# Compact Aggregate Signature from Module-Lattices

Toi Tomita[1] and Junji Shikata[1]

Yokohama National University, Yokohama, Japan.
{tomita-toi-sk, shikata-junji-rb}@ynu.ac.jp

**Abstract.** We propose the first aggregate signature scheme such that: (1) its security is based on the standard lattice assumptions in the random oracle model; (2) the aggregate signature size is logarithmic; (3) it is not one-time; and (4) it supports non-interactive aggregation. To obtain such a scheme, we combine the most compact SNARK (Succinct Non-interactive ARgument of Knowledge) system and a SNARK-friendly signature scheme. As a result, our aggregated signature size is sufficiently compact. For example, the size required to aggregate $2^{20}$ signatures is only a few hundred kilobytes. This result shows that our scheme is superior to the existing lattice-based schemes in compressing many signatures.

## 1 Introduction

### 1.1 Background

The notion of aggregate signature schemes, introduced by Boneh, Gentry, Lynn, and Shacham [7], allows individual signatures $\sigma_1, \ldots, \sigma_N$ for *different* messages $\mathsf{M}_1, \ldots, \mathsf{M}_N$ created by $N$ signers to be aggregated into a compact signature $\sigma_{\mathsf{agg}}$. The aggregated signature $\sigma_{\mathsf{agg}}$ gives the verifier confidence that all the signatures aggregated into $\sigma_{\mathsf{agg}}$ are valid. The original motivation for signature aggregation was to compress certificate chains and aggregate signatures in secure BGP. More recently, it has gained significant practical interest in the context of blockchains.

There is a plethora of work on constructing aggregate signature schemes by using bilinear maps [7,23,6,4,17] or trapdoor permutations [24,30,12,19]. On the other hand, post-quantum, in particular, lattice-based aggregate signature schemes have not been widely proposed. The first lattice-based scheme was proposed by Döroz et al. [16]. However, this scheme was found to be either less efficient than a trivial concatenation of signatures or more vulnerable to attack by compression techniques, as pointed out by Boudgoust and Roux-Langlois [9,10]. Boudgoust and Roux-Langlois also presented in [9,10] a (module) lattice-based scheme following the Fiat-Shamir with aborts paradigm. Boneh and Kim [8] proposed two types of lattice-based schemes such that the security is based on the standard Short Integer Solution (SIS) assumption and the aggregate signature size is logarithmic in the number of signatures to be aggregated. However, the first scheme is a *one-time* scheme, and the second scheme requires *interactions* *for aggregation.* Sato and Shikata [32] presented the first identity-based aggregate signatures scheme, although interaction is necessary for aggregation. Recently, Jeudy et al. [20] proposed a (module) lattice-based scheme following the new hash-and-sign with aborts technique. Unfortunately, the aggregated signature size of the schemes [9,10,20] is *linear* in the number of signatures being aggregated. Based on the above literature, we raise the following natural question in this paper:

*Can we construct a lattice-based aggregate signature scheme such that: (1) its security is based on the standard lattice-based assumptions; (2) the aggregate signature size is logarithmic; (3) it is not one-time (i.e., many-time); and (4) it supports non-interactive aggregation?*

### 1.2 Our Contributions

In this paper, we answer the above question in the affirmative: in this paper, we construct a lattice-based aggregate signature scheme that meets all the conditions (1)–(4). Table 7 provides a comparison between our aggregate signature scheme and the existing ones. As we can see from the table, our scheme is the first one that meets all the conditions (1)–(4). The main idea of our construction is to construct a succinct non-interactive argument of knowledge ($\mathsf{SNARK}$) system for Batch NP such that a lattice-based signature can be combined with it to obtain the resultant aggregate signature scheme satisfying the conditions (1)–(4).

**Table 1.** Comparison of lattice-based aggregate signature schemes. The column $|\sigma_{\mathsf{agg}}|$ indicates the size of the aggregate signature. $N$ is the number of signatures to be aggregated.

| Scheme | Aggregated Sig. Size | Many-time | Non-interactive | Assumption |
|---|---|---|---|---|
| [8, Sec. 4] | $O(\log N)$ | - | ✓ | SIS |
| [8, Sec. 6] | $O(\log N)$ | ✓ | - | SIS |
| [32] | $O(\log N)$ | ✓ | - | SIS |
| [9,10,20] | $\tilde{O}(N)$ | ✓ | ✓ | MSIS & MLWE |
| **Ours** | $\tilde{O}(\log N)$ | ✓ | ✓ | MSIS & MLWE |

Here, a $\mathsf{SNARK}$ system for Batch NP allows a prover to construct a proof of $N$ NP statements, where the size of the proof grows sublinearly with $N$, and to convince the verifier that all these statements are true. By the following straightforward construction, a $\mathsf{SNARK}$ system for Batch NP directly yields an aggregate signature scheme. Consider the NP relation $\mathcal{R}$, which takes the verification key-message pair $(\mathsf{vk}, \mathsf{M})$ as an NP statement and the signature $\sigma$ as an NP witness, and $((\mathsf{vk}, \mathsf{M}), \sigma) \in \mathcal{R}$ if and only if $\sigma$ is a valid signature on $\mathsf{M}$ under $\mathsf{vk}$. An aggregate signature on $(\mathsf{vk}_1, \mathsf{M}_1, \sigma_1), \ldots, (\mathsf{vk}_N, \mathsf{M}_N, \sigma_N)$ is a $\mathsf{SNARK}$ proof that $((\mathsf{vk}_i, \mathsf{M}_i), \sigma_i) \in \mathcal{R}$ for all $i = 1, \ldots, N$. The compactness of the $\mathsf{SNARK}$ system ensures that the size of the aggregate signature is sublinear in $N$. Recently, several lattice-based SNARK systems for Batch NP have been proposed [14,2,15,31,21,11]. However, these SNARK systems are not so practical in the real world though they are asymptotically efficient from a theoretical viewpoint.

In order to construct a practical aggregate signature scheme, we take the following approach in this paper. We adopt LaBRADOR [5], currently the most compact, as our SNARK system. We then combine the SNARK system and a SNARK-friendly variant of Lyubashevsky's signature scheme [13] to construct an aggregate signature scheme. Here, "SNARK-friendly" means that the signature verification equations can be described by simple NP relations, in particular, the signature scheme does not require the computation of a (complex) cryptographic hash function with the signature as input during verification. This property is similar to the so-called "structure-preserving" property. Thanks to this friendliness, we can avoid the overheads incurred by converting the verification circuit to a quadratic format.

Consequently, the resultant aggregate signature scheme is concretely compact as shown in Table 2. Table 2 shows the aggregated signature size in our scheme for varying the number $N$ of signatures to be aggregated from $2^{10}$ to $2^{20}$. For example, our aggregated signature size requires 63 KB and 132 KB for approximately $N = 2^{10}$ and $N = 2^{20}$, respectively. This result shows that our scheme is superior to the existing lattice-based schemes in compressing many signatures.

**Table 2.** Aggregated signature size of our scheme. The first row indicates the number of signatures to be aggregated. The second row indicates the aggregated signature sizes of our scheme.

| $N$ | Aggregated Sig. Size |
|---|---|
| $2^{10}$ | 63.48 KB |
| $2^{12}$ | 65.02 KB |
| $2^{14}$ | 69.38 KB |
| $2^{16}$ | 77.46 KB |
| $2^{18}$ | 103.42 KB |
| $2^{20}$ | 131.54 KB |

# 2 Preliminaries

## 2.1 Notation

For a positive integer $n \in \mathbb{N}$, let $[n]$ denote the set of integers $\{1, \ldots, n\}$. For a distribution $\mathcal{X}$, let $x \xleftarrow{\$} \mathcal{X}$ denote the process of sampling the value $x$ according to the distribution $\mathcal{X}$. Let $x \xleftarrow{\$} \mathcal{S}$ denote the process of sampling $x$ according to a uniform distribution on a finite set $\mathcal{S}$. Let $\mathsf{negl}(\lambda)$ be a negligible function.

**Rings.** Let $q \in \mathbb{N}$ be a modulus and $\mathbb{Z}_q$ be the ring of integers modulo $q$. For a positive integer $n \in \mathbb{N}$, we denote by $\vec{a} \in \mathbb{Z}_q^n$ a vector over $\mathbb{Z}_q$ and by $a_i \in \mathbb{Z}_q$ the

$i$-th entry of $\vec{a}$, i.e., $\vec{a} = (a_1, \dots, a_n)^\top$. Let $I_n \in \{0,1\}^{n \times n}$ be the $n$-by-$n$ identity matrix. Let $d \in \mathbb{N}$ be a power of two and let $\mathcal{R}$ and $\mathcal{R}_q$ be the polynomial rings $\mathbb{Z}[X]/(X^d+1)$ and $\mathbb{Z}_q[X]/(X^d+1)$, respectively. We denote column vectors over $\mathcal{R}$ or $\mathcal{R}_q$ by bold lowercase letters such as $\mathbf{a}$, and matrices over $\mathcal{R}$ or $\mathcal{R}_q$ by bold uppercase letters such as $\mathbf{A}$. If $a = a_0 + a_1 X + \cdots + a_{d-1} X^{d-1} \in \mathcal{R}_q$, then we denote by $\mathsf{ct}(a)$ the constant term of $a$, i.e., $\mathsf{ct}(a) = a_0 \in \mathbb{Z}_q$.

**Norms.** For $a = a_0 + a_1 X + \cdots + a_{d-1} X^{d-1} \in \mathcal{R}$, we have the coefficient norm $\|a\|_2 = \sqrt{\sum_{i=0}^{d-1} |a_i|^2}$ and the infinity norm $\|a\|_\infty := \max_i a_i$. The norms are naturally extended to vectors $\mathbf{a} \in \mathcal{R}_q^n$ of polynomials, i.e., $\|\mathbf{a}\|_2 = \sqrt{\sum_{i=1}^{n} \|a_i\|_2^2}$ and $\|\mathbf{a}\|_\infty := \max_i \|a_i\|_\infty$. For $a \in \mathcal{R}$, we also have the operator norm $\|a\|_{\mathsf{op}} = \sup_{r \in \mathcal{R}} \|ar\|_2 / \|r\|_2$.

**The Conjugation Automorphism.** The ring $\mathcal{R}_q$ has a group of automorphisms $\mathsf{Aut}(\mathcal{R}_q)$ that is isomorphic to $\mathbb{Z}_{2d}^\times$. Let $\Sigma_i \in \mathsf{Aut}(\mathcal{R}_q)$ be defined by $\Sigma_i(X) = X^i$. For readability, we denote for an arbitrary vector $\mathbf{a} \in \mathcal{R}^n$:

$$\Sigma_i(\mathbf{a}) := (\Sigma_i(a_1), \dots, \Sigma_i(a_k)).$$

Let $\Sigma_{-1} \in \mathsf{Aut}(\mathcal{R}_q)$ be defined by $\Sigma_{-1}(X) = X^{-1}$. This was introduced in [26]. For coefficient vectors $\vec{a}, \vec{b} \in \mathbb{Z}_q^{nd}$ and its corresponding polynomial vectors $\mathbf{a}, \mathbf{b} \in R_q^n$, we have $\langle \vec{a}, \vec{b} \rangle = \mathsf{ct}(\langle \sigma_{-1}(\mathbf{a}), \mathbf{b} \rangle)$.

### 2.2 Lattices

**Gaussian.** For $\vec{x} \in \mathbb{Z}^d$, let $\rho_s(\vec{x}) := \exp(-\pi \|\vec{x}\|_2^2 / s^2)$ be a Gaussian function of parameter $s \in \mathbb{R}$. The discrete Gaussian distribution $\mathcal{D}_s^n$ is

$$\mathcal{D}_s^n(\vec{x}) := \frac{\rho_s(\vec{x})}{\sum_{\vec{y} \in \mathcal{R}^n} \rho_s(\vec{y})}.$$

To simplify notations, we occasionally use $a \xleftarrow{\$} \mathcal{D}_s$ to mean that the coefficient vector of $a \in R$ is sampled from $\mathcal{D}_s^d$. The definitions naturally extend to vectors over $\mathcal{R}^n$. Finally, let $\mathcal{S}_\eta$ denote the set of all elements in $a \in \mathcal{R}_q$ such that $\|a\|_\infty \le \eta$.

The following are useful lemmas for bounding the norm of an element sampled from a discrete Gaussian distribution.

**Lemma 1 ([29,25,1]).** *For any real $t > 0$ and $t' > 1$, we have*

$$\Pr_{\vec{x} \xleftarrow{\$} \mathcal{D}_s^n} [\|\vec{x}\|_\infty > ts] < 2n \cdot 2^{-\frac{\log e}{2} \cdot t^2},$$

$$\Pr_{\vec{x} \xleftarrow{\$} \mathcal{D}_s^n} [\|\vec{x}\|_2 > t' s \sqrt{n}] < 2^{n \cdot \left(\frac{\log e}{2}(1 - t'^2) + \log t'\right)}.$$

The following is the rejection sampling lemma.

**Lemma 2 (Rejection Sampling [25, Lemmas 4.3, 4.6]).** *Let $\mathcal{V} \subset \mathbb{Z}^m$ in which all elements have norm less than $T$, $\mathcal{H}$ be a distribution over $\mathcal{V}$, $\phi, \mathsf{err} \in \mathbb{R}$ be positive reals with $\mathsf{err} < 1$, and set $y \coloneqq \phi \cdot T$. Now, sample $\vec{e} \xleftarrow{\$} \mathcal{H}$ and $\vec{y} \xleftarrow{\$} \mathcal{D}_y^m$, set $\vec{z} \coloneqq \vec{e} + \vec{y}$, and run $b \xleftarrow{\$} \mathsf{Rej}(\vec{z}, \vec{e}, \phi, T, \mathsf{err})$ in Figure 1. Then, the probability that $b = \top$ is at least $(1 - \mathsf{err})/\mu(\phi, \mathsf{err})$ for*

$$\mu(\phi, \mathsf{err}) = \exp\left(\sqrt{\frac{-2\log \mathsf{err}}{\log e}} \cdot \frac{1}{\phi} + \frac{1}{2\phi^2}\right)$$

*and the distribution of $(\vec{e}, \vec{z})$ conditioned on $b = \top$ is within statistical distance of $\mathsf{err}/\mu(\phi, \mathsf{err})$ of the product distribution $\mathcal{H} \times \mathcal{D}_y^m$.*

$\mathsf{Rej}(\vec{z}, \vec{e}, \phi, T, \mathsf{err})$

---

$u \xleftarrow{\$} [0, 1)$

**if** $u > \dfrac{1}{\mu(\phi, \mathsf{err})} \cdot \exp\left(\dfrac{-2\vec{z}^\top \vec{e} + \|\vec{e}\|_2^2}{2y^2}\right)$ :

  **then return** $\perp$

**else return** $\top$

**Fig. 1.** Rejection Sampling.

As a concrete example often used, by setting $\phi = 11$ and $\mathsf{err} = 2^{-100}$, we get $\mu(\phi, \mathsf{err}) \approx 3$. We can also set for example $\phi = 14$ and $\mathsf{err} = 2^{-256}$ to obtain $\mu(\phi, \mathsf{err}) \approx 4$ if we want better statistical bounds.

**Gadget.** For any integer $k \geq 1$, we define the gadget matrix [28]

$$\mathbf{G}_{b,k} \coloneqq \mathbf{I}_k \otimes \mathbf{g}^\top \in \mathcal{R}_q^{k \times k\lceil \log_b q \rceil},$$

where $\mathbf{g} \coloneqq (1\|b\|\cdots\|b^{\lceil \log_b q \rceil - 1})^\top \in \mathcal{R}_q^{\lceil \log_b q \rceil}$. The function $\mathbf{G}_{b,k}^{-1} : \mathcal{R}_q^k \to \mathcal{R}_b^{k\lceil \log_b q \rceil}$ is the base-$b$ decomposition function. Then, for any vector $\mathbf{c} \in \mathcal{R}_q^k$, we have

$$\mathbf{G}_{b,k}\mathbf{G}_{b,k}^{-1}(\mathbf{c}) = \mathbf{c} \mod q.$$

**Hardness Assumptions.** We define the module short integer solutions ($\mathsf{MSIS}$) and module learning with errors ($\mathsf{MLWE}$) assumptions, first introduced in [22].

**Definition 1 (MSIS Assumption).** *Let $n, m \in \mathbb{N}$ be positive integers and $\beta \in \mathbb{R}$ be a positive real with $0 < \beta < q$. For an algorithm $\mathcal{A}$, the advantage of the module short integer solutions $\mathsf{MSIS}_{n,\beta,q}$ problem of $\mathcal{A}$ is defined as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{msis}}(\lambda) \coloneqq \Pr\left[\mathbf{A}\mathbf{s} = \mathbf{0} \bmod q \wedge 0 < \|\mathbf{s}\|_2 \leq \beta \mid \mathbf{s} \leftarrow \mathcal{A}(1^\lambda, \mathbf{A})\right],$$

*where* $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$*. We say the* $\mathsf{MSIS}_{n,\beta,q}$ *assumption holds if the above advantage is negligible for all probabilistic polynomial time (PPT) algorithms* $\mathcal{A}$*.*

**Definition 2 (MLWE Assumption).** *Let* $n, l \in \mathbb{N}$ *be a positive integer and* $\eta \in \mathbb{R}$ *be a positive real. For an algorithm* $\mathcal{A}$*, the advantage of the module short integer solutions* $\mathsf{MLWE}_{n,\eta,q}$ *problem of* $\mathcal{A}$ *is defined as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{mlwe}}(\lambda) := \left|\Pr\left[\mathcal{A}(1^\lambda, \mathbf{A}, \mathbf{AS}+\mathbf{E}) \to 1\right] - \Pr\left[\mathcal{A}(1^\lambda, \mathbf{A}, \mathbf{B}) \to 1\right]\right|,$$

*where* $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n\times n}$*,* $\mathbf{S}, \mathbf{E} \xleftarrow{\$} \mathcal{S}_\eta^{n\times m}$*, and* $\mathbf{B} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$*. We say the* $\mathsf{MLWE}_{n,\eta,q}$ *assumption holds if the above advantage is negligible for all PPT algorithms* $\mathcal{A}$*.*

**Challenge Space.** Let $\tau_{\mathsf{Ch}}, T_{\mathsf{Ch}} \in \mathbb{R}$ be positive reals. Let $\mathsf{Ch} \subset \mathcal{R}$ be a challenge space such that $\mathbf{c}-\mathbf{c}'$ is invertible for any pair of distinct $\mathbf{c}, \mathbf{c}' \in \mathsf{Ch}$ and $\|\mathbf{c}\|_2 \le \tau_{\mathsf{Ch}}$ and $\|\mathbf{c}\|_{\mathsf{op}} \le T_{\mathsf{Ch}}$ for all $\mathbf{c} \in \mathsf{Ch}$.

In our concrete instantiations, we use the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(X^{64}+1)$, and as challenges we use ring elements with 23 zero coefficients, 31 coefficients that are $\pm 1$, and 10 coefficients that are $\pm 2$. There are over $2^{128}$ such elements. All these polynomials have the norm 8.43, and we use the rejection sampling [25] to restrict to challenges with operator norm at most 15. (On average, we need to sample about 6 elements before sampling an element $\mathbf{c}$ with $\|\mathbf{c}\|_{\mathsf{op}} < 15$). Differences in different challenges are invertible according to [27].

**Approximate Proofs of Smallness.** The following lemma, provided in [18], can be used to efficiently prove the smallness of a long vector.

**Lemma 3 ([18, Lemma 3.4]).** *Let* $q, d \in \mathbb{N}$ *be positive integers. Let* $\mathsf{Ch}_{\mathsf{mJL}}$ *be a distribution on* $\{-1, 0, 1\}$ *with* $\Pr[\mathsf{Ch}_{\mathsf{mJL}} = 0] = 1/2$ *and* $\Pr[\mathsf{Ch}_{\mathsf{mJL}} = 1] = \Pr[\mathsf{Ch}_{\mathsf{mJL}} = -1] = 1/4$*. Then, for any vector* $\vec{w} \in [\pm q/2]^d$*, we have*

$$\Pr\left[\|P\vec{w} \bmod q\|_\infty < \frac{1}{2}\|\vec{w}\|_\infty : P \xleftarrow{\$} \mathsf{Ch}_{\mathsf{mJL}}^{128\times d}\right] < 2^{-128}.$$

### 2.3 Succinct Non-Interactive Argument of Knowledge in the Random Oracle Model

We consider the succinct non-interactive argument of knowledge ($\mathsf{SNARK}$) system for NP in the random oracle model (ROM). Before defining $\mathsf{SNARK}$, we introduce some NP relations that we use in this paper.

**NP Relations.** Let $\ell_x, \ell_w \in \mathbb{N}$ be positive integers. Let $\mathsf{Rel} \subseteq \{0,1\}^{\ell_x} \times \{0,1\}^{\ell_w}$ be an NP relation and $\mathcal{L}_{\mathsf{Rel}}$ be an NP language corresponding to $\mathsf{Rel}$, i.e.,

$$\mathcal{L}_{\mathsf{Rel}} := \{\mathsf{X} : \exists \mathsf{W} \in \{0,1\}^{\ell_w} \text{ s.t. } (\mathsf{X}, \mathsf{W}) \in \mathsf{Rel}\}.$$

We call $\mathsf{X}$ a statement and $\mathsf{W}$ a witness.

Then, we define the $\mathsf{SNARK}$ system for NP in the ROM. In the random oracle model, algorithms have black-box access to an oracle $\mathsf{RO} : \{0,1\}^* \to \mathcal{Y}$, called a random oracle. The oracle $\mathsf{RO}$ is instantiated by a uniform random function with domain $\{0,1\}^*$ and range $\mathcal{Y}$. We denote by $\mathcal{A}^{\mathsf{RO}}$ an algorithm that has black-box access to $\mathsf{RO}$, and we may occasionally omit the superscript $\mathsf{RO}$ for simplicity if the meaning is clear from the context.

**Syntax.** Let $\ell_x, \ell_w, \ell \in \mathbb{N}$ and let $\mathsf{Rel}, \tilde{\mathsf{Rel}} \subseteq \{0,1\}^{\ell_x} \times \{0,1\}^{\ell_w}$ be NP relations with $\mathsf{Rel} \subseteq \tilde{\mathsf{Rel}}$. A succinct non-interactive argument of knowledge ($\mathsf{SNARK}$) system $\Pi_{\mathsf{SNARK}}$ for the relations $\mathsf{Rel}$ and $\tilde{\mathsf{Rel}}$ and a common random string $\mathsf{crs} \in \{0,1\}^{\ell}$ consists of the following oracle-calling PPT algorithms.

- $\mathsf{Prove}^{\mathsf{RO}}_{\mathsf{Rel},\tilde{\mathsf{Rel}}}(\mathsf{crs}, \mathsf{X}, \mathsf{W}) \to \pi$: On input of the $\mathsf{crs} \in \{0,1\}^{\ell}$, a statement $\mathsf{X}$, and a witness $\mathsf{W}$, the prover algorithm outputs a proof $\pi$.
- $\mathsf{Verify}^{\mathsf{RO}}_{\mathsf{Rel},\tilde{\mathsf{Rel}}}(\mathsf{crs}, \mathsf{X}, \pi) \to \top/\bot$: On input of the $\mathsf{crs}$, a statement $\mathsf{X}$, and a proof $\pi$, the verifier algorithm outputs either $\top$ (accept) of $\bot$ (reject).

**Definition 3.** *A* $\mathsf{SNARK}$ *system* $\Pi_{\mathsf{SNARK}} = (\mathsf{Prove}^{\mathsf{RO}}_{\mathsf{Rel},\tilde{\mathsf{Rel}}}, \mathsf{Verify}^{\mathsf{RO}}_{\mathsf{Rel},\tilde{\mathsf{Rel}}})$ *is required to satisfy the following properties:*

**Completeness:** *For any* $\lambda \in \mathbb{N}$, $\mathsf{crs} \in \{0,1\}^{\ell}$, *and* $(\mathsf{X}, \mathsf{W}) \in \mathsf{Rel}$, *it holds that*

$$\Pr\left[\mathsf{Verify}^{\mathsf{RO}}_{\mathsf{Rel},\tilde{\mathsf{Rel}}}(\mathsf{crs}, \mathsf{X}, \pi) = \top : \pi \xleftarrow{\$} \mathsf{Prove}^{\mathsf{RO}}_{\mathsf{Rel},\tilde{\mathsf{Rel}}}(\mathsf{crs}, \mathsf{X}, \mathsf{W})\right] = 1 - \mathsf{negl}(\lambda).$$

**Argument of knowledge:** *For any* $\lambda \in \mathbb{N}$, $\mathsf{crs} \in \{0,1\}^{\ell}$, $\mathsf{X} \in \{0,1\}^{\ell_x}$, *and any PPT adversary* $\mathcal{A}$, *there exists a PPT algorithm* $\mathsf{Extract}$, *called an extractor, such that*

$$\Pr\left[(\mathsf{X}, \mathsf{W}) \in \tilde{\mathsf{Rel}} : \mathsf{W} \xleftarrow{\$} \mathsf{Extract}^{\mathcal{A}}(\mathsf{crs}, \mathsf{X})\right] \geq \epsilon(\mathcal{A}, \mathsf{X}) - \mathsf{negl}(\lambda),$$

*where* $\epsilon(\mathcal{A}, \mathsf{X})$ *is the success probability of* $\mathcal{A}$ *for the statement* $\mathsf{X}$, *which is defined as*

$$\epsilon(\mathcal{A}, \mathsf{X}) := \Pr\left[\mathsf{Verify}^{\mathsf{RO}}_{\mathsf{Rel},\tilde{\mathsf{Rel}}}(\mathsf{crs}, \mathsf{X}, \pi) = \top : \pi \xleftarrow{\$} \mathcal{A}^{\mathsf{RO}}(\mathsf{crs}, \mathsf{X})\right].$$

*Here,* $\mathsf{Extract}$ *implements* $\mathsf{RO}$ *for* $\mathcal{A}$, *in particular,* $\mathsf{Extract}$ *can program* $\mathsf{RO}$ *arbitrarily.*

**Succinctness:** *The length of the proof* $\pi$ *is at most* $\mathsf{poly}(\lambda, \log \ell_x, \log \ell_w)$[1].

### 2.4 Digital Signature

Here, we recall the standard digital signature ($\mathsf{DS}$) scheme.

**Syntax.** A digital signature scheme $\Pi_{\mathsf{DS}}$ with message space $\mathcal{M}$ consists of the following PPT algorithms.

- $\mathsf{KGen}(1^{\lambda}) \to (\mathsf{sk}, \mathsf{vk})$: On input of the security parameter $\lambda$, the key generation algorithm outputs a signing key $\mathsf{sk}$ and a verification key $\mathsf{vk}$.

[1] In this work, we consider only the succinctness of the proof size, not the running time of the verification time.

– $\mathsf{Sign}(\mathsf{sk}, \mathsf{M}) \to \sigma$: On input of the signing key $\mathsf{sk}$ and a message $\mathsf{M} \in \mathcal{M}$, the signing algorithm outputs a signature $\sigma$.
– $\mathsf{Verify}(\mathsf{vk}, \mathsf{M}, \sigma) \to \top/\bot$: On input of the verification key $\mathsf{vk}$, a message $\mathsf{M} \in \mathcal{M}$, and a signature $\sigma$, the verification algorithm outputs either $\top$ (accept) or $\bot$ (reject). The verification algorithm is deterministic.

**Definition 4.** *A* DS *scheme* $\Pi_{\mathsf{DS}} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ *is required to satisfy the following properties:*

**Correctness:** *For any* $\lambda \in \mathbb{N}$ *and any* $\mathsf{M} \in \mathcal{M}$*, it holds that*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{M}, \sigma) = \top : \begin{array}{l}(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda), \\ \sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, \mathsf{M})\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

**Unforgeability:** *For any* $\lambda \in \mathbb{N}$ *and any PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{M}^*, \sigma^*) = \top : \begin{array}{l}(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda), \\ (\mathsf{M}^*, \sigma^*) \leftarrow \mathcal{A}^{O_{\mathsf{sk}}(\cdot)}(\mathsf{vk})\end{array}\right] = \mathsf{negl}(\lambda),$$

*where an oracle* $O_{\mathsf{sk}}(\mathsf{M})$ *returns* $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, \mathsf{M})$ *for* $\mathsf{M} \neq \mathsf{M}^*$*.*

### 2.5 Aggregate Signature

Here, we provide the definition of the aggregate signature ($\mathsf{AS}$) scheme.

**Syntax.** A (bounded) aggregate signature scheme $\Pi_{\mathsf{AS}}$ with message space $\mathcal{M}$ consists of the following PPT algorithms.

– $\mathsf{Setup}(1^\lambda, 1^N) \to \mathsf{pp}$: On input of the security parameter $\lambda$ and an aggregation bound $N$, the setup algorithm outputs the public parameter $\mathsf{pp}$.
– $\mathsf{KGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$: On input of the public parameter $\mathsf{pp}$, the key generation algorithm outputs a signing key $\mathsf{sk}$ and a verification key $\mathsf{vk}$.
– $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{M}) \to \sigma$: On input of the public parameter $\mathsf{pp}$, the signing key $\mathsf{sk}$, and a message $\mathsf{M} \in \mathcal{M}$, the signing algorithm outputs a signature $\sigma$.
– $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathsf{M}, \sigma) \to \top/\bot$: On input of the public parameter $\mathsf{pp}$, the verification key $\mathsf{vk}$, a message $\mathsf{M} \in \mathcal{M}$, and a signature $\sigma$, the verification algorithm outputs either $\top$ (accept) or $\bot$ (reject).
– $\mathsf{Agg}(\mathsf{pp}, \{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i \in [N']}) \to \sigma_{\mathsf{agg}}$: On input of the public parameter $\mathsf{pp}$ and a collection of up to $N' \leq N$ verification keys $\mathsf{vk}_i$, messages $\mathsf{M}_i$, and signatures $\sigma_i$, the aggregation algorithm outputs an aggregated signature $\sigma_{\mathsf{agg}}$.
– $\mathsf{AggVer}(\mathsf{pp}, \vec{\mathsf{vk}}, \vec{\mathsf{M}}, \sigma_{\mathsf{agg}}) \to \top/\bot$: On input of the public parameter $\mathsf{pp}$, a collection of $N' \leq N$ verification keys $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_{N'})$, messages $\vec{\mathsf{M}} = (\mathsf{M}_1, \ldots, \mathsf{M}_{N'})$, and an aggregated signature $\sigma_{\mathsf{agg}}$, the aggregate verification algorithm outputs either $\top$ (accept) or $\bot$ (reject).

**Definition 5 ([7,33]).** *An* $\mathsf{AS}$ *scheme* $\Pi_{\mathsf{AS}} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Agg}, \mathsf{AggVer})$ *is required to satisfy the following properties:*

**Correctness:** *For any* $\lambda, N \in \mathbb{N}$ *and any* $\mathsf{M} \in \mathcal{M}$*, it holds that*

$$\Pr\left[\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathsf{M}, \sigma) = \top : \begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N), \\ (\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}), \\ \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{M}) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

*In addition, for any* $N' \in \mathbb{N}$ *with* $N' \leq N$ *and any* $\mathsf{M}_1, \ldots, \mathsf{M}_{N'} \in \mathcal{M}$*, it holds that*

$$\Pr\left[\mathsf{AggVer}(\mathsf{pp}, (\mathsf{vk}_1, \ldots, \mathsf{vk}_{N'}), (\mathsf{M}_1, \ldots, \mathsf{M}_{N'}), \sigma_{\mathsf{agg}}) = \top\right] = 1 - \mathsf{negl}(\lambda),$$

*where* $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N)$*,* $(\mathsf{sk}_i, \mathsf{vk}_i) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp})$ *and* $\sigma_i \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_i, \mathsf{M}_i)$ *for all* $i \in [N']$*, and* $\sigma_{\mathsf{agg}} \xleftarrow{\$} \mathsf{Agg}(\mathsf{pp}, \{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i \in [N']})$*.*

**Unforgeability:** *For any* $\lambda \in \mathbb{N}$ *and any PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l} \exists i \in [N'] \ s.t. \ \mathsf{vk}_i = \mathsf{vk} \\ \wedge \mathsf{AggVer}(\mathsf{pp}, \vec{\mathsf{vk}}, \vec{\mathsf{M}}, \sigma^*) = \top \end{array} : \begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda), \\ (\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}), \\ (\vec{\mathsf{vk}}, \vec{\mathsf{M}}, \sigma^*) \leftarrow \mathcal{A}^{O_{\mathsf{sk}}(\cdot)}(\mathsf{pp}, \mathsf{vk}) \end{array}\right] = \mathsf{negl}(\lambda),$$

*where* $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_{N'})$*,* $\vec{\mathsf{M}} = (\mathsf{M}_1, \ldots, \mathsf{M}_{N'})$*, an oracle* $O_{\mathsf{sk}}(\mathsf{M})$ *returns* $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, \mathsf{M})$ *for* $\mathsf{M} \neq \mathsf{M}_i$*.*

**Efficiency:** *The length of the aggregated signature* $\sigma_{\mathsf{agg}}$ *is at most* $\mathsf{poly}(\lambda, \log N)$*.*

## 3 Building Blocks

In this section, we describe the building blocks used in our main construction. We respectively provide in Sections 3.1 and 3.2, our concrete building blocks for the SNARK system $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$ and the digital signature scheme $\Pi^{\mathsf{CLMQ}}_{\mathsf{DS}}$.

### 3.1 Main Protocol for LaBRADOR

The SNARK system $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$ used in our main construction is the non-interactive variant (via Fiat-Shamir) of the interactive proof system for the principal relations, proposed by Beullens and Seiler [5]. To do so, we first define the principal relations and then provide the non-interactive protocol for the principal relations.

**Principal Relations.** The principal relation is parameterized by a rank $n \geq 1$, a multiplicity $r \geq 1$, and a norm bound $\beta > 0$. It consists of short solutions to dot product constraints over $\mathcal{R}_Q$. Specifically, a statement consists of a family $\mathcal{F} := \{f^{(1)}, \ldots, f^{(K)}\}$ of quadratic dot product functions $f : (\mathcal{R}_Q^n)^r \to \mathcal{R}_Q$ of the form

$$f(\mathbf{s}_1, \ldots, \mathbf{s}_r) = \sum_{i=1}^{r}\sum_{j=1}^{r} a_{i,j}\langle \mathbf{s}_i, \mathbf{s}_j\rangle + \sum_{i=1}^{r}\langle \boldsymbol{\varphi}_i, \mathbf{s}_i\rangle - b,$$

where $a_{i,j}, b \in \mathcal{R}_Q$ and $\boldsymbol{\varphi}_i \in \mathcal{R}_Q^n$. Without loss of generality, we assume $a_{i,j} = a_{j,i}$. We now define the principal relation. In the following, we identify a quadratic dot product function $f^{(k)}$ and its coefficients $(\{a_{i,j}^{(k)}\}_{i,j\in[r]}, \{\boldsymbol{\varphi}_i^{(k)}\}_{i\in[r]}, b^{(k)})$ for all $k \in [K]$.

**Definition 6 (Principal Relation).** *Let $n, r \in \mathbb{N}$ and let $\beta > 0$ be a positive real. The principal relation $\mathsf{Rel}_{\mathsf{pr}}$ is defined by*

$$\mathsf{Rel}_{\mathsf{pr},\beta} := \left\{ (\mathsf{X} = (\mathcal{F}, \hat{\mathcal{F}}), \mathsf{W} = (\mathbf{s}_1, \ldots, \mathbf{s}_r)) : \begin{array}{l} f(\mathbf{s}_1, \ldots, \mathbf{s}_r) = 0 \ \forall f \in \mathcal{F}, \\ \mathsf{ct}(\hat{f}(\mathbf{s}_1, \ldots, \mathbf{s}_r)) = 0 \ \forall \hat{f} \in \hat{\mathcal{F}}, \\ \sum_{i=1}^{r} \|\mathbf{s}_i\|_2^2 \leq \beta^2 \end{array} \right\},$$

*where $\mathcal{F}$ and $\hat{\mathcal{F}}$ are two families of quadratic dot product functions.*

**Non-Interactive Protocol.** Here, we provide the non-interactive protocol $\Pi_{\mathsf{SNARK}}^{\mathsf{pr}}$ for the principal relations $(\mathsf{Rel}_{\mathsf{pr},\beta}, \mathsf{Rel}_{\mathsf{pr},\beta'})$ proposed in [5], where $\beta' = \sqrt{128/30}\beta$. For reference, we give in Table 3 the parameters used in the $\Pi_{\mathsf{SNARK}}^{\mathsf{pr}}$. The prove and verify algorithms of $\Pi_{\mathsf{SNARK}}^{\mathsf{pr}}$ are described in Figure 2 and Figure 3, respectively. The following algorithms use the common random string $\mathsf{crs}$ that is sampled as follows:

$$\mathsf{crs} = \left(\mathbf{A}, (\mathbf{B}_i)_{i\in[r]}, (\mathbf{C}_{i,j})_{1\leq i\leq j\leq r}, (\mathbf{D}_{i,j})_{1\leq i\leq j\leq r}\right),$$

where $\mathbf{A} \xleftarrow{\$} \mathcal{R}_Q^{\kappa_0\times n}$, $\mathbf{B}_i \xleftarrow{\$} \mathcal{R}_Q^{\kappa_1\times\kappa_0\lceil\log_{b_1} Q\rceil}$, $\mathbf{C}_{i,j} \xleftarrow{\$} \mathcal{R}_Q^{\kappa_1\times\lceil\log_{b_2} Q\rceil}$, and $\mathbf{D}_{i,j} \xleftarrow{\$} \mathcal{R}_Q^{\kappa_2\times\lceil\log_{b_1} Q\rceil}$.

**Table 3.** Overview of parameters used in $\Pi_{\mathsf{SNARK}}^{\mathsf{pr}}$.

| Parameter | Explanation |
|---|---|
| $\mathcal{R}_Q$ | Ring $\mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^d+1)$ |
| $n$ | A rank for $\mathsf{Rel}_{\mathsf{pr},\beta}$ and $\mathsf{Rel}_{\mathsf{pr},\beta'}$ |
| $r$ | A multiplicity for $\mathsf{Rel}_{\mathsf{pr},\beta}$ and $\widetilde{\mathsf{Rel}}_{\mathsf{pr},\beta'}$ |
| $\beta$ | A norm bound for $\mathsf{Rel}_{\mathsf{pr},\beta}$ |
| $\beta'(=\sqrt{128/30}\beta)$ | A norm bound for $\mathsf{Rel}_{\mathsf{pr},\beta'}$ |
| $K$ | The number of quadratic dot product functions in $\mathcal{F}$ |
| $L$ | The number of quadratic dot product functions in $\hat{\mathcal{F}}$ |
| $K'(=\lceil 128/\log Q\rceil)$ | The number of functions after aggregation |
| $(\kappa_0, \kappa_1, \kappa_2)$ | The sizes of matrices in $\mathsf{crs}$ |
| $(b, b_1, b_2)$ | The decomposition parameters |
| $\mathsf{H}_{\mathsf{pr}}$ | The hash function $\mathsf{H}_{\mathsf{pr}} : \{0,1\}^* \to \{0,1\}^*$ |

Beullens and Seiler showed the following lemma for $\Pi_{\mathsf{SNARK}}^{\mathsf{pr}}$.

**Lemma 4 ([5, Theorem 5.1]).** *Let $\mathsf{Ch}$ be the challenge space $\mathsf{Ch} \subset \mathcal{R}_Q$ from Sec. 2.2 consisting of polynomials with norm $\tau_{\mathsf{Ch}}$ and operator norm $T_{\mathsf{Ch}}$.*

$$\mathsf{Prove}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta'}}(\mathsf{crs},\mathsf{X},\mathsf{W}) \to \pi_{\mathsf{pr}}$$

$$\textbf{for } i \in [r] : \mathbf{t}_i := \mathbf{A}\mathbf{s}_i$$

$$\textbf{for } (i,j) \in [r] \times [r] : g_{i,j} := \langle \mathbf{s}_i, \mathbf{s}_j \rangle$$

$$\mathbf{u}_1 := \sum_{i=1}^{r} \mathbf{B}_i \mathbf{G}^{-1}_{b_1,\kappa}(\mathbf{t}_i) + \sum_{i \le j} \mathbf{C}_{i,j} \mathbf{G}^{-1}_{b_2,1}(g_{i,j})$$

$$(\vec{\pi}_{i,j})_{i\in[r],j\in[256]} \leftarrow \mathsf{H}_{\mathsf{pr}}(1,\mathsf{X},\mathbf{u}_1), \text{ where } \vec{\pi}_{i,j} \in \mathsf{Ch}^{nd}_{\mathsf{mJL}} \subseteq \{-1,0,1\}^{nd}$$

$$\textbf{for } j \in [256] : p_j := \sum_{i=1}^{r} \langle \vec{\pi}_{i,j}, \vec{s}_i \rangle$$

$$(\vec{\psi}_k, \vec{\omega}_k)_{k\in[K']} \leftarrow \mathsf{H}_{\mathsf{pr}}(2,\mathsf{X},\mathbf{u}_1,(p_j)_{j\in[256]}), \text{ where } \vec{\psi}_k \in \mathbb{Z}_Q^L,\ \vec{\omega}_k \in \mathbb{Z}_Q^{256}$$

$$\textbf{for } k \in [K'] :$$

$$\quad a'^{(k)}_{i,j} := \sum_{l=1}^{L} \psi^{(k)}_l \hat{a}_{i,j}$$

$$\quad \boldsymbol{\varphi}'^{(k)}_i := \sum_{l=1}^{L} \psi^{(k)}_l \hat{\boldsymbol{\varphi}}^{(l)}_i + \sum_{j=1}^{256} \omega^{(k)}_j \sigma_{-1}(\vec{\boldsymbol{\pi}}_{i,j})$$

$$\quad b'^{(k)} := \sum_{i,j=1}^{r} a'^{(k)}_{i,j} \langle \mathbf{s}_i, \mathbf{s}_j \rangle + \sum_{i=1}^{r} \langle \boldsymbol{\varphi}^{(k)}_i, \mathbf{s}_i \rangle$$

$$(\boldsymbol{\alpha},\boldsymbol{\beta}) \leftarrow \mathsf{H}_{\mathsf{pr}}(3,\mathsf{X},\mathbf{u}_1,(p_j)_{j\in[256]},(b'^{(k)})_{k\in[K']}), \text{ where } \boldsymbol{\alpha} \in \mathcal{R}_Q^K,\ \boldsymbol{\beta} \in \mathcal{R}_Q^{K'}$$

$$\textbf{for } i \in [r] : \vec{\boldsymbol{\varphi}}_i := \sum_{k=1}^{K} \alpha_k \boldsymbol{\varphi}^{(k)}_i + \sum_{k=1}^{K'} \beta_k \boldsymbol{\varphi}'^{(k)}_i$$

$$\textbf{for } (i,j) \in [r] \times [r] : h_{i,j} := \frac{1}{2}(\langle \boldsymbol{\varphi}_i, \mathbf{s}_j \rangle + \langle \boldsymbol{\varphi}_j, \mathbf{s}_i \rangle)$$

$$\mathbf{u}_2 := \sum_{i \le j} \mathbf{D}_{i,j} \mathbf{G}^{-1}_{b_1,1}(h_{i,j})$$

$$(c_i)_{i\in[r]} \leftarrow \mathsf{H}_{\mathsf{pr}}(4,\mathsf{X},\mathbf{u}_1,(p_j)_{j\in[256]},(b'^{(k)})_{k\in[K']},\mathbf{u}_2), \text{ where } c_i \in \mathsf{Ch}$$

$$\mathbf{z} := c_1\mathbf{s}_1 + \cdots + c_r\mathbf{c}_r$$

$$\textbf{return } \pi_{\mathsf{pr}} := (\mathbf{u}_1,(p_j)_{j\in[256]},(b'^{(k)})_{k\in[K']},\mathbf{u}_2,\mathbf{z},(\mathbf{t}_i)_{i\in[r]},(g_{i,j},h_{i,j})_{i,j\in[r]})$$

**Fig. 2.** Prove algorithm of $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$.

$\mathsf{Verify}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta}}(\mathsf{crs}, \mathsf{X}, \pi_{\mathsf{pr}}) \to \top/\bot$

**for** $k \in [K']$ :

$$a'^{(k)}_{i,j} \coloneqq \sum_{l=1}^{L} \psi_l^{(k)} \hat{a}_{i,j},$$

$$\boldsymbol{\varphi}'^{(k)}_i \coloneqq \sum_{l=1}^{L} \psi_l^{(k)} \hat{\boldsymbol{\varphi}}_i^{(l)} + \sum_{j=1}^{256} \omega_j^{(k)} \sigma_{-1}(\vec{\boldsymbol{\pi}}_{i,j})$$

**for** $(i,j) \in [r] \times [r] : a_{i,j} \coloneqq \sum_{k=1}^{K} \alpha_k a_{i,j}^{(k)} + \sum_{j=1}^{256} \beta_k a'^{(k)}_{i,j}$

**for** $i \in [r] : \boldsymbol{\varphi}_i \coloneqq \sum_{k=1}^{K} \alpha_k \boldsymbol{\varphi}_i^{(k)} + \sum_{j=1}^{256} \beta_k \boldsymbol{\varphi}'^{(k)}_i$

$b \coloneqq \sum_{k=1}^{K} \alpha_k b^{(k)} + \sum_{j=1}^{256} \beta_k b'^{(k)}$

$\mathbf{z} \coloneqq \mathbf{z}^{(0)} + \mathbf{z}^{(1)} b,$ where $\|\mathbf{z}^{(0)}\|_\infty \leq \frac{b}{2}$

**for** $i \in [r] : \mathbf{t}'_i \coloneqq \mathbf{G}^{-1}_{b_1,\kappa}(\mathbf{t}_i)$

**for** $(i,j) \in [r] \times [r] : \mathbf{g}'_{i,j} \coloneqq \mathbf{G}^{-1}_{b_2,1}(g_{i,j}),\ \mathbf{h}'_{i,j} \coloneqq \mathbf{G}^{-1}_{b_1,1}(h_{i,j})$

$$\textbf{if} \begin{cases} \|\vec{p}\| \leq \sqrt{128}\beta \\ \wedge\ b'^{(k)}_0 = \langle \vec{\omega^{(j)}}, \vec{p} \rangle + \sum_{l=1}^{L} \varphi_l^{(k)} \hat{b}_0^{(l)} \\ \wedge\ g_{i,j} = g_{j,i} \\ \wedge\ h_{i,j} = h_{j,i} \\ \wedge\ (\beta')^2 \geq \|\mathbf{z}^{(0)}\|^2 + \|\mathbf{z}^{(1)}\|^2 + \sum_{i=1}^{r} \left( \|\mathbf{t}'_i\|^2 + \sum_{j=1}^{r} (\|\mathbf{g}'_{i,j}\|^2 + \|\mathbf{h}'_{i,j}\|^2) \right) \\ \wedge\ \mathbf{A}\mathbf{z} = \sum_{i=1}^{r} c_i \mathbf{t}_i \\ \wedge\ \langle \mathbf{z}, \mathbf{z} \rangle = \sum_{i=1}^{r} \sum_{j=1}^{r} g_{i,j} c_i c_j \\ \wedge\ \sum_{i=1}^{r} \langle \boldsymbol{\varphi}_i, \mathbf{z} \rangle c_i = \sum_{i=1}^{r} \sum_{j=1}^{r} h_{i,j} c_i c_j \\ \wedge\ \sum_{i=1}^{r} \sum_{j=1}^{r} a_{i,j} g_{i,j} + \sum_{i=1}^{r} h_{i,i} - b = 0 \\ \wedge\ \mathbf{u}_1 = \sum_{i=1}^{r} \mathbf{B}_i \mathbf{t}'_i + \sum_{i \leq j} \mathbf{C}_{i,j} \mathbf{g}'_{i,j} \\ \wedge\ \mathbf{u}_2 = \sum_{i \leq j} \mathbf{D}_{i,j} \mathbf{h}_{i,j} \end{cases}$$

**then return** $\top$

**else return** $\bot$

**Fig. 3.** Verify algorithm of $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$.

*Suppose that* $\mathsf{MSIS}_{\kappa_1,2\beta',Q}$*,* $\mathsf{MSIS}_{\kappa_2,2\beta',Q}$*, and* $\mathsf{MSIS}_{\kappa_0,\beta'',Q}$ *are hard, where* $\beta'' = \max\{8T_{\mathsf{Ch}}(b+1)\beta', 2(b+1)\beta' + 4T_{\mathsf{Ch}}\sqrt{128/30}\beta\}$*. Further suppose that* $\beta \le \sqrt{30/128}Q/125$*. Then,* $\mathit{\Pi}^{\mathsf{pr}}_{\mathsf{SNARK}}$ *in Figures 2 and 3 is a* SNARK *for* $\mathsf{Rel}_{\mathsf{pr},\beta}$ *and* $\mathsf{Rel}_{\mathsf{pr},\beta'}$ *in the random oracle model.*

*Remark 1 (Recursion and proof size.).* The target relation of the above main protocol is almost another instance of the dot product constraint. To reduce the size of the proof, we recursively apply the protocol several times. Asymptotically, we need only $O(\log\log n)$ iterations of the main protocol. Finally, the proof size can be logarithmic in the witness size. See [5] for details.

### 3.2 CLMQ Signature Scheme

We rely on a module version of the CLMQ signature scheme $\mathit{\Pi}^{\mathsf{CLMQ}}_{\mathsf{DS}}$ by Chen et al. [13]. For reference, we give in Table 4 the parameters used in the $\mathit{\Pi}^{\mathsf{CLMQ}}_{\mathsf{DS}}$. We present the algorithms of $\mathit{\Pi}^{\mathsf{CLMQ}}_{\mathsf{DS}}$ in Figure 4.

**Table 4.** Overview of parameters and notations used in $\mathit{\Pi}^{\mathsf{CLMQ}}_{\mathsf{DS}}$.

| Parameter | Explanation |
|---|---|
| $\mathcal{R}_q$ | A Ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ |
| $n$ | The size of $\mathbf{A}$ in $\mathsf{vk}$ |
| $m(=n\lceil\log q\rceil)$ | The length of challenge vector $\mathbf{c}$ |
| $\eta$ | The norm bound of $\mathbf{S}$ |
| $(\phi, T, \mathsf{err})$ | Parameters for rejection sampling (Lemma 2) |
| $t(=\sqrt{2(1+\log nd+\lambda)/\log e})$ | Parameter for Lemma 1 |
| $\mathsf{H_M}$ | The hash function $\mathsf{H_M} : \{0,1\}^* \to \mathcal{R}_q^n$ |

From the results of [25,13], we obtain the following lemma.

**Lemma 5.** *Let* $\beta_0 = \sqrt{(t\phi T)^2 2nd + md}$*. Suppose that* $T = m\eta\sqrt{2nd}$ *and* $t\phi T \le q/2$*. Assuming* $\mathsf{MSIS}_{n,\beta,q}$ *and* $\mathsf{MLWE}_{n,\eta,q}$ *assumptions hold, then* $\mathit{\Pi}^{\mathsf{CLMQ}}_{\mathsf{DS}}$ *is a* DS *scheme in the random oracle model.*

## 4 Our Aggregate Signature Scheme

In this section, we give our lattice-based AS scheme $\mathit{\Pi}_{\mathsf{AS}}$. To construct $\mathit{\Pi}_{\mathsf{AS}}$, we follow the approach of [33, Sec. 7]. $\mathit{\Pi}_{\mathsf{AS}}$ is obtained by combining the SNARK system $\mathit{\Pi}^{\mathsf{pr}}_{\mathsf{SNARK}}$ in Sec. 3.1 and the CLMQ signature scheme $\mathit{\Pi}^{\mathsf{CLMQ}}_{\mathsf{DS}}$ in Sec. 3.2.

### 4.1 Construction of Aggregate Signature

**Parameters.** For reference, we provide in Table 5 the parameters used in the scheme. We require that these parameters satisfy certain conditions for correctness and security to hold.

$\mathsf{KGen}(1^\lambda)$

$\mathbf{S} \xleftarrow{\$} \mathcal{S}_\eta^{2n\times m}$
$\mathbf{A}_0 \xleftarrow{\$} \mathcal{R}_q^{n\times n}$
$\mathbf{A} := (\mathbf{A}_0 \| \mathbf{I}_n)$
$\mathbf{B} := \mathbf{AS}$
$\mathsf{sk} := (\mathbf{S}, \mathbf{A})$
$\mathsf{vk} := (\mathbf{A}, \mathbf{B})$
**return** $(\mathsf{sk}, \mathsf{vk})$

$\mathsf{Sign}(\mathsf{sk}, \mathsf{M})$

$\mathbf{y} \xleftarrow{\$} \mathcal{D}_{\phi T}^{2n}$
$\mathbf{u} := \mathbf{Ay}$
$\mathbf{c} := \mathbf{G}_{2,n}^{-1}(\mathbf{u} - \mathsf{H}_\mathsf{M}(\mathsf{vk}, \mathsf{M}))$
$\mathbf{z} := \mathbf{y} + \mathbf{Sc}$
**if** $\mathsf{Rej}(\mathbf{z}, \mathbf{y}, \phi, T, \mathsf{err}) = \top$ :
  **then return** $\sigma := (\mathbf{z}, \mathbf{c})$
**else** restart

$\mathsf{Verify}(\mathsf{vk}, \mathsf{M}, \sigma)$

**if** $(\mathbf{G}_{2,n} + \mathbf{B})\mathbf{c} = \mathbf{Az} - \mathsf{H}_\mathsf{M}(\mathsf{vk}, \mathsf{M}) \wedge \|\mathbf{z}\|_\infty \le t\phi T$ :
  **then return** $\top$
**else return** $\bot$

**Fig. 4.** CLMQ signature scheme $\Pi_{\mathsf{DS}}^{\mathsf{CLMQ}}$.

**Building Blocks.** Our AS scheme $\Pi_{\mathsf{AS}}$ relies on the following building blocks.

- A SNARK system $\Pi_{\mathsf{SNARK}}^{\mathsf{pr}} = (\mathsf{pr.Prove}^{\mathsf{H}_\mathsf{pr}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta}}, \mathsf{pr.Verify}^{\mathsf{H}_\mathsf{pr}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta}})$ with $\mathsf{crs}_\mathsf{pr}$ for the principal relations $\mathsf{Rel}_{\mathsf{pr},\beta}$, $\mathcal{R}_{\mathsf{pr},\beta'}$ in Sec. 3.1. Here, $\mathsf{Rel}_\mathsf{pr}$ (resp. $\mathcal{R}_{\mathsf{pr},\beta'}$) is a principal relation of a rank $m$, a multiplicity $r$, and a norm bound $\beta = \sqrt{Q}$ (resp. $\beta' = \sqrt{128/30}\sqrt{Q}$).
- The CLMQ signature scheme $\Pi_{\mathsf{DS}}^{\mathsf{CLMQ}} = (\mathsf{DS.KGen}^{\mathsf{H}_\mathsf{M}}, \mathsf{DS.Sign}^{\mathsf{H}_\mathsf{M}}, \mathsf{DS.Verify}^{\mathsf{H}_\mathsf{M}})$ in Sec. 3.2.
- Four hash functions $\mathsf{H}_\mathsf{M}$, $\mathsf{H}_\mathsf{pr}$, $\mathsf{H}$, and $\mathsf{H}_\mathsf{crs}$ modeled as a random oracle. $\mathsf{H}_\mathsf{M}$ and $\mathsf{H}_\mathsf{pr}$ are hash functions used by $\Pi_{\mathsf{DS}}^{\mathsf{CLMQ}}$ and $\Pi_{\mathsf{SNARK}}^{\mathsf{pr}}$, respectively. A hash function $\mathsf{H} : \{0,1\}^* \to \{-1,0,1\}^{128\times 2Nnd}$ is used to generate challenge vectors. We assume that the distribution of the output of $\mathsf{H}$ is $\mathsf{Ch}_{\mathsf{mJL}}^{128\times 2Nnd}$. $\mathsf{H}_\mathsf{crs}$ is a special hash function and $\mathsf{H}_\mathsf{crs}(0) = \mathsf{crs}_\mathsf{pr}$ contains the common random string $\mathsf{crs}_\mathsf{pr}$ used by $\Pi_{\mathsf{SNARK}}^{\mathsf{pr}}$.

**Construction.** Below, we give the construction of our AS scheme $\Pi_{\mathsf{AS}} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Agg}, \mathsf{AggVer})$ with message space $\{0,1\}^*$. We assume that $\mathsf{H}_\mathsf{crs}(0) = (\mathsf{crs}_\mathsf{pr}, \mathbf{U}_1, \ldots, \mathbf{U}_N)$ is correctly derived by all the algorithms and omits the process of generating them.

- $\mathsf{Setup}(1^\lambda, 1^N) \to \mathsf{pp} := 1^\lambda$:
- $\mathsf{KGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$: (Same as $\mathsf{DS.KGen}^{\mathsf{H}_\mathsf{M}}(1^\lambda)$.)
  1. Sample $\mathbf{S} \xleftarrow{\$} \mathcal{S}_\eta^{2n\times m}$ and $\mathbf{A}_0 \xleftarrow{\$} \mathcal{R}_q^{n\times n}$.

**Table 5.** Overview of parameters and notations which are used in $\Pi_{\mathsf{AS}}$.

| Parameter | Description |
|---|---|
| $\mathcal{R}_q$ | A Ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ for $\Pi_{\mathsf{DS}}^{\mathsf{CLMQ}}$ |
| $n$ | The size of $\mathbf{A}$ in $\mathsf{vk}$ |
| $m(= n\lceil \log q \rceil)$ | The length of challenge vector $\mathbf{c}$ |
| $\eta$ | The norm bound of $\mathbf{S}$ |
| $(\phi, T, \mathsf{err})$ | Parameters for rejection sampling (Lemma 2) |
| $t(= \sqrt{2(1+\log nd + \lambda)/\log e})$ | Parameter for Lemma 1 |
| $\mathsf{H}_{\mathsf{M}}$ | The hash function $\mathsf{H}_{\mathsf{M}} : \{0,1\}^* \to \mathcal{R}_q^n$ for $\Pi_{\mathsf{DS}}^{\mathsf{CLMQ}}$ |
| $N$ | The number of signatures to be aggregated |
| $q'$ | Integer to increase modulus |
| $Q$ | Modulus s. t. $Q = qq'$ |
| $\mathsf{H}$ | The hash function $\mathsf{H} : \{0,1\}^* \to \{-1,0,1\}^{128\times 2Nnd}$ |

2. Set $\mathbf{A} := (\mathbf{A}_0 \| \mathbf{I}_n) \in \mathcal{R}_q^{n\times 2n}$ and $\mathbf{B} := \mathbf{AS} \in \mathcal{R}_q^{n\times m}$.
3. Output $(\mathsf{sk}, \mathsf{vk}) := ((\mathbf{S}, \mathbf{A}), (\mathbf{A}, \mathbf{B}))$.

– $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{M}) \to \sigma$: (Same as $\mathsf{DS.Sign}^{\mathsf{H}_{\mathsf{M}}}(\mathsf{sk}, \mathsf{M})$.)
1. Sample $\mathbf{y} \xleftarrow{\$} \mathcal{D}_{\phi T}^{2n}$.
2. Set $\mathbf{u} := \mathbf{Ay} \in \mathcal{R}_q^n$, $\mathbf{c} := \mathbf{G}_{2,n}^{-1}(\mathbf{Ay} - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}, \mathsf{M})) \in \mathcal{R}_2^m$, and $\mathbf{z} := \mathbf{y} + \mathbf{Sc} \in \mathcal{R}^{2n}$.
3. Output $\sigma := (\mathbf{z}, \mathbf{c})$ if $\mathsf{Rej}(\mathbf{z}, \mathbf{y}, \phi, T, \mathsf{err}) = \top$.
4. Otherwise, restart.

– $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathsf{M}, \sigma) \to \top/\bot$: (Same as $\mathsf{DS.Verify}^{\mathsf{H}_{\mathsf{M}}}(\mathsf{vk}, \mathsf{M}, \sigma)$.)
1. Output $\top$ if $(\mathbf{G}_{2,n} + \mathbf{B})\mathbf{c} = \mathbf{Az} - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}, \mathsf{M}) \wedge \|\mathbf{z}\|_\infty \le t\phi T$.
2. Otherwise, output $\bot$.

– $\mathsf{Agg}(\mathsf{pp}, \{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i\in[N']}) \to \sigma_{\mathsf{agg}}$:
1. If there exists $i \in [N']$ such that $\mathsf{Verify}(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i) = \bot$, then output $\bot$.
2. For $i \in [N]$, parse $\mathsf{vk}_i = (\mathbf{A}_i, \mathbf{B}_i) \in \mathcal{R}_q^{n\times 2n} \times \mathcal{R}_q^{n\times m}$ and $\sigma_i = (\vec{\mathbf{z}}_i, \vec{\mathbf{c}}_i) \in \mathcal{R}_q^{2n} \times \mathcal{R}_2^m$.
3. For $i \in [N]$, compute $\mathbf{c}_i' := \Sigma_{-1}(\mathbf{c}_i)$.
4. Compute $P_z := \mathsf{H}((\mathsf{vk}_i, \mathsf{M}_i)_{i\in[N]})$ and

$$\vec{p}_z := P_z \begin{pmatrix} \vec{z}_1 \\ \vdots \\ \vec{z}_N \end{pmatrix} \bmod q \in \mathbb{Z}_q^{128},$$

where $P_z \in \mathsf{Ch}_{\mathsf{mJL}}^{128\times 2Nnd} \subseteq \{-1,0,1\}^{128\times 2Nnd}$.

5. Define two families of quadratic dot product function $(\mathcal{F}, \hat{\mathcal{F}})$ as follows:

$$\mathcal{F} := \{q'(\mathbf{A}_i\mathbf{z}_i - (\mathbf{G}_{2,n} + \mathbf{B}_i)\mathbf{c}_i - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}_i, \mathsf{M}_i)), \langle \mathbf{c}_i, \mathbf{c}_i' - \mathbf{1}_m\rangle\}_{i\in[N]}$$

$$\hat{\mathcal{F}} := \left\{ q'\left(\Sigma_{-1}(\mathbf{P}_z)\begin{pmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_N \end{pmatrix} - \mathbf{p}_z\right)\right\} \cup \{\mathbf{c}_i' - \Sigma_{-1}(\mathbf{c}_i)\}_{i\in[N]},$$

where $\mathbf{1}_l \in R^l$ is a vector all of whose coefficients are 1.

6. Set $\mathsf{X}_{\mathsf{pr}} := (\mathcal{F}, \hat{\mathcal{F}})$ and $\mathsf{W}_{\mathsf{pr}} := (\mathbf{z}_1, \mathbf{c}_1, \mathbf{c}'_1, \ldots, \mathbf{z}_N, \mathbf{c}_N, \mathbf{c}'_N)$.
7. Compute $\pi_{\mathsf{pr}} \xleftarrow{\$} \mathsf{pr.Prove}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta}, \mathcal{R}_{\mathsf{pr},\beta'}}(\mathsf{crs}_{\mathsf{pr}}, \mathsf{X}_{\mathsf{pr}}, \mathsf{W}_{\mathsf{pr}})$.
8. Output $\sigma_{\mathsf{agg}} := (\vec{p}_z, \pi_{\mathsf{pr}})$.

– $\mathsf{AggVer}(\mathsf{pp}, (\mathsf{vk}_1, \ldots, \mathsf{vk}_T), (\mathsf{M}_1, \ldots, \mathsf{M}_T), \sigma_{\mathsf{agg}}) \to \top/\bot$:
  1. Parse $\sigma_{\mathsf{agg}} = (\vec{p}_z, \pi_{\mathsf{pr}})$.
  2. Compute $P_z := \mathsf{H}((\mathsf{vk}_i, \mathsf{M}_i)_{i \in [N]})$.
  3. Define $\mathsf{X}_{\mathsf{pr}} := (\mathcal{F}, \hat{\mathcal{F}})$ as Item 5 in $\mathsf{Agg}(\mathsf{pp}, \{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i \in [T]})$.
  4. Output $\top$ if $\mathsf{pr.Verify}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta}, \mathsf{Rel}_{\mathsf{pr},\beta'}}(\mathsf{crs}_{\mathsf{pr}}, \mathsf{X}_{\mathsf{pr}}, \pi_{\mathsf{pr}}) = \top \wedge \|\vec{p}_z\|_\infty \le t\phi T/2$.
  5. Otherwise, output $\bot$.

### 4.2 Correctness

The following establishes the correctness of our aggregate signature $\Pi_{\mathsf{AS}}$.

**Lemma 6.** *The aggregate signature $\Pi_{\mathsf{AS}}$ is correct if $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$ is complete and $\Pi^{\mathsf{CLMQ}}_{\mathsf{DS}}$ is correct.*

*Proof.* We first verify the correctness of the pre-aggregated signature of $\Pi_{\mathsf{AS}}$. The relationship that the verifier checks within the $\mathsf{Verify}$ algorithm can be expanded as in

$$\begin{aligned}
(\mathbf{G}_{2,n} + \mathbf{B})\mathbf{c} &= \mathbf{G}_{2,n}\mathbf{c} + \mathbf{B}\mathbf{c} \\
&= \mathbf{G}_{2,n}\mathbf{G}^{-1}_{2,n}(\mathbf{u} - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}, \mathsf{M})) + \mathbf{A}\mathbf{S}\mathbf{c} \\
&= \mathbf{u} - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}, \mathsf{M}) + \mathbf{A}\mathbf{S}\mathbf{c} \\
&= \mathbf{A}\mathbf{y} - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}, \mathsf{M}) + \mathbf{A}\mathbf{S}\mathbf{c} \\
&= \mathbf{A}(\mathbf{y} + \mathbf{S}\mathbf{c}) - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}, \mathsf{M}) \\
&= \mathbf{A}\mathbf{z} - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}, \mathsf{M}).
\end{aligned}$$

We then check the correctness of the aggregated signature of $\Pi_{\mathsf{AS}}$. We assume that all signatures $(\mathbf{z}_1, \mathbf{c}_1), \ldots, (\mathbf{z}_N, \mathbf{c}_N) \in \mathcal{R}_q^{2n} \times \mathcal{R}_2^m$ are valid signature of $\Pi^{\mathsf{CLMQ}}_{\mathsf{DS}}$. That is, for all $i \in [N]$ we have

$$(\mathbf{G}_{2,n} + \mathbf{B}_i)\mathbf{c}_i = \mathbf{A}_i\mathbf{z}_i - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}_i, \mathsf{M}_i) \bmod q, \tag{1}$$

$$\|\mathbf{z}_i\|_\infty \le t\phi T. \tag{2}$$

Furthermore, we assume that the aggregator correctly processes the $\mathsf{Agg}$ algorithm. That is, we have

$$\mathbf{c}'_i = \Sigma_{-1}(\mathbf{c}_i) \text{ for all } i \in [N], \tag{3}$$

$$\vec{p}_z = P_z \begin{pmatrix} \vec{z}_1 \\ \vdots \\ \vec{z}_N \end{pmatrix} \bmod q. \tag{4}$$

Since $Q = qq'$, Equations (1) and (4) implies

$$q'(\mathbf{A}_i\mathbf{z}_i - (\mathbf{G}_{2,n} + \mathbf{B}_i)\mathbf{c}_i - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}_i, \mathsf{M}_i)) = \mathbf{0} \bmod Q \text{ for all } i \in [N], \tag{5}$$

$$\mathsf{ct}\left(q'\left(\Sigma_{-1}(\mathbf{P}_z)\begin{pmatrix}\mathbf{z}_1\\ \vdots \\ \mathbf{z}_N\end{pmatrix} - \mathbf{p}_z\right)\right) = 0 \bmod Q. \tag{6}$$

In addition, since $\mathbf{c} \in \mathcal{R}_2^m$, we have

$$\langle \mathbf{c}_i, \mathbf{c}_i' - \mathbf{1}_l \rangle = \mathbf{0} \tag{7}$$

for all $i \in [N]$. From Equations (3) and (5) to (7), we have

$$(\mathsf{X}_{\mathsf{pr}}, \mathsf{W}_{\mathsf{pr}}) = \Big((\mathcal{F}, \hat{\mathcal{F}}), (\mathbf{z}_1, \mathbf{c}_1, \mathbf{c}_1', \ldots, \mathbf{z}_N, \mathbf{c}_N, \mathbf{c}_N')\Big) \in \mathsf{Rel}_{\mathsf{pr},\beta}.$$

Therefore, by the completeness of $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$, $\pi_{\mathsf{pr}}$ outputted by $\mathsf{pr.Prove}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta'}}$ will be accepted by $\mathsf{pr.Verify}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta'}}$. Furthermore, by Lemma 3, Equations (2) and (4) implies $\|\vec{p}_z\|_\infty \le t\phi T/2$ with overwhelming probability.

Putting these together, any aggregated signature $\sigma_{\mathsf{agg}}$ outputted by $\mathsf{Agg}$ will be accepted by $\mathsf{AggVer}$ with overwhelming probability. We conclude that $\Pi_{\mathsf{AS}}$ is correct. □

### 4.3 Security: Unforgeability

**Theorem 1.** *The aggregate signature $\Pi_{\mathsf{AS}}$ is unforgeable if $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$ is an argument of knowledge and $\Pi^{\mathsf{CLMQ}}_{\mathsf{DS}}$ is unforgeable.*

*Proof.* Assume there exists a PPT adversary $\mathcal{A}$ with non-negligible probability $\epsilon$ against the unforgeability experiment. We consider a sequence of games, where we denote $\mathsf{E}_i$ as the event $\mathcal{A}$ that succeeds in generating a forgery in $\mathsf{Hyb}_i$.

$\mathsf{Hyb}_0$**:** This game is the real unforgeability experiment:

- At the beginning of the experiment, we set $\mathsf{pp} := 1^\lambda$, sample $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp})$, and gives $\mathsf{pp}$ and $\mathsf{vk}$ to $\mathcal{A}$.
- Upon receiving signing queries on messages $\mathsf{M}$, we computes $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{M})$ and replies with $\sigma$ to $\mathcal{A}$.
- At the end of the experiment, $\mathcal{A}$ outputs a tuple of verification keys $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$, a tuple of messages $\vec{\mathsf{M}} = (\mathsf{M}_1, \ldots, \mathsf{M}_N)$, and an aggregated signature $\sigma^*_{\mathsf{agg}} = (\vec{p}^*_z, \pi^*_{\mathsf{pr}})$.
- If there exists an index $i \in [N]$ such that $\mathsf{vk}_i = \mathsf{vk}$, $\|\vec{p}^*_z\|_\infty \le t\phi T/2$, and $\mathsf{pr.Verify}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta'}}(\mathsf{crs}_{\mathsf{pr}}, \mathsf{X}^*_{\mathsf{pr}}, \pi^*_{\mathsf{pr}}) = \top$, then $\sigma^*_{\mathsf{agg}}$ is a valid forgery. Here $\mathsf{X}_{\mathsf{pr}} = (\mathcal{F}, \hat{\mathcal{F}})$ is two families of quadratic dot product function derived from $\vec{\mathsf{vk}}$, $\vec{\mathsf{M}}$, and $\vec{p}^*_z$.

By definition, we have $\Pr[\mathsf{E}_0] = \epsilon$.

$\mathsf{Hyb}_1$**:** This game is the same as $\mathsf{Hyb}_0$ except at the end of the experiment, we additional computes

$$\mathsf{pr.Extract}(\mathsf{crs}_{\mathsf{pr}}, \mathsf{X}_{\mathsf{pr}}, \pi^*_{\mathsf{pr}}) \to \tilde{\mathsf{W}}_{\mathsf{pr}} = (\mathbf{z}_1, \mathbf{c}_1, \mathbf{c}'_1, \ldots, \mathbf{z}_N, \mathbf{c}_N, \mathbf{c}'_N),$$

where $\mathsf{pr.Extract}$ is an extractor of $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$. If $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_i, \mathsf{M}_i, (\mathbf{z}_i, \mathbf{c}_i)) = \perp$, i.e.,

$$(\mathbf{G}_{2,n} + \mathbf{B}_i)\mathbf{c}_i \neq \mathbf{A}_i\mathbf{z}_i - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}_i, \mathsf{M}_i) \bmod q \vee \|\mathbf{z}_i\|_\infty > t\phi T,$$

then the experiment is aborted, where $\mathsf{vk}_i = (\mathbf{A}_i, \mathbf{B}_i)$.

**Lemma 7.** *If $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$ is an argument of knowledge, then we have* $\Pr[\mathsf{E}_1] \geq \Pr[\mathsf{E}_0] - \mathsf{negl}(\lambda)$.

*Proof of Lemma.* The only difference between $\mathsf{Hyb}_0$ and $\mathsf{Hyb}_1$ is the extra check performed in $\mathsf{Hyb}_1$. That is, for $\mathsf{Hyb}_0$ to output $\top$ and $\mathsf{Hyb}_1$ to be aborted, it must be the case that

- $\mathsf{pr.Verify}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta'}}(\mathsf{crs}_{\mathsf{pr}}, \mathsf{X}^*_{\mathsf{pr}}, \pi^*_{\mathsf{pr}}) = \top \wedge \|\vec{p}^*_z\|_\infty \leq t\phi T/2$ and
- $(\mathbf{G}_{2,n} + \mathbf{B}_i)\mathbf{c}_i \neq \mathbf{A}_i\mathbf{z}_i - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}_i, \mathsf{M}_i) \bmod q \vee \|\mathbf{z}_i\|_\infty > t\phi T$.

We first assume that $\mathsf{pr.Verify}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta'}}(\mathsf{crs}_{\mathsf{pr}}, \mathsf{X}^*_{\mathsf{pr}}, \pi^*_{\mathsf{pr}}) = \top \wedge \|\vec{p}^*_z\|_\infty \leq t\phi T/2$. Since $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$ is an argument of knowledge for the relations $\mathsf{Rel}_{\mathsf{pr},\beta}$ and $\mathsf{Rel}_{\mathsf{pr},\beta'}$ the extracted witness $\tilde{\mathsf{W}}_{\mathsf{pr}} = (\mathbf{z}_1, \mathbf{c}_1, \mathbf{c}'_1, \ldots, \mathbf{z}_N, \mathbf{c}_N, \mathbf{c}'_N)$ satisfies

$$\Big((\mathcal{F}, \hat{\mathcal{F}}), (\mathbf{z}_1, \mathbf{c}_1, \mathbf{c}'_1, \ldots, \mathbf{z}_N, \mathbf{c}_N, \mathbf{c}'_N)\Big) \in \mathsf{Rel}_{\mathsf{pr},\beta'}$$

with overwhelming probability. This implies that

$$\begin{aligned}
\langle \mathbf{c}_i, \mathbf{c}'_i - \mathbf{1}_m \rangle &= 0, \\
\mathbf{c}'_i - \Sigma_{-1}(\mathbf{c}_i) &= 0, \\
\mathbf{A}_i\mathbf{z}_i - (\mathbf{G}_{2,n} + \mathbf{B}_i)\mathbf{c}_i - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}_i, \mathsf{M}_i) &= \mathbf{0} \bmod q, \\
P_z \begin{pmatrix} \vec{z}_1 \\ \vdots \\ \vec{z}_N \end{pmatrix} - \vec{p}^*_z &= 0 \bmod q, \\
\sum_{j=1}^{N} (\|\mathbf{z}_j\|_2^2 + \|\mathbf{c}_j\|_2^2 + \|\mathbf{c}'_j\|_2^2) \leq \beta'^2 &= 128\beta^2/30.
\end{aligned}$$

If $md \leq 15Q/128$, the $\ell_2$-norm of $\left(\mathbf{c}_1^\top \| \mathbf{c}_1'^\top \| \cdots \| \mathbf{c}_N^\top \| \mathbf{c}_N'^\top\right)^\top$ is smaller than $\sqrt{Q}$ by at least the slack factor $\sqrt{128/30}$. Hence, the above first and second equations imply that $\mathbf{c}_i$ has binary coefficients. Next, the third equation immediately implies $(\mathbf{G}_{2,n} + \mathbf{B}_i)\mathbf{c}_i = \mathbf{A}_i\mathbf{z}_i - \mathsf{H}_{\mathsf{M}}(\mathsf{vk}_i, \mathsf{M}_i) \bmod q$. Finally, by Lemma 3, the forth equation and the fact that $\|\vec{p}^*_z\|_\infty \leq t\phi T/2$ imply that $\|\mathbf{z}_i\|_\infty \leq t\phi T$ with overwhelming probability.

Putting these together, if $\mathsf{pr.Verify}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta'}}(\mathsf{crs}_{\mathsf{pr}}, \mathsf{X}^*_{\mathsf{pr}}, \pi^*_{\mathsf{pr}}) = \top \wedge \|\vec{p}^*_z\|_\infty \leq t\phi T/2$, then we have $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_i, \mathsf{M}_i, (\mathbf{z}_i, \mathbf{c}_i)) = \top$. Therefore, we have $\Pr[\mathsf{E}_1] \geq \Pr[\mathsf{E}_0] - \mathsf{negl}(\lambda)$. $\square$

**Lemma 8.** *If $\Pi_{\mathsf{DS}}^{\mathsf{CLMQ}}$ is unforgeable, then we have* $\Pr[\mathsf{E}_1] = \mathsf{negl}(\lambda)$.

*Proof of Lemma 8.* This proof is (almost) the same as the proof of [33, Lemma 7.10].

Assuming that there exists a PPT algorithm $\mathcal{A}$ with a non-negligible advantage $\epsilon$ in $\mathsf{Hyb}_1$. We use $\mathcal{A}$ to build an efficient algorithm $\mathcal{B}$ that breaks the unforgeability of $\Pi_{\mathsf{DS}}^{\mathsf{CLMQ}}$.

1. $\mathcal{B}$ receives the verification key $\mathsf{vk}^*$ from its challenger.
2. $\mathcal{B}$ gives $\mathsf{pp} := 1^\lambda$ and $\mathsf{vk}^*$ to $\mathcal{A}$.
3. Whenever $\mathcal{A}$ makes a signing query on a message $\mathsf{M}$, $\mathcal{B}$ makes a signing query on $\mathsf{M}$ and gets a signature $\sigma$. It replies to $\mathcal{A}$ with $\sigma$.
4. At the end of the experiment, $\mathcal{A}$ outputs $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \dots, \mathsf{vk}_N)$, $\vec{\mathsf{M}} = (\mathsf{M}_1, \dots, \mathsf{M}_N)$, and $\sigma^* = (\vec{p}_z^*, \pi_{\mathsf{pr}}^*)$. $\mathcal{B}$ check that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathcal{A}$ did not issue a signing query on $\mathsf{M}_i$, and that
$$\mathsf{pr.Verify}^{\mathsf{H}_{\mathsf{pr}}}_{\mathsf{Rel}_{\mathsf{pr},\beta},\mathsf{Rel}_{\mathsf{pr},\beta'}}(\mathsf{crs}_{\mathsf{pr}}, \mathsf{X}^*_{\mathsf{pr}}, \pi^*_{\mathsf{pr}}) = \top \wedge \|\vec{p}_z^*\|_\infty \le t\phi T/2.$$
If any checks do not pass, $\mathcal{B}$ aborts. Otherwise, it computes
$$\mathsf{pr.Extract}(\mathsf{crs}_{\mathsf{pr}}, \mathsf{X}_{\mathsf{pr}}, \pi^*_{\mathsf{pr}}) \to \tilde{\mathsf{W}}_{\mathsf{pr}} = (\mathbf{z}_1, \mathbf{c}_1, \mathbf{c}'_1, \dots, \mathbf{z}_N, \mathbf{c}_N, \mathbf{c}'_N)$$
and outputs $(\mathsf{M}_i, (\mathbf{z}_i, \mathbf{c}_i))$ as its forgery.

By construction, $\mathcal{B}$ simulates an execution of $\mathsf{Hyb}_1$ for $\mathcal{A}$. Thus, with a probability of at least $\epsilon$, $\mathcal{A}$ outputs $\vec{\mathsf{vk}}$, $\vec{\mathsf{M}}$, and $\sigma^* = (\vec{p}_z^*, \pi^*_{\mathsf{pr}})$, where $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathcal{A}$ never queried the signing oracle on $\mathsf{M}_i$, and $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_i, \mathsf{M}_i, (\mathbf{z}_i, \mathbf{c}_i)) = \top$. Therefore, $\mathcal{B}$ succeeds with the advantage $\epsilon$. □

By Lemmas 7 and 8, we have $\Pr[\mathsf{E}_0] = \mathsf{negl}(\lambda)$. □

### 4.4 Parameter Selection and Efficiency

**Parameter Selection.** Here, we first summarize the conditions that our parameters in Table 5 must satisfy for the correctness and unforgeability of $\Pi_{\mathsf{AS}}$. These conditions are only asymptotic. We then show a set of concrete parameters in Table 6 for 128 bits of security.

- For $\Pi_{\mathsf{AS}}$:
  - $N(nd(t\phi T)^2 + md) < 15Q/128$.
  - $\beta = \sqrt{Q}$ and $\beta' = \sqrt{128/30\sqrt{Q}}$.
- For $\Pi_{\mathsf{SNARK}}^{\mathsf{pr}}$:
  - The $\mathsf{MSIS}_{\kappa_1, 2\beta', Q}$, $\mathsf{MSIS}_{\kappa_2, 2\beta', Q}$, and $\mathsf{MSIS}_{\kappa_0, \beta'', Q}$ assumptions hold.
  - $\beta'' = \max\{8T_{\mathsf{Ch}}(b+1)\beta', 2(b+1)\beta' + 4T_{\mathsf{Ch}}\sqrt{128/30}\beta\}$.
- For $\Pi_{\mathsf{DS}}^{\mathsf{CLMQ}}$:
  - $m = n\lceil \log q \rceil$.
  - $t = \sqrt{2(1 + \log nd + \lambda)/\log e}$.
  - $T = m\eta\sqrt{2nd}$.

**Table 6.** Concrete parameters for our scheme.

| Parameter | Value |
|---|---|
| $q$ | $\approx 2^{30}$ |
| $d$ | 64 |
| $n$ | 9 |
| $m$ | 270 |
| $\eta$ | 2 |
| $\phi$ | 14 |
| $T$ | 18328.21 |
| $\mathsf{err}$ | $2^{-256}$ |
| $t$ | 13.84 |
| $N$ | $2^{10} \sim 2^{20}$ |
| $q'$ | $\approx 2^{46}$ |
| $Q$ | $\approx 2^{76}$ |

- $t\phi T \leq q/2$.
- $\beta_0 = \sqrt{2nd(t\phi T)^2 + md}$.
- The $\mathsf{MSIS}_{n,\beta_0,q}$ and $\mathsf{MLWE}_{n,\eta,q}$ assumptions hold.

**Concrete Efficiency.** We evaluate concrete aggregated signature sizes of $\Pi_{\mathsf{AS}}$. The size of the aggregated signature is given by the size of the proof $\pi$ of $\Pi_{\mathsf{SNARK}}$. Furthermore, the size of the proof $\pi$ is given by $\vec{p} \in \mathbb{Z}_q^{128}$ and $\pi_{\mathsf{pr}}$, which is a proof of $\Pi^{\mathsf{pr}}_{\mathsf{SNARK}}$. Then, we have the following proof size in bits:

$$\underbrace{128 \log q}_{|\vec{p}|} + |\pi_{\mathsf{pr}}|.$$

To evaluate the concrete size of $\pi_{\mathsf{pr}}$, we refer to [3, Sec. F]. Table 7 contains the aggregated signature sizes of the comparison of our scheme with other lattice-based aggregate signature schemes in the literature. Table 8 contains the aggregated signature sizes of our scheme for the number of signatures to be aggregated $N$ varying between $2^{10}$ and $2^{20}$.

**Table 7.** Comparison of various lattice-based many-time and non-interactive aggregate signature schemes, and their aggregation over $2^{10}$ signatures. We assume 128-bit security. Schemes that do not support aggregation are marked with *.

| Scheme | Individual Sig. Size | Aggregated Sig. Size |
|---|---:|---:|
| Dilithium3* | 3.3 KB | 3300 KB |
| Falcon-512* | 0.6 KB | 618 KB |
| [9,10] | 8.9 KB | 4400 KB |
| [20] | 7.9 KB | 7444 KB |
| Ours | 6.3 KB | 63 KB |

**Table 8.** Aggregate signature sizes for our scheme with a varying $N$.

| $N$ | Aggregated Sig. Size |
|---|---|
| $2^{10}$ | 63.48 KB |
| $2^{12}$ | 65.02 KB |
| $2^{14}$ | 69.38 KB |
| $2^{16}$ | 77.46 KB |
| $2^{18}$ | 103.42 KB |
| $2^{20}$ | 131.54 KB |

We note that we can also obtain an aggregate signature scheme by simply combining LaBRADOR for rank 1 constraint systems (R1CS) and the CLMQ signature scheme. However, this construction is less efficient than ours in Sec. 4.1. This is because we have to convert its verification algorithm into an R1CS. This conversion induces a quite larger statement and witness. As a result, the simple combination has a larger aggregated signature size.

## 5 Conclusion

In this paper, we presented the first aggregate signature scheme such that: (1) its security is based on the standard lattice-based assumptions (MSIS and MLWE) in the random oracle model, (2) the size of the aggregated signature is logarithmic in $N$, (3) it is many-time, and (4) it can be aggregated non-interactively. In addition, our scheme is quite compact because the size of the aggregated signature required to aggregate $2^{20}$ signatures is only a few hundred kilobytes. This result shows that our scheme is superior to the existing lattice-based aggregate signature schemes in compressing many signatures.

**Acknowledgment.** This research was in part conducted under a contract of ”Research and development on new generation cryptography for secure wireless communication services” among “Research and Development for Expansion of Radio Wave Resources (JPJ000254)”, which was supported by the Ministry of Internal Affairs and Communications, Japan. This work was in part supported by JSPS KAKENHI Grant Numbers JP22H03590, JP22K19773.

## References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_28 (Cited on page 4.)
2. Albrecht, M.R., Cini, V., Lai, R.W.F., Malavolta, G., Thyagarajan, S.A.K.: Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 102–132. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_4 (Cited on page 3.)

3. Albrecht, M.R., Davidson, A., Deo, A., Gardham, D.: Crypto dark matter on the torus: Oblivious PRFs from shallow PRFs and FHE. Cryptology ePrint Archive, Report 2023/232 (2023), https://eprint.iacr.org/2023/232 (Cited on page 20.)
4. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (Jul 2007). https://doi.org/10.1007/978-3-540-73420-8_37 (Cited on page 1.)
5. Beullens, W., Seiler, G.: LaBRADOR: Compact proofs for R1CS from module-SIS. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 518–548. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_17 (Cited on page 3, 9, 10, 13.)
6. Boldyreva, A., Gentry, C., O'Neill, A., Yum, D.H.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (eds.) ACM CCS 2007. pp. 276–285. ACM Press (Oct 2007). https://doi.org/10.1145/1315245.1315280 (Cited on page 1.)
7. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_26 (Cited on page 1, 8.)
8. Boneh, D., Kim, S.: One-time and interactive aggregate signatures from lattices. preprint (2020) (Cited on page 1, 2.)
9. Boudgoust, K., Roux-Langlois, A.: Non-interactive half-aggregate signatures based on module lattices-a first attempt. Cryptology ePrint Archive (2021) (Cited on page 1, 2, 20.)
10. Boudgoust, K., Roux-Langlois, A.: Overfull: Too large aggregate signatures based on lattices. The Computer Journal p. bxad013 (2023) (Cited on page 1, 2, 20.)
11. Brakerski, Z., Brodsky, M.F., Kalai, Y.T., Lombardi, A., Paneth, O.: SNARGs for monotone policy batch NP. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 252–283. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_9 (Cited on page 3.)
12. Brogle, K., Goldberg, S., Reyzin, L.: Sequential aggregate signatures with lazy verification from trapdoor permutations - (extended abstract). In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 644–662. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34961-4_39 (Cited on page 1.)
13. Chen, Y., Lombardi, A., Ma, F., Quach, W.: Does fiat-shamir require a cryptographic hash function? In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 334–363. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_12 (Cited on page 3, 13.)
14. Choudhuri, A.R., Jain, A., Jin, Z.: SNARGs for $\mathcal{P}$ from LWE. In: 62nd FOCS. pp. 68–79. IEEE Computer Society Press (Feb 2022). https://doi.org/10.1109/FOCS52979.2021.00016 (Cited on page 3.)
15. Devadas, L., Goyal, R., Kalai, Y., Vaikuntanathan, V.: Rate-1 non-interactive arguments for batch-NP and applications. In: 63rd FOCS. pp. 1057–1068. IEEE Computer Society Press (Oct / Nov 2022). https://doi.org/10.1109/FOCS54457.2022.00103 (Cited on page 3.)
16. Doröz, Y., Hoffstein, J., Silverman, J.H., Sunar, B.: MMSAT: A scheme for multimessage multiuser signature aggregation. Cryptology ePrint Archive, Report 2020/520 (2020), https://eprint.iacr.org/2020/520 (Cited on page 1.)

17. Fischlin, M., Lehmann, A., Schröder, D.: History-free sequential aggregate signatures. In: Visconti, I., Prisco, R.D. (eds.) SCN 12. LNCS, vol. 7485, pp. 113–130. Springer, Heidelberg (Sep 2012). https://doi.org/10.1007/978-3-642-32928-9_7 (Cited on page 1.)
18. Gentry, C., Halevi, S., Lyubashevsky, V.: Practical non-interactive publicly verifiable secret sharing with thousands of parties. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 458–487. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-06944-4_16 (Cited on page 6.)
19. Gentry, C., O'Neill, A., Reyzin, L.: A unified framework for trapdoor-permutation-based sequential aggregate signatures. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 34–57. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_2 (Cited on page 1.)
20. Jeudy, C., Roux-Langlois, A., Sanders, O.: Phoenix: Hash-and-sign with aborts from lattice gadgets. Cryptology ePrint Archive (2023) (Cited on page 2, 20.)
21. Kalai, Y., Lombardi, A., Vaikuntanathan, V., Wichs, D.: Boosting batch arguments and ram delegation. In: Proceedings of the 55th Annual ACM Symposium on Theory of Computing. pp. 1545–1552 (2023) (Cited on page 3.)
22. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015) (Cited on page 5.)
23. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_28 (Cited on page 1.)
24. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_5 (Cited on page 1.)
25. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_43 (Cited on page 4, 5, 6, 13.)
26. Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 71–101. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_3 (Cited on page 4.)
27. Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 204–224. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9_8 (Cited on page 6.)
28. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_41 (Cited on page 5.)
29. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: 45th FOCS. pp. 372–381. IEEE Computer Society Press (Oct 2004). https://doi.org/10.1109/FOCS.2004.72 (Cited on page 4.)
30. Neven, G.: Efficient sequential aggregate signed data. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 52–69. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_4 (Cited on page 1.)

31. Paneth, O., Pass, R.: Incrementally verifiable computation via rate-1 batch arguments. In: 63rd FOCS. pp. 1045–1056. IEEE Computer Society Press (Oct / Nov 2022). https://doi.org/10.1109/FOCS54457.2022.00102 (Cited on page 3.)
32. Sato, S., Shikata, J.: Identity-based interactive aggregate signatures from lattices. In: Seo, S.H., Seo, H. (eds.) Information Security and Cryptology – ICISC 2022. pp. 408–432. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-29371-9_20 (Cited on page 2.)
33. Waters, B., Wu, D.J.: Batch arguments for sfNP and more from standard bilinear group assumptions. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 433–463. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_15 (Cited on page 8, 13, 19.)

# Table of Contents