Non-Interactive Half-Aggregate Signatures Based on Module Lattices
A First Attempt

Katharina Boudgoust[1] and Adeline Roux-Langlois[2]
katharina.boudgoust@cs.au.dk, adeline.roux-langlois@irisa.fr

[1] Aarhus University
[2] Univ Rennes, CNRS, IRISA

Abstract. The Fiat-Shamir with Aborts paradigm of Lyubashevsky 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. Informally, it can be seen as a lattice analogue of the well-known discrete-logarithm-based Schnorr signature. 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 significantly smaller than the trivial concatenation of all signatures. Ideally, the aggregation can be done offline by a third party, called *public aggregation*. Previous works have shown that it is possible to half-aggregate Schnorr signatures, but it was left unclear if the underlying techniques can be adapted to the lattice setting.
In this work, we show that, indeed, we can use similar strategies to obtain a signature scheme allowing for public aggregation whose hardness is proven 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).
Unfortunately, our scheme produces aggregated signatures that are *larger* than the trivial solution of concatenating. This is due to peculiarities that seem inherent to lattice-based cryptography. Its motivation is thus mainly *pedagogical*, as we explain the subtleties when designing lattice-based aggregate signatures that are supported by a proper security proof.

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 Π_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[3] 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. [Bon+03] 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.

Constructing aggregate signature schemes based on the discrete logarithm problem (without bilinear maps) turned out to be much more harder, and so far, only solutions that partly aggregate the signatures are known. Chalkias et al. [Cha+21] build a half-aggregate scheme for the well-known Schnorr signature [Sch91]. It produces aggregate signatures of half the size compared to the trivial solution of concatenating. Its security is proven in the Random Oracle Model (ROM) assuming the hardness of the discrete logarithm problem. It was left unclear if the underlying techniques can be adapted to the lattice setting.

Contributions. 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 thus based on worst-case module lattice problems [LS15]. Earlier proposals either only

[3] 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`

offered security based on (non-standard) average-case lattice problems, or didn't allow for public aggregation (cf. Related Works). From a high level perspective, we take the practical signature from Güneysu et al. [GLP12] as a starting point. It follows the Fiat-Shamir with Aborts (FSwA) paradigm for lattice-based schemes [Lyu12], which is also used in the signature Dilithium [Duc+18], a finalist in the ongoing NIST standardization process[4].

Due to peculiarities that seem inherent to lattice-based cryptography, our scheme produces aggregated signatures whose sizes are *larger* than the size of the trivial solution (that is concatenating all the single signatures together). The motivation of our work thus is *pedagogical* in order to demonstrate the subtleties when designing lattice-based aggregate signatures that are supported by a proper security proof, in a security model we explain below. We would like to remark that most issues we came across also apply to the MMSA(TK) aggregate signature [Dor+20] (cf. Related Works).

Technical Details. Let us quickly recall the FSwA paradigm for lattice-based signatures in the module setting. 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 $\mathbf{t} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{s} \in R_q^k$, 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 $\mathbf{I}_k$ 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. The challenge c is a polynomial with coefficients in $\{-1, 0, 1\}$. For verification, one checks that $\mathbf{z}$ is short and that $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} = \mathbf{t} \cdot c + \mathbf{u}$, 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-attacks* [Bon+03, Sec. 3.2].

An intuitive 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 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 $\sigma_{agg} = ((\mathbf{u}_j)_j, \sum_j \mathbf{z}_j)$.[5] This is essentially how our (half-)aggregate signature looks like. In order to prevent again rogue-type attacks, we use a linear combination (instead of the simple sum), where the coefficients come from some random oracle (which was queried on all signatures that are aggregated). This technique is also used in the Schnorr-analogue by Chalkias et al. [Cha+21]. We formally present our aggregate signature scheme in Section 3.2 and the rogue-attack for the simple-sum solution in Section 4.3.

The size of a single signature can be significantly reduced by replacing the commitment $\mathbf{u} \in R_q^k$ by the challenge $c \in R$. This does not only reduce the

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

[5] Chalkias et al. [Cha+21, Sec. 6] provide evidence that it is *necessary* to transmit all the commitments.

dimension of the vector from k to 1, but also the total bit-length from $nk \log_2 q$ to $n \log_2 3$, as $\mathbf{u}$ can be any vector in R_q^k but c is a polynomial with ternary coefficients. Unfortunately, this can't be done in the aggregate setting, as we only have knowledge of the *aggregated* value of all $\mathbf{z}_j$'s. This is the main reason of our failure in constructing aggregate signatures schemes on lattices that are shorter than the trivial concatenation. Note that in the discrete-logarithm setting of the Schnorr aggregate signature in [Cha+21], the challenge c and the commitment $\mathbf{u}$ are elements of the same space. This explains why there is a real improvement for Schnorr signatures, but not for FSwA signatures.

In our aggregate signature, we have to transmit all N vectors $(\mathbf{u}_j)_j$ (and the smallish linear combination of all the $\mathbf{z}_j$), whereas in the trivial concatenation we transmit all N challenges $(c_j)_j$ (and the N small vectors $\mathbf{z}_j$). The size of the $\mathbf{u}_j$ is so large that it annihilates the compressing effect of combining all the $\mathbf{z}_j$. More concretely, taking the level III parameters of Dilithium [Duc+18] and $N = 1000$ signatures to aggregate, then $(\mathbf{u}_1, \ldots, \mathbf{u}_N) \in (R_q^k)^N$ is of size ca. 4400 KB.[6] However, simply concatenating N single Dilithium signatures produces an aggregate signature of a smaller size of ca. 3300 KB. Note that both earlier versions of this paper on e-print (3 Mar 2021 and 8 Apr 2021) we further compressed the aggregated signature via some linear function T to obtain a solution that was indeed smaller than the trivial concatenation. As we elaborate in Section 3.3, this allowed for simple lattice attacks.

In Section 4.2, we provide a rigorous security proof (Theorem 1), where the proof idea follows the one of Damgård et al. [Dam+21] for their inter-active multi-signature (cf. Related Works). 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. [Bon+03]. 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 (twice) the General Forking Lemma from Bellare and Neven [BN06] we can use four different responses of a successful adversary against the scheme in the last game to solve an instance of M-SIS. This "double forking technique" has been used in the setting of multi-signatures, see Maxwell et al. [Max+19]. As we don't need trapdoor commitment schemes, the proof is less technical than the one in [Dam+21].

Related Works. A first attempt to build lattice-based aggregate signatures with public aggregation was made by Doröz et al. [Dor+20]. Their construction builds upon the signature scheme PASS Sign, introduced by Hoffstein et al. [Hof+14]. As a warm-up, they introduce a simple linear (half-)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, they first compress the sig-

[6] This calculation does not even take into account the fact that in our aggregate signature $\log_2 q$ has to be increased by some factor $\log_2 \sqrt{N}$.

nature, 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 several disadvantages: First, the linear compression used in MMSAT(K) is prone to simple forgery attacks (similar to Section 3.3), making those constructions insecure. Second, they only provide a security proof for the first variant MMSA 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 bounded distance decoding problem over some ideal lattice. This may rise security concerns, as problems over ideal lattices have been shown to be in specific parameter settings easier than their counterparts over arbitrary lattices, e.g., [CDW21]. Furthermore, up to today, there are no connections to worst-case lattice problems, which may be seen as an additional 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 FSwA paradigm providing lattice-based inter-active aggregate signatures, see for instance [Dam+21] and references therein.

Open Problems. We leave as an open problem the construction of an aggregate signature scheme based on standard lattice-problems which allows public aggregation, produces aggregate signatures that are smaller than the simple concatenation and at the same time is provably secure in the aggregate chosen-key security model.

Change Log. This is the second revision of the paper originally submitted to e-print in March 2021. The first revision (April 2021) took into account the attack against the simple sum solution in the chosen key model (Section 4.3). As, at the time of writing this version, we weren't aware of any other way to fix the scheme, we proved its security in a more restricted security model, that we called the *aggregate independent-chosen-key model.* The latest revision (May 2022) now uses a randomized linear combination of the single signatures (instead of a simple sum), where the scalars come from a large enough space. The security of the scheme can now be proven in the standard security model of aggregate signatures. However, we had to remove the compressing function T that we originally applied to the input to the hash function H_c as linear compressing is prone to simple lattice attacks (cf. Section 3.3). To avoid misunderstanding, we accordingly changed the title of the paper.

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 $\beta \in \mathbb{N}$.

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 1 (M-LWE)**.** *Let $k, \ell, \beta \in \mathbb{N}$. The* Module Learning With Errors *problem $\text{M-LWE}_{k,\ell,\beta}$ 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 [Bou+20] at least as hard as the approximate shortest vector problem over module lattices.

Definition 2 (M-SIS)**.** *Let $k, \ell, b \in \mathbb{N}$. The* Module Short Integer Solution problem *$\text{M-SIS}_{k,\ell,b}$ 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 3. *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).*

$\mathsf{AggSig}(\mathsf{VK}, M, \Sigma) \to \sigma_{agg}$: *Given as 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}.*

$\mathsf{AggVf}(\mathsf{VK}, M, \sigma_{agg}) \to \{0, 1\}$: *Given as 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 4. *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 1 (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 $\rho$ 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 the idea of Doröz et. al [Dor+20], 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. Lastly, we explain in Section 3.3 why the linear compression idea that we used in an earlier version and that we adapted from $\mathsf{MMSA(TK)}$ [Dor+20] leads to efficient lattice attacks.

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 $\mathcal{M} = \{0,1\}^*$. Finally, let $\mathcal{D}$ denote a distribution over $R^{\ell+k}$ providing (with overwhelming probability) vectors of norm at most B and to which we associate a rejection probability $\Pr_{rej}$.

The signature scheme $\Pi_S = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ from [GLP12] with minor modifications 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

```
KGen(1^λ) :    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 $(\mathsf{sk}, \mathsf{vk})$
Sig(sk, m) :   set $\mathbf{z} = \perp$
               while $\mathbf{z} = \perp$ do:
                 sample $\mathbf{y} \leftarrow D$
                 set $\mathbf{u} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y} \in R_q^k$
                 compute $c = H_c(\mathbf{u}, \mathbf{t}, m) \in C$
                 set $\mathbf{z} = \mathbf{s} \cdot c + \mathbf{y}$
                 with probability $1 - \Pr_{rej}$
                   set $\mathbf{z} = \perp$
               return $\sigma = (\mathbf{u}, \mathbf{z})$
Vf(vk, σ, m) : re-construct $c = H_c(\mathbf{u}, \mathbf{t}, m)$
               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.

vector $\mathbf{y}$ following the distribution D 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 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 $\Pr_{rej}$. This step is called rejection sampling. In order to verify σ, the verifier first re-constructs the hash value $c = H_c(\mathbf{u}, \mathbf{t}, m)$ and then checks if the norm of $\mathbf{z}$ is smaller than B and that $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} = \mathbf{t} \cdot c + \mathbf{u}$. The parameters β, d, B and $\Pr_{rej}$ have to be set strategically such that the scheme is correct, efficient and secure, see [Lyu12; Duc+18].

Distribution $\mathcal{D}$. For simplicity, we leave the concrete implementation of the distribution $\mathcal{D}$ open. In the literature, mainly two instantiations have been studied: the discrete Gaussian distribution over R and the uniform distribution over the set of elements of small norms [Lyu09; Lyu12]. The literature provides concrete formulas for the rejection probability $\Pr_{rej}$ and the bound B. For example, for $\mathcal{D} = D_s^{k+\ell}$ the discrete Gaussian distribution of width s, the bound B comes from the Gaussian tail bound and the rejection probability can be computed as $\min(1, D_s^{\ell+k}(\mathbf{z})/M \cdot D_{c\cdot\mathbf{s},s}^{\ell+k}(\mathbf{z}))$, 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).

Modifications. A first difference to the signature scheme in [GLP12] 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 $\mathbf{u}$.

Another 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. [Bon+03, Sec. 3.2] and implemented for $\mathsf{MMSA(TK)}$ in [Dor+20, 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.

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 Non-Interactive Half-Aggregation of Signatures

In the following we describe how to aggregate signatures from the scheme above. 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 $\Sigma = (\sigma_j)_{j\in[N]} = (\mathbf{u}_j, \mathbf{z}_j)_j$. For the aggregation, we need another random oracle $H_e\colon \{0,1\}^* \to C$, where C is the same challenge space as in the single signature scheme, characterized by some positive integer d. Further, we let B' denote the bound on the norm of an aggregated signature. It's concrete value depends on the distribution $\mathcal{D}$ (and its bound B) used for the single signature. Informally speaking, if $\mathcal{D}$ is a discrete Gaussian, so is $B' = O(\sqrt{N}B)$. If $\mathcal{D}$ is the uniform distribution over vectors of bounded norm, so is $B' = O(NB)$. For simplicity, we set $B' = N\sqrt{d}B$ in the following. The non-interactive half-aggregation of signatures is illustrated in Figure 3. The aggregate signature scheme is given by $\Pi_{AS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{AggSig}, \mathsf{AggVf})$.

```
AggSig(VK, $M$, $\Sigma$) :   For $j \in [N]$ compute $c_j = H_c(\mathbf{u}_j, \mathbf{t}_j, m_j)$
                    Then query $e_j \leftarrow H_e(c_1, \ldots, c_N, j)$ for all $j \in [N]$
                    Set $\mathbf{z} = \sum_j e_j \mathbf{z}_j \in R_q^{\ell+k}$
                    if $\|\mathbf{z}\|_2 \le B'$,
                      then return $\sigma_{agg} = ((\mathbf{u}_j)_j, \mathbf{z})$;
                    else
                      return $\perp$;
AggVf(VK, $M$, $\sigma_{agg}$) : Re-construct $c_j = H_c(\mathbf{u}_j, \mathbf{t}_j, m_j)$ for all $j \in [N]$
                    And $e_j \leftarrow H_e(c_1, \ldots, c_N, j)$ for all $j \in [N]$
                    If $\|\mathbf{z}\|_2 < B'$
                    and if $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} = \sum_j e_j(\mathbf{t}_j \cdot c_j + \mathbf{u}_j)$
                    return 1; else return 0;
```

Fig. 3. Non-Interactive Half-Aggregation of Signatures.

In order to aggregate N signatures $(\sigma_j)_{j\in[N]}$ the algorithm AggSig first reconstructs the challenges $c_j = H_c(\mathbf{u}_j, \mathbf{t}_j, m_j)$ for every $j \in [N]$ and queries H_e to obtain $(e_j = H_e(c_1, \ldots, c_N, j)$ for every index $j \in [N]$. It then computes the sum $\mathbf{z} = \sum_j e_j \mathbf{z}_j$ and outputs the aggregated signature $\sigma_{agg} = ((\mathbf{u}_j)_j, \mathbf{z})$, if the Euclidean norm of $\mathbf{z}$ is bounded above by B'. Else the algorithm outputs $\perp$. Note that $\|\mathbf{z}\|_2 = \left\|\sum_j e_j \mathbf{z}_j\right\|_2 \leq \sum_j \|e_j \mathbf{z}_j\|_2 \leq \sum_j \|e_j\|_2 \cdot \|\mathbf{z}_j\|_2 \leq N \cdot \sqrt{d} \cdot B = B'$ (at least with overwhelming probability). Thus, the probability that AggSig outputs $\perp$ is negligible. 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 commitment $\mathbf{u}_j$ provided in σ_{agg}, the verification key $\mathbf{t}_j$ and the message m_j. Then it re-constructs the scalars $e_j = H_e(c_1, \ldots, c_N, j)$ for every j. The algorithm then checks if the norm of $\mathbf{z}$ lies within the correct bound. Finally it verifies the equations $[\mathbf{A}, \mathbf{I}_k] \cdot \mathbf{z} = \sum_j (\mathbf{t}_j \cdot c_j + \mathbf{u}_j)$. If all checks go through, it outputs 1, else 0.

Remark 1. 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 [Dam+21]. 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 2 (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 check if $\|\mathbf{z}\|_2 \leq B'$ succeeds by the construction of AggSig. For the last check, we compute

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

where we used the linearity over R_q and the correctness of Π_S. $\square$

Remark 2. 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 [Dam+21]. In order to gain in efficiency, it can instead also be computed by some compact random seed.

3.3 Why Linear Compression Is A Bad Idea

In earlier versions of this work we followed the idea of $\mathsf{MMSA(TK)}$ [Dor+20] to further compress the aggregated signature. For the compression we used a linear function $T\colon R_q^k \to \mathbb{Z}_q^{n_0}$ and applied it to the commitment $\mathbf{u}_j$ before inputting it to the random oracle H_c. An aggregate signature was given by $\sigma_{agg} = (\sum_j \mathbf{u}_j, T(\mathbf{u}_j)_j, \sum_j \mathbf{z}_j)$.[7] The key idea behind the compression was that $(\sum_j \mathbf{u}_j, T(\mathbf{u}_j)_{j\in[N]})$ is much smaller than the complete vector $(\mathbf{u}_j)_{j\in[N]}$. At the same time, n_0 is chosen large enough to guarantee combinatorial security (that is $2^\lambda \leq q^{n_0}$) 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.

In order to make aggregation compatible with the single signatures, the signing procedure (for a single signature) has to be modified. Instead of inputting $\mathbf{u}_j$ to H_c, only the compressed vector $T(\mathbf{u}_j)$ serves as input, see Figure 4. As we elaborate in the following, this defines an *insecure* signature scheme.[8]

```
$\mathsf{Sig}(\mathsf{sk}, m)$ :  set $\mathbf{z} = \perp$
               *while* $\mathbf{z} = \perp$ *do*:
                 sample $\mathbf{y} \leftarrow D$
                 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 $1 - \Pr_{rej}$
                   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. 4. The signature scheme from [GLP12] with linear compression.

More precisely, we define an attacker who is able to forge signatures on arbitrary messages for any given public key $\mathbf{t}$ without knowing the corresponding secret key $\mathbf{s}$. First, the attacker initiates the signing procedure by sampling $\mathbf{y}' \leftarrow D$

[7] For the sake of simplicity we omit here the scalars coming from the hash function H_e, as they are irrelevant for the attack. Further, it makes the presentation closer to the previous version.

[8] This attack has been described to us by Thomas Prest.

and computes $\mathbf{u}' = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y}'$. They then query $c = H_c(T(\mathbf{u}'), \mathbf{t}, m)$ to obtain the challenge for an arbitrary message m. Now, they exploit the fact that T reduces the lattice dimension from nk to n_0. More precisely, they use standard lattice reduction to find a short solution $\mathbf{z}$ to the equation $T[\mathbf{A}|\mathbf{I}_k]\mathbf{z} = T(\mathbf{u}' + \mathbf{t} \cdot c)$ and set $\mathbf{u} := [\mathbf{A}|\mathbf{I}_k]\mathbf{z} - \mathbf{t} \cdot c$. That is, they solve inhomogeneous SIS for the integer matrix $\mathbf{A}' = T[\mathbf{A}|\mathbf{I}_k] \in \mathbb{Z}_q^{n_0 \times (n(\ell+k))}$ (ignoring the structure of $\mathbf{A}'$). By the *linearity* of T, it yields $T(\mathbf{u}) = T(\mathbf{u}')$ and thus $\sigma = (\mathbf{u}, \mathbf{z})$ defines a valid forgery for the public key $\mathbf{t}$. Note that there may not exist a short $\mathbf{y}$ such that $[\mathbf{A}, \mathbf{I}_k] \cdot \mathbf{y} = \mathbf{u}$, but assuming the hardness of decisional M-LWE, both $\mathbf{u}$ and $\mathbf{u}'$ are computationally close to uniform random vectors.

4 Security of Our Aggregate Signature Scheme

We recall in Sec. 4.1 the security model for aggregate signatures from Boneh et al. [Bon+03], before proving in Sec. 4.2 the security of our scheme from Sec. 3.

4.1 The Aggregate Chosen-Key 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. [Bon+03]. 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 3 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 5. *Let H denote 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 independent-chosen-key model. 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, H_e be two hash function providing the challenges and scalars in our aggregate signature, modeled as random oracles both with image space C. Assume the hardness of* $\text{M-LWE}_{k,\ell,\beta}$ *and of* $\text{M-SIS}_{k,\ell+1,b}$*, where $b = 16d \cdot B' + 8d^{3/2}$. Then, the aggregate signature Π_{AS} with parameters (ℓ, k, β, B', d) 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*

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

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

Proof Sketch. Let us first describe the high level idea of the proof, which follows the "double forking technique" of [Max+19]. To begin, note that, assuming the hardness of $\text{M-LWE}_{k,\ell,\beta}$, the honestly generated challenge key $\mathsf{vk}_N = \mathbf{t} = [\mathbf{A}, \mathbf{I}_k] \cdot \mathbf{s}_N$ is computationally indistinguishable from a uniformly random vector over R_q. Thus, one can replace the input $(\mathbf{A}, \mathbf{t})$ by a real uniform element, which itself defines an instance of $\text{M-SIS}_{k,\ell+1,b}$. Now, we construct a so-called wrapper $\mathcal{B}$ around the adversary $\mathcal{A}$, that internally invokes $\mathcal{A}$ to obtain a signature forgery. The wrapper has the important role to simulate the random oracles H_c and H_e and the signing queries with respect to the challenge key. They output the received forgery, together with some extra information about the forger execution. We then apply twice the General Forking Lemma (Lemma 1). First, we construct an algorithm $\mathcal{C}$ that runs the forking algorithm $\mathcal{F}_{\mathcal{B}}$ on $\mathcal{B}$, where the fork is with respect to the answer of H_e to the forgery and the challenge index N. Second, we construct another algorithm $\mathcal{D}$ that runs now the forking algorithm $\mathcal{F}_{\mathcal{C}}$ on $\mathcal{C}$, where the fork happens regarding the answer of H_c to the forgery and the challenge index N. At the end, the wrapper $\mathcal{B}$ has been run a total of 4 times, leading to four forgeries. The way how $\mathcal{D}, \mathcal{C}$ and $\mathcal{B}$ are defined allows to derive a solution to M-SIS.

Proof. Let $\mathcal{A}$ be an adversary against Π_{AS} with advantage $\mathsf{Adv\ AggSig}_{\mathcal{A}}$. 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 then apply twice the General Forking Lemma, leading to four different forgeries. Those then allow to construct a solution to M-SIS.

G_0: Set $N_q = N_{H_c} + N_{H_e} + N_{\mathsf{Sig}}$, where $\mathcal{A}$ makes at most N_{H_c} queries to H_c, N_{H_e} queries to H_e and N_{Sig} queries to the signing oracle on vk_N. Recall that C denotes the challenge space from Sig (and the scalar space from AggSig). Let $\mathcal{B}$ be a second algorithm that is provided with some randomly chosen $h_j, h'_j \leftarrow U(C)$ 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. Similarly, for the random oracle H_e, a table HT_e is maintained. Further $\mathcal{B}$ stores two counters ctr_c and ctr_e, both 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 $x = (\mathbf{u}, \mathbf{t}, m)$, 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_c and set $\mathsf{HT}_c[x] = h_{\mathsf{ctr}_c}$. Else they sample $\mathsf{HT}_c[x] \leftarrow U(C)$. Finally, they output $c = \mathsf{HT}_c[x]$.

Queries on H_e. On input $x = (c_1, \ldots, c_N, j)$ for some $j \in [N]$, if $\mathsf{HT}_e[x]$ is already set, $\mathcal{B}$ returns $\mathsf{HT}_c[x]$. Else, $\mathcal{B}$ does the following. First, for all $i \in [N-1]$, they sample $\mathsf{HT}_e[x_i] \leftarrow U(C)$ for $x_i = (c_1, \ldots, c_N, i)$ (independently of j). Only then, they increment ctr_e and set $\mathsf{HT}_e[x_N] = h'_{\mathsf{ctr}_e}$ for $x_N = (c_1, \ldots, c_N, N)$. Finally, they output $c = \mathsf{HT}_c[x]$ (where $x = x_j$ for some j). The order is important here, as we use later that e_i has been fixed before e_N for all $i \in [N-1]$ for a fixed tuple $(c_1, \ldots, c_N)$.

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 $\sigma_{agg} = ((\mathbf{u}_j)_j, \mathbf{z},)$ 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 $x = (\mathbf{u}_N, \mathbf{t}_N, m_N)$ 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. With the same argumentation, we assume that HT_e was programmed on $x = (c_1, \ldots, c_N, N)$ and thus $e_N = h'_{j'_f}$ for some counter index j'_f. If $\mathsf{AggVf}(\mathsf{VK}, M, \sigma_{agg}) = 1$, then $\mathcal{B}$ outputs $(j'_f, j_f, (\mathsf{VK}, M, \sigma_{agg}, \mathbf{C}, \mathbf{E}))$, with $\mathbf{C} = (c_j)_{j \in [N]}$ such that $c_j = H_c(\mathbf{u}_j, \mathbf{t}_j, m_j)$ and $\mathbf{E} = (e_j)_j$ such that $e_j = H_e(c_1, \ldots, c_N, j)$. Else, $\mathcal{B}$ outputs $(0, 0, \perp)$.

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

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 $x = (\mathbf{u}, \mathbf{t}_N, m)$ 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 [Dam+21, 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 $|\Pr[G_2] - \Pr[G_1]| \leq \mathsf{Adv}_{\text{M-LWE}}$.

Using the above, we obtain $\mathsf{Adv}\ \mathsf{AggSig}_{\mathcal{A}} \leq \Pr[G_2] + \mathsf{Adv}_{\text{M-LWE}_{k,\ell,\beta}} + \mathsf{negl}(\lambda)$. Our aim for the reminder of the proof is to bound $\Pr[G_2]$. To do so, we apply twice the General Forking Lemma. Note that in game G_2 the matrix $\mathbf{A}' = [\mathbf{A}|\mathbf{t}_N]$ follows the uniform distribution over $R_q^{k\times(\ell+1)}$.

First Fork. We construct an algorithm $\mathcal{C}$ around $\mathcal{B}$ who invokes the General Forking Lemma (Lemma 1), where the input generator IGen is defined to output $\mathbf{A}'$. Let $\mathsf{acc}_{\mathcal{B}}$ denote the accepting probability of $\mathcal{B}$ and $\mathsf{frk}_{\mathcal{B}}$ the forking probability of $\mathsf{F}_{\mathcal{B}}$ as defined in Lemma 1. Further, set $\mathsf{out} = (j_f, \mathsf{VK}, M, \sigma_{agg}, \mathbf{C}, \mathbf{E})$ and feed $\mathcal{B}$ with the input coins $h'_1, \ldots, h'_{N_q}$. The fork happens with respect to the counter index from ctr_e. Thus, the forking algorithm $\mathsf{F}_{\mathcal{B}}$ outputs with probability $\mathsf{frk}_{\mathcal{B}}$ two different outputs $\mathsf{out}, \widetilde{\mathsf{out}} \neq (\perp, \perp)$, where $\mathsf{acc}_{\mathcal{B}} \leq N_q/|C| + \sqrt{N_q \cdot \mathsf{frk}_{\mathcal{B}}}$. Let $\mathsf{out} = (j_f, \mathsf{VK}, M, \sigma_{agg}, \mathbf{C}, \mathbf{E})$ and $\widetilde{\mathsf{out}} = (\tilde{j}_f, \widetilde{\mathsf{VK}}, \tilde{M}, \tilde{\sigma}_{agg}, \tilde{\mathbf{C}}, \tilde{\mathbf{E}})$. During the forking, neither $\mathcal{B}$ nor $\mathcal{A}$ is aware of being rewound. In particular, before arriving at the fork, they behave exactly the same in both executions. As the random coins of $\mathcal{B}$ are the same in both executions and as the simulation of H_e is for $j \neq N$ independent of the input $(h'_j)_{j\in[N_q]}$ to $\mathcal{B}$, we have $\mathbf{t}_j = \tilde{\mathbf{t}}_j$, $\mathbf{u}_j = \tilde{\mathbf{u}}_j$ and $c_j = \tilde{c}_j$ for all $j \in [N]$ and $e_j = \tilde{e_j}$ for all $j \in [N-1]$. Further it yields $M = \tilde{M}$. As in both cases, the forgery passes validation, it yields $\|\mathbf{z}\|_2, \|\tilde{\mathbf{z}}\|_2 < B'$. Additionally, $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} = \sum_{j\in[N]} e_j(\mathbf{t}_j \cdot c_j + \mathbf{u}_j)$ and $[\mathbf{A}|\mathbf{I}_k] \cdot \tilde{\mathbf{z}} = \sum_{j\in[N]} \tilde{e}_j(\tilde{\mathbf{t}}_j \cdot \tilde{c}_j + \tilde{\mathbf{u}}_j)$. Hence, we can deduce that

$$[\mathbf{A}|\mathbf{I}_k] \cdot (\mathbf{z} - \tilde{\mathbf{z}}) = (e_N - \tilde{e}_N)(\mathbf{t}_N c_N + \mathbf{u}_N). \tag{1}$$

Note that Equation 1 is not yet sufficient to extract a solution to M-SIS and we thus fork a second time.

Second Fork. We now construct another algorithm $\mathcal{D}$ around $\mathcal{C}$ who invokes the General Forking Lemma, where the input generator IGen is again defined to output $\mathbf{A}'$. Let $\mathsf{acc}_{\mathcal{C}}$ denote the accepting probability of $\mathcal{C}$ and $\mathsf{frk}_{\mathcal{C}}$ the forking probability of $\mathsf{F}_{\mathcal{C}}$ as defined in Lemma 1. We observe that $\mathsf{acc}_{\mathcal{C}} = \mathsf{frk}_{\mathcal{B}}$. Further, set $\mathsf{out} = (\mathsf{VK}, M, \sigma_{agg}, \mathbf{C}, \mathbf{E})$ and feed $\mathcal{C}$ with the input coins $h_1, \ldots, h_{N_q}$. This time, the fork is on the counter ctr_c. Thus, the forking algorithm $\mathsf{F}_{\mathcal{C}}$ outputs with probability $\mathsf{frk}_{\mathcal{C}}$ two different outputs $\mathsf{out}, \widetilde{\mathsf{out}} \neq (\perp, \perp)$, where $\mathsf{acc}_{\mathcal{C}} \leq N_q / |C| + \sqrt{N_q \cdot \mathsf{frk}_{\mathcal{C}}}$. More precisely, let $\mathsf{out} = (\mathsf{VK}, M, \sigma_{agg}, \mathbf{C}, \mathbf{E})$ and $\widetilde{\mathsf{out}} = (\widetilde{\mathsf{VK}}, \tilde{M}, \tilde{\sigma}_{agg}, \tilde{\mathbf{C}}, \tilde{\mathbf{E}})$. With the same argumentation as above, we have $m_j = m'_j$, $\mathbf{u}_j = \mathbf{u}'_j$ and $\mathbf{t}_j = \mathbf{t}'_j$ for all $j \in [N]$ and $c_j = c'_j$ for all $j \in [N-1]$. Furthermore, we have $c_N \neq c'_N$. We hence obtain

$$[\mathbf{A}|\mathbf{I}_k] \cdot (\mathbf{z}' - \tilde{\mathbf{z}}') = (e'_N - \tilde{e}'_N)(\mathbf{t}_N c'_N + \mathbf{u}_N) \tag{2}$$

Multiplying Equation 1 by $\varepsilon' := (e'_N - \tilde{e}'_N)$ and Equation 2 by $\varepsilon := (e_N - \tilde{e}_N)$ and then combining both, we obtain

$$[\mathbf{A}|\mathbf{I}_k] \cdot (\varepsilon'(\mathbf{z} - \tilde{\mathbf{z}}) - \varepsilon(\mathbf{z}' - \tilde{\mathbf{z}}')) = \varepsilon\varepsilon'(\mathbf{t}_N(c_N - c'_N)). \tag{3}$$

In other words, $\mathcal{D}$ can compute the vector $\mathbf{x} = (\varepsilon'(\mathbf{z} - \tilde{\mathbf{z}}) - \varepsilon(\mathbf{z}' - \tilde{\mathbf{z}}'), \varepsilon\varepsilon'(c'_N - c_N))^T$ which is a solution to the M-SIS problem for the matrix $\mathbf{A}' = [\mathbf{A}|\mathbf{t}_N]$. The Euclidean norm of the vector $\mathbf{x}$ is bounded above by

$$\|\mathbf{x}\|_2 \leq 4 \cdot \|\varepsilon\mathbf{z}\|_2 + 2 \cdot \left\|\varepsilon^2 c\right\|_2 \leq 4 \cdot 4d \cdot B' + 2 \cdot 4d^{3/2} = b,$$

where B' is the bound on $\mathbf{z}$. This implies that $\mathsf{frk}_{\mathcal{C}} \leq \mathsf{Adv}_{\text{M-SIS}}$. Overall, we get

$$\Pr[G_2] = \mathsf{acc}_{\mathcal{B}} \leq \frac{N_q}{|C|} + \sqrt{N_q \cdot \left(\frac{N_q}{|C|} + \sqrt{N_q \cdot \mathsf{Adv}_{\text{M-SIS}_{\mathsf{k}, \ell+1, \mathsf{b}}}}\right)},$$

completing the proof. □

Remark 3. Using twice 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. [Abd+16] 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. We leave it as an open problem to further investigate if those techniques can be applied to our aggregate signature scheme.

4.3 Rogue-Attack Against the Simple Sum

We just showed that the aggregate signature scheme Π_{AS} as presented in Section 3 is proven secure in the aggregate chosen-key model as introduced by Boneh et al. [Bon+03]. When aggregating, it is important to use random scalars for the sum. Instead of the simple (deterministic) sum, we hence obtain a (randomized) linear combination of the single signature parts $\mathbf{z}_j$.[9]

As we think this is important to understand this motivation, we now describe a simple attack if one uses only the simple sum (or a linear combination where the challenge space is too small, as done in the current version of $\mathsf{MMSA(TK)}$), which was first observed by Zhao [Zha19] in the context of a Schnorr-based aggregate signature scheme on elliptic curves.

1. Let $\mathbf{t}_N$ be the challenge key given tot he adversary $\mathcal{A}$.
2. $\mathcal{A}$ generates the remaining key pairs $(\mathbf{s}_j, \mathbf{t}_j) \leftarrow \mathsf{KGen}$ for $j \in [N-1]$ and selects arbitrary messages m_j for $j \in [N]$.
3. They sample $\mathbf{y}_N \leftarrow D$, set $\mathbf{u}_N = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y}_N$ and query the random oracle to obtain $c_N = H_c(\mathbf{u}_N, \mathbf{t}_N, m_N)$.
4. Now, $\mathcal{A}$ can prepare a *rogue-commitment* $\mathbf{u}_{N-1}$ by sampling $\mathbf{y}_{N-1} \leftarrow D$ and setting $\mathbf{u}_{N-1} = -\mathbf{t}_N \cdot c_N + [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y}_{N-1}$, depending on the challenge c_N for the challenge key $\mathbf{t}_N$.
5. They compute $c_{N-1} = H_c(\mathbf{u}_{N-1}, \mathbf{t}_{N-1}, m_{N-1})$.
6. They set $\mathbf{z}_{N-1} = \mathbf{y}_{N-1} + \mathbf{s}_{N-1} \cdot c_{N-1}$ and apply the rejection sampling in order to make the distribution of $\mathbf{z}_{N-1}$ independent of $\mathbf{s}_{N-1}$.
7. For the remaining $j \in [N-2]$, they follow the honest signature procedure: they sample $\mathbf{y}_j \leftarrow D$, set $\mathbf{u}_j = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y}_j$, compute $c_j = H_c(\mathbf{u}_j, \mathbf{t}_j, m_j)$ and set $\mathbf{z}_j = \mathbf{s}_j \cdot c_j + \mathbf{y}_j$. Then they apply the rejection sampling in order to make the distribution of $\mathbf{z}_j$ independent of $\mathbf{s}_j$.
8. Finally, they output the forgery $\sigma_{agg} = (\hat{\mathbf{u}}, (\mathbf{u}_j)_j, \mathbf{z})$, where $\hat{\mathbf{u}} = \sum_{j=1}^{N} \mathbf{u}_j$ and $\mathbf{z} = \mathbf{y}_N + \sum_{j=1}^{N-1} \mathbf{z}_j$.

We claim that the forgery passes verification. As all $\mathbf{z}_j \sim D$ and $\mathbf{y} \sim D$, the norm of the sum $\mathbf{z}$ is (with overwhelming probability) bounded above by $N \cdot B =: B'$. Further, it yields $\hat{\mathbf{u}} = \sum_j \mathbf{u}_j$, as the $\mathbf{u}_j$ are computed by honestly compressing

[9] For the deterministic sum, one can only show security in a much more restricted security model where the adversary has to commit themselves to the signatures they will use in the aggregate signature scheme *before* receiving the challenge key. More details can be found in an older version of this e-print (8 Apr 2021).

the corresponding $\mathbf{u}_j$. It remains to show that $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} = \sum_{j=1}^{N}(\mathbf{t}_j \cdot c_j) + \hat{\mathbf{u}}$.

$$\begin{aligned}
[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z} &= [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y}_N + \sum_{j=1}^{N-1} [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{z}_j \\
&= \mathbf{u}_N + [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y}_{N-1} + \mathbf{t}_{N-1} \cdot c_{N-1} + \sum_{j=1}^{N-2} \mathbf{t}_j \cdot c_j + \mathbf{u}_j \\
&= \mathbf{u}_N + \mathbf{u}_{N-1} + \mathbf{t}_N \cdot c_N + \mathbf{t}_{N-1} \cdot c_{N-1} + \sum_{j=1}^{N-2} \mathbf{t}_j \cdot c_j + \mathbf{u}_j \\
&= \sum_{j=1}^{N} \mathbf{t}_j \cdot c_j + \hat{\mathbf{u}},
\end{aligned}$$

where we used that $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{y}_{N-1} = \mathbf{u}_{N-1} + \mathbf{t}_N \cdot c_N$. In this attack the adversary exploits the fact that they can define a rogue-commitment with respect to the challenge c_N. In our security model, this is prevented as the adversary has to fix the signatures they are going to use in the forgery *before* receiving the challenge key $\mathbf{t}_N$.

Acknowledgments

This work was supported by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701). It has also received a French government support managed by the National Research Agency in the "Investing for the Future" program, under the national project RISQ P141580-2660001 / DOS0044216. Katharina Boudgoust was funded by the Direction Générale de l'Armement (Pôle de Recherche CYBER). We warmly thank Akira Takahashi for making us aware of the attacks against Schnorr-based aggregate signatures in the chosen-key model and discussions with him and Claudio Orlandi. We also would like to thank Olivier Sanders for an interesting exchange regarding the security proof. Lastly, we also would like to thank Thomas Prest who explained us his attack on the linear compression function used in an earlier version of this paper.

References

[Abd+16] Michel Abdalla et al. "Tightly Secure Signatures From Lossy Identification Schemes". In: *J. Cryptol.* 29.3 (2016), pp. 597–631.

[BN06] Mihir Bellare and Gregory Neven. "Multi-signatures in the plain public-Key model and a general forking lemma". In: *CCS*. ACM, 2006, pp. 390–399.

[Bon+03] Dan Boneh et al. "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps". In: *EUROCRYPT*. Vol. 2656. Lecture Notes in Computer Science. Springer, 2003, pp. 416–432.

[Bou+20] Katharina Boudgoust et al. “Towards Classical Hardness of Module-LWE: The Linear Rank Case”. In: *ASIACRYPT (2)*. Vol. 12492. Lecture Notes in Computer Science. Springer, 2020, pp. 289–317.

[BS16] Rachid El Bansarkhani and Jan Sturm. “An Efficient Lattice-Based Multisignature Scheme with Applications to Bitcoins”. In: *CANS*. Vol. 10052. Lecture Notes in Computer Science. 2016, pp. 140–155.

[CDW21] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. “Mildly Short Vectors in Cyclotomic Ideal Lattices in Quantum Polynomial Time”. In: *J. ACM* 68.2 (2021), 8:1–8:26.

[Cha+21] Konstantinos Chalkias et al. “Non-interactive Half-Aggregation of EdDSA and Variants of Schnorr Signatures”. In: *CT-RSA*. Vol. 12704. Lecture Notes in Computer Science. Springer, 2021, pp. 577–608.

[Dam+21] Ivan Damgård et al. “Two-Round n-out-of-n and Multi-signatures and Trapdoor Commitment from Lattices”. In: *Public Key Cryptography (1)*. Vol. 12710. Lecture Notes in Computer Science. Springer, 2021, pp. 99–130.

[Dor+20] Yarkin Doröz et al. “MMSAT: A Scheme for Multimessage Multiuser Signature Aggregation”. In: *IACR Cryptol. ePrint Arch.* (2020), p. 520.

[Duc+18] Léo Ducas et al. “CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.1 (2018), pp. 238–268.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. “Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems”. In: *CHES*. Vol. 7428. Lecture Notes in Computer Science. Springer, 2012, pp. 530–547.

[Hof+14] Jeffrey Hoffstein et al. “Practical Signatures from the Partial Fourier Recovery Problem”. In: *ACNS*. Vol. 8479. Lecture Notes in Computer Science. Springer, 2014, pp. 476–493.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. “A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model”. In: *EUROCRYPT (3)*. Vol. 10822. Lecture Notes in Computer Science. Springer, 2018, pp. 552–586.

[LS15] Adeline Langlois and Damien Stehlé. “Worst-case to average-case reductions for module lattices”. In: *Des. Codes Cryptogr.* 75.3 (2015), pp. 565–599.

[Lyu09] Vadim Lyubashevsky. “Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures”. In: *ASIACRYPT*. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 598–616.

[Lyu12] Vadim Lyubashevsky. “Lattice Signatures without Trapdoors”. In: *EUROCRYPT*. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 738–755.

[Max+19] Gregory Maxwell et al. “Simple Schnorr multi-signatures with applications to Bitcoin”. In: *Des. Codes Cryptogr.* 87.9 (2019), pp. 2139–2164.

[Sch91] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards". In: *J. Cryptol.* 4.3 (1991), pp. 161–174.

[Zha19] Yunlei Zhao. "Practical Aggregate Signature from General Elliptic Curves, and Applications to Blockchain". In: *AsiaCCS*. ACM, 2019, pp. 529–538.