Compressed Linear Aggregate Signatures Based on Module Lattices

Katharina Boudgoust and Adeline Roux-Langlois

katharina.boudgoust@irisa.fr, adeline.roux-langlois@irisa.fr

Univ Rennes, CNRS, IRISA

Abstract. The Fiat-Shamir with Aborts paradigm of Lyubashevsky (Asiacrypt'09) has given rise to efficient lattice-based signature schemes. One popular implementation is Dilithium which is a finalist in an ongoing standardization process run by the NIST. An interesting research question is whether it is possible to combine several unrelated signatures, issued from different signing parties on different messages, into one single aggregated signature. Of course, its size should be much smaller than the trivial concatenation of all signatures. Ideally, the aggregation can be done offline by a third party, called *public aggregation*. Doröz et al. (IACR eprint 2020/520) proposed a first lattice-based aggregate signature scheme allowing public aggregation. However, its security is based on the hardness of the Partial Fourier Recovery problem, a structured lattice problem which neither benefits from worst-to-average reductions nor wasn't studied extensively from a cryptanalytic point of view.

In this work we give a first instantiation of an aggregate signature allowing public aggregation whose hardness is proven in the aggregate chosen-key model assuming the intractability of two well-studied problems on module lattices: The Module Learning With Errors problem (M-LWE) and the Module Short Integer Solution problem (M-SIS). Both benefit from worst-case to average-case hardness reductions. Our protocol can be seen as an aggregated variant of Dilithium. Alternatively, it can be seen as a transformation of the protocol from Doröz et al. to the M-LWE/M-SIS framework.

Keywords. Lattice-Based Cryptography, Module Lattices, Signature Aggregation

1 Introduction

For a long time, the main focus of cryptology was on secure encryption. With the invention of public key cryptology in the 1970s and the spread of the internet, the need of secure key exchange and authentication of data became more and more important. This is why nowadays the focus of public key cryptology increasingly shifts towards digital signatures. A digital signature scheme $\mathsf{\Pi}_S$ with message space $\mathcal{M}$ is composed of three algorithms $\mathsf{KGen}, \mathsf{Sig}$ and Vf. The algorithm KGen

generates a key pair $(\mathsf{sk}, \mathsf{vk})$ for a given user, who can then use their[1] secret key sk to generate a signature σ on a given message $m \in \mathcal{M}$ by running $\mathsf{Sig}(\mathsf{sk}, m)$. Afterwards, this signature can be verified by anyone using the verification key vk, which is publicly available, by running $\{0,1\} \leftarrow \mathsf{Vf}(\mathsf{vk}, m, \sigma)$. If the verification procedure outputs 1, the signature passes validation.

An interesting research question is whether it is possible to define an additional algorithm $\sigma_{agg} \leftarrow \mathsf{AggSig}(\mathsf{VK}, M, \Sigma)$ which takes as input a vector of $N \in \mathbb{Z}$ verification keys $\mathsf{VK} = (\mathsf{vk}_j)_{j\in[N]}$, a vector of N messages $M = (m_j)_{j\in[N]}$ and a vector of N signatures $\Sigma = (\sigma_j)_{j\in[N]}$, that were generated by the N different signing parties with corresponding verification keys vk_j, and outputs a single signature σ_{agg}. We further require a way for others to verify that σ_{agg} is indeed an aggregation of valid signatures. Thus, we need to provide a second additional algorithm $\{0,1\} \leftarrow \mathsf{AggVf}(\mathsf{VK}, M, \sigma_{agg})$, that outputs 1 if σ_{agg} is a valid aggregation of N valid signatures. All five algorithms define a so-called *aggregate signature scheme* $\Pi_{AS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{AggSig}, \mathsf{AggVf})$, where we require that it must satisfy correctness and unforgeability properties. A trivial solution is to set σ_{agg} as the concatenation of all the N different signatures and verify one after the other. In the following we are searching for an aggregate scheme that produces a σ_{agg} which is significantly shorter than this trivial solution. Ideally, the aggregation algorithm AggSig can be performed by a third, even untrusted party without needing to communicate with the N signing parties. We call this *public aggregation*. The concept and a first realization of aggregate signatures with public aggregation were given by Boneh et al. [BGLS03] by using bilinear maps constructed over elliptic curves in the generic group model. Aggregate signatures are a useful tool to save communication costs in settings where different users have to authenticate their communication, for instance in consensus protocols or certificate chains. More recently, they attracted increased interest as they help to reduce the size of blockchains such as the Bitcoin blockchain.

A first attempt to build lattice-based aggregate signatures with public aggregation was recently made by Doröz et al. [DHSS20]. Their construction builds upon the signature scheme PASS Sign, introduced by Hoffstein et al. [HPS$^+$14]. As a warm-up, they introduce a simple linear aggregate signature, which they call MMSA (multi-message, multi-user signature aggregation). However, in this version, the aggregate signature is larger than the trivial concatenation of N different signatures. In order to improve the efficiency and thus to get something meaningful, they first compress the signature, leading to MMSAT (the T stands for a linear compression function T), and then compress the verification keys, leading to MMSATK. Unfortunately, their construction has two disadvantages: First, the size of an aggregated signature is still linear in the number N of involved signatures, even though it is much smaller than simply concatenating them all thanks to the compression. Second, they only provide a security proof

[1] Throughout the paper, the neutral singular pronouns *they/their* are used in order to keep the language as inclusive as possible. See also `https://www.acm.org/diversity-inclusion/words-matter`

for the first variant MMSA[2] by showing that it inherits the security of the underlying PASS Sign, and subsequently its security can be based on the hardness of the Partial Fourier Recovery problem (PFR). The PFR asks to recover a polynomial in the ring $\mathbb{Z}[x]/\langle x^n - 1\rangle$ of small norm having access only to a partial list of its Fourier transform. It can be formulated as a shortest vector problem over some structured lattices. However, up to date there are no connections to worst-case lattice problems, which may be seen as a security concern.

In a parallel line of research, aggregate signature schemes that only allow for *private aggregation* have been proposed. In this setting, the different signing parties interact with each other to generate an aggregated signature on one message, which can be the concatenation of different messages. Those are also known as *multi-signature schemes* and there have been several recent protocols following the Fiat-Shamir with Aborts (FSwA) paradigm [Lyu09] providing lattice-based inter-active aggregate signatures, see for instance [BS16], [DOTT20] and [BK20].

Contributions. In this paper, we propose an aggregate signature allowing public aggregation, whose security is proven assuming simultaneously the hardness of Module Learning With Errors (M-LWE) and Module Short Integer Solution (M-SIS) and therefore on worst-case module lattice problems [LS15]. Earlier proposals either only offered security based on (non-standard) average-case lattice problems, or didn't allow for public aggregation. From a high level perspective, we take the practical signature from Güneysu et al. [GLP12] as a starting point. It follows the FSwA paradigm for lattice-based schemes [Lyu09,Lyu12], which is also used in the signature Dilithium [DKL+18], a finalist in the ongoing NIST standardization process[3]. The emphasis of our work lies on a proper security proof, also for the compressed version of the aggregate signature. We think that it is important to adapt the interesting ideas of the MMSA(TK) aggregate signature to the setting of lattice-based signatures within the M-LWE/M-SIS framework to stimulate further research in this direction.

Technical Details. Let us quickly recall the FSwA paradigm for lattice-based signatures. In the following, all computations are done over the ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, where n is a power of two and q is some prime modulus. A verification key is given as $\mathbf{t} = [\mathbf{A}|\mathbf{I}] \cdot \mathbf{s} \in R_q^k$, where $\mathbf{s} \in R_q^{\ell+k}$ is a vector of small norm (defining sk), $\mathbf{A} \in R_q^{k\times\ell}$ is a public uniform matrix and $\mathbf{I}$ the identity matrix of order k. A signature is provided by $\sigma = (\mathbf{u}, \mathbf{z}) \in R_q^k \times R_q^{\ell+k}$, where $\mathbf{u}$ is a commitment that via some hash function H_c defines a challenge c, and $\mathbf{z}$ is the answer to this challenge. For verification, one checks that $\mathbf{z}$ is small and that $[\mathbf{A}|\mathbf{I}] \cdot \mathbf{z} = \mathbf{t} \cdot c + \mathbf{u}$, where $c = H_c(\mathbf{u}, \mathbf{t}, m)$ for the verification key $\mathbf{t}$ and a message m. Adding $\mathbf{t}$ to the input of H_c is a simple countermeasure to prevent so-called *rogue key attacks* [BGLS03, Sec. 3.2]. The size of a single signature can be reduced by replacing $\mathbf{u} \in R_q^k$ by $c \in R$. As we detail out later in Section 3 this can't be done in the aggregate setting. A naive way to aggregate N different signatures $(\sigma_j)_{j\in[N]}$ with $\sigma_j = (\mathbf{u}_j, \mathbf{z}_j)$ into one signature σ_{agg} would be to compute

[2] The authors announced that a security proof for MMSAT will appear in a full version which was not available at the time of writing.

[3] `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/`

the sum of all components $(\sum_j \mathbf{u}_j, \sum_j \mathbf{z}_j)$. However, we wouldn't be able to verify this aggregated signature as we can't re-compute the different challenges c_j as we don't know the inputs $\mathbf{u}_j$ to H_c, originally used by the signing parties. Thus, we can only sum up the $\mathbf{z}_j$ parts and still have to transmit all the $\mathbf{u}_j$, which produces an aggregate signature of the form $\sigma_{agg} = ((\mathbf{u}_j)_j, \sum_j \mathbf{z}_j)$. This is essentially how the aggregate signature in MMSA looks like [DHSS20, Sec. 4]. In order to gain efficiency, we use the following observation made in [DHSS20, Sec. 5]. For the security of the signature scheme, we need the collision property of the random oracle H_c to be negligible in our security parameter λ, say $2^{-2\lambda}$. However, a commitment $\mathbf{u}$ is an element of R_q^k and thus the commitment space is much bigger than we need for combinatorial security. By taking a linear map $T\colon R_q^k \to \mathbb{Z}_q^{n_0}$ with $n_0 \log_2(q) \approx 2\lambda$, we can compress the input to H_c to $T(\mathbf{u}_j)$. For the verification, we only need the sum of the $\mathbf{u}_j$, i.e., $\hat{\mathbf{u}} = \sum_j \mathbf{u}_j$. Now, the aggregate signature is set to $\sigma_{agg} = (\hat{\mathbf{u}}, T(\mathbf{u}_j)_j, \mathbf{z})$, where $\mathbf{z} = \sum_j \mathbf{z}_j$. In order to pass verification, the norm of $\mathbf{z}$ has to be small enough and the equation $[\mathbf{A}|\mathbf{I}] \cdot \mathbf{z} = \sum_j \mathbf{t}_j \cdot c_j + \hat{\mathbf{u}}$ has to be true, where $c_j = H_c(T(\mathbf{u}_j), \mathbf{t}_j, m_j)$. Further, one checks that $T(\hat{\mathbf{u}}) = \sum_j T(\mathbf{u}_j)$, to make sure that the $T(\mathbf{u}_j)$ and $\hat{\mathbf{u}}$ are well defined. We formally present our aggregate signature scheme in Section 3. An aggregated signature is composed of $N+2$ elements, which is linear in the number of involved signatures. Still, it is much smaller than the trivial concatenation, as the size of the vector $T(\mathbf{u}_j)_j$ is only linear in the security parameter λ and not linear in the lattice dimension kn, where $\lambda \ll kn$.

To illustrate the gain in efficiency, we consider the concrete parameters of Dilithium for the security level III with security parameter $\lambda = 128$. In this setting, one signature $\sigma = (\mathbf{z}, c) \in R_q^\ell \times R_q$ with $\|\mathbf{z}\|_\infty \leq B$ and $\|c\|_\infty = 1$ is of size 2 701 Bytes. Taking the concatenation of $N = 1000$ signatures results in an aggregate signature of size 2 701 000 Bytes. In our construction, an aggregate signature $\sigma_{agg} = (\hat{\mathbf{u}}, (T(\mathbf{u}_j))_{j \in [N]}, \mathbf{z}) \in R_q^k \times (\mathbb{Z}_q^{n_0})^N \times R_q^{k+\ell}$ with $\|\mathbf{z}\|_\infty \leq \sqrt{N} \cdot B$ and $n_0 \cdot \log_2(q) \approx 2\lambda$. In order to keep the same ratio between the bound and the modulus, in our scheme $\log_2(q)$ has to be larger than in Dilithium by an additive factor $\log_2(\sqrt{N})$. The resulting σ_{agg} is of size $\approx$ 43 700 Bytes, which is more than 60 times smaller than the trivial solution.

In Section 4, we provide a rigorous security proof (Theorem 1), where the proof idea follows the one of Damgård et al. [DOTT20] for their inter-active multi-signature. It is composed of a sequence of indistinguishable games (assuming the hardness of M-LWE), where the starting one is the security game of our aggregate signature. The game is specified by the aggregate chosen key model, as introduced by Boneh et al. [BGLS03]. In the last game, the signing procedure is simulated by some algorithm that doesn't depend on the secret key and the verification key is sampled uniformly at random. By applying the General Forking Lemma from Bellare and Neven [BN06] we can use two different responses of a successful adversary against the scheme in the last game to solve an instance of M-SIS. As we don't need trapdoor commitment schemes, the proof is less technical than the one in [DOTT20]. We use a Gaussian distribution for the masking vectors and the rejection sampling, as done in [DOTT20]. This leads to tighter norm bounds of an aggregate signature, see Remark 2.

Open Problems. The General Forking Lemma induces two problems: First, we currently don't know how to extend it to the quantum setting. And second, it leads to non-tight security proofs. To circumvent both issues, one could try to adapt the techniques of Abdalla et al. [AFLT16] and of Kiltz et al. [KLS18]. As we point out in Remark 3 the signature compression impedes their straightforward adaption. More generally, we leave as an open problem the design of an aggregate signature scheme based on standard lattice-problems that allows for public aggregation and at the same time has length (almost) independent of the number N of aggregated signatures.

2 Preliminaries

For $k \in \mathbb{N}$, we represent the set $\{1, \ldots, k\}$ by $[k]$. Vectors are denoted in bold lowercase and matrices in bold capital letters and the identity matrix of order k is denoted by $\mathbf{I}_k$. The concatenation of two matrices $\mathbf{A}$ and $\mathbf{B}$ with the same number of rows is denoted by $[\mathbf{A}|\mathbf{B}]$. For any set S, we denote by $U(S)$ the uniform distribution over S. We write $x \leftarrow D$ to denote the process of sampling an element x following the distribution D. Throughout the paper $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ denotes the ring of integers of the $2n$-th cyclotomic field, where n is a power of two. Further, q is a prime such that $q = 1 \bmod 2n$ defining the quotient ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. An element $a = \sum_{j=1}^{n} a_j x^{j-1}$ of R is identified with its coefficient vector $\mathbf{a} = (a_j)_{j\in[n]} \in \mathbb{Z}^n$. For any ring element $a \in R$, we set $\|a\|_\infty$, $\|a\|_2$ and $\|a\|_1$ as the infinity, the Euclidean and the ℓ_1-norm of its coefficient vector, respectively. All norms can be generalized to vectors $\mathbf{a} \in R^k$ for $k \in \mathbb{N}$, by considering the coefficient vector of dimension kn defined by $\mathbf{a}$. We rely on the key set $S_\beta = \{a \in R\colon \|a\|_\infty \leq \beta\}$ with $\beta \in \mathbb{N}$ and use the following upper bound on the norm of the product of two ring elements in R.

Lemma 1. *For any two elements $a, b \in R$ it yields $\|a \cdot b\|_\infty \leq 2 \cdot \|a\|_1 \cdot \|b\|_\infty$.*

Proof. Using the notion of expansion factor, as introduced in [LM06], we know that in R it holds $\|a \cdot b\|_\infty \leq 2 \cdot \|a \star b\|_\infty$, where $\star$ denotes the convolution product of two polynomials in the ring $\mathbb{Z}[x]$, without modulo $x^n + 1$. It yields

$$\|a \star b\|_\infty \leq \max_{k\in[2n-1]} \sum_{j\in[k+1]} |a_j| \cdot \|b\|_\infty \leq \|a\|_1 \cdot \|b\|_\infty. \qquad \square$$

We define continuous and discrete Gaussian distributions over R^m, for $m \in \mathbb{Z}$.

Definition 1. *For $\mathbf{z} \in R^m$ let $\rho_{\mathbf{v},s}(\mathbf{z}) = (1/\sqrt{2\pi}s)^m \exp(-\|\mathbf{z} - \mathbf{v}\|_2^2/2s^2)$ be the continuous Gaussian distribution centered at $\mathbf{v} \in R^m$ with standard deviation $s > 0$. Its discrete analogue is defined as $D^m_{\mathbf{v},s}(\mathbf{z}) = \frac{\rho_{\mathbf{v},s}(\mathbf{z})}{\rho_{\mathbf{v},s}(R^m)}$, where we set $\rho_{\mathbf{v},s}(R^m) = \sum_{\mathbf{x}\in R^m} \rho_{\mathbf{v},s}(\mathbf{x})$. If $\mathbf{v} = \mathbf{0}$, we simply write $D^m_s(\mathbf{z})$.*

We restate a result on the distribution of the sum of discrete Gaussians over R^m. It uses the so-called smoothing parameter $\eta(R^m)$ of R^m, which was introduced in [MR07] and can be bounded by $\eta(R^m) \leq \sqrt{\omega(\log(nm))}$ [MR07, Lem. 3.3]. Informally, it describes a threshold above which many properties of the continuous Gaussian distribution also hold for its discrete analogue.

Lemma 2 (Thm. 3.3 [MP13]). *Suppose that $s \geq \eta(R^m)/\sqrt{\pi}$. Let $m, N \in \mathbb{Z}$ and for $j \in [N]$ let $\mathbf{z}_j \leftarrow D_s^m$ be independent samples. Then the distribution of $\mathbf{z} = \sum_{j\in[N]} \mathbf{z}_j$ is statistically close to $D^m_{s\sqrt{N}}$.*

Finally, we use the following tail bound on discrete Gaussians.

Lemma 3 (Lem. 4.4 [Lyu12]). *For any parameter $\gamma > 1$ it yields $\Pr[\|\mathbf{z}\|_2 > \gamma s\sqrt{mn} \colon \mathbf{z} \leftarrow D_s^m] < \gamma^{mn} \exp(mn(1-\gamma^2)/2)$.*

Throughout the paper we choose γ such that this probability is negligibly small for a fixed m.

2.1 Module Lattice Problems

We also recall two lattice problems and refer to [LS15] for more details. We state them in their discrete, primal and HNF form.

Definition 2 (M-LWE)**.** *Let $k, \ell, \beta \in \mathbb{N}$. The* Module Learning With Errors problem M-LWE$_{k,\ell,\beta}$ *is defined as follows. Given $\mathbf{A} \leftarrow U(R_q^{k\times\ell})$ and $\mathbf{t} \in R_q^k$. Decide whether $\mathbf{t} \leftarrow U(R_q^k)$ or if $\mathbf{t} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{s}$, where $\mathbf{s} \leftarrow U(S_\beta^{\ell+k})$.*

The M-LWE assumption states that no PPT distinguisher can distinguish the two distributions with non-negligible advantage. Worst-case to average-case reductions guarantee that M-LWE is quantumly [LS15] and classically [BJRW20] at least as hard as the approximate shortest vector problem over module lattices.

Definition 3 (M-SIS)**.** *Let $k, \ell, b \in \mathbb{N}$. The* Module Short Integer Solution problem M-SIS$_{k,\ell,b}$ *is as follows. Given a uniformly random matrix $\mathbf{A} \leftarrow U(R_q^{k\times\ell})$. Find a non-zero vector $\mathbf{s} \in R_q^{k+\ell}$ such that $\|\mathbf{s}\|_2 \leq b$ and $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{s} = \mathbf{0} \in R_q^k$.*

The M-SIS assumption states that no PPT adversary can solve this problem with non-negligible probability. Worst-case to average-case reductions guarantee that M-SIS is classically [LS15] at least as hard as the approximate shortest independent vector problem over module lattices.

2.2 Aggregate Signature Schemes

We present the formal definition of aggregate signature schemes and their property of correctness.

Definition 4. *An aggregate signature scheme Π_{AS} for a message space $\mathcal{M}$ consists of a tuple of* PPT *algorithms $\Pi_{AS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{AggSig}, \mathsf{AggVf})$, proceeding as specified in the following protocol:*

$\mathsf{KGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$: *On input a security parameter λ, the key generation algorithm returns a secret signing key* sk *and a public verification key* vk.

$\mathsf{Sig}(\mathsf{sk}, m) \to \sigma$: *On input a signing key* sk *and a message $m \in \mathcal{M}$, the signing algorithm returns a signature σ.*

$\mathsf{Vf}(\mathsf{vk}, m, \sigma) \to \{0, 1\}$: *On input a verification key* vk*, a message $m \in \mathcal{M}$ and a signature σ, the verification algorithm either accepts (i.e. outputs* 1*) or rejects (i.e. outputs* 0*).*

$\mathsf{AggSig}(\mathsf{VK}, M, \Sigma) \to \sigma_{agg}$: *On input a vector of verification keys $\mathsf{VK} = (\mathsf{vk}_j)_{j\in[N]}$, a vector of messages $M = (m_j)_{j\in[N]}$ and a vector of signatures $\Sigma = (\sigma_j)_{j\in[N]}$, the signature aggregation algorithm returns a aggregated signature σ_{agg}.*

$\mathsf{AggVf}(\mathsf{VK}, M, \sigma_{agg}) \to \{0, 1\}$ *On input a vector of verification keys $\mathsf{VK} = (\mathsf{vk}_j)_{j\in[N]}$, a vector of messages $M = (m_j)_{j\in[N]}$ and an aggregated signature σ_{agg}, the aggregated verification algorithm either accepts (i.e. outputs* 1*) or rejects (i.e. outputs* 0*).*

Definition 5. *Let $\Pi_{AS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{AggSig}, \mathsf{AggVf})$ be an aggregate signature scheme for a message space $\mathcal{M}$. It is called* correct *if for all security parameters $\lambda \in \mathbb{N}$ and number of signers $N \in \mathbb{N}$ it yields*

$$\Pr[\mathsf{AggVf}(\mathsf{VK}, M, \mathsf{AggSig}(\mathsf{VK}, M, \Sigma)) = 1] = 1,$$

where $m_j \in \mathcal{M}$, $(\mathsf{sk}_j, \mathsf{vk}_j) \leftarrow \mathsf{KGen}(1^\lambda)$ and $\sigma_j \leftarrow \mathsf{Sig}(\mathsf{sk}_j, m_j)$ for $j \in [N]$ and $\mathsf{VK} = (\mathsf{vk}_j)_{j\in[N]}$, $M = (m_j)_{j\in[N]}$ and $\Sigma = (\sigma_j)_{j\in[N]}$.

2.3 General Forking Lemma

For the sake of completeness and to fix notations, we restate the General Forking Lemma from Bellare and Neven [BN06].

Lemma 4 (General Forking Lemma). *Let $N_q \geq 1$ be an integer and let C be a set of size $|C| \geq 2$. Let $\mathcal{B}$ be a randomized algorithm that on input $x, h_1, \ldots, h_{N_q}$ returns a pair (j, out), where $j \in \{0, \ldots, N_q\}$ and a side output* out*. Let* IGen *be a randomized algorithm called the input generator, parametrized by some security parameter λ. We define the accepting probability of $\mathcal{B}$ as*

$$\mathsf{acc} = \Pr[j \neq 0 \colon x \leftarrow \mathsf{IGen}(1^\lambda); h_1, \ldots, h_{N_q} \leftarrow U(H); (j, \mathsf{out}) \leftarrow \mathcal{B}(x, h_1, \ldots, h_{N_q})].$$

Let $\mathcal{F}_{\mathcal{B}}$ be a forking algorithm that works as in Figure 1, given x as input and black-box access to $\mathcal{B}$. We define the forking probability of $\mathcal{F}_{\mathcal{B}}$ as

$$\mathsf{frk} = \Pr[(\mathsf{out}, \widetilde{\mathsf{out}}) \neq (\bot, \bot) \colon x \leftarrow \mathsf{IGen}(1^\lambda); (\mathsf{out}, \widetilde{\mathsf{out}}) \leftarrow \mathcal{F}_{\mathcal{B}}(x)].$$

Then it yields $\mathsf{acc} \leq N_q / |C| + \sqrt{N_q \cdot \mathsf{frk}}$.

Upon input x, the algorithm $\mathcal{F}_\mathcal{B}$ does the following:
1. Pick a random coin ρ for $\mathcal{B}$
2. Generate $h_1, \ldots, h_{N_q} \leftarrow U(C)$
3. $(j, \mathsf{out}) \leftarrow \mathcal{B}(x, h_1, \ldots, h_{N_q}, \rho)$
4. If $j = 0$, then return $(\perp, \perp)$
5. Regenerate $\widetilde{h}_j, \ldots, \widetilde{h}_{N_q} \leftarrow U(C)$
6. $(\tilde{j}, \widetilde{\mathsf{out}}) \leftarrow \mathcal{B}(x, h_1, \ldots, h_{j-1}, \tilde{h}_j, \ldots, \tilde{h}_{N_q}, \rho)$
7. If $j = \tilde{j}$ and $h_j \neq \tilde{h}_j$, then return $(\mathsf{out}, \widetilde{\mathsf{out}})$
8. Else return $(\perp, \perp)$.

Fig. 1: The forking algorithm $\mathsf{F}_\mathcal{B}$.

3 Our Lattice-Based Aggregate Signature Scheme

In this section we first present the underlying single signature scheme (Section 3.1) before introducing our aggregate signature scheme in Section 3.2. From a high level perspective, we take the practical signature from Güneysu et al. [GLP12] as a starting point. The linear aggregation follows Doröz et. al [DHSS20], where the main difference is that we moved to the M-LWE/M-SIS framework instead of the Partial Fourier Recovery framework of the original scheme. We think that M-LWE and M-SIS are more standard lattice problems and thus they increase our confidence in the security of the proposed scheme.

3.1 The Single Signature Scheme

In the following we describe the underlying single signature scheme, which is essentially the signature scheme from Güneysu et al. [GLP12] with minor modifications. Let $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, with n a power of two and q a prime such that $q = 1 \bmod 2n$. For $k, \ell \in \mathbb{N}$, let $\mathbf{A} \in R_q^{k \times \ell}$ follow the uniform distribution and be a public shared parameter of the system. The number of columns ℓ and the number of rows k should be adapted to the required security level, but usually they are small constants. Let $H_c \colon \{0,1\}^* \to C = \{c \in R \colon \|c\|_1 = d, \|c\|_\infty = 1\}$ be a random oracle with d such that $|C| > 2^{2\lambda}$, where λ denotes the required security level. Let $s, \beta, M \in \mathbb{Z}$ and the message space $\mathcal{M} = \{0,1\}^*$. Further, let T denote a linear map $T \colon R_q^k \to \mathbb{Z}_q^{n_0}$, such that $n_0 \cdot \log_2 q \approx 2\lambda$.

The signature scheme $\Pi_S = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ from [GLP12] with minor modifications and allowing signature compression is illustrated in Figure 2.

Description. The algorithm KGen samples a secret key vector $\mathbf{s}$, composed of elements of R with coefficients of size at most β, and sets the verification key to $\mathbf{t} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{s} \in R_q^k$. At the beginning of the signing procedure, a masking vector $\mathbf{y}$ following the Gaussian distribution $D_s^{\ell+k}$ is sampled. The signing party then computes $\mathbf{u} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y} \in R_q^k$, which serves together with the verification key $\mathbf{t}$ and the message m as input to the random oracle H_c. The output c of H_c is a polynomial in R with exactly d coefficients that are ± 1 and the remaining coefficients are 0. The second part of a potential signature is defined

$\mathsf{KGen}(1^\lambda)$: sample $\mathbf{s} \leftarrow U(S_\beta^{\ell+k})$
set $\mathsf{sk} = \mathbf{s}$ and $\mathsf{vk} = \mathbf{t} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{s} \in R_q^k$
return $(\mathsf{sk}, \mathsf{vk})$

$\mathsf{Sig}(\mathsf{sk}, m)$: set $\mathbf{z} = \perp$
while $\mathbf{z} = \perp$ *do*:
  sample $\mathbf{y} \leftarrow D_s^{\ell+k}$
  set $\mathbf{u} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y} \in R_q^k$
  compute $c = H_c(T(\mathbf{u}), \mathbf{t}, m) \in C$
  set $\mathbf{z} = \mathbf{s} \cdot c + \mathbf{y}$
  with probability $1 - \min(1, D_s^{\ell+k}(\mathbf{z})/M \cdot D_{c\cdot\mathbf{s},s}^{\ell+k}(\mathbf{z}))$
    set $\mathbf{z} = \perp$
return $\sigma = (\mathbf{u}, \mathbf{z})$

$\mathsf{Vf}(\mathsf{vk}, \sigma, m)$: re-construct $c = H_c(T(\mathbf{u}), \mathbf{t}, m)$
if $\|\mathbf{z}\|_2 < B$ and $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} = \mathbf{t} \cdot c + \mathbf{u}$,
  then return 1
else return 0

Fig. 2: The signature scheme from [GLP12] with minor modifications, allowing signature compression.

as $\mathbf{z} = \mathbf{s} \cdot c + \mathbf{y}$. In order to make the distribution of the signature independent of the secret key, the algorithm only outputs the potential signature with probability $\min(1, D_s^{\ell+k}(\mathbf{z})/M \cdot D_{c\cdot\mathbf{s},s}^{\ell+k}(\mathbf{z}))$, where M is a constant that depends on β (the Euclidean norm of the secret $\mathbf{s}$) and d (the ℓ_1-norm of the challenge c). This step is called rejection sampling. In order to verify σ, the verifier first re-constructs the hash value $c = H_c(T(\mathbf{u}), \mathbf{t}, m)$ and then checks if the norm of $\mathbf{z}$ is smaller than B, where $B = \gamma s\sqrt{n(\ell + k)}$ is as in Lemma 3 with $m = \ell + k$, and that $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} = \mathbf{t} \cdot c + \mathbf{u}$. The parameters β, d, M and γ have to be set strategically such that the scheme is correct, efficient and secure, see [Lyu12,DKL+18].

Modifications. A first difference to the signature scheme in [GLP12] is that we use a Gaussian distribution for $\mathbf{y}$ (and thus for $\mathbf{z}$ and the rejection sampling) as originally done in [Lyu12], instead of a bounded uniform distribution. This change is motivated by the fact that the sum of independent Gaussians provides better bounds on its norm than the sum of uniformly distributed elements. This becomes crucial when aggregating the signatures in Section 3.2.

Another minor difference is that instead of transmitting c in the signature, we send $\mathbf{u}$. For a single signature, both cases are equivalent, as $\mathbf{u}$ defines c via the hash function H_c (and the verification key $\mathbf{t}$ and the message m) and c defines $\mathbf{u}$ via the equation $\mathbf{u} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} - \mathbf{t} \cdot c$ over R_q. In Section 3.2 we see that this is not the case for an aggregate signature scheme and we thus need to transmit the information $\mathbf{u}$ (and some additional information on the compressed values, see Sec. 3.2). Note that the size of $\mathbf{u}$ is much larger than the one of c, but for a large number of aggregated signatures, this additional cost will be amortized.

A third modification is that we add the verification key $\mathbf{t}$ to the input of the hash function H_c to compute the challenge c. As proposed by Boneh et al. [BGLS03, Sec. 3.2] and implemented for $\mathsf{MMSA(TK)}$ in [DHSS20, Sec. 8.2], adding $\mathbf{t}$ to the input of H_c ties the hash value to the $(\mathsf{sk}, \mathsf{vk})$-pair, which prevents so-called *rogue key attacks* (also called *key swap attacks*) on aggregate signatures.

A forth modification is that we compress the input to the hash function via the linear map T, which will later help to compress the size of an aggregated signature. Concretely, one can take any matrix $\mathbf{T} \in \mathbb{Z}_q^{n_0 \times nk}$ to define the linear map T. Taking T as the identity map (and $n_0 = nk$) gives the usual uncompressed signature. Note that the verification of a signature still requires computations over R_q, so that we keep the same hardness guarantees for the underlying lattice problem and thus the same security level.

Security. Overall, the security of the scheme $\Pi_S = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ as specified in Figure 2 is based on the hardness of M-LWE and M-SIS. For the reason of space limits, we don't go into detail here, but refer the interested reader to the original security proofs in [Lyu12] and [GLP12] in the ROM.

3.2 Linear Signature Aggregation With Compression

In the following we describe how to aggregate signatures from the scheme above. We use the linear aggregation with compression approach from [DHSS20]. Assume that we have N different users with corresponding secret keys $\mathbf{s}_1, \ldots, \mathbf{s}_N$ and verification keys $\mathsf{VK} = (\mathsf{vk}_j)_{j \in [N]}$, where $\mathsf{vk}_j = \mathbf{t}_j = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{s}_j$ using the same public matrix $\mathbf{A}$. The N users signed N different messages $M = (m_j)_{j \in [N]}$, producing N independent signatures $\Sigma = (\sigma_j)_{j \in [N]} = (\mathbf{u}_j, \mathbf{z}_j)_j$. The signature aggregation with compression is illustrated in Figure 3. The aggregate signature scheme is given by $\Pi_{AS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{AggSig}, \mathsf{AggVf})$.

$\mathsf{AggSig}(\mathsf{VK}, M, \Sigma)$:	For all $j \in [N]$ compute $T(\mathbf{u}_j)$
	set $\mathbf{z} = \sum_j \mathbf{z}_j \in R_q^{\ell+k}$
	and $\hat{\mathbf{u}} = \sum_j \mathbf{u}_j \in R_q^k$
	if $\\|\mathbf{z}\\|_2 \leq \sqrt{N} \cdot B$,
	  then return $\sigma_{agg} = (\hat{\mathbf{u}}, (T(\mathbf{u}_j))_j, \mathbf{z})$;
	else
	  return $\perp$;

$\mathsf{AggVf}(\mathsf{VK}, M, \sigma_{agg})$:	Re-construct $c_j = H_c(T(\mathbf{u}_j), \mathbf{t}_j, m_j)$ for all $j \in [N]$ If $\\|\mathbf{z}\\|_2 < \sqrt{N} \cdot B$ and if $[\mathbf{A}\|\mathbf{I}_k] \cdot \mathbf{z} = \sum_j (\mathbf{t}_j \cdot c_j) + \hat{\mathbf{u}}$ and if $T(\hat{\mathbf{u}}) = \sum_j T(\mathbf{u}_j)$ return 1; else return 0;

Fig. 3: Linear Signature Aggregation with Compression.

In order to aggregate N signatures $(\sigma_j)_{j \in [N]}$ the algorithm AggSig simply computes the two sums $\mathbf{z} = \sum_j \mathbf{z}_j$ and $\hat{\mathbf{u}} = \sum_j \mathbf{u}_j$, together with the compressed values $T(\mathbf{u}_j)_{j \in [N]}$ and outputs the aggregated signature $\sigma_{agg} = (\hat{\mathbf{u}}, (T(\mathbf{u}_j))_j, \mathbf{z})$, if the Euclidean norm of $\mathbf{z}$ is bounded above by $\sqrt{N} \cdot B$. Else the algorithm outputs $\perp$. The probability that AggSig outputs $\perp$ is negligible by Lemma 2. The aggregation can be done by anyone, even by untrusted parties, as long as they have access to the random oracle H_c. Thus, public aggregation is enabled.

To verify an aggregated signature σ_{agg}, AggVf first re-constructs the challenges c_j for $j \in [N]$ by using the compressed commitment $T(\mathbf{u}_j)$ provided in σ_{agg}, the verification key $\mathbf{t}_j$ and the message m_j. The algorithm then checks if the norm of $\mathbf{z}$ lies within the correct bound. Finally it verifies the equations $[\mathbf{A}, \mathbf{I}_k] \cdot \mathbf{z} = \sum_j (\mathbf{t}_j \cdot c_j) + \hat{\mathbf{u}}$ and $T(\hat{\mathbf{u}}) = \sum_j T(\mathbf{u}_j)$. If all checks go through, it outputs 1, else 0. It now becomes clear that transmitting c_j wouldn't be sufficient to verify the aggregated signature: as the verifier only knows the term $\mathbf{z}$, but not all the $\mathbf{z}_j$, they cannot reconstruct $\mathbf{u}_j$ from c_j. Without knowing $\mathbf{u}_j$, however, the verifier would not be able to verify the aggregated signature. Inter-active multi-signatures circumvent this issue by generating inter-actively one common input $\mathbf{u}$ to the random oracle that depends on all the $\mathbf{u}_j$, before sending the multi-signature, see for instance [BS16] or [DOTT20]. As a public aggregation is our main objective, we accept a larger aggregated signature size.

The correctness of our protocol Π_{AS} simply follows from the linearity of matrix-vector multiplication over R_q.

Lemma 5 (Correctness). *Let* $\Pi_{AS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{AggSig}, \mathsf{AggVf})$ *be the aggregate signature scheme for a message space* $\mathcal{M}$ *with the algorithms as in Figures 2 and 3. Assuming the correctness of the corresponding single signature scheme* $\Pi_S = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$*, the aggregate signature is correct, i.e.,*

$$\Pr[\mathsf{AggVf}(\mathsf{VK}, M, \mathsf{AggSig}(\mathsf{VK}, M, \Sigma)) = 1 | \mathsf{AggSig}(\mathsf{VK}, M, \Sigma) \neq \perp] = 1,$$

where $m_j \in \mathcal{M}$*,* $(\mathsf{sk}_j, \mathsf{vk}_j) \leftarrow \mathsf{KGen}(1^\lambda)$ *and* $\sigma_j \leftarrow \mathsf{Sig}(\mathsf{sk}_j, m_j)$ *for* $j \in [N]$ *and* $\mathsf{VK} = (\mathsf{vk}_j)_{j \in [N]}$*,* $M = (m_j)_{j \in [N]}$ *and* $\Sigma = (\sigma_j)_{j \in [N]}$*.*

Proof. Let $\sigma_{agg} \neq \perp$ be the aggregate signature produced by $\mathsf{AggSig}(\mathsf{VK}, M, \Sigma)$. The first two checks if $\|\mathbf{z}\|_2 \leq \sqrt{N} \cdot B$ and if $T(\hat{\mathbf{u}}) = \sum_{j \in [N]} T(\mathbf{u}_j)$ succeed by the construction of AggSig. For the last check, we compute

$$\begin{aligned}
[\mathbf{A} | \mathbf{I}_k] \cdot \mathbf{z} = [\mathbf{A} | \mathbf{I}_k] \cdot \left(\sum_{j=1}^{N} \mathbf{z}_j \right) &= \sum_{j=1}^{N} [\mathbf{A} | \mathbf{I}_k] \cdot \mathbf{z}_j \\
= \sum_{j=1}^{N} \mathbf{t}_j \cdot c_j + \mathbf{u}_j &= \sum_{j=1}^{N} \mathbf{t}_j \cdot c_j + \hat{\mathbf{u}},
\end{aligned}$$

where we used the linearity over R_q and the correctness of Π_S. □

Remark 1. In order to guarantee the correctness of the scheme, it is important that all key pairs $(\mathsf{sk}, \mathsf{vk})$ share the same public matrix $\mathbf{A}$. It can be computed interactively by all, or by a reasonable large subset of all parties together during a setup-phase as in [DOTT20]. In order to gain in efficiency, it can instead also be computed by some compact random seed.

The key idea behind the compression is that $(\hat{\mathbf{u}}, T(\mathbf{u}_j)_{j\in[N]})$ is much smaller than transmitting the complete $(\mathbf{u}_j)_{j\in[N]}$. At the same time, n_0 is large enough to guarantee combinatorial security as we set n_0 such that it is hard to find a collision of H_c for fixed verification key $\mathbf{t}$ and message m. By verifying that $T(\hat{\mathbf{u}}) = \sum_j T(\mathbf{u}_j)$, we know that $\hat{\mathbf{u}}$ was correctly generated. If we set T as the identity map (and $n_0 = kn$), then $T(\mathbf{u}_j) = \mathbf{u}_j$ defines the full vector and we can delete the sum $\hat{\mathbf{u}}$ from the aggregate signature.

Remark 2. Throughout the paper we use the Gaussian distribution for the masking vectors $\mathbf{y}_j$ and thus the resulting signature components $\mathbf{z}_j$. Alternatively, one can use a bounded uniform distribution, as originally done in [GLP12], as well as in Dilithium. In this case, the bound on the norm of the aggregate signature component $\mathbf{z}$ increases by a factor of N (instead of a factor of $\sqrt{N}$ in the Gaussian setting). To mitigate this loss, one can use a randomized weighted sum and the Central Limit Theorem as done for MMSA in [DHSS20, Sec. 4], which requires the introduction of a second hash function, increasing the aggregation complexity.

4 Security of Our Aggregate Signature Scheme

We recall in Sec. 4.1 the security model for aggregate signatures from Boneh et al. [BGLS03], before proving in Sec. 4.2 the security of our scheme from Sec. 3.

4.1 The Security Model

Informally speaking, the security notion we use within this paper of an aggregate signature scheme captures that there exists no efficient adversary who is able to existentially forge an aggregate signature, within a specified game. We use the aggregate chosen-key security model as introduced by Boneh et al. [BGLS03]. Let $\Pi_{AS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{AggSig}, \mathsf{AggVf})$ be an aggregate signature scheme with message space $\mathcal{M}$ as in Definition 4 and let N be the number of aggregated signatures. An adversary $\mathcal{A}$ attacking Π_{AS} is given a single verification key vk_N, the *challenge key*. Their goal is the existential forgery of an aggregate signature involving N signatures, where we oblige $\mathcal{A}$ to include a signature that can be verified using the challenge key. The remaining $N-1$ verification keys can be chosen freely by $\mathcal{A}$. The adversary is also given access to a signing oracle on the challenge key vk_N. Their advantage, denoted by $\mathsf{Adv}\ \mathsf{AggSig}_{\mathcal{A}}$, is defined to be their probability of success in the following game.

Setup. The aggregate forger $\mathcal{A}$ is provided with a challenge verification key vk_N.

Queries. Proceeding adaptively, $\mathcal{A}$ queries signatures on messages of their choice that can be verified using the challenge key vk_N.

Response. Finally, $\mathcal{A}$ outputs an aggregate signature σ_{agg}, together with a message vector $M = (m_j)_{j\in[N]}$ and verification key vector $\mathsf{VK} = (\mathsf{vk}_j)_{j\in[N]}$.

Result. The forger $\mathcal{A}$ wins the game if the aggregate signature σ_{agg} is a valid aggregate on the verification key-message pairs $(\mathsf{vk}_j, m_j)_{j\in[N]}$, i.e., if $1 \leftarrow \mathsf{AggVf}(\mathsf{VK}, M, \sigma_{agg})$. In order to avoid trivial solutions, $\mathcal{A}$ is not allowed to hand in a pair (vk_N, m_N), which was queried on the signing oracle before.

If we show the security in the *Random Oracle Model* (ROM), we also have to give $\mathcal{A}$ the possibility to query the used random oracles.

Definition 6. *Let H denote a hash function that is modeled as a random oracle. An aggregate signature scheme Π_{AS} is called $(N_H, N_{\mathsf{Sig}}, N)$*-secure against existential forgery in the aggregate chosen-key model *in the ROM, if there exists no PPT algorithm $\mathcal{A}$ that existentially forges an aggregate signature on N verification keys in the aggregate chosen-key model, where $\mathcal{A}$ has non-negligible advantage, makes at most N_H queries to the random oracle H and at most N_{Sig} queries to the signing oracle on the challenge key.*

4.2 Proof of Security

We now prove that our scheme is secure against existential forgery in the chosen-key model. The proof follows the one of Damgård et al. [DOTT20] for their interactive multi-signature. As we don't need trapdoor commitment schemes, our proof is less technical. Let $\Pi_{AS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{AggSig}, \mathsf{AggVf})$ be the aggregate signature from Section 3.

Theorem 1. *Let H_c be the hash function from our aggregate signature, modeled as a random oracle, with image space C. Assume the hardness of* $\text{M-LWE}_{k,\ell,\beta}$ *and of* $\text{M-SIS}_{k,\ell+1,b}$*, where $b = 2(\sqrt{N} \cdot B + \sqrt{d})$ with $B = \gamma s \sqrt{n(k+\ell)}$. Then, the aggregate signature Π_{AS} with parameters $(\ell, k, \beta, s, \gamma, d, N)$ is secure against existential forgery in the aggregate chosen-key model in the ROM. The advantage of some PPT adversary $\mathcal{A}$ against Π_{AS} is bounded above by*

$$\mathsf{Adv}\ \mathsf{AggSig}_{\mathcal{A}} \le \mathsf{Adv}_{\text{M-LWE}_{k,\ell,\beta}} + N_q / |C| + \sqrt{N_q \cdot \mathsf{Adv}_{\text{M-SIS}_{k,\ell+1,b}}} + \mathsf{negl}(\lambda)\,,$$

where λ denotes the security parameter and $\mathcal{A}$ makes at most N_{H_c} queries to H_c and N_{Sig} queries to the signing oracle and we set $N_q = N_{H_c} + N_{\mathsf{Sig}}$.

Proof. Let $\mathcal{A}$ be an adversary against Π_{AS} with advantage $\mathsf{Adv}\ \mathsf{AggSig}_{\mathcal{A}}$. Our high level goal is to show that their advantage is negligible in the security parameter λ by providing a sequence of games G_0, G_1 and G_2, where G_0 is the original chosen-key security game as in Section 4.1. Assuming the hardness of M-LWE, the adversary $\mathcal{A}$ can distinguish between those games only with negligible advantage. In the last game G_2 we apply the General Forking Lemma in order to obtain two forgeries $\sigma_{agg}, \tilde{\sigma}_{agg}$, with distinct challenges for the challenge key vk_N. The two forgeries then allow to construct a solution to M-SIS.

G_0**:** Set $N_q = N_{H_c} + N_{\mathsf{Sig}}$, where $\mathcal{A}$ makes at most N_{H_c} queries to H_c and N_{Sig} queries to the signing oracle on vk_N. Recall that C denotes the challenge space from Sig. Let $\mathcal{B}$ be a second algorithm that is provided with some randomly chosen $h_j \leftarrow U(C)$ for $j \in [N_q]$. For the random oracle H_c, the algorithm $\mathcal{B}$ maintains a table HT_c which is empty at the beginning. Further $\mathcal{B}$ stores a counter ctr, initially set to 0.

Setup. $\mathcal{B}$ generates $(\mathsf{sk}_N, \mathsf{vk}_N) \leftarrow \mathsf{KGen}(1^\lambda)$ and sends vk_N to $\mathcal{A}$.

Queries on H_c. On input $x = (T(\mathbf{u}), \mathbf{t}, m)$, if $\mathsf{HT}_c[x]$ is already set, $\mathcal{B}$ returns $\mathsf{HT}_c[x]$. Else, if $\mathbf{t} = \mathbf{t}_N$, they increment ctr and set $\mathsf{HT}_c[x] = h_{\mathsf{ctr}}$. Else they sample $\mathsf{HT}_c[x] \leftarrow U(C)$. Finally, they output $c = \mathsf{HT}_c[x]$.

Signing queries. $\mathcal{B}$ follows the honest signing procedure Sig from Π_{AS} for sk_N on input message m.

Forgery. Suppose that $\mathcal{A}$ outputs a forgery $\sigma_{agg} = (\hat{\mathbf{u}}, (T(\mathbf{u}_j))_j, \mathbf{z},)$ on the message vector $M = (m_j)_{j\in[N]}$ and the verification key vector $\mathsf{VK} = (\mathsf{vk}_j)_{j\in[N]}$. Without loss of generality we assume that HT_c was programmed on the input $x = (T(\mathbf{u}_N), \mathbf{t}_N, m_N)$ and thus $c_N = h_{j_f}$ for some counter index j_f. If not, $\mathcal{A}$ would only have a probability of $1 - 1/|C|$ to guess the correct c_N. If $\mathsf{AggVf}(\mathsf{VK}, M, \sigma_{agg}) = 1$, then $\mathcal{B}$ outputs (j_f, out), where $\mathsf{out} = (\mathsf{VK}, M, \sigma_{agg}, \mathbf{C})$, with $\mathbf{C} = (c_j)_{j\in[N]}$ such that $c_j = H_c(T(\mathbf{u}_j), \mathbf{t}_j, m_j)$. Else, $\mathcal{B}$ outputs $(0, \perp)$.

For $j = 0, 1, 2$, let $\Pr[G_j]$ denote the probability that $\mathcal{B}$ doesn't output $(0, \perp)$ in game G_j. It yields $\Pr[G_0] = \mathsf{Adv}\ \mathsf{AggSig}_{\mathcal{A}}$.

G_1**:** The game G_1 is identical to the previous game G_0 except that $\mathcal{B}$ doesn't generate the signature honestly, but instead simulates the transcript without using the secret key sk_N.

Signing queries. On input message m, $\mathcal{B}$ samples $c \leftarrow U(C)$ and $\mathbf{z} \leftarrow D_s^{\ell+k}$. They compute $\mathbf{u} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} - \mathbf{t}_N \cdot c$ and program $c = \mathsf{HT}_c[x]$ with $x = (T(\mathbf{u}), \mathbf{t}_N, m)$ afterwards. Finally, $\mathcal{B}$ outputs $\sigma = (\mathbf{u}, \mathbf{z})$ with probability $1/M$.

Due to the rejection sampling, the distribution of $\mathbf{z}$ is identical in both signing variants (see [DOTT20, Lem. 4] for more details). The only difference between the actual and the original signing algorithm is that now the output of H_c is programmed at the end, without checking whether it has already been set for x. Following the same argument as in [Lyu12, Lem. 5.3] this happens only with negligible probability and thus $|\Pr[G_1] - \Pr[G_0]| \leq \mathsf{negl}(\lambda)$.

G_2**:** The game G_2 is identical to the previous game G_1 except how $\mathcal{B}$ generates the vk_N during the setup phase.

Setup. $\mathcal{B}$ samples $\mathbf{t}_N \leftarrow U(R_q^k)$, sets $\mathsf{vk}_N = \mathbf{t}_N$ and outputs vk_N to $\mathcal{A}$.

As the signing queries are answered without using the corresponding secret key sk_N, $\mathcal{B}$ can replace vk_N by a random vector without $\mathcal{A}$ noticing, assuming the hardness of M-LWE$_{k,\ell,\beta}$. Thus $|\Pr[G_2] - \Pr[G_1]| \leq \mathsf{Adv}_{\text{M-LWE}}$.

Now comes the final step of the proof. Note that in game G_2 the matrix $\mathbf{A}' = [\mathbf{A}|\mathbf{t}_N]$ follows the uniform distribution over $R_q^{k\times(\ell+1)}$. We invoke the General Forking Lemma (Lemma 4), where the input generator IGen is defined to output $\mathbf{A}'$. Let acc denote the accepting probability of $\mathcal{B}$ and frk the forking probability of $\mathsf{F}_{\mathcal{B}}$ as defined in Lemma 4. Thus, the forking algorithm $\mathsf{F}_{\mathcal{B}}$ outputs with probability frk two different outputs $\mathsf{out}, \widetilde{\mathsf{out}} \neq (\perp, \perp)$, where $\Pr[G_2] = \mathsf{acc} \leq$

$N_q/|C| + \sqrt{N_q \cdot \mathsf{frk}}$. Let $\mathsf{out} = (\mathsf{VK}, M, \sigma_{agg}, \mathbf{C})$ and $\widetilde{\mathsf{out}} = (\widetilde{\mathsf{VK}}, \tilde{M}, \tilde{\sigma}_{agg}, \tilde{\mathbf{C}})$. As the random coins in both executions of $\mathcal{B}$ are the same and as the simulation of H_c is for $\mathbf{t}_j \neq \mathbf{t}_N$ independent of the input $(h_j)_{j\in[N_q]}$ to $\mathcal{B}$, we have $\mathbf{t}_j = \tilde{\mathbf{t}}_j$ and $c_j = \tilde{c}_j$ for all $j \in [N-1]$. Further it yields $M = \tilde{M}$ and $\hat{\mathbf{u}} = \tilde{\hat{\mathbf{u}}}$. As in both cases, the forgery passes validation, we know that $\|\mathbf{z}\|_2, \|\tilde{\mathbf{z}}\|_2 < \sqrt{N} \cdot B$. Additionally, $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} = \sum_{j\in[N]} \mathbf{t}_j \cdot c_j + \hat{\mathbf{u}}$ and $[\mathbf{A}|\mathbf{I}_k] \cdot \tilde{\mathbf{z}} = \sum_{j\in[N]} \tilde{\mathbf{t}}_j \cdot \tilde{c}_j + \tilde{\hat{\mathbf{u}}}$.

Hence, we can deduce that $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} - \mathbf{t}_N c_N = [\mathbf{A}|\mathbf{I}_k] \cdot \tilde{\mathbf{z}} - \mathbf{t}_N \tilde{c}_N$. In other words, the vector $\mathbf{x} = (\mathbf{z} - \tilde{\mathbf{z}}, \tilde{c}_N - c_N)^T$ is a solution to the M-SIS problem for the matrix $\mathbf{A}'$. The Euclidean norm of the vector $\mathbf{x}$ is bounded above by $\|\mathbf{x}\|_2 \leq 2(\sqrt{N}B + \sqrt{d}) = b$. This implies that $\mathsf{frk} \leq \mathsf{Adv}_{\text{M-SIS}}$. Overall, we get

$$\mathsf{Adv\ AggSig}_{\mathcal{A}} \leq \mathsf{Adv}_{\text{M-LWE}_{k,\ell,\beta}} + N_q/|C| + \sqrt{N_q \cdot \mathsf{Adv}_{\text{M-SIS}_{k,\ell+1,b}}} + \mathsf{negl}(\lambda)\,.$$

□

Remark 3. Using the General Forking Lemma introduces two disadvantages: On the one hand, it causes an important loss in the reduction. On the other hand, we currently don't know how to extend it to the so-called *quantum* ROM, where an adversary has quantum access to the random oracle (and classical access to the signing oracle). Abdalla et al. [AFLT16] proposed a much tighter reduction for lattice-based signature schemes following the FSwA paradigm by introducing *lossy* identification schemes. Kiltz et al. [KLS18] used their techniques to construct a generic framework for tightly secure signatures in the quantum ROM. The key idea behind lossy identification schemes is that verification keys can be replaced by random, so-called lossy, keys. Then, using a lossy key, for a fixed commitment (in our scheme this is the vector $\mathbf{u}$) with overwhelming probability there exists at most one transcript that verifies. In our case, however, it is not clear that lossiness holds due to the compression. More precisely, fixing the commitments $T(\mathbf{u}_j)_j$ to the hash function H_c, does not fix the underlying vectors $(\mathbf{u}_j)_j$ as one can easily generate other vectors $(\mathbf{v}_j)_j$ that are mapped to the same values by T, i.e., $T(\mathbf{u}_j) = T(\mathbf{v}_j)$ for all j. Hence, the sum $\hat{\mathbf{u}} = \sum_j \mathbf{u}_j$ in general doesn't equal the sum $\hat{\mathbf{v}} = \sum_j \mathbf{v}_j$ and thus an adversary may find two valid transcripts.

References

AFLT16. M. Abdalla, P.-A. Fouque, V. Lyubashevsky, and M. Tibouchi. Tightly secure signatures from lossy identification schemes. *J. Cryptol.*, 29(3):597–631, 2016.

BGLS03. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

BJRW20. K. Boudgoust, C. Jeudy, A. Roux-Langlois, and W. Wen. Towards classical hardness of module-lwe: The linear rank case. In *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 289–317. Springer, 2020.

BK20. D. Boneh and S. Kim. One-time and interactive aggregate signatures from lattices, 2020. `https://crypto.stanford.edu/~skim13/agg_ots.pdf`.

BN06. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 390–399. ACM, 2006.

BS16. R. El Bansarkhani and J. Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. In *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, volume 10052 of *Lecture Notes in Computer Science*, pages 140–155, 2016.

DHSS20. Y. Doröz, J. Hoffstein, J. H. Silverman, and B. Sunar. MMSAT: A scheme for multimessage multiuser signature aggregation. *IACR Cryptol. ePrint Arch.*, 2020:520, 2020.

DKL$^+$18. L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

DOTT20. I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *IACR Cryptol. ePrint Arch. to appear at PKC 2021*, 2020:1110, 2020.

GLP12. T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 530–547. Springer, 2012.

HPS$^+$14. J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, and W. Whyte. Practical signatures from the partial fourier recovery problem. In *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, pages 476–493, 2014.

KLS18. E. Kiltz, V. Lyubashevsky, and C. Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586. Springer, 2018.

LM06. V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 144–155, 2006.

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

Lyu09. V. Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *Advances in Cryptology - ASIACRYPT 2009,* *15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

Lyu12. V. Lyubashevsky. Lattice signatures without trapdoors. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, 2012.

MP13. D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2013.

MR07. D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.