user i basically provides a valid non-threshold signature with $(L_{\mathsf{SS},i} \cdot \mathbf{s}_i, \mathbf{e}_i)$ as the corresponding signing key. Thus, the adversary obtains information about the corresponding scaled partial public key $\mathbf{T}_{\mathsf{SS},i} = L_{\mathsf{SS},i} \cdot \mathbf{A}\mathbf{s}_i$. Afterwards, by specifying a different signer set SS, the adversary adaptively asks user i to sign on a scaled public key of its choice. Finally, with specially crafted Lagrange coefficients $L_{\mathsf{SS},i}$, the partial signing key $\mathbf{s}_i$ can be recovered from standard linear algebra. The lattice peculiarity that allows the attack to unfold is that all the obtained scaled partial public keys would be linearly dependent, if there was not the noise vector $\mathbf{e}_i$, which doesn't exist outside of the lattice setting.

Although several "quick" fixes have been suggested in the lattice bibliography, a truly effective solution has not yet been found. For example, a simple approach could be to make the modulus q big enough that the Lagrange coefficients become relatively small to q [ABV+12, BGG+18, CCK23], something completely inefficient as q should grow with at least $O(N!^2)$. Another intuitive approach has been to use an alternative secret sharing scheme instead of Shamir's secret shares. One of them is known as the {0,1}-linear secret sharing scheme [DT06, LST18, BGG+18, CCK23], but it is still inefficient since the reconstruction algorithm is more costly and the individual secret shares grow by at least $O(N^4)$ [Val84].

Raccoon's solution Threshold Raccoon [DPKM+24] manages to overcome the aforementioned problem, simply by exploiting the fact that threshold signatures are interactive. Intuitively, each signer additively masks their partial signature with a random vector. In the end, the aggregator is able to publicly remove the sum of the masks. As we will explain below, it is crucial to point out that only the sum of the masks can be removed in the final signature, not each individual mask in the partial signatures.

In Threshold Raccoon, every two pairs of users $i, j \in \mathsf{SS}$ privately share also a pair-wise seed for a pseudorandom function (PRF). In the first round, signer i computes the *row* mask $\mathbf{m}_i := \sum_{j \in \mathsf{SS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{sid})$ where sid is a unique string defined per session, and broadcasts it to the other users. In the third round, it computes a *column* mask $\mathbf{m}^*_i := \sum_{j \in \mathsf{SS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{sid})$ and adds this to its response (partial signature) $\mathbf{z}_i = (c \cdot L_{\mathsf{SS},i} \cdot \mathbf{s}_i + \mathbf{y}_i + \mathbf{m}^*_i)$. We emphasize that while the row masks $(\mathbf{m}_i)_{i \in \mathsf{SS}}$ are broadcasted and therefore *public*, the column masks $(\mathbf{m}^*_i)_{i \in \mathsf{SS}}$ are *private*. In Fig. 3, one can see a pictorial example and where the terms row/column mask come from.

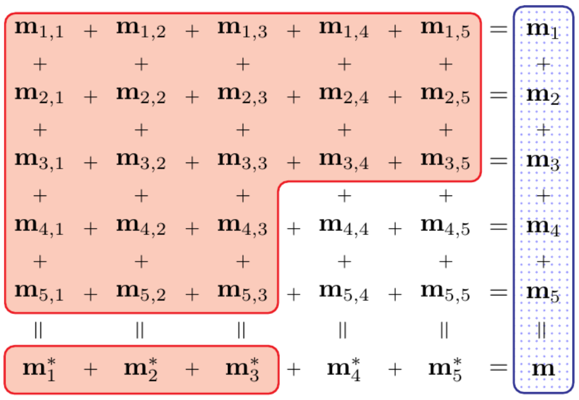
The aggregator now, since we have $\sum_{j \in \mathsf{SS}} \mathbf{m}_j = \sum_{j \in \mathsf{SS}} \mathbf{m}^*_j$, can subtract the sum of the row masks $\sum_{j \in \mathsf{SS}} \mathbf{m}_j$ from the partial responses $\mathbf{z}_i$ to finally compute the aggregated response $\mathbf{z} = \sum_{j \in \mathsf{SS}} (\mathbf{z}_j - \mathbf{m}_j) = \sum_{j \in \mathsf{SS}} (c \cdot L_{\mathsf{SS},j} \cdot \mathbf{s}_j + \mathbf{y}_j + (\mathbf{m}^*_j - \mathbf{m}_j))$, managing to remove the previously added masks.

We denote HS (resp. CS) the set of honest (resp. corrupt) users. We remind that the adversary can corrupt up to $T-1$ users, i.e., $|\mathsf{CS}| \leq T-1$. The individual row masks $(\mathbf{m}_j)_{j \in \mathsf{HS}}$ are all known to the adversary. Since the adversary controls CS, he also knows the column masks $(\mathbf{m}^*_j)_{j \in \mathsf{CS}}$. However, the only knowledge the adversary gains on the column masks of the honest users $(\mathbf{m}^*_j)_{j \in \mathsf{HS}}$ are their sum $\sum_{j \in \mathsf{HS}} \mathbf{m}^*_j$. Because $\mathbf{m}^*_i = \sum_{j \in \mathsf{HS}} \mathbf{m}_{i,j} + \sum_{j \in \mathsf{CS}} \mathbf{m}_{i,j}$, $(\mathbf{m}^*_j)_{j \in \mathsf{HS}}$ are distributed randomly, conditioned on the corresponding sum. Each $\mathbf{m}_{i,j}$ with users i, j being *both* honest, looks random to the adversary under the pseudorandomness of the PRF.

Conceptually, this masking trick will allow the security reduction (the security proof) to simulate the response using only the full signing key, without the partial signing key. This ultimately means that the Lagrange coefficients in the reduction can be removed, leading to a reduction similar to a standard non-thresholdized signature scheme.

[EKT24] two-round approach As previously mentioned, it is standard practice for threshold signature schemes [BN06, CKM23] to exchange a *commitment* in a commit-reveal[8] manner. This approach prevents rushing adversaries from maliciously generating their commitment after learning the commitments of all honest users. Threshold Raccoon follows this pattern. In the first round,

[8] Also known as hash-and-open.



$\mathbf{m}_{1,1} + \mathbf{m}_{1,2} + \mathbf{m}_{1,3} + \mathbf{m}_{1,4} + \mathbf{m}_{1,5} = \mathbf{m}_1$
$\mathbf{m}_{2,1} + \mathbf{m}_{2,2} + \mathbf{m}_{2,3} + \mathbf{m}_{2,4} + \mathbf{m}_{2,5} = \mathbf{m}_2$
$\mathbf{m}_{3,1} + \mathbf{m}_{3,2} + \mathbf{m}_{3,3} + \mathbf{m}_{3,4} + \mathbf{m}_{3,5} = \mathbf{m}_3$
$\mathbf{m}_{4,1} + \mathbf{m}_{4,2} + \mathbf{m}_{4,3} + \mathbf{m}_{4,4} + \mathbf{m}_{4,5} = \mathbf{m}_4$
$\mathbf{m}_{5,1} + \mathbf{m}_{5,2} + \mathbf{m}_{5,3} + \mathbf{m}_{5,4} + \mathbf{m}_{5,5} = \mathbf{m}_5$
$\mathbf{m}^*_1 + \mathbf{m}^*_2 + \mathbf{m}^*_3 + \mathbf{m}^*_4 + \mathbf{m}^*_5 = \mathbf{m}$

Fig. 3. Figure taken from [DPKM$^+$24]. Relationships between the individual masks $\mathbf{m}_{i,j}$, the row masks $\mathbf{m}_i$, and column masks $\mathbf{m}^*_j$, with $\mathsf{SS} = \{1, 2, 3, 4, 5\}$.

- 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).
- By construction, it holds that $\sum_{j\in\mathsf{SS}} \mathbf{m}_j = \mathbf{m} = \sum_{j\in\mathsf{SS}} \mathbf{m}^*_j$.

all users output the hash of their commitment share, and in the second round, they reveal their commitment share $\mathbf{w}_j$.

However, [EKT24] effectively addresses this issue by eliminating the extra "opening the hash commitment" round. They adapt the "random linear combination trick" from the (discrete logarithm-based) threshold signature scheme FROST [KG20] to the lattice setting. In this approach, during the first round, each signer broadcasts a list of commitments $\overrightarrow{\mathbf{w}_i} := [\mathbf{w}_{i,1}, \ldots, \mathbf{w}_{i,\mathsf{rep}}]$ openly. To prevent rushing adversaries, in the second round, they compute a random weight $(\beta_b)_{b\in[\mathsf{rep}]}$ and locally set the partial commitment $\mathbf{w}_j := \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{w}_{j,b}$. These random weights are generated by a hash function G, modeled as a random oracle, and being dependent on the message, the current participating users, and their commitment shares. This approach ensures that the adversary cannot maliciously construct a commitment to their advantage, as it is protected by the collision-resistance of cryptographic hash functions.

Nonetheless, the drawback is that they prove their protocol under a new, non-standard computational assumption called Algebraic One-More LWE (AOM-LWE). In an One-More type assumption, the adversary is given with Q challenge instances, such as Q MLWE instances, and their goal is to output Q solutions to these Q challenge instances while having access to the solving oracle for at most $Q-1$ instances. In their algebraic setting[9], the adversary is allowed to access the solving oracle only on an *algebraic* combination of the provided challenge instances.

Espitau et al. in [EKT24] introduce AOM-LWE as the lattice analog of the Algebraic One-More Discrete-Logarithm problem (AOM-DL), which was introduced in [NRS21] to prove the security of the multi-signature scheme MuSig2.

Nevertheless, aside from the concern that the AOM-LWE assumption is relatively new, accurately translating discrete-logarithm problems to lattices remains challenging. This difficulty arises because MLWE instances posses more structure compared to their discrete-logarithm counterparts.

Remark 28. In [EKT24], the row and column masks $(\mathbf{m}_i, \mathbf{m}^*_i)$ are generated in the second round using the PRF evaluated on the input $\mathsf{ctnt} := [\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_j})_{j\in\mathsf{SS}}]$. This results in two key efficiency improvements: First, the output of the first round, $\overrightarrow{\mathbf{w}_i}$, does not depend on any specific message or set of signers, which introduces the *offline-online* property. Second, signers no longer need to maintain a session-specific identifier sid. Due to the high min-entropy of $\overrightarrow{\mathbf{w}_j}$, if the honest signers correctly follow the protocol, no adversary can trick them into using the same input to the PRF.

[9] This is different from the Algebraic Group Model [FKL18], where the adversary is restricted to be algebraic. Since lattices inherently support non-algebraic operations, restricting the adversary to algebraic operations would be too limiting.

Reducing to standard assumptions When proving the security of signature schemes, the simulator needs to perform a forgery by simulating the signing procedure without possessing the secret key. If the simulator then successfully "breaks" the underlying computational assumption, such as computing a discrete logarithm or finding a valid MSIS instance, we can then argue that such a forgery should be infeasible under the given assumptions. Consequently, this leads to the conclusion that the scheme is secure.

To achieve this, the simulator has "extra" powers that a normal signer does not. For example, a simulator could "rewind time" and generate parts of the protocol's transcript in different ways. In the case of individual signing, the simulator can generate the commitment, the challenge, and the response in any way that is convenient, enabling it to produce a forgery.

However, in a threshold setting, things change. For instance, when the simulator simulates an honest party, it must give a commitment in the first round of interaction. But the challenge hasn't been determined yet, as it depends on the commitments provided by the other signers. Afterwards, the simulator will then need to produce a valid response once the challenge is decided. This shows that, in the threshold setting, the simulator can't exploit the "extra power" it would typically use in a standard signature scheme. Nevertheless, it still needs to perform a forgery, which a "normal" honest signer would not be able to do. Therefore, in threshold signature schemes, the so-called *straight-line simulation*, i.e. commitment, challenge and last response, should be intentional, and the protocol should provide the simulator with additional powers in an alternative way.

This can occur in various ways. One approach is to use trapdoor-based simulation techniques, as explained in works such as [GPV08, MP12]. With the trapdoor, the simulator can produce a valid forgery without possessing the secret key while simulating the protocol "in-order". However, the use of a trapdoor can have negative effects, such as worsening certain parameters, for example, by increasing signature or key sizes, or by adding communication complexity.

Following the construction of DualMS [Che23], we prove the security of our scheme using trapdoor-free (straight-line) simulation. Intuitively, we introduce another short vector $\mathbf{u}$ that acts as a "dual" secret key, with the public key containing an extra matrix $\mathbf{B}$. The commitment is then defined as such $\mathbf{w} := \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{p} + \mathbf{e}'$. There are two valid responses corresponding to this commitment: the normal response using the original secret key $\mathbf{s}$, and a "dual" response, using the "dual" secret key $\mathbf{u}$.

The extra power that aids the simulator in generating a signature lies in its ability to generate $\mathbf{B}$ in such a way that it "contains" the dual secret $\mathbf{u}$. Afterwards, the simulator can perform the straight-line simulation. Possessing the "dual" secret key, the simulator can produce a valid response, the "dual" one, thereby creating a signature *without* using the original secret key $\mathbf{s}$. Crucially, the technical generation of $\mathbf{B}$ to "contain" the dual secret key $\mathbf{u}$ is computationally indistinguishable from a uniformly random generation of $\mathbf{B}$ under the MLWE assumption.

As is common in signature schemes derived from identification protocols, the security reduction afterwards involves invoking the forking lemma [PS00, BN06] after the adversary generates a forgery. This allows the simulator to extract a valid MSIS solution, thereby establishing the unforgeability of our signature scheme under the MLWE and MSIS assumptions.

2.2 Construction

The formal construction of our two-round threshold signature scheme is presented in Fig. 4. We provide in Table 1 the parameters and short descriptions of them used in the scheme. Our scheme uses three hash functions H_{com}, H_{sig} and G modeled as random oracles in the security proof.

3 Correctness

Theorem 29. *The proposed two round threshold signature in Fig. 4 is correct if* $(W \cdot 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}}) \cdot \sqrt{nk} + e^{1/4} \cdot (W \cdot \sigma_{\mathbf{t}} + \sigma_{\mathbf{w}} \cdot \sqrt{\mathsf{rep} \cdot N} \cdot \sqrt{n} \cdot (\sqrt{k} + \sqrt{\ell})) + \eta \cdot \sigma_{\mathbf{w}} \cdot \sqrt{\mathsf{rep} \cdot N \cdot \ell'} \leq B,\ \sigma_{\mathbf{w}} > \sqrt{\frac{\log(2nk)+\lambda}{\pi}}$, *and assuming* $(q, \nu_{\mathbf{t}}, \nu_{\mathbf{w}})$ *satisfy the conditions in Table 1.*

It is clear that when the signatures are generated honestly, the check $[\![c = c']\!]$ inside the verification algorithm always holds. More concretely, we check that:

$$\mathbf{A}\widetilde{\mathbf{z}} - \mathbf{T}c + \mathbf{B}\widetilde{\mathbf{p}} + \mathbf{h} = \mathbf{x} + \widetilde{\mathbf{w}} - \mathbf{x} = \widetilde{\mathbf{w}}$$

Now, we have yet to bound the L_2-norm of the signature. Due to minimal changes, the same proof idea holds as in prior works [DPKM⁺24, EKT24]. Thus, because of space limitations, we refer the reader to Appendix B. Correctness for the detailed proof.

4 Straight-Line Simulation

In this section, we prove a technical theorem from DualMS [Che23] that will be key inside the security proof, allowing the simulator to simulate "in-order" the signing procedure *without* the secret key.

We will bound the statistical distance between the output distributions of the normal signing oracle and the simulated one. We note that an adversary can query TS.PP We now proceed to describe how the reduction performs straight-line simulation. The intuition is as such:

When the adversary queries H_{com}, the reduction answers with $\mathbf{B} = [\mathbf{b}|\hat{\mathbf{B}}]$, where $\hat{\mathbf{B}}$ is uniformly chosen from $\mathcal{R}_q^{k\times(\ell'-1)}$ and $\mathbf{b} = \mathbf{T} - [\hat{\mathbf{B}}|\mathbf{I}]\mathbf{u}$ with $\mathbf{u} \leftarrow\$\ \mathcal{S}^{\ell'-1}$ with $\mathcal{S}$ the dual key space. Now, the reduction "knows" a dual secret key $\bar{\mathbf{u}} := [1, \mathbf{u}^\intercal]^\intercal$ satisfying $\bar{\mathbf{B}}\bar{\mathbf{u}} = \mathbf{T}$.

In the signing protocol, the reduction computes its commitment as $\mathbf{w}_1 := \bar{\mathbf{A}}\mathbf{z}_1 + \bar{\mathbf{B}}\mathbf{r}$ with $\mathbf{z}_1 \leftarrow\$\ \mathcal{D}_{\mathbf{t}}^{\ell+k}$ and $\mathbf{r} \leftarrow\$\ \mathcal{D}_{\mathcal{S}}^{\ell'+k}$. In the second stage, the reduction generates its individual signature with $\mathbf{p}_1 := c\bar{\mathbf{u}} + \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{r}_b + \mathbf{m}_1^*$.

Theorem 30. *We define two procedures* Trans *and* Sim *corresponding to the normal signing procedure and the dual signing procedure respectively, with specific and public inputs as described in the figure below. Then, with probability* $1 - 2^{-\Omega(N)}$ *over the choices* $\mathbf{A}$ *and* $\hat{\mathbf{B}}$ *uniformly over* $\mathcal{R}_q^{k\times\ell} \times \mathcal{R}_q^{k\times(\ell'-1)}$*, for any* $\mathbf{s} \leftarrow\$\ \mathcal{D}_{\mathbf{t}}^{\ell+k}$, $\mathbf{u} \leftarrow\$\ \mathcal{D}_{\mathcal{S}}^{\ell'-1+k}$*, the following claims hold:*

1. *The distributions of* out_1 *(i.e.,* $\mathbf{w}$*) in* Trans *and* Sim *are identical.*
2. *For any* $c \in \mathcal{C}'$ *and conditioned on any* out_1 *(i.e.,* $\mathbf{w}$*), the distinguishing advantage between the distributions of* out_2 *in* Trans *and* Sim *is negligible.*

Specific Inputs:
```
1 : $\mathbf{s} \leftarrow\$\ \mathcal{D}_{\mathbf{t}}^{\ell+k}$; $\bar{\mathbf{u}} := [1, \mathbf{u}^\intercal]^\intercal$ where $\mathbf{u} \in \mathcal{S}^{\ell'-1}$
2 : $(\beta_b)_{b\in[\mathsf{rep}]} := G(\mathsf{vk}, \mathsf{ctnt})$
```
Public Inputs:
```
1 : $c \in \mathcal{C}$
2 : $\bar{\mathbf{A}} := [\mathbf{A}|\mathbf{I}]$ where $\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$; $\mathbf{T} := \bar{\mathbf{A}}\mathbf{s}$
3 : $\bar{\mathbf{B}} := [\mathbf{b}|\hat{\mathbf{B}}|\mathbf{I}]$ where $\hat{\mathbf{B}} \in \mathcal{R}_q^{k\times(\ell'-1)}$ and $\mathbf{b} := \mathbf{T} - [\hat{\mathbf{B}}|\mathbf{I}]\mathbf{u}$
```
Trans(s)
```
1 : $\widetilde{\mathbf{y}} \leftarrow\$\ \mathcal{D}_s^{\ell+k}$
2 : $\mathbf{p} \leftarrow\$\ \mathcal{D}_{s'}^{\ell'+k}$
3 : $\mathsf{out}_1 := \mathbf{w} := \bar{\mathbf{A}}\mathbf{y} + \bar{\mathbf{B}}\mathbf{p}$
4 : $\mathbf{z} := c\mathbf{s} + \widetilde{\mathbf{y}}$   // $\mathsf{Game}_0$
5 : $\mathbf{z} \leftarrow\$\ \mathcal{D}_s^{\ell+k}$   // $\mathsf{Game}_1$
6 : $\mathsf{out}_2 := (\mathbf{z}, \mathbf{p})$
```
Sim($\bar{\mathbf{u}}$)
```
1 : $\mathbf{z} \leftarrow\$\ \mathcal{D}_s^{\ell+k}$
2 : $\widetilde{\mathbf{r}} \leftarrow\$\ \mathcal{D}_{s'}^{\ell'+k}$
3 : $\mathsf{out}_1 := \mathbf{w} := \bar{\mathbf{A}}\mathbf{z} + \bar{\mathbf{B}}\mathbf{r}$
4 : $\mathbf{p} := c\bar{\mathbf{u}} + \widetilde{\mathbf{r}}$   // $\mathsf{Game}_0$
5 : $\mathbf{p} \leftarrow\$\ \mathcal{D}_{s'}^{\ell'}$   // $\mathsf{Game}_1$
6 : $\mathsf{out}_2 := (\mathbf{z}, \mathbf{p})$
```

Proof. Let $\widetilde{\mathbf{y}} := \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{y}_b$ and $\widetilde{\mathbf{r}} := \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{r}_b$. Since, the $(\beta_b)_b \in [\mathsf{rep}]$ are fixed and the same in both Trans and Sim, we can work over $\widetilde{\mathbf{y}}$ and $\widetilde{\mathbf{r}}$ for simplicity. Since $\widetilde{\mathbf{y}}, \mathbf{p}$ in Trans are identical to $\mathbf{z}, \widetilde{\mathbf{r}}$ in Sim, claim 1 is obvious.

We shall prove claim 2. Firstly, we will examine the distribution of $\mathsf{out}_1 = \mathbf{w}$ in Trans. Split $\mathbf{p}$ into $\mathbf{p} = [p_1, \mathbf{p}_2^\intercal]^\intercal$. Then, we have,

$$\bar{\mathbf{B}}\mathbf{p} = p_1(\mathbf{T} - [\hat{\mathbf{B}}|\mathbf{I}]\mathbf{u}) + [\hat{\mathbf{B}}|\mathbf{I}]\mathbf{p_2}$$

Using lemma 6, a useful regularity result that gives the minimum Gaussian width of $\mathbf{x}$ to make $[\mathbf{A}|\mathbf{I}]\mathbf{x}$ statistically close to the uniform distribution, with probability $1 - 2^{-\Omega(N)}$ over the choice of $\hat{\mathbf{B}}$, we obtain that $[\hat{\mathbf{B}}|\mathbf{I}]\mathbf{p}_2$ is within statistical distance $2^{-\Omega(N)}$ of the uniform distribution. Hence, $\bar{\mathbf{B}}\mathbf{p}$ and

$\mathbf{w}$ are also within distance $2^{-\Omega(N)}$ of the uniform distribution. Similarly, in Sim, $\bar{\mathbf{A}}\mathbf{z}$ and $\mathbf{w}$ are within statistical distance $2^{-\Omega(N)}$ of the uniform distribution.

The Rényi Divergence (RD) is a measure of closeness of any two probability distributions. The RD can be used as an alternative to the statistical distance, which leads to security proofs for schemes with smaller parameters, and sometimes to simpler security proofs than the existing ones [BLR+18]. By performing minimal changes on Theorem 4.1 in [ASY22], we can independently prove that both in Trans and Sim, the distinguishing advantage between Game_0 and Game_1 is negligible. However, we will omit the proof since the technical proof via Rényi Divergence is beyond the scope of the report.

Since the distributions of out_1 in Trans and Sim are identical and the distinguishing advantage between the distributions of out_2 in Trans and Sim is negligible, the distinguishing advantage between $\mathsf{Trans}(\mathbf{s})$ and $\mathsf{Sim}(\bar{\mathbf{u}})$ is also negligible.

5 Unforgeability

Theorem 31. *The two-round threshold signature is unforgeable under the* MLWE*,* MSIS *assumptions and the pseudorandomness of* PRF*.*
For any N and $|\mathsf{SS}| = T$ with $T \leq N$ and an adversary $\mathcal{A}$ against the unforgeability game making at most Q_H, Q_G and Q_S queries to the random oracles H and G, and the signing oracle, respectively, there exists adversaries $\mathcal{B}$, $\mathcal{C}$ $\mathcal{D}$ against the the pseudorandomness of PRF*,* MLWE *problem, and* MSIS *problem such that*

$$\mathsf{Adv}^{\mathsf{TS-uf}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T) \leq N^2 \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}}(1^\lambda) + \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{C}}(1^\lambda) + \mathsf{Adv}^{\mathsf{MSIS}}_{\mathcal{D}}(1^\lambda) + \frac{Q_S^2}{n-1} + \mathsf{negl}(\lambda)$$

To prove unforgeability, we proceed using a series of hybrid games. The aim is to arrive at a hybrid game in which the simulator can extract from the adversary's view a valid MSIS solution. The following first 7 games, from Game_1 to Game_7, are taken almost verbatim from [EKT24] with the appropriate modifications to suit our scheme. Thus, due to space constraints, we refer the reader to Appendix C. Unforgeability for the formally detailed games, their corresponding figures in Appendix D. Games, and here we shall present only the informal intuition.

Proof. Let $\mathcal{A}$ be an adversary against the unforgeability game. When performing its forgery attack, we grant $\mathcal{A}$ the role of the signature aggregator. Moreover, without loss of generality, we assume $\mathcal{A}$ controls $T-1$ participants, and has full power over how these participants behave. We relate the advantage of $\mathcal{A}$ for each adjacent games, where ϵ_i denotes the advantage of $\mathcal{A}$ in Game_i.

Game_1 : This is the real unforgeability game, presented in Fig. 5. By definition, we have

$$\epsilon_1 := \mathsf{Adv}^{\mathsf{TS-uf}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T)$$

Game_2 : In this game the simulator modifies how he maintains the random oracles G and H_{sig}, presented in Fig. 6. As shown in the appendix, we have

$$\epsilon_2 = \epsilon_1$$

Game_3 : In this game the only modifications are syntactical to aid readability, presented in Fig. 7 thus,

$$\epsilon_3 = \epsilon_2$$

Game_4 : In this game, the simulator samples a mask $\mathbf{m}_{i,j} \leftarrow\!\$\ \mathcal{R}_q^\ell$ instead of computing $\mathbf{m}_{i,j} = \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})$ when i, j are honest users, presented in Fig. 7. As shown in the appendix, we have

$$|\epsilon_4 - \epsilon_3| \leq N^2 \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}}(1^\lambda)$$

Game_5 : In this game, the simulator adds an abort condition when queried the signing oracle $\mathcal{O}_{\mathsf{TS.Sign}}$, presented in Fig 8. We bound the probability that an honest user i sings the same ctnt more than once and obtain,

$$|\epsilon_5 - \epsilon_4| \leq \frac{Q_S^2}{2^{n-1}} + \mathsf{negl}(1^\lambda)$$

Game_6 : In this game, the simulator changes how it generates the intermediate masking terms of the honest users $(\mathbf{m}_{i,\mathsf{sHS}}, \mathbf{m}^*_{\mathsf{sHS},i})$, presented in Fig. 8 and prove that the view of the two games are identically distributed to the adversary $\mathcal{A}$ to conclude that,

$$\epsilon_6 = \epsilon_5$$

Game_7 : In this game, the simulator modifies how it computes the responses $\mathbf{z}_i$, presented in Fig. 9. When the adversary $\mathcal{A}$ queries the signing oracle on user $i \in \mathsf{sHS}$, the simulator constructs $\mathsf{ctnt} := [\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_j})_{j \in \mathsf{SS}}]$ and checks whether user i is the last user in sHS to sign with ctnt. If not, the simulator removes the partial secret key $\mathsf{sk}_i = \mathbf{s}_i$ from the response $\mathbf{z}_i$. Otherwise, if user i is the last user, then it uses $\mathbf{s} - \sum_{j \in \mathsf{sCS}} L_{\mathsf{SS},j} \cdot \mathbf{s}_j$ in place of the partial secret key $\mathbf{s}_i$ to generate the response $\mathbf{z}_i$.
In the appendix we show that view of $\mathcal{A}$ remains identical to the previous game and obtain,

$$\epsilon_7 = \epsilon_6$$

Game_8 : In this game, we generate secret shares for the corrupted signers $(\mathbf{s}_i)_{i \in \mathsf{CS}}$ by uniformly sampling $\mathbf{s}_i \leftarrow\!\$\; \mathcal{R}_q^\ell$, presented in Fig. 10. Since $\mathbf{s}_i$ of the linear Shamir secret sharing scheme is uniformly distributed over $\mathcal{R}_q^\ell$, the view of $\mathcal{A}$ remains identical to the previous game.
Moreover, the verification key $\mathsf{vk} = (\mathbf{A}, \mathbf{A}\mathbf{s}{+}\mathbf{e})$ is replaced by $(\mathbf{A}, \widetilde{\mathbf{T}})$ where $\widetilde{\mathbf{T}}$ is sampled uniformly at random from $\mathcal{R}_q^k$. We now show that we can construct an adversary $\mathcal{C}$ against the MLWE problem such that

$$|\epsilon_8 - \epsilon_7| \leq \mathsf{Adv}_{\mathcal{C}}^{\mathsf{MLWE}}(1^\lambda)$$

Proof. Let us provide the description of $\mathcal{C}$. $\mathcal{C}$ is given $(\mathbf{A}, \mathbf{d})$ as the MLWE problem, where $\mathbf{d}$ is either $\mathbf{A}\mathbf{s} + \mathbf{e}$ or random over $\mathcal{R}_q^\ell$. It simulates the view of the Game_7 simulator to $\mathcal{A}$ by

- Setting $\widetilde{\mathbf{T}} := \mathbf{d}$
- Sampling secret shares $\mathbf{s}_i \leftarrow\!\$\; \mathcal{R}_q^\ell$ for $i \in \mathsf{CS}$
- Sampling $(\mathsf{vk}_{\mathsf{sig}}, \mathsf{sk}_{\mathsf{sig}}) \leftarrow \mathsf{TS.KeyGen}(1^\lambda)$ for $i \in [N]$
- Sampling $\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i} \leftarrow\!\$\; \{0,1\}^\lambda$ for $i \in \mathsf{CS}, j \in [N]$
- 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 $i \in \mathsf{CS}$

and running $\mathcal{A}^{\mathcal{O}_{\mathsf{TS.PP}}, \mathcal{O}_{\mathsf{TS.Sign}}, H, G}(\mathsf{vk}, (\mathsf{sk}_i)_{i \in \mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$. All random oracle queries are simulated identically to the simulator.

Game_9 : Here, the simulator modifies how he maintains the random oracle H_{com}, presented in Fig. 11. Our goal is to have an MLWE instance in $\mathbf{B}$. Right now, in any fresh query, H_{com} returns a randomly chosen $\mathbf{B} \leftarrow\!\$\; \mathcal{R}_q^{k \times \ell'}$.
We have a randomly chosen $\hat{\mathbf{B}} \leftarrow\!\$\; \mathcal{R}_q^{k \times (\ell'-1)}$ and set $\mathbf{v} \leftarrow\!\$\; \mathcal{R}_q^k$. We set $\mathbf{b} = \widetilde{\mathbf{T}} - \mathbf{v}$ and the the oracle returns $\mathbf{B} := [\mathbf{b} | \hat{\mathbf{B}}]$. Since the time on which H_{com} is set cannot be detected by $\mathcal{A}$, the two games are identical. Thus, we have

$$\epsilon_9 = \epsilon_8$$

Game_{10} : Now, instead of $\mathbf{v}$ being uniformly chosen, we set $\mathbf{v} := \hat{\mathbf{B}}\mathbf{u}$ with a short $\mathbf{u}$ uniformly chosen from the dual key space $\mathcal{S}^{\ell'-1}$, presented in Fig. 11. We now can observe that $\hat{\mathbf{B}}$ and $\mathbf{v}$ define an MLWE instance. Setting now $\mathbf{B} := [\mathbf{b} | \hat{\mathbf{B}}]$, we obtain that

$$|\epsilon_{10} - \epsilon_9| \leq \mathsf{Adv}_{\mathcal{C}}^{\mathsf{MLWE}}(1^\lambda)$$

Game_{11} : In this final game, the simulator is finally ready to simulate the signing procedure without the secret key, presented in Fig. 12. Notice that now it knows a dual secret key $\bar{\mathbf{u}} := [1, \mathbf{u}^\intercal]^\intercal$ satisfying $\bar{\mathbf{B}}\bar{\mathbf{u}} = \mathbf{T}$ and is able to generate its individual signature.
Using Theorem 30, we show that the distinguishing advantage between the output distributions of "normal" signing procedure and the dual signing simulation is negligible. Thus, we obtain

$$|\epsilon_{11} - \epsilon_{10}| \leq \mathsf{negl}(\lambda)$$

Theorem 32. *There exists an adversary $\mathcal{D}$ against the* MSIS *problem such that*

$$\epsilon_{11} \leq \mathsf{Adv}_{\mathcal{D}}^{\mathsf{MSIS}}(1^\lambda)$$

with time $\mathsf{Time}(\mathcal{D}) \approx \mathsf{Time}(\mathcal{A})$

Now that the public key is a random public vector and the simulator no longer requires the signing key for the unforgeability game, we can invoke the MSIS problem.

Proof. Let $\mathcal{D}$ the forking algorithm that runs $\mathcal{A}$ twice. If adversary $\mathcal{A}$ wins in the first time, $\mathcal{D}$ can obtain $\widetilde{\mathbf{T}}, \mu, \widetilde{\mathbf{w}}, \widetilde{\mathbf{z}}$ and $\widetilde{\mathbf{p}}$ satisfying

$$\mathbf{A}\widetilde{\mathbf{z}} + \mathbf{B}\widetilde{\mathbf{p}} - c\widetilde{\mathbf{T}} = \widetilde{\mathbf{w}}$$

where $c = H_{\mathsf{sig}}(\widetilde{\mathbf{T}}, \mu, \widetilde{\mathbf{w}})$ is what we call "crucial" query corresponding to the forgery. Algorithm $\mathcal{D}$ then forks the adversary at the crucial query, assigning another hash value c' to $H_{\mathsf{sig}}(\widetilde{\mathbf{T}}, \mu, \widetilde{\mathbf{w}})$ in the second execution. In this execution, with probability lower-bounded by the forking lemma [PS00, BN06], we have $c \neq c'$, $\mathcal{A}$ wins, and $H_{\mathsf{sig}}(\widetilde{\mathbf{T}}, \mu, \widetilde{\mathbf{w}})$ is again the crucial query. In this case, $\mathcal{D}$ obtains another tuple of responses $\widetilde{\mathbf{z}}'$ and $\widetilde{\mathbf{p}}'$ satisfying

$$\mathbf{A}\widetilde{\mathbf{z}}' + \mathbf{B}\widetilde{\mathbf{p}}' - c'\widetilde{\mathbf{T}} = \widetilde{\mathbf{w}}$$

Combining the two equations, we obtain

$$\mathbf{A}\hat{\mathbf{z}} + \mathbf{B}\hat{\mathbf{p}} - \hat{c}\widetilde{\mathbf{T}} = 0$$

where $\hat{\mathbf{z}} = \widetilde{\mathbf{z}} - \widetilde{\mathbf{z}}',\ \hat{\mathbf{p}} = \widetilde{\mathbf{p}} - \widetilde{\mathbf{p}}'$ and $\hat{c} = c - c' \neq 0$

Decomposing $\mathbf{B} = [\mathbf{b}|\hat{\mathbf{B}}]$ and since $\mathbf{b} = \widetilde{\mathbf{T}} - \hat{\mathbf{B}}\mathbf{u}$, with $\mathbf{v} := \hat{\mathbf{B}}\mathbf{u}$ defining an MLWE instance, we have that $\mathcal{D}$ obtains an MSIS solution since

$$\mathbf{A}\hat{\mathbf{z}} + (\widetilde{\mathbf{T}} - \hat{\mathbf{B}}\mathbf{u})\hat{\mathbf{p}} - \hat{c}\widetilde{\mathbf{T}} = \mathbf{A}\hat{\mathbf{z}} + \widetilde{\mathbf{T}}(\hat{\mathbf{p}} - \hat{c}) - \mathbf{v}\hat{\mathbf{p}} = 0$$

and therefore

$$\left[\mathbf{A} \,\middle|\, \widetilde{\mathbf{T}} \,\middle|\, \mathbf{v}\right] \left[\begin{array}{c} \hat{\mathbf{z}} \\ \hline \hat{\mathbf{p}} - \hat{c} \\ \hline \hat{\mathbf{p}} \end{array}\right] = 0$$

with $\hat{\mathbf{z}}$ and $\hat{\mathbf{p}}$ small, $\widetilde{\mathbf{T}}$ and $\mathbf{v}$ both indistinguishable from random by MLWE and $\mathbf{A}$ randomly sampled.

References

ABV+12. S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *PKC 2012*, *LNCS* 7293, pages 280–297. Springer, Heidelberg, May 2012.

ADRS15. D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete Gaussian sampling: Extended abstract. In *47th ACM STOC*, pages 733–742. ACM Press, June 2015.

Ajt96. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

AKS01. M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd ACM STOC*, pages 601–610. ACM Press, July 2001.

ASY22. S. Agrawal, D. Stehlé, and A. Yadav. Round-optimal lattice-based threshold signatures, revisited. LIPIcs, pages 8:1–8:20. Schloss Dagstuhl, 2022.

BGG+18. D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO 2018, Part I*, *LNCS* 10991, pages 565–596. Springer, Heidelberg, August 2018.

BGV11. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. `https://eprint.iacr.org/2011/277`.

BKL+24. C. Boschini, D. Kaviani, R. W. F. Lai, G. Malavolta, A. Takahashi, and M. Tibouchi. Ringtail: Practical two-round threshold signatures from learning with errors. Cryptology ePrint Archive, Paper 2024/1113, 2024. `https://eprint.iacr.org/2024/1113`.

BKP13. R. Bendlin, S. Krehbiel, and C. Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In *ACNS 13*, *LNCS* 7954, pages 218–236. Springer, Heidelberg, June 2013.

BLR+18. S. Bai, T. Lepoint, A. Roux-Langlois, A. Sakzad, D. Stehlé, and R. 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.

BN06. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

BP23. L. Brandão and R. Peralta. Nistir 8214c (ipd): Nist first call for multi-party threshold schemes (initial public draft), 01 2023.

BR93. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

BS99. J. Blömer and J.-P. Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *31st ACM STOC*, pages 711–720. ACM Press, May 1999.

BTT22. C. Boschini, A. Takahashi, and M. Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. LNCS, pages 276–305. Springer, Heidelberg, 2022.

CCJ$^+$16. L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. A. Perlner, and D. Smith-Tone. *Report on post-quantum cryptography.* US Department of Commerce, National Institute of Standards and Technology . . . , 2016.

CCK23. J. H. Cheon, W. Cho, and J. Kim. Improved universal thresholdizer from iterative shamir secret sharing. Cryptology ePrint Archive, Paper 2023/545, 2023. `https://eprint.iacr.org/2023/545`.

CGH98. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. Cryptology ePrint Archive, Report 1998/011, 1998. `https://eprint.iacr.org/1998/011`.

Che23. Y. Chen. Dualms: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In *Annual International Cryptology Conference*, pages 716–747. Springer, 2023.

CKM23. E. C. Crites, C. Komlo, and M. Maller. Fully adaptive Schnorr threshold signatures. LNCS, pages 678–709. Springer, Heidelberg, 2023.

CS19. D. Cozzo and N. P. Smart. Sharing the LUOV: Threshold post-quantum signatures. In *17th IMA International Conference on Cryptography and Coding*, *LNCS* 11929, pages 128–153. Springer, Heidelberg, December 2019.

DEF$^+$19. M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven, and I. Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1084–1101. IEEE, 2019.

Des90. Y. Desmedt. Abuses in cryptography and how to fight them. In *CRYPTO'88*, *LNCS* 403, pages 375–389. Springer, Heidelberg, August 1990.

DF90. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO'89*, *LNCS* 435, pages 307–315. Springer, Heidelberg, August 1990.

dPEK$^+$23. R. del Pino, T. Espitau, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, M. Rossi, and M.-J. 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.

DPKM$^+$24. R. Del Pino, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, and M.-J. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 219–248. Springer, 2024.

DT06. I. Damgård and R. Thorbek. Linear integer secret sharing and distributed exponentiation. In *PKC 2006*, *LNCS* 3958, pages 75–90. Springer, Heidelberg, April 2006.

EKT24. T. Espitau, S. Katsumata, and K. Takemure. Two-round threshold signature from algebraic one-more learning with errors. Cryptology ePrint Archive, Paper 2024/496, 2024. `https://eprint.iacr.org/2024/496`.

FHK$^+$18. P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, Z. Zhang, et al. Falcon: Fast-fourier lattice-based compact signatures over ntru. *Submission to the NIST's post-quantum cryptography standardization process*, 36(5):1–75, 2018.

FKL18. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *CRYPTO 2018, Part II*, *LNCS* 10992, pages 33–62. Springer, Heidelberg, August 2018.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, *LNCS* 263, pages 186–194. Springer, Heidelberg, August 1987.

GKPV10. S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. 2010.

GMPW20. N. Genise, D. Micciancio, C. Peikert, and M. Walter. Improved discrete gaussian and subgaussian analysis for lattice cryptography. In *PKC 2020, Part I*, *LNCS* 12110, pages 623–651. Springer, Heidelberg, May 2020.

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

Kan83. R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 193–206, 1983.

KG20. C. Komlo and I. Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In *SAC 2020*, LNCS, pages 34–65. Springer, Heidelberg, 2020.

Kia. A. Kiayias. Cryptography, primitives and protocols. Lecture Notes.

LLL82. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261:515–534, 1982.

LM06. V. Lyubashevsky and D. Micciancio. Generalized compact Knapsacks are collision resistant. In *ICALP 2006, Part II*, *LNCS* 4052, pages 144–155. Springer, Heidelberg, July 2006.

LPR10. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, *LNCS* 6110, pages 1–23. Springer, Heidelberg, May / June 2010.

LPR13. V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT 2013*, *LNCS* 7881, pages 35–54. Springer, Heidelberg, May 2013.

LS15. A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. 75(3):565–599, 2015.

LST18. B. Libert, D. Stehlé, and R. Titiu. Adaptively secure distributed PRFs from LWE. In *TCC 2018, Part II*, *LNCS* 11240, pages 391–421. Springer, Heidelberg, November 2018.

Lyu09. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, *LNCS* 5912, pages 598–616. Springer, Heidelberg, December 2009.

Lyu12. V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT 2012*, *LNCS* 7237, pages 738–755. Springer, Heidelberg, April 2012.

Mic02. D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *43rd FOCS*, pages 356–365. IEEE Computer Society Press, November 2002.

Mic07. D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16:365–411, 2007. Full Version.

MP12. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, *LNCS* 7237, pages 700–718. Springer, Heidelberg, April 2012.

MR04. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

MR09. D. Micciancio and O. Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

MV10. D. Micciancio and P. Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *42nd ACM STOC*, pages 351–358. ACM Press, June 2010.

NRS21. J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple two-round Schnorr multi-signatures. LNCS, pages 189–221. Springer, Heidelberg, 2021.

Pei16. C. Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, 2016.

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

Reg05. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

Reg09. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009. Full Version.

Sch90. C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, *LNCS* 435, pages 239–252. Springer, Heidelberg, August 1990.

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

Sha79. A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

SSTX09. D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT 2009*, *LNCS* 5912, pages 617–635. Springer, Heidelberg, December 2009.

Val84. L. G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984.

Appendix A. Two-Round threshold signature scheme

```
TS.Setup($1^\lambda, N, T$)
 1 : $\mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k\times\ell}$
 2 : $\mathsf{tspar} := (\mathbf{A}, N, T)$
 3 : return $\mathsf{tspar}$

TS.KeyGen($\mathsf{tspar}$)
 1 : parse $(\mathbf{A}, N, T) \leftarrow \mathsf{tspar}$
 2 : $(\mathbf{s}, \mathbf{e}) \leftarrow\!\!\$\ \mathcal{D}_{\mathbf{t}}^{\ell} \times \mathcal{D}_{\mathbf{t}}^{k}$
 3 : $\mathbf{T} := \lfloor \mathbf{A}\mathbf{s} + \mathbf{e} \rceil_{\nu_{\mathbf{t}}} \in \mathcal{R}_{q_{\nu_{\mathbf{t}}}}^{k}$
 4 : for $(i, j) \in [N] \times [N]$ do
 5 :     $\mathsf{seed}_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
 6 : $\overrightarrow{P} \leftarrow\!\!\$\ \mathcal{R}_q^{\ell}[X]$ with $deg(\overrightarrow{P}) = T - 1,\ \overrightarrow{P}(0) = \mathbf{s}$
 7 : $(\mathbf{s}_i)_{i\in[N]} := (\overrightarrow{P}(i))_{i\in[N]}$
 8 : $\mathbf{B} \leftarrow H_{\mathsf{com}}(\mathbf{A}, \mathbf{T}) \in \mathcal{R}_q^{k\times\ell'}$
 9 : $\mathsf{tspar} := (\mathbf{A}, \mathbf{B}, N, T)$
10 : $\mathsf{vk} := (\mathsf{tspar}, \mathbf{T})$
11 : $(\mathsf{sk}_i)_{i\in[N]} := \big((\mathbf{s}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[N]}\big)_{i\in[N]}$
12 : return $(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]})$

TS.Agg($\mathsf{vk}, \mathsf{SS}, M, (\widehat{\mathsf{sig}}_j)_{j\in\mathsf{SS}}$)
 1 : parse $(\mathsf{tspar}, \mathbf{T}) \leftarrow \mathsf{vk}$
 2 : parse $(\mathbf{A}, \mathbf{B}, N, T) \leftarrow \mathsf{tspar}$
 3 : parse $(\mathbf{w}_j, \mathbf{m}_j, \mathbf{z}_j, \mathbf{p}_j)_{j\in\mathsf{SS}} \leftarrow (\widehat{\mathsf{sig}}_j)_{j\in\mathsf{SS}}$
 4 : $\widetilde{\mathbf{w}} := \left\lfloor \sum_{j\in\mathsf{SS}} \mathbf{w}_j \right\rceil_{\nu_{\mathbf{w}}}$
 5 : $\widetilde{\mathbf{p}} := \sum_{j\in\mathsf{SS}} \mathbf{p}_j$
 6 : $\widetilde{\mathbf{z}} := \sum_{j\in\mathsf{SS}} (\mathbf{z}_j - \mathbf{m}_j)$
 7 : $c := H_{\mathsf{sig}}(\mathsf{vk}, M, \widetilde{\mathbf{w}})$
 8 : $\mathbf{x} := \lfloor \mathbf{A}\widetilde{\mathbf{z}} - 2^{\nu_{\mathbf{t}}}\mathbf{T}c + \mathbf{B}\widetilde{\mathbf{p}} \rceil_{\nu_{\mathbf{w}}} \in \mathcal{R}_{q_{\nu_{\mathbf{w}}}}^{k}$
 9 : $\mathbf{h} := \widetilde{\mathbf{w}} - \mathbf{x} \in \mathcal{R}_{q_{\nu_{\mathbf{w}}}}^{k}$
10 : return $\mathsf{sig} := (c, \widetilde{\mathbf{z}}, \mathbf{h}, \widetilde{\mathbf{p}})$

TS.Verify($\mathsf{vk}, M, \mathsf{sig}$)
 1 : parse $(c, \widetilde{\mathbf{z}}, \mathbf{h}, \widetilde{\mathbf{p}}) \leftarrow \mathsf{sig}$
 2 : $c' := H_{\mathsf{sig}}(\mathsf{vk}, M, \lfloor \mathbf{A}\widetilde{\mathbf{z}} - 2^{\nu_{\mathbf{t}}}\mathbf{T}c + \mathbf{B}\widetilde{\mathbf{p}} \rceil_{\nu_{\mathbf{w}}} + \mathbf{h})$
 3 : if $\llbracket c = c' \rrbracket \wedge \llbracket \|(\widetilde{\mathbf{z}}, 2^{\nu_{\mathbf{w}}} \cdot \mathbf{h}, \widetilde{\mathbf{p}})\|_2 \leq B \rrbracket$ then
 4 :     return 1
 5 : return 0
```

```
TS.PP($\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i$)
 1 : parse $(\mathsf{tspar}, \mathbf{T}) \leftarrow \mathsf{vk}$
 2 : parse $(\mathbf{A}, \mathbf{B}, N, T) \leftarrow \mathsf{tspar}$
 3 : for $b \in [\mathsf{rep}]$ do
 4 :     $(\mathbf{y}_{i,b}, \mathbf{p}_{i,b}, \mathbf{e}'_{i,b}) \leftarrow\!\!\$\ \mathcal{D}_{\mathbf{w}}^{\ell} \times \mathcal{D}_{\mathbf{w}}^{\ell'} \times \mathcal{D}_{\mathbf{w}}^{k}$
 5 :     $\mathbf{w}_{i,b} := \mathbf{A}\mathbf{y}_{i,b} + \mathbf{B}\mathbf{p}_{i,b} + \mathbf{e}'_{i,b} \in \mathcal{R}_q^{k}$
 6 : $\overrightarrow{\mathbf{w}_i} := [\mathbf{w}_{i,1}, \cdots, \mathbf{w}_{i,\mathsf{rep}}]$
 7 : $\mathsf{pp}_i := \overrightarrow{\mathbf{w}_i}$
 8 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \cup \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b\in[\mathsf{rep}]})\}$
 9 : return $\mathsf{pp}_i$

TS.Sign($\mathsf{vk}, \mathsf{SS}, M, i, (\mathsf{pp}_j)_{j\in\mathsf{SS}}, \mathsf{sk}_i, \mathsf{st}_i$)
 1 : parse $(\mathbf{s}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[N])} \leftarrow \mathsf{sk}_i$
 2 : assert $\llbracket \mathsf{SS} \subseteq [N] \rrbracket \wedge \llbracket i \in \mathsf{SS} \rrbracket \wedge \llbracket (\mathsf{pp}_i, \cdot) \in \mathsf{st}_i \rrbracket$
 3 : parse $(\overrightarrow{\mathbf{w}_j})_{j\in\mathsf{SS}\setminus\{i\}} \leftarrow (\mathsf{pp}_j)_{j\in\mathsf{SS}}$
 4 : pick $(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b\in[\mathsf{rep}]})$ from $\mathsf{st}_i$ with $\mathsf{pp}_i = \overrightarrow{\mathbf{w}_i}$
 5 : $\mathsf{ctnt} := [\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_j})_{j\in\mathsf{SS}}]$
 6 : $(\beta_b)_{b\in[\mathsf{rep}]} := G(\mathsf{vk}, \mathsf{ctnt})$   // $\beta_1 = 1,\ \beta_b \in \mathbb{T}$
 7 : for $j \in \mathsf{SS}$ do
 8 :     parse $[\mathbf{w}_{j,1} \mid \cdots \mid \mathbf{w}_{j,\mathsf{rep}}] \leftarrow \overrightarrow{\mathbf{w}_j}$
 9 :     $\mathbf{w}_j := \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{w}_{j,b}$
10 : $\widetilde{\mathbf{w}} := \left\lfloor \sum_{j\in\mathsf{SS}} \mathbf{w}_j \right\rceil_{\nu_{\mathbf{w}}} \in \mathcal{R}_{q_{\nu_{\mathbf{w}}}}^{k}$
11 : $c := H_{\mathsf{sig}}(\mathsf{vk}, M, \widetilde{\mathbf{w}}) \in \mathcal{C}$
12 : $\mathbf{m}_i := \sum_{j\in\mathsf{SS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})) \in \mathcal{R}_q^{\ell}$
13 : $\mathbf{m}_i^* := \sum_{j\in\mathsf{SS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ctnt})) \in \mathcal{R}_q^{\ell}$
14 : $\mathbf{p}_i := \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{p}_{i,b}$
15 : $\mathbf{z}_i := c \cdot L_{\mathsf{SS},i} \cdot \mathbf{s}_i + \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{y}_{i,b} + \mathbf{m}_i^* \in \mathcal{R}_q^{\ell}$
16 : $\widehat{\mathsf{sig}}_i := (\mathbf{w}_i, \mathbf{m}_i, \mathbf{z}_i, \mathbf{p}_i)$
17 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \setminus \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b\in[\mathsf{rep}]})\}$
18 : return $(\widehat{\mathsf{sig}}_i, \mathsf{st}_i)$
```

Fig. 4. Our two-round offline-online threshold signature scheme.

Parameter	Description
λ	Security parameter
N	Number of parties
T	Threshold, i.e. Number of parties needed to sign
SS	Number of parties that are actively participating in the signature
q	Ring modulus
$\mathcal{R}_q$	Polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n+1)$
(k,ℓ)	Dimension of the public matrix $\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$
(k,ℓ')	Dimension of the matrix $\mathbf{B} \in \mathcal{R}_q^{k\times\ell'}$
$(\mathcal{D}_{\mathbf{t}},\sigma_{\mathbf{t}})$	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 the aggregated commitment
$(q_{\nu_{\mathbf{t}}},q_{\nu_{\mathbf{w}}})$	Rounded moduli satisfying $(q_{\nu_{\mathbf{t}}},q_{\nu_{\mathbf{w}}}) := (\lfloor q/2^{\nu_{\mathbf{t}}}\rfloor, \lfloor q/2^{\nu_{\mathbf{w}}}\rfloor) = (\lfloor q/2^{\nu_{\mathbf{t}}}\rceil, \lfloor q/2^{\nu_{\mathbf{w}}}\rceil)$
$\mathbb{T}\subset\mathcal{R}_q$	Set of all signed monomials, i.e., $\{(-1)^b\cdot X^i \mid (b,i)\in\{0,1\}\times[n]\}$
rep	An integer s.t. $\lvert\mathbb{T}^{\mathsf{rep}-1}\rvert \geq 2^\lambda$
$(\mathcal{C}\subset\mathcal{R}_q, W)$	Challenge set $\{c\in\mathcal{R}_q \mid \lVert c\rVert_\infty = 1 \wedge \lVert c\rVert_1 = W\}$ s.t. $\lvert\mathcal{C}\rvert\geq 2^\lambda$
$\mathcal{S}$	Dual key set
B	Two-norm bound on the signature
Q_S	Maximum number of signing queries

Table 1. Overview of the parameters used in the two round threshold signature scheme

Appendix B. Correctness

Fix any $(N,T,\mathsf{SS}\subset[N])$ such that $\lvert\mathsf{SS}\rvert = T$. The aggregated response satisfies:

$$\begin{aligned}
\widetilde{\mathbf{z}} &= \sum_{j\in\mathsf{SS}} (\mathbf{z}_j - \mathbf{m}_j) \\
&= \sum_{j\in\mathsf{SS}} \left(c\cdot L_{\mathsf{SS},j}\cdot\mathbf{s}_j + \sum_{b\in[\mathsf{rep}]}\beta_b\cdot\mathbf{y}_{j,b} + \mathbf{m}_j^* - \mathbf{m}_j \right) \\
&= c\cdot\mathbf{s} + \sum_{j\in\mathsf{SS}}\sum_{b\in[\mathsf{rep}]}\beta_b\cdot\mathbf{y}_{j,b} \in \mathcal{R}_q^\ell
\end{aligned}$$

where the last equality follows from the correctness of the linear Shamir secret sharing scheme as well as the observation that $\sum_{j\in\mathsf{SS}}\mathbf{m}_j^* = \sum_{j\in\mathsf{SS}}\mathbf{m}_j$.

Moreover, it also holds that:

$$\begin{aligned}
\mathbf{x} &= \lfloor \mathbf{A}\mathbf{z} - 2^{\nu_{\mathbf{t}}}\cdot\bar{\mathbf{T}}c + \mathbf{B}\widetilde{\mathbf{p}} \rceil_{\nu_{\mathbf{w}}} \\
&= \left\lfloor c\cdot\mathbf{A}\mathbf{s} + \sum_{j\in\mathsf{SS}}\sum_{b\in[\mathsf{rep}]}\beta_b\cdot\mathbf{A}\mathbf{y}_{j,b} - 2^{\nu_{\mathbf{t}}}\cdot\mathbf{T}c + \sum_{j\in\mathsf{SS}}\sum_{b\in[\mathsf{rep}]}\beta_b\cdot\mathbf{B}\mathbf{p}_{j,b} \right\rceil_{\nu_{\mathbf{w}}} \\
&= \left\lfloor c\cdot(\bar{\mathbf{T}} - \mathbf{e}) + \sum_{j\in\mathsf{SS}}\left(\mathbf{w}_j - \sum_{b\in[\mathsf{rep}]}\beta_b\cdot\mathbf{e}'_{j,b}\right) - 2^{\nu_{\mathbf{t}}}\cdot\mathbf{T}c \right\rceil_{\nu_{\mathbf{w}}} \\
&= \left\lfloor \bar{\mathbf{w}} + c\cdot\underbrace{\left(\bar{\mathbf{T}} - 2^{\nu_{\mathbf{t}}}\cdot\lfloor\bar{\mathbf{T}}\rceil_{\nu_{\mathbf{t}}}\right)}_{=:\boldsymbol{\alpha}_{\mathbf{t}}\in\mathcal{R}_q^k} - \underbrace{\left(c\cdot\mathbf{e} + \sum_{j\in\mathsf{SS}}\sum_{b\in[\mathsf{rep}]}\beta_b\cdot\mathbf{e}'_{j,b}\right)}_{=:\boldsymbol{\alpha}\in\mathcal{R}_q^k} \right\rceil_{\nu_{\mathbf{w}}}
\end{aligned}$$

where $\bar{\mathbf{T}} := \mathbf{A}\mathbf{s} + \mathbf{e}\in\mathcal{R}_q^k$, $\bar{\mathbf{w}} := \sum_{j\in\mathsf{SS}}\mathbf{w}_j\in\mathcal{R}_q^k$ and $\widetilde{\mathbf{w}} = \lfloor\bar{\mathbf{w}}\rceil_{\nu_{\mathbf{w}}}\in\mathcal{R}_{q_{\nu_{\mathbf{w}}}}^k$.

Replacing $\mathbf{x}$ calculated above and using Lemma 22, Eq. (2), we have:

$$\begin{aligned}\left\|2^{\nu_{\mathbf{w}}} \cdot \bar{\mathbf{h}} \mod q\right\|_2 &= \left\|2^{\nu_{\mathbf{w}}} \cdot \overline{\widetilde{\mathbf{w}} - \mathbf{x}} \mod q\right\|_2 \\ &= \left\|2^{\nu_{\mathbf{w}}} \cdot \overline{\lfloor \bar{\mathbf{w}} \rceil_{\nu_{\mathbf{w}}} - \lfloor \bar{\mathbf{w}} + c \cdot \boldsymbol{\alpha}_{\mathbf{t}} - \boldsymbol{\alpha} \rceil_{\nu_{\mathbf{w}}}} \mod q\right\|_2 \\ &\leq \left\|-c \cdot \boldsymbol{\alpha}_{\mathbf{t}} + \boldsymbol{\alpha} \mod q\right\|_2 + \sqrt{nk} \cdot 2^{\nu_{\mathbf{w}}}\end{aligned}$$

Using the triangular inequality, we obtain $\|-c \cdot \boldsymbol{\alpha}_{\mathbf{t}} + \boldsymbol{\alpha} \mod q\|_2 \leq \|c \cdot \boldsymbol{\alpha}_{\mathbf{t}} \mod q\|_2 + \|\boldsymbol{\alpha} \mod q\|_2$. Using the Minkowski inequality and Lemma 22, Eq. (1), we have $\|c \cdot \boldsymbol{\alpha}_{\mathbf{t}} \mod q\|_2 \leq W \cdot \sqrt{nk} \cdot (2^{\nu_{\mathbf{t}}} - 1)$, where $W = \|c\|_1$ since $c \in \mathcal{C}$. Moreover, we have $\|\boldsymbol{\alpha} \mod q\|_2 \leq e^{1/4} \cdot (W \cdot \sigma_{\mathbf{t}} + \sigma_{\mathbf{w}} \cdot \sqrt{\mathsf{rep} \cdot |\mathsf{SS}|}) \cdot \sqrt{nk}$ with overwhelming probability from Lemma 7 and Lemma 8, where note that we use the fact $\beta_b \in \mathbb{T}$ for the latter. As before, we can also obtain $\|\widetilde{\mathbf{z}} \mod q\|_2 \leq e^{1/4} \cdot (W \cdot \sigma_{\mathbf{t}} + \sigma_{\mathbf{w}} \cdot \sqrt{\mathsf{rep} \cdot |\mathsf{SS}|}) \cdot \sqrt{n\ell}$. The only difference with [EKT24] is that we have to also bound $\|\widetilde{p} \mod q\|_2$.

We can do such using lemma 8 and 9 to obtain for $\eta > 1$, $\|\widetilde{\mathbf{p}} \mod q\|_2 \leq \eta \cdot \sigma_{\mathbf{w}} \cdot \sqrt{\mathsf{rep} \cdot |\mathsf{SS}|} \cdot \sqrt{\ell'}$ with overwhelming probability.

Combining the bounds presented above results in the desired bound.

Appendix C. Unforgeability

In this appendix, we present detailed the first seven games for our game-based proof in order to prove unforgeability under the MLWE and MSIS assumptions.

Game_1 : This is the real unforgeability game. By definition, we have

$$\epsilon_1 := \mathsf{Adv}^{\mathsf{TS-uf}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T)$$

Game_2 : In this game, the simulator modifies how he maintains the random oracles G and H_{sig}. In detail, when G is queried on a pair $(\mathsf{vk}, \mathsf{ctnt})$ such that ctnt correctly parses as $[\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_j})_{j\in\mathsf{SS}}]$, it computes the aggregated commitment $\widetilde{\mathbf{w}}$ and sets $H_{\mathsf{sig}}(\mathsf{vk}, M, \widetilde{\mathbf{w}}) \leftarrow c$ for a random $c \leftarrow\!\!\$\ \mathcal{C}$ if it hasn't been set yet. Since the time on which H_{sig} is set cannot be detected by $\mathcal{A}$, the two games are identical. Thus, we have

$$\epsilon_2 = \epsilon_1$$

This step prevents malicious parties from adaptively influencing inputs to the next random oracle G, deriving "joint challenge" $c \in \mathcal{C}$ that all parties must agree on. Due to the Drijvers et al. attack [DEF+19], each signer must ensure that their secret nonce changes unpredictably whenever $c = H_{\mathsf{sig}}(vk, M, \widetilde{\mathbf{w}})$ changes.

Game_3 : In this game we divide the signer set SS into the set of honest users $\mathsf{sHS} := \mathsf{SS}\cap\mathsf{HS}$ and corrupt users $\mathsf{sCS} := \mathsf{SS}\cap\mathsf{CS}$, and define intermediate masking terms $(\mathbf{m}_{i,\mathsf{sHS}}, \mathbf{m}^*_{\mathsf{sHS},i}, \mathbf{m}_{i,\mathsf{sCS}}, \mathbf{m}^*_{\mathsf{sCS},i})$. Since the only modifications were syntactical to aid readability, the advantage remains the same and we have

$$\epsilon_3 = \epsilon_2$$

Game_4 : In this game, the simulator samples a mask $\mathbf{m}_{i,j} \leftarrow\!\!\$\ \mathcal{R}_q^\ell$ instead of computing $\mathbf{m}_{i,j} = \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})$ when $i, j \in \mathsf{HS}$. The simulator prepares an empty list $\mathsf{Rand}[\cdot]$ at the beginning of the game and assigns $\mathsf{Rand}[\mathsf{ctnt}, \mathsf{i}, \mathsf{j}]$ random masks. Since $\mathsf{seed}_{i,j}$ is never revealed to the adversary $\mathcal{A}$ when $i, j \in \mathsf{HS}$, we can go over at most $(N-1)^2$ hybrids using the pseudorandomness of the PRF to establish indistinguishability of the two games. That is that there exists an adversary $\mathcal{B}$ against the pseudorandomness of PRF such that

$$|\epsilon_4 - \epsilon_3| \leq N^2 \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}}(1^\lambda)$$

Game_5 : In this game, the simulator adds an abort condition when queried the signing oracle $\mathcal{O}_{\mathsf{TS.Sign}}$. The simulator first prepares an empty list $\mathsf{Signed}[\cdot]$ at the beginning of the game. When the adversary $\mathcal{A}$ queries the signing oracle $\mathcal{O}_{\mathsf{TS.Sign}}$ on user $i \in \mathsf{SS}$, it sets $\mathsf{ctnt} := [\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_\mathbf{j}})_{j\in\mathsf{SS}}]$ and checks if $\mathsf{Signed}[\mathsf{ctnt}, \mathsf{i}] = \perp$, that is, the simulator checks whether user i has already signed with ctnt. If so, it aborts the game and otherwise, it proceed identically to Game_4 and the simulator updates $\mathsf{Signed}[\mathsf{ctnt}, i] \leftarrow \top$.

Let us bound the probability that an honest user i signs the same ctnt more than once. Notice that ctnt includes $\overrightarrow{\mathbf{w}_i}$; the vector of commitments that user i generated in the pre-processing phase. By construction, $\overrightarrow{\mathbf{w}_i}$ is stored in st_i, which is discarded once user i signs with ctnt. In particular, for an honest user i to have signed on the same ctnt more than twice, then it must have generated the same vector of commitments $\overrightarrow{\mathbf{w}_i}$ in the pre-processing phase. Since these commitments are generated honestly, the probability of such an event can be bounded by Lemma 3.20 from [EKT24]. Thus, we have

$$|\epsilon_5 - \epsilon_4| \leq \frac{Q_S^2}{2^{n-1}} + \mathsf{negl}(1^\lambda)$$

Game_6 : In this game, the simulator changes how it generates the intermediate masking terms of the honest users $(\mathbf{m}_{i,\mathsf{sHS}}, \mathbf{m}^*_{\mathsf{sHS},i})$. Throughout the proof, we will call $\mathbf{m}_{i,\mathsf{sHS}}$ and $\mathbf{m}^*_{\mathsf{sHS},i}$ as the *row* and *column* masks, respectively. The simulator prepares a new object $\mathsf{Mask}[\cdot]$ containing two fields: $\mathsf{Mask}[\cdot].\mathbf{m}$ to store row masks and $\mathsf{Mask}[\cdot].\mathbf{m}^*$ to store column masks. The simulator directly generates the row and column masks $\mathbf{m}_{i,\mathsf{sHS}}$ and $\mathbf{m}^*_{\mathsf{sHS},i}$, rather than generating the individual masks $(\mathbf{m}_{i,j})_{i,j\in\mathsf{sHS}}$, and stores them in $\mathsf{Mask}[\cdot].\mathbf{m}$ and $\mathsf{Mask}[\cdot].\mathbf{m}^*$.

We prove the view of the two games are identically distributed to the adversary $\mathcal{A}$. Assume $\mathcal{A}$ queries the signing oracle on user $i \in \mathsf{HS}$ with signer set SS. We first focus on the row masks. Since user i has never signed with $\mathsf{ctnt} = [\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_j})_{j \in \mathsf{SS}}]$ (i.e., $\mathsf{Signed}[\mathsf{ctnt}, i] =\perp$), we have $\mathsf{Rand}[\mathsf{ctnt}, i, i] =\perp$ in Game_5. This implies $\mathbf{m}_{i,i} \leftarrow\!\!\$\; \mathcal{R}_q^\ell$, and in particular, $\mathbf{m}_{i,\mathsf{sHS}} := \sum_{j \in \mathsf{sHS}} \mathsf{Rand}[\mathsf{ctnt}, i, j]$ is distributed uniformly random over $\mathcal{R}_q^\ell$ in Game_5. On the other hand, $\mathbf{m}_{i,\mathsf{sHS}} := \mathsf{Mask}[\mathsf{ctnt}, i].\mathbf{m}$ in Game_6 is also distributed uniformly random over $\mathcal{R}_q^\ell$ since $\mathsf{Mask}[\mathsf{ctnt}, i].\mathbf{m} =\perp$ if $\mathsf{Signed}[\mathsf{ctnt}, i] =\perp$. Therefore, the view of $\mathcal{A}$ remains identical in both games.

Next, we look at the column masks. We first assume user $i \in \mathsf{sHS}$ is not the last user to sign with ctnt. In this case, there exists at least one $j \in \mathsf{sHS}$ distinct from i for which $\mathsf{Rand}[\mathsf{ctnt}, j, i]$ is not set yet. This implies $\mathbf{m}_{j,i} \leftarrow\!\!\$\; \mathcal{R}_q^\ell$, and in particular, $\mathbf{m}^*_{\mathsf{sHS},i} := \sum_{j \in \mathsf{sHS}} \mathsf{Rand}[\mathsf{ctnt}, j, i]$ is distributed uniformly random over $\mathcal{R}_q^\ell$ in Game_5. Similarly to the previous argument, $\mathbf{m}^*_{\mathsf{sHS},i} := \mathsf{Mask}[\mathsf{ctnt}, i].\mathbf{m}^*$ in Game_6 is also distributed uniformly random over $\mathcal{R}_q^\ell$, and thus, the view of $\mathcal{A}$ remains identical in both games.

Lastly, let us assume user $i \in \mathsf{sHS}$ is the last user to sign with ctnt. That is, $\mathsf{Signed}[\mathsf{ctnt}, j] = \top$ for all $j \in \mathsf{sHS} \backslash \{i\}$. In Game_5, this implies that everything expect $\mathsf{Rand}[\mathsf{ctnt}, i, i]$ is already set. Namely, after $\mathbf{m}_{i,i}$ is sampled, the set $(\mathbf{m}_{i,j})_{i,j \in \mathsf{sHS}}$ is fully determined. Moreover, by construction, we have

$$\sum_{j \in \mathsf{sHS}} \mathbf{m}_{j,\mathsf{sHS}} = \sum_{j \in \mathsf{sHS}} \mathbf{m}^*_{\mathsf{sHS},j}$$

Combining the arguments, the column mask $\mathbf{m}^*_{\mathsf{sHS},i}$ of the final user is uniquely defined as

$$\mathbf{m}^*_{\mathsf{sHS},i} = \sum_{j \in \mathsf{sHS}} \mathbf{m}_{j,\mathsf{sHS}} - \sum_{j \in \mathsf{sHS} \backslash \{i\}} \mathbf{m}^*_{\mathsf{sHS},j}$$

This is identical to how $\mathbf{m}^*_{\mathsf{sHS},i}$ is set in Game_6. Therefore, the view of $\mathcal{A}$ remains identical in both games. We conclude that,

$$\epsilon_6 = \epsilon_5$$

Game_7 : In this game, the simulator modifies how it computes the responses $\mathbf{z}_i$. When the adversary $\mathcal{A}$ queries the signing oracle on user $i \in \mathsf{sHS}$, the simulator constructs $\mathsf{ctnt} := [\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_j})_{j \in \mathsf{SS}}]$ and checks whether user i is the last user in sHS to sign with ctnt. If not, the simulator removes the partial secret key $\mathsf{sk}_i = \mathbf{s}_i$ from the response $\mathbf{z}_i$. Otherwise, if user i is the last user, then it uses $\mathbf{s} - \sum_{j \in \mathsf{sCS}} L_{\mathsf{SS},j} \cdot \mathbf{s}_j$ in place of the partial secret key $\mathbf{s}_i$ to generate the response $\mathbf{z}_i$.

We show that view of $\mathcal{A}$ remains identical to the previous game. The key observation is that up until the last user in sHS, denoted as i^*, signs with ctnt, the row and column masks $(\mathbf{m}_{j,\mathsf{sHS}}, \mathbf{m}^*_{\mathsf{sHS},j})_{j \in \mathsf{sHS} \backslash \{i^*\}}$ are independently and uniformly distributed over $\mathcal{R}_q^\ell$. Moreover, until i^* signs with ctnt, all the column masks $(\mathbf{m}^*_{\mathsf{sHS},j})_{j \in \mathsf{sHS} \backslash \{i^*\}}$ remain information theoretically hidden from $\mathcal{A}$.

With this observation, we can equally define the simulator of Game_7 to first sample $\widetilde{\mathbf{m}}^*_{\mathsf{sHS},j} \leftarrow\!\!\$\; \mathcal{R}_q^\ell$ and set the row masks as $\mathbf{m}^*_{\mathsf{sHS},j} := c \cdot L_{\mathsf{SS},j} \cdot \mathbf{s}_j + \widetilde{\mathbf{m}}^*_{\mathsf{sHS},j}$ for all users $j \in \mathsf{sHS} \backslash \{i^*\}$. This induces the same distribution as simply sampling $\mathbf{m}^*_{\mathsf{sHS},j} \leftarrow\!\!\$\; \mathcal{R}_q^\ell$. Then, we can rewrite the response $\mathbf{z}_j$ as

$$\mathbf{z}_j := c \cdot L_{\mathsf{SS},j} \cdot \mathbf{s}_j + \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{y}_{j,b} + \widetilde{\mathbf{m}}^*_{\mathsf{sHS},j} + \mathbf{m}^*_{\mathsf{sCS},j}$$

Since, $\widetilde{\mathbf{m}}^*_{\mathsf{sHS},j}$ is distributed identically to the column mask of user j sampled in Game_6, the response $\mathbf{z}_j$ for $j \in \mathsf{sHS} \backslash \{i^*\}$ is identically distributed to Game_6.

It remains to analyze the response $\mathbf{z}_{i^*}$ for the last user i^*. First notice that the column masks

$\mathbf{m}^*_{\mathsf{sHS},i}$ of user i^* can be rewritten as

$$\begin{aligned}
\mathbf{m}^*_{\mathsf{sHS},i} &:= \sum_{j\in\mathsf{sHS}} \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m} - \sum_{j\in\mathsf{sHS}\setminus\{i^*\}} \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m}^* \\
&= \sum_{j\in\mathsf{sHS}} \mathbf{m}_{\mathsf{sHS},j} - \sum_{j\in\mathsf{sHS}\setminus\{i^*\}} \mathbf{m}^*_{j,\mathsf{sHS}} \\
&= \sum_{j\in\mathsf{sHS}} \mathbf{m}_{\mathsf{sHS},j} - \sum_{j\in\mathsf{sHS}\setminus\{i^*\}} (c\cdot L_{\mathsf{SS},j}\cdot \mathbf{s}_j + \widetilde{\mathbf{m}}^*_{j,\mathsf{sHS}})
\end{aligned}$$

Plugging this into $\mathbf{z}_{i^*}$, we have

$$\begin{aligned}
\mathbf{z}_{i^*} &= c\cdot\mathbf{s} - c\sum_{j\in\mathsf{sCS}} L_{\mathsf{SS},j}\cdot\mathbf{s}_j + \sum_{b\in[\mathsf{rep}]} \beta_b\cdot\mathbf{y}_{i^*,b} + \mathbf{m}^*_{\mathsf{sHS},i^*} + \mathbf{m}^*_{\mathsf{sCS},i^*} \\
&= c\cdot\left(\mathbf{s} - \sum_{j\in\mathsf{sCS}\cup\mathsf{sHS}\setminus\{i^*\}} L_{\mathsf{SS},j}\cdot\mathbf{s}_j\right) + \sum_{b\in[\mathsf{rep}]} \beta_b\cdot\mathbf{y}_{i^*,b} + \sum_{j\in\mathsf{sHS}} \mathbf{m}_{\mathsf{sHS},j} - \sum_{j\in\mathsf{sHS}\setminus\{i^*\}} \widetilde{\mathbf{m}}^*_{j,\mathsf{sHS}} + \mathbf{m}^*_{\mathsf{sCS},i^*} \\
&= c\cdot L_{\mathsf{SS},i^*}\cdot\mathbf{s}_{i^*} + \sum_{b\in[\mathsf{rep}]} \beta_b\cdot\mathbf{y}_{i^*,b} + \sum_{j\in\mathsf{sHS}} \mathbf{m}_{\mathsf{sHS},j} - \sum_{j\in\mathsf{sHS}\setminus\{i^*\}} \widetilde{\mathbf{m}}^*_{j,\mathsf{sHS}} + \mathbf{m}^*_{\mathsf{sCS},i^*}
\end{aligned}$$

where the second equality comes from the correctness of the linear Shamir secret sharing scheme. That is exactly how $\mathbf{z}_{i^*}$ is generated in Game_6.

Combining all the arguments, the response $\mathbf{z}_j$ for all users $j \in \mathsf{sHS}$ are distributed identically in both games. Thus, we have

$$\epsilon_7 = \epsilon_6$$

Appendix D. Games

```
$Game_1 := Game^{ts-uf}_{TS_{2-\mathsf{round}},\mathcal{A}}(1^\lambda, N, T)$
 1 : $Q_M := \emptyset,\ Q_{H_{\mathsf{sig}}}[\cdot] := \perp,\ Q_G[\cdot] := \perp$
 2 : $Q_{H_{\mathsf{com}}}[\cdot] := \perp,\ Q_{\mathbf{u}}[\cdot] := \perp$
 3 : $\mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k\times\ell}$
 4 : $(\mathbf{s}, \mathbf{e}) \leftarrow\!\!\$\ \mathcal{D}_{\mathbf{t}}^{\ell} \times \mathcal{D}_{\mathbf{t}}^{k}$
 5 : $\mathbf{T} := \lfloor \mathbf{A}\mathbf{s} + \mathbf{e} \rceil_{\nu_{\mathbf{t}}}$
 6 : $\mathbf{B} \leftarrow H_{\mathsf{com}}(\mathbf{A}, \mathbf{T}) \in \mathcal{R}_q^{k\times\ell'}$
 7 : $\mathsf{tspar} := (\mathbf{A}, \mathbf{B}, N, T)$
 8 : $(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow\!\!\$\ \mathcal{A}^{H_{\mathsf{sig}}, G}(\mathbf{A}, \mathbf{B}, N, T)$
 9 : assert $[\![\mathsf{CS} \subseteq [N]]\!] \wedge [\![|\mathsf{CS}| < T]\!]$
10 : $\mathsf{HS} := [N] \backslash \mathsf{CS}$
11 : for $i \in \mathsf{HS}$ do $\mathsf{st}_i := \emptyset$
12 : for $(i, j) \in [N] \times [N]$ do
13 :     $\mathsf{seed}_{i,j} \leftarrow\!\!\$\ \{0,1\}^\lambda$
14 : $\overrightarrow{P} \leftarrow\!\!\$\ \mathcal{R}_q^{\ell}[X]$ with $deg(\overrightarrow{P}) = T - 1,\ \overrightarrow{P}(0) = \mathbf{s}$
15 : $(\mathbf{s}_i)_{i\in[N]} := (\overrightarrow{P}(i))_{i\in[N]}$
16 : $\mathsf{vk} := (\mathsf{tspar}, \mathbf{T})$
17 : $(\mathsf{sk}_i)_{i\in[N]} := \big((\mathbf{s}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[N]}\big)_{i\in[N]}$
18 : $(\mathsf{sig}^*, M^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_{\mathsf{TS.PP}}, \mathcal{O}_{\mathsf{TS.Sign}}, H, G}(\mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
19 : if $[\![M^* \in Q_M]\!]$ then return $0$
20 : return $\mathsf{TS.Verify}(\mathsf{tspar}, \mathsf{vk}, M^*, \mathsf{sig}^*)$
```

```
$\mathcal{O}_{\mathsf{TS.PP}}(i)$
 1 : assert $[\![i \in \mathsf{HS}]\!]$
 2 : for $b \in [\mathsf{rep}]$ do
 3 :     $(\mathbf{y}_{i,b}, \mathbf{p}_{i,b}, \mathbf{e}'_{i,b}) \leftarrow\!\!\$\ \mathcal{D}_{\mathbf{w}}^{\ell} \times \mathcal{D}_{\mathbf{w}}^{\ell'} \times \mathcal{D}_{\mathbf{w}}^{k}$
 4 :     $\mathbf{w}_{i,b} := \mathbf{A}\mathbf{y}_{i,b} + \mathbf{B}\mathbf{p}_{i,b} + \mathbf{e}'_{i,b} \in \mathcal{R}_q^k$
 5 : $\overrightarrow{\mathbf{w}_i} := [\mathbf{w}_{i,1}, \cdots, \mathbf{w}_{i,\mathsf{rep}}]$
 6 : $\mathsf{pp}_i := \overrightarrow{\mathbf{w}_i}$
 7 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \cup \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b\in[\mathsf{rep}]})\}$
 8 : return $\mathsf{pp}_i$
```

```
$H_{\mathsf{com}}(\mathbf{A}, \mathbf{T})$
 1 : if $[\![Q_{H_{\mathsf{com}}}[\mathbf{A}, \mathbf{T}] = \perp]\!]$ then
 2 :     $\mathbf{B} \leftarrow\!\!\$\ \mathcal{R}_q^{k\times\ell'}$
 3 :     $Q_{H_{\mathsf{com}}}[\mathbf{A}, \mathbf{T}] \leftarrow \mathbf{B}$
 4 : return $Q_{H_{\mathsf{com}}}[\mathbf{A}, \mathbf{T}]$
```

```
$H_{\mathsf{sig}}(\mathsf{vk}, M, \mathbf{w})$
 1 : if $[\![Q_{H_{\mathsf{sig}}}[\mathsf{vk}, M, \mathbf{w}] = \perp]\!]$ then
 2 :     $c \leftarrow\!\!\$\ \mathcal{C}$
 3 :     $Q_{H_{\mathsf{sig}}}[\mathsf{vk}, M, \mathbf{w}] \leftarrow c$
 4 : return $Q_{H_{\mathsf{sig}}}[\mathsf{vk}, M, \mathbf{w}]$
```

```
$G(\mathsf{vk}, \mathsf{ctnt})$
 1 : if $[\![Q_G[\mathsf{vk}, \mathsf{ctnt}] = \perp]\!]$ then
 2 :     $(\beta_b)_{b\in[2,\mathsf{rep}]} \leftarrow\!\!\$\ \mathbb{T}^{\mathsf{rep}-1}$
 3 :     $Q_G[\mathsf{vk}, \mathsf{ctnt}] \leftarrow (1, (\beta_b)_{b\in[2,\mathsf{rep}]})$
 4 : return $Q_G[\mathsf{vk}, \mathsf{ctnt}]$
```

```
$\mathcal{O}_{\mathsf{TS.Sign}}(\mathsf{SS}, M, i, (pp_j)_{j\in\mathsf{SS}})$
 1 : assert $[\![\mathsf{SS} \subseteq [N]]\!] \wedge [\![i \in \mathsf{HS} \cap \mathsf{SS}]\!] \wedge [\![(\mathsf{pp}_i, \cdot) \in \mathsf{st}_i]\!]$
 2 : parse $(\overrightarrow{\mathbf{w}_j})_{j\in\mathsf{SS}\backslash\{i\}} \leftarrow (\mathsf{pp}_j)_{j\in\mathsf{SS}}$
 3 : pick $(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b\in[\mathsf{rep}]})$ from $\mathsf{st}_i$ with $\mathsf{pp}_i = \overrightarrow{\mathbf{w}}_i$
 4 : $\mathsf{ctnt} := [\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_j})_{j\in\mathsf{SS}}]$
 5 : $(\beta_b)_{b\in[\mathsf{rep}]} := G(\mathsf{vk}, \mathsf{ctnt})$
 6 : for $j \in \mathsf{SS}$ do
 7 :     parse $[\mathbf{w}_{j,1} \mid \cdots \mid \mathbf{w}_{j,\mathsf{rep}}] \leftarrow \overrightarrow{\mathbf{w}_j}$
 8 :     $\mathbf{w}_j := \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{w}_{j,b}$
 9 : $\widetilde{\mathbf{w}} := \left\lfloor \sum_{j\in\mathsf{SS}} \mathbf{w}_j \right\rceil_{\nu_{\mathbf{w}}}$
10 : $c := H_{\mathsf{sig}}(\mathsf{vk}, M, \widetilde{\mathbf{w}})$
11 : $\mathbf{m}_i := \sum_{j\in\mathsf{SS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt}))$
12 : $\mathbf{m}_i^* := \sum_{j\in\mathsf{SS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ctnt}))$
13 : $\mathbf{p}_i := \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{p}_{i,b}$
14 : $\mathbf{z}_i := c \cdot L_{\mathsf{SS},i} \cdot \mathbf{s}_i + \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{y}_{i,b} + \mathbf{m}_i^*$
15 : $\widehat{\mathsf{sig}}_i := (\mathbf{w}_i, \mathbf{m}_i, \mathbf{z}_i, \mathbf{p}_i)$
16 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \backslash \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b\in[\mathsf{rep}]})\}$
17 : $Q_M := Q_M \cup \{M\}$
18 : return $\widehat{\mathsf{sig}}_i$
```

Fig. 5. The first game, identical to the real unforgeability game.

Game$_2$:

$G(\mathsf{vk}, \mathsf{ctnt})$

```
1 :  if ⟦$Q_G[\mathsf{vk}, \mathsf{ctnt}] = \perp$⟧ then
2 :    $(\beta_b)_{b \in [2,\mathsf{rep}]} \leftarrow\$ \mathbb{T}^{\mathsf{rep}-1}$
3 :    $Q_G[\mathsf{vk}, \mathsf{ctnt}] \leftarrow (1, (\beta_b)_{b \in [2,\mathsf{rep}]})$
4 :    if ⟦$\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_j})_{j \in \mathsf{SS}} \leftarrow \mathsf{ctnt}$ correctly parses⟧ then
5 :      for $j \in \mathsf{SS}$ do
6 :        parse $[\mathbf{w}_{j,1} \mid \cdots \mid \mathbf{w}_{j,\mathsf{rep}}] \leftarrow \overrightarrow{\mathbf{w}_j}$
7 :        $\mathbf{w}_j := \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{w}_{j,b}$
8 :      $\widetilde{\mathbf{w}} := \left\lfloor \sum_{j \in \mathsf{SS}} \mathbf{w}_j \right\rceil_{\nu_{\mathbf{w}}}$
9 :      if ⟦$Q_{H_{\mathsf{sig}}}[\mathsf{vk}, M, \mathbf{w}] \neq \perp$⟧ then
10 :       $c \leftarrow\$ \mathcal{C}$
11 :       $Q_{H_{\mathsf{sig}}}[\mathsf{vk}, M, \mathbf{w}] \leftarrow c$
12 : return $Q_G[\mathsf{vk}, \mathsf{ctnt}]$
```

Fig. 6. In the Game 2, we change the simulation of the random oracle G. The changes between the previous game are highlighted in blue.

```
Game$_3$:

$\mathcal{O}_{\mathsf{TS.Sign}}(\mathsf{SS}, M, i, (\mathsf{pp}_j)_{j \in \mathsf{SS}})$

1 :    // Identical to Lines 1 to 10 in $\mathcal{O}_{\mathsf{TS.Sign}}$ in Game$_1$
2 :  $\mathbf{m}_{i,\mathsf{sCS}} := \sum_{j \in \mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})$
3 :  $\mathbf{m}^*_{\mathsf{sCS},i} := \sum_{j \in \mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ctnt})$
4 :  $\mathbf{m}_{i,\mathsf{sHS}} := \sum_{j \in \mathsf{sHS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})$
5 :  $\mathbf{m}^*_{\mathsf{sHS},i} := \sum_{j \in \mathsf{sHS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ctnt})$
6 :  $\mathbf{m}_i := \mathbf{m}_{i,\mathsf{sHS}} + \mathbf{m}_{i,\mathsf{sCS}}$
7 :  $\mathbf{m}^*_i := \mathbf{m}^*_{\mathsf{sHS},i} + \mathbf{m}^*_{\mathsf{sCS},i}$
8 :  $\mathbf{p}_i := \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{p}_{i,b}$
9 :  $\mathbf{z}_i := c \cdot L_{\mathsf{SS},i} \cdot \mathbf{s}_i + \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{y}_{i,b} + \mathbf{m}^*_i$
10 : $\widehat{\mathsf{sig}}_i := (\mathbf{w}_i, \mathbf{m}_i, \mathbf{z}_i, \mathbf{p}_i)$
11 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \backslash \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b \in [\mathsf{rep}]})\}$
12 : $Q_M := Q_M \cup \{M\}$
13 : return $\widehat{\mathsf{sig}}_i$
```

```
Game$_4$:

$\mathcal{O}_{\mathsf{TS.Sign}}(\mathsf{SS}, M, i, (\mathsf{pp}_j)_{j \in \mathsf{SS}})$

1 :    // Identical to Lines 1 to 10 in $\mathcal{O}_{\mathsf{TS.Sign}}$ in Game$_1$
2 :  $\mathbf{m}_{i,\mathsf{sCS}} := \sum_{j \in \mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})$
3 :  $\mathbf{m}^*_{\mathsf{sCS},i} := \sum_{j \in \mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ctnt})$
4 :  for $j \in \mathsf{sHS}$ do
5 :    if $[\![\mathsf{Rand}[\mathsf{ctnt}, i, j] = \perp]\!]$ then
6 :      $\mathbf{m}_{i,j} \leftarrow\!\$\ \mathcal{R}_q^\ell$, $\mathsf{Rand}[\mathsf{ctnt}, i, j] \leftarrow \mathbf{m}_{i,j}$
7 :    if $[\![\mathsf{Rand}[\mathsf{ctnt}, j, i] = \perp]\!]$ then
8 :      $\mathbf{m}_{j,i} \leftarrow\!\$\ \mathcal{R}_q^\ell$, $\mathsf{Rand}[\mathsf{ctnt}, j, i] \leftarrow \mathbf{m}_{j,i}$
9 :  $\mathbf{m}_{i,\mathsf{sHS}} := \sum_{j \in \mathsf{sHS}} \mathsf{Rand}[\mathsf{ctnt}, i, j]$
10 : $\mathbf{m}^*_{\mathsf{sHS},i} := \sum_{j \in \mathsf{sHS}} \mathsf{Rand}[\mathsf{ctnt}, j, i]$
11 : $\mathbf{m}_i := \mathbf{m}_{i,\mathsf{sHS}} + \mathbf{m}_{i,\mathsf{sCS}}$
12 : $\mathbf{m}^*_i := \mathbf{m}^*_{\mathsf{sHS},i} + \mathbf{m}^*_{\mathsf{sCS},i}$
13 : $\mathbf{p}_i := \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{p}_{i,b}$
14 : $\mathbf{z}_i := c \cdot L_{\mathsf{SS},i} \cdot \mathbf{s}_i + \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{y}_{i,b} + \mathbf{m}^*_i$
15 : $\widehat{\mathsf{sig}}_i := (\mathbf{w}_i, \mathbf{m}_i, \mathbf{z}_i, \mathbf{p}_i)$
16 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \backslash \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b \in [\mathsf{rep}]})\}$
17 : $Q_M := Q_M \cup \{M\}$
18 : return $\widehat{\mathsf{sig}}_i$
```

Fig. 7. In Games 3 and 4, we introduce the notion of honest/corrupt users and sample masks randomly for honest users. The changes between the previous game are highlighted in blue.

Game$_5$:

$\mathcal{O}_{\mathsf{TS.Sign}}(\mathsf{SS}, M, i, (\mathsf{pp}_j)_{j \in \mathsf{SS}})$

```
1 :      // Identical to Lines 1 to 10 in $\mathcal{O}_{\mathsf{TS.Sign}}$ in Game$_1$
2 :  $\mathbf{m}_{i,\mathsf{sCS}} := \sum_{j \in \mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})$
3 :  $\mathbf{m}^*_{\mathsf{sCS},i} := \sum_{j \in \mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ctnt})$
4 :  $\textbf{abort if } [\![\mathsf{Signed}[\mathsf{ctnt}, i] = \top]\!]$
5 :  $\textbf{for } j \in \mathsf{sHS} \textbf{ do}$
6 :    $\textbf{if } [\![\mathsf{Rand}[\mathsf{ctnt}, i, j] = \bot]\!] \textbf{ then}$
7 :      $\mathbf{m}_{i,j} \leftarrow\!\!\$\ \mathcal{R}_q^\ell,\ \mathsf{Rand}[\mathsf{ctnt}, i, j] \leftarrow \mathbf{m}_{i,j}$
8 :    $\textbf{if } [\![\mathsf{Rand}[\mathsf{ctnt}, i, j] = \bot \textbf{ then}$
9 :      $\mathbf{m}_{j,i} \leftarrow\!\!\$\ \mathcal{R}_q^\ell,\ \mathsf{Rand}[\mathsf{ctnt}, j, i] \leftarrow \mathbf{m}_{j,i}$
10 : $\mathbf{m}_{i,\mathsf{sHS}} := \sum_{j \in \mathsf{sHS}} \mathsf{Rand}[\mathsf{ctnt}, i, j]$
11 : $\mathbf{m}^*_{\mathsf{sHS},i} := \sum_{j \in \mathsf{sHS}} \mathsf{Rand}[\mathsf{ctnt}, j, i]$
12 : $\mathsf{Signed}[\mathsf{ctnt}, i] \leftarrow \top$
13 : $\mathbf{m}_i := \mathbf{m}_{i,\mathsf{sHS}} + \mathbf{m}_{i,\mathsf{sCS}}$
14 : $\mathbf{m}^*_i := \mathbf{m}^*_{\mathsf{sHS},i} + \mathbf{m}^*_{\mathsf{sCS},i}$
15 : $\mathbf{p}_i := \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{p}_{i,b}$
16 : $\mathbf{z}_i := c \cdot L_{\mathsf{SS},i} \cdot \mathbf{s}_i + \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{y}_{i,b} + \mathbf{m}^*_i$
17 : $\widehat{\mathsf{sig}}_i := (\mathbf{w}_i, \mathbf{m}_i, \mathbf{z}_i, \mathbf{p}_i)$
18 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \backslash \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b \in [\mathsf{rep}]})\}$
19 : $Q_M := Q_M \cup \{M\}$
20 : $\textbf{return } \widehat{\mathsf{sig}}_i$
```

Game$_6$:

$\mathcal{O}_{\mathsf{TS.Sign}}(\mathsf{SS}, M, i, (\mathsf{pp}_j)_{j \in \mathsf{SS}})$

```
1 :      // Identical to Lines 1 to 10 in $\mathcal{O}_{\mathsf{TS.Sign}}$ in Game$_1$
2 :  $\mathbf{m}_{i,\mathsf{sCS}} := \sum_{j \in \mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})$
3 :  $\mathbf{m}^*_{\mathsf{sCS},i} := \sum_{j \in \mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ctnt})$
4 :  $\textbf{abort if } [\![\mathsf{Signed}[\mathsf{ctnt}, i] = \top]\!]$
5 :  $\mathbf{m}_{i,\mathsf{sHS}} \leftarrow\!\!\$\ \mathcal{R}_q^\ell,\ \mathsf{Mask}[\mathsf{ctnt}, i].\mathbf{m} \leftarrow \mathbf{m}_{i,\mathsf{sHS}}$
6 :  $\textbf{if } [\![\forall\ j \in \mathsf{sHS} \backslash \{i\},\ \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m}^* \neq \bot]\!] \textbf{ then}$
7 :    $\mathbf{m}^*_{\mathsf{sHS},i} := \sum_{j \in \mathsf{sHS}} \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m} - \sum_{j \in \mathsf{sHS} \backslash \{i\}} \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m}^*$
8 :  $\textbf{else}$
9 :    $\mathbf{m}^*_{\mathsf{sHS},i} \leftarrow\!\!\$\ \mathcal{R}_q^\ell$
10 : $\mathsf{Mask}[\mathsf{ctnt}, i].\mathbf{m}^* \leftarrow \mathbf{m}^*_{\mathsf{sHS},i}$
11 : $\mathsf{Signed}[\mathsf{ctnt}, i] \leftarrow \top$
12 : $\mathbf{m}_i := \mathbf{m}_{i,\mathsf{sHS}} + \mathbf{m}_{i,\mathsf{sCS}}$
13 : $\mathbf{m}^*_i := \mathbf{m}^*_{\mathsf{sHS},i} + \mathbf{m}^*_{\mathsf{sCS},i}$
14 : $\mathbf{p}_i := \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{p}_{i,b}$
15 : $\mathbf{z}_i := c \cdot L_{\mathsf{SS},i} \cdot \mathbf{s}_i + \sum_{b \in [\mathsf{rep}]} \beta_b \cdot \mathbf{y}_{i,b} + \mathbf{m}^*_i$
16 : $\widehat{\mathsf{sig}}_i := (\mathbf{w}_i, \mathbf{m}_i, \mathbf{z}_i, \mathbf{p}_i)$
17 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \backslash \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b \in [\mathsf{rep}]})\}$
18 : $Q_M := Q_M \cup \{M\}$
19 : $\textbf{return } \widehat{\mathsf{sig}}_i$
```

Fig. 8. In Game 5 and 6, the simulator adds an abort condition and changes how it generates the intermediate masking terms of the honest users. The changes between the previous game are highlighted in blue.

```
Game$_7$:

$\mathcal{O}_{\mathsf{TS.Sign}}(\mathsf{SS}, M, i, (\mathsf{pp}_j)_{j\in\mathsf{SS}})$
1 :    // Identical to Lines 1 to 10 in $\mathcal{O}_{\mathsf{TS.Sign}}$ in Game$_1$
2 :  $\mathbf{m}_{i,\mathsf{sCS}} := \sum_{j\in\mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})$
3 :  $\mathbf{m}^*_{\mathsf{sCS},i} := \sum_{j\in\mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ctnt})$
4 :  abort if $[\![\mathsf{Signed}[\mathsf{ctnt}, i] = \top]\!]$
5 :  $\mathbf{m}_{i,\mathsf{sHS}} \leftarrow\!\!\$\ \mathcal{R}_q^\ell$, $\mathsf{Mask}[\mathsf{ctnt}, i].\mathbf{m} \leftarrow \mathbf{m}_{i,\mathsf{sHS}}$
6 :  if $[\![\forall\ j \in \mathsf{sHS}\backslash\{i\},\ \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m}^* \neq \perp]\!]$ then
7 :    $\mathbf{m}^*_{\mathsf{sHS},i} := \sum_{j\in\mathsf{sHS}} \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m} - \sum_{j\in\mathsf{sHS}\backslash\{i\}} \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m}^*$
8 :    $\mathbf{z}_i := c\cdot\mathbf{s} - c\sum_{j\in\mathsf{sCS}} L_{\mathsf{SS},j}\cdot\mathbf{s}_j + \sum_{b\in[rep]} \beta_b\cdot\mathbf{y}_{i,b} + \mathbf{m}^*_{\mathsf{sHS},i} + \mathbf{m}^*_{\mathsf{sCS},i}$
9 :  else
10 :   $\mathbf{m}^*_{\mathsf{sHS},i} \leftarrow\!\!\$\ \mathcal{R}_q^\ell$
11 :   $\mathbf{z}_i := \sum_{b\in[rep]} \beta_b\cdot\mathbf{y}_{i,b} + \mathbf{m}^*_{\mathsf{sHS},i} + \mathbf{m}^*_{\mathsf{sCS},i}$
12 : $\mathsf{Mask}[\mathsf{ctnt}, i].\mathbf{m}^* \leftarrow \mathbf{m}^*_{\mathsf{sHS},i}$
13 : $\mathsf{Signed}[\mathsf{ctnt}, i] \leftarrow \top$
14 : $\mathbf{m}_i := \mathbf{m}_{i,\mathsf{sHS}} + \mathbf{m}_{i,\mathsf{sCS}}$
15 : $\mathbf{p}_i := \sum_{b\in[\mathsf{rep}]} \beta_b\cdot\mathbf{p}_{i,b}$
16 : $\widehat{\mathsf{sig}}_i := (\mathbf{w}_i, \mathbf{m}_i, \mathbf{z}_i, \mathbf{p}_i)$
17 : $\mathsf{st}_i \leftarrow \mathsf{st}_i\backslash\{(\overrightarrow{\mathbf{w}_i}, (\mathbf{y}_{i,b}, \mathbf{p}_{i,b})_{b\in[\mathsf{rep}]})\}$
18 : $Q_M := Q_M \cup \{M\}$
19 : return $\widehat{\mathsf{sig}}_i$
```

Fig. 9. In Game 7, the simulator modifies how it computes the responses $\mathbf{z}_i$. The changes between the previous game are highlighted in blue.

```
Game$_8$:
1 : $Q_M := \emptyset,\ Q_{H_{\mathsf{sig}}}[\cdot] := \perp,\ Q_G[\cdot] := \perp$
2 : $Q_{H_{\mathsf{com}}}[\cdot] := \perp,\ Q_{\mathbf{u}}[\cdot] := \perp$
3 : $\mathsf{Signed}[\cdot] := \perp,\ \mathsf{Mask}[\cdot] := \perp$
4 : $\mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k \times \ell}$
5 : $\widetilde{\mathbf{T}} \leftarrow\!\!\$\ \mathcal{R}_q^{\ell}$
6 : $\mathbf{B} \leftarrow H_{\mathsf{com}}(\mathbf{A}, \widetilde{\mathbf{T}}) \in \mathcal{R}_q^{k \times \ell'}$
7 : $(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow\!\!\$\ \mathcal{A}^{H_{\mathsf{sig}}, G}(\mathbf{A}, \mathbf{B}, N, T)$
8 : assert $[\![\mathsf{CS} \subseteq [N]]\!] \wedge [\![|\mathsf{CS}| < T]\!]$
9 : $\mathsf{HS} := [N] \backslash \mathsf{CS}$
10 : for $i \in \mathsf{HS}$ do $\mathsf{st}_i := \emptyset$
11 : for $(i, j) \in [N] \times [N]$ do
12 :     $\mathsf{seed}_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
13 : for $i \in \mathsf{CS}$ do $\mathbf{s}_i \leftarrow\!\!\$\ \mathcal{R}_q^{\ell}$
14 : $\mathsf{vk} := (\mathsf{tspar}, \widetilde{\mathbf{T}})$
15 : $(\mathsf{sk}_i)_{i \in \mathsf{CS}} := \big((\mathbf{s}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j \in [N]})\big)_{i \in \mathsf{CS}}$
16 : $(\mathsf{sk}_i)_{i \in \mathsf{HS}} := \big((\perp, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j \in [N]})\big)_{i \in \mathsf{HS}}$
17 : $(\mathsf{sig}^*, M^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_{\mathsf{TS.PP}}, \mathcal{O}_{\mathsf{TS.Sign}}, H, G}(\mathsf{vk}, (\mathsf{sk}_i)_{i \in \mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
18 : if $[\![M^* \in Q_M]\!]$ then return $0$
19 : return $\mathsf{TS.Verify}(\mathsf{tspar}, \mathsf{vk}, M^*, \mathsf{sig}^*)$
```

Fig. 10. In Game 8, the verification key $\mathsf{vk} = (\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$ is replaced by $(\mathbf{A}, \widetilde{\mathbf{T}})$ where $\widetilde{\mathbf{T}}$ is sampled uniformly at random, remaining indistinguishable by the MLWE assumption. The changes between the previous game are highlighted in blue.

```
Game$_9$:

$H_{\mathsf{com}}(\mathbf{A}, \widetilde{\mathbf{T}})$
1 : if $[\![Q_{H_{\mathsf{com}}}[\mathbf{A}, \mathbf{T}] = \perp]\!]$ then
2 :     $\hat{\mathbf{B}} \leftarrow\!\!\$\ \mathcal{R}_q^{k \times (\ell' - 1)}$
3 :     $\mathbf{v} \leftarrow\!\!\$\ \mathcal{R}_q^{k}$
4 :     $\mathbf{b} := \widetilde{\mathbf{T}} - \mathbf{v}$
5 :     $\mathbf{B} := [\mathbf{b} | \hat{\mathbf{B}}]$
6 :     $Q_{H_{\mathsf{com}}}[\mathbf{A}, \widetilde{\mathbf{T}}] \leftarrow \mathbf{B}$
7 : return $Q_{H_{\mathsf{com}}}[\mathbf{A}, \mathbf{T}]$
```

```
Game$_{10}$:

$H_{\mathsf{com}}(\mathbf{A}, \widetilde{\mathbf{T}})$
1 : if $[\![Q_{H_{\mathsf{com}}}[\mathbf{A}, \mathbf{T}] = \perp]\!]$ then
2 :     $\hat{\mathbf{B}} \leftarrow\!\!\$\ \mathcal{R}_q^{k \times (\ell' - 1)}$
3 :     $\mathbf{u} \leftarrow\!\!\$\ \mathcal{S}_{\eta'}^{\ell' - 1 + k}$
4 :     $Q_{\mathbf{u}}[\mathbf{A}, \widetilde{\mathbf{T}}] := \mathbf{u}$
5 :     $\mathbf{v} := \hat{\mathbf{B}}\mathbf{u}$
6 :     $\mathbf{b} := \widetilde{\mathbf{T}} - \mathbf{v}$
7 :     $\mathbf{B} := [\mathbf{b} | \hat{\mathbf{B}}]$
8 :     $Q_{H_{\mathsf{com}}}[\mathbf{A}, \widetilde{\mathbf{T}}] \leftarrow \mathbf{B}$
9 : return $Q_{H_{\mathsf{com}}}[\mathbf{A}, \mathbf{T}]$
```

Fig. 11. In Game 9 and 10, the simulator modifies how he maintains the random oracle H_{com}. The changes between the previous game are highlighted in blue.

Game$_{11}$:

$\mathcal{O}_{\mathsf{TS.Sign}}(\mathsf{SS}, M, i, (\mathsf{pp}_j)_{j\in\mathsf{SS}})$

```
1 :  $\textbf{assert } [\![\mathsf{SS} \subseteq [N]]\!] \wedge [\![\in \mathsf{HS} \cap \mathsf{SS}]\!] \wedge [\![(\mathsf{pp}_i, \cdot) \in \mathsf{st}_i]\!]$
2 :  $\textbf{parse } (\overrightarrow{\mathbf{w}_j})_{j\in\mathsf{SS}\setminus\{i\}} \leftarrow (\mathsf{pp}_j)_{j\in\mathsf{SS}}$
3 :  $\textbf{pick } (\overrightarrow{\mathbf{w}_i}, (\mathbf{z}_{i,b}, \mathbf{r}_{i,b})_{b\in[\mathsf{rep}]}) \textbf{ from } \mathsf{st}_i \text{ with } \mathsf{pp}_i = \overrightarrow{\mathbf{w}}_i$
4 :  $\mathsf{ctnt} := [\mathsf{SS}, M, (\overrightarrow{\mathbf{w}_j})_{j\in\mathsf{SS}}]$
5 :  $(\beta_b)_{b\in[\mathsf{rep}]} := G(\mathsf{vk}, \mathsf{ctnt})$
6 :  $\textbf{for } j \in \mathsf{SS} \textbf{ do}$
7 :      $\textbf{parse } [\mathbf{w}_{j,1} \mid \cdots \mid \mathbf{w}_{j,\mathsf{rep}}] \leftarrow \overrightarrow{\mathbf{w}_j}$
8 :      $\mathbf{w}_j := \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{w}_{j,b}$
9 :  $\widetilde{\mathbf{w}} := \left\lfloor \sum_{j\in\mathsf{SS}} \mathbf{w}_j \right\rceil_{\nu_{\mathbf{w}}}$
10 : $c := H_{\mathsf{sig}}(\mathsf{vk}, M, \widetilde{\mathbf{w}})$
11 : $\mathbf{u} := Q_{\mathbf{u}}[\mathbf{A}, \widetilde{\mathbf{T}}]$
12 : $\bar{\mathbf{u}} := [1, \mathbf{u}^\intercal]^\intercal$
13 : $\mathbf{m}_{i,\mathsf{sCS}} := \sum_{j\in\mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{i,j}, \mathsf{ctnt})$
14 : $\mathbf{m}^*_{\mathsf{sCS},i} := \sum_{j\in\mathsf{sCS}} \mathsf{PRF}(\mathsf{seed}_{j,i}, \mathsf{ctnt})$
15 : $\textbf{abort if } [\![\mathsf{Signed}[\mathsf{ctnt}, i] = \top]\!]$
16 : $\mathbf{m}_{i,\mathsf{sHS}} \leftarrow\!\!\$\, \mathcal{R}_q^\ell,\ \mathsf{Mask}[\mathsf{ctnt}, i].\mathbf{m} \leftarrow \mathbf{m}_{i,\mathsf{sHS}}$
17 : $\textbf{if } [\![\forall\, j \in \mathsf{sHS}\setminus\{i\},\ \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m}^* \neq \perp]\!] \textbf{ then}$
18 :     $\mathbf{m}^*_{\mathsf{sHS},i} := \sum_{j\in\mathsf{sHS}} \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m} - \sum_{j\in\mathsf{sHS}\setminus\{i\}} \mathsf{Mask}[\mathsf{ctnt}, j].\mathbf{m}^*$
19 :     $\mathbf{z}_i := c \sum_{j\in\mathsf{sCS}} L_{\mathsf{SS},j} \cdot \mathbf{s}_j + \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{y}_{i,b} + \mathbf{m}^*_{\mathsf{sHS},i} + \mathbf{m}^*_{\mathsf{sCS},i}$
20 : $\textbf{else}$
21 :     $\mathbf{m}^*_{\mathsf{sHS},i} \leftarrow\!\!\$\, \mathcal{R}_q^\ell$
22 :     $\mathbf{z}_i := \sum_{b\in[rep]} \beta_b \cdot \mathbf{y}_{i,b} + \mathbf{m}^*_{\mathsf{sHS},i} + \mathbf{m}^*_{\mathsf{sCS},i}$
23 : $\mathsf{Mask}[\mathsf{ctnt}, i].\mathbf{m}^* \leftarrow \mathbf{m}^*_{\mathsf{sHS},i}$
24 : $\mathsf{Signed}[\mathsf{ctnt}, i] \leftarrow \top$
25 : $\mathbf{m}_i := \mathbf{m}_{i,\mathsf{sHS}} + \mathbf{m}_{i,\mathsf{sCS}}$
26 : $\mathbf{p}_i := \sum_{b\in[\mathsf{rep}]} \beta_b \cdot \mathbf{p}_{i,b} + c\bar{\mathbf{u}}$
27 : $\widehat{\mathsf{sig}}_i := (\mathbf{w}_i, \mathbf{m}_i, \mathbf{z}_i, \mathbf{p}_i)$
28 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \setminus \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{r}_{i,b}, \mathbf{z}_{i,b})_{b\in[\mathsf{rep}]})\}$
29 : $Q_M := Q_M \cup \{M\}$
30 : $\textbf{return } \widehat{\mathsf{sig}}_i$
```

Game$_{11}$:

$\mathcal{O}_{\mathsf{TS.PP}}(i)$

```
1 : $\textbf{assert } [\![i \in \mathsf{HS}]\!]$
2 : $\textbf{for } b \in [\mathsf{rep}] \textbf{ do}$
3 :     $(\mathbf{z}_{i,b}, \mathbf{r}_{i,b}, \mathbf{e}'_{i,b}) \leftarrow\!\!\$\, \mathcal{D}^\ell_{\mathbf{w}'} \times \mathcal{D}^{\ell'}_{\mathbf{w}'} \times \mathcal{D}^k_{\mathbf{w}'}$
4 :     $\mathbf{w}_{i,b} := \mathbf{A}\mathbf{z}_{i,b} + \mathbf{B}\mathbf{r}_{i,b} + \mathbf{e}'_{i,b} \in \mathcal{R}_q^k$
5 : $\overrightarrow{\mathbf{w}_i} := [\mathbf{w}_{i,1}, \cdots, \mathbf{w}_{i,\mathsf{rep}}]$
6 : $\mathsf{pp}_i := \overrightarrow{\mathbf{w}_i}$
7 : $\mathsf{st}_i \leftarrow \mathsf{st}_i \cup \{(\overrightarrow{\mathbf{w}_i}, (\mathbf{z}_{i,b}, \mathbf{r}_{i,b})_{b\in[\mathsf{rep}]})\}$
8 : $\textbf{return } \mathsf{pp}_i$
```

Fig. 12. In the final Game 11, the simulator is ready to generate its individual signature in order to invoke the MSIS problem. The changes between the previous game are highlighted in blue.