Compact Signature Aggregation from Modulo-Lattices

Toi Tomita[1] and Junji Shikata[1]

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

Abstract

An aggregate signature scheme allows multiple signatures generated by different people for different messages to be aggregated into a compact aggregate signature. We propose the first signature aggregation scheme that (1) grows the size of the aggregate signature only logarithmically in the number of signatures to be aggregated, (2) is many-time, (3) supports non-interactive aggregation, (4) its security is based on the standard lattice assumption in the random oracle model. To obtain the result, we construct a new compact non-interactive batch argument (BARG) for NP. Our BARG has a very compact proof and its security is based on the standard modulo lattice assumptions in the random oracle model.

1 Introduction

1.1 Background

The notion of aggregate signature schemes, introduced by Boneh, Gentry, Lynn, and Shacham [BGLS03], allows individual signatures $\sigma_1, \dots, \sigma_N$ for different messages $\mathsf{M}_1, \dots, \mathsf{M}_N$ created by N signers to be aggregated into a compact signature $\hat{\sigma}$. The aggregated signature $\hat{\sigma}$ gives the verifier confidence that all the signatures aggregated into $\hat{\sigma}$ are valid. The original motivation for signature aggregation was to compress certificate chains and aggregate signatures in secure BGP. Recently, it has gained significant practical interest in the context of blockchains.

A plethora of work proposed highly efficient aggregate signature schemes using bilinear maps [BGLS03, LOS$^+$06, BGOY07, BNN07, FLS12] or trapdoor permutations [LMRS04, Nev08, BGR12, GOR18]. On the other hand, post-quantum, especially lattice-based aggregate signature schemes have not been proposed much. Boneh and Kim [BK20] proposed two lattice-based aggregate signature schemes whose security is based on the standard Short Integer Solution (SIS) assumption, and where the aggregated signature size grows at most logarithmically with the number of signatures being aggregated. However, the first scheme is a *one-time* scheme, and the second scheme requires *interactions for aggregation.* Subsequently, several works [DHSS20, BR21, BT23] proposed lattice-based aggregate signature schemes that are many-time and support non-interactive aggregation. However, the aggregate signature size of their schemes grows *linearly* with the number of the number of signatures being aggregated. Based on the above, the following questions are addressed in this paper:

Can we construct a lattice-based aggregate signature scheme that (1) grows the aggregate signature size sublinearly, (2) is many-time, and (3) supports non-interactive aggregation?

1.2 Our Contributions

In this paper, we answer the above question in the affirmative. More precisely, we construct the first lattice-based aggregate signature scheme that only grows the aggregate signature size logarithmically, is not one-time, and supports non-interactive aggregation. Our scheme is obtained by a general approach using a

Table 1: Comparison of lattice-based aggregate signature schemes. The column $|\hat{\sigma}|$ indicates the size of aggregate signature. λ is the security parameter and N is the number of signatures to be aggregated. †PFR stands for the Partial Fourier Recovery assumption, which is a non-standard lattice assumption introduced in [DHSS20].

Scheme	$\|\hat{\sigma}\|$	Many-time	Non-interactive	Assumption
[BK20, Sec. 4]	$O(\log N) \cdot \mathsf{poly}(\lambda)$	-	✓	SIS
[BK20, Sec. 6]	$O(\log N) \cdot \mathsf{poly}(\lambda)$	✓	-	SIS
[DHSS20]	$O(N) \cdot \mathsf{poly}(\lambda)$	✓	✓	PFR†
[BR21, BT23]	$O(N) \cdot \mathsf{poly}(\lambda)$	✓	✓	MSIS & MLWE
Ours	$O(\log N) \cdot \mathsf{poly}(\lambda)$	✓	✓	MSIS & MLWE

non-interactive batch argument (BARG) for NP.

A BARG for NP allows a prover to construct a proof of m NP statements, where the size of the proof grows sublinearly with m, and to convince the verifier that all these statements are true. By the following straightforward construction, a BARG for NP directly yields an aggregate signature scheme. Consider the NP relation $R((\mathsf{vk}, \mathsf{M}), \sigma)$, which takes the verification key-message pair $(\mathsf{vk}, \mathsf{M})$ as an NP statement and the signature σ as an NP witness, and returns 1 if σ is a valid signature on M under vk. An aggregate signature on $(\mathsf{vk}_1, \mathsf{M}_1, \sigma_1), \ldots, (\mathsf{vk}_N, \mathsf{M}_N, \sigma_N)$ is a BARG proof that $R((\mathsf{vk}_i, \mathsf{M}_i), \sigma_i) = 1$ for all $i = 1, \ldots, N$. The compactness of the BARG ensures that the size of the aggregate signature is sublinear in N.

The appeal of this general approach is that BARG can be used to lift *any* ordinary signature scheme into an aggregate signature scheme. This means that a lattice-based (full-fledged) aggregate signature can be constructed from a lattice-based signature scheme and a lattice-based BARG for NP. Recently, several lattice-based BARGs have been proposed [CJJ22, ACL$^+$22, DGKV22], and we can use these BARGs to construct lattice-based aggregate signature schemes. Unfortunately, these BARGs have some drawbacks. The construction of [ACL$^+$22] relies on a new, non-standard lattice assumption. The constructions of [CJJ22] and [DGKV22] have the proof of size $\mathsf{poly}(\log N, |C|)$, where C is the circuit for the NP relation.

For our purposes, we construct a new efficient lattice-based BARG for NP. Our BARG has proof size $\mathsf{poly}(\log N, \log |C|)$, and its security is based on the modulo short integer solution (MSIS) and modulo learning with errors (MLWE) assumptions, which are known as standard lattice assumptions, in the random oracle model. Our BARG is obtained by combining LaBRADOR, a highly compact proof system proposed by Beullens and Seiler [BS22], and the partially-decryptable commitment scheme by Esgin, Steinfeld, and Zhao [ESZ22].

2 Preliminaries

Notations. For positive integer n, let $[n]$ denote the set of integers $\{1, \ldots, n\}$. For positive integers d and q, let $\mathcal{R}_q$ denote the polynomial ring $\mathbb{Z}_q[X]/(X^d + 1)$. If $a = a_0 + \sum_{i=1}^{d-1} a_i X^i \in \mathcal{R}_q$, then we denote by $\mathsf{const}(a)$ the constant term of a, i.e., $\mathsf{const}(a) = a_0$. The ring $\mathcal{R}_q$ has a group of automorphisms $\mathsf{Aut}(\mathcal{R}_q)$ that is isomorphic to $\mathbb{Z}_{2d}^\times$. Let $\sigma_{-1} \in \mathsf{Aut}(\mathcal{R}_q)$ be defined by $\sigma_{-1}(X) = X^{-1}$.

2.1 Digital Signature and Aggregate Signature

Here, we recall the definition of standard and aggregate signature schemes.

Definition 2.1 (Digital Signature). *A digital signature (Sig) scheme with message space $\mathcal{M}$ is a tuple of probabilistic polynomial time (PPT) algorithms $\Pi_{\mathsf{Sig}} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ with the following properties:*

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

- $\mathsf{Sign}(\mathsf{sk}, \mathsf{M}) \to \sigma$: *On input of the signing key* sk *and a message* $\mathsf{M} \in \mathcal{M}$, *the signing algorithm outputs a signature* σ.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{M}, \sigma) \to b$: *On input of the verification key* vk, *a message* $\mathsf{M} \in \mathcal{M}$, *and a signature* σ, *the verification algorithm outputs a bit* $b \in \{0, 1\}$. *The verification algorithm is deterministic.*

In addition, the above algorithms should have the following properties.

Definition 2.2 (Correctness). *A Sig* Π_{Sig} *is correct if for all* $\lambda \in \mathbb{N}$ *and* $\mathsf{M} \in \mathcal{M}$, *it holds that*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{M}, \mathsf{Sign}(\mathsf{sk}, \mathsf{M})) = 1\right] = 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{sk}, \mathsf{vk}) \overset{\$}{\leftarrow} \mathsf{KGen}(1^\lambda)$ *and the probability is taken over the randomness of all algorithms.*

Definition 2.3 (Unforgeability). *Define the signature unforgeability game between an adversary* $\mathcal{A}$ *and a challenger as follows:*

1. *The challenger samples* $(\mathsf{sk}, \mathsf{vk}) \overset{\$}{\leftarrow} \mathsf{KGen}(1^\lambda)$ *and gives* vk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *can now make signing queries on message* $\mathsf{M} \in \mathcal{M}$ *of its choosing. On each query* M, *the challenger replies with* $\sigma \overset{\$}{\leftarrow} \mathsf{Sign}(\mathsf{sk}, \mathsf{M})$.
3. *At the end of the game,* $\mathcal{A}$ *outputs a message-signature pair* (M^*, σ^*). *The output of the game is* $b = 1$ *if* $\mathsf{Verify}(\mathsf{vk}, \mathsf{M}^*, \sigma^*) = 1$ *and* $\mathcal{A}$ *did not make a signing query on* M^*. *Otherwise, the output is* $b = 0$.

A Sig Π_{Sig} *is unforgeable if for all PPT adversaries* $\mathcal{A}$, *it holds that* $\Pr[b = 1] = \mathsf{negl}(\lambda)$ *in the above unforgeability game.*

Definition 2.4 (Aggregate Signature [BGLS03, adapted]). *A bounded aggregate signature (AggSig) scheme with message space* $\mathcal{M}$ *is a tuple of PPT algorithms* $\Pi_{\mathsf{AggSig}} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Aggregate}, \mathsf{AggVerify})$ *with the following properties:*

- $\mathsf{Setup}(1^\lambda, 1^N) \to \mathsf{pp}$: *On input of the security parameter* λ *and an aggregation bound* N, *the setup algorithm outputs the public parameter* pp.
- $\mathsf{KGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$: *On input of the public parameter* pp, *the key-generation algorithm outputs a signing key* sk *and a verification key* vk.
- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{M}) \to \sigma$: *On input of the public parameter* pp, *the signing key* sk, *and a message* $\mathsf{M} \in \mathcal{M}$, *the signing algorithm outputs a signature* σ.
- $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathsf{M}, \sigma) \to b$: *On input of the public parameter* pp, *the verification key* vk, *a message* $\mathsf{M} \in \mathcal{M}$, *and a signature* σ, *the verification algorithm outputs a bit* $b \in \{0, 1\}$.
- $\mathsf{Aggregate}(\mathsf{pp}, \{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i \in [T]}) \to \sigma_{\mathsf{agg}}$: *On input of the public parameter* pp *and a collection of up to* $T \le N$ *verification keys* vk_i, *messages* M_i, *and signatures* σ_i, *the aggregation algorithm outputs an aggregate signature* σ_{agg}.
- $\mathsf{AggVerify}(\mathsf{pp}, (\mathsf{vk}_1, \ldots, \mathsf{vk}_T), (\mathsf{M}_1, \ldots, \mathsf{M}_T), \sigma_{\mathsf{agg}}) \to b$: *On input of the public parameter* pp, *a collection of* $T \le N$ *verification keys* vk_i, *messages* M_i, *and an aggregate signature* σ_{agg}, *the aggregate verification algorithm outputs a bit* $b \in \{0, 1\}$.

In addition, the above algorithms should have the following properties.

Definition 2.5 (Correctness). *An AggSig* Π_{AggSig} *is correct if for all* $\lambda, N \in \mathbb{N}$ *and* $\mathsf{M} \in \mathcal{M}$, *it holds that*

$$\Pr\left[\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathsf{M}, \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{M})) = 1\right] = 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N)$, $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp})$, *and the probability is taken over the randomness of all algorithms. In addition, for all collections* $\{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i\in[T]}$ *where* $T \leq N$ *and* $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_i, \mathsf{M}_i, \sigma_i) = 1$ *for all* $i \in [T]$,

$$\Pr\left[\mathsf{AggVerify}(\mathsf{pp}, (\mathsf{vk}_1, \dots, \mathsf{vk}_T), (\mathsf{M}_1, \dots, \mathsf{M}_T), \mathsf{Aggregate}(\mathsf{pp}, \{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i\in[T]}))\right] = 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N)$, $(\mathsf{sk}_i, \mathsf{vk}_i) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp})$ *for all* $i \in [T]$, *and the probability is taken over the randomness of all algorithms.*

Definition 2.6 (Unforgeability). *Define the unforgeability game for the aggregate signature between an adversary* $\mathcal{A}$ *and a challenger as follows:*

1. *The challenger samples* $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N)$ *and* $(\mathsf{sk}^*, \mathsf{vk}^*) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp})$ *and gives* pp *and* vk^* *to* $\mathcal{A}$.
2. $\mathcal{A}$ *can now make signing queries on messages* $\mathsf{M} \in \mathcal{M}$ *of its choosing. On each query* M, *the challenger replies with* $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}^*, \mathsf{M})$.
3. *At the end of the game,* $\mathcal{A}$ *outputs a tuple of verification keys* $(\mathsf{vk}_1, \dots, \mathsf{vk}_T)$, *a tuple of messages* $(\mathsf{M}_1, \dots, \mathsf{M}_T)$ *with* $T \leq N$, *and a signature* σ^*.
4. *The output of the game is* $b = 1$ *if there exists an index* $i^* \in [T]$, *where* $\mathsf{vk}_{i^*} = \mathsf{vk}^*$, $\mathcal{A}$ *did not make a signing query on* M_{i^*}, *and* $\mathsf{AggVerify}(\mathsf{pp}, (\mathsf{vk}_1, \dots, \mathsf{vk}_T), (\mathsf{M}_1, \dots, \mathsf{M}_T), \sigma^*) = 1$. *Otherwise, the output is* $b = 0$.

An AggSig Π_{AggSig} *is unforgeable if for all PPT adversaries* $\mathcal{A}$ *and all polynomials* $N = N(\lambda)$, *it holds that* $\Pr[b = 1] = \mathsf{negl}(\lambda)$ *in the above unforgeability game.*

Definition 2.7 (Efficiency). *An AggSig* Π_{AggSig} *is efficient if there exists a fixed polynomial* $\mathsf{poly}(\cdot,\cdot)$, *the size of the aggregate signature* σ_{agg} *satisfies* $|\sigma_{\mathsf{agg}}| = \mathsf{poly}(\lambda, \log T)$.

2.2 Non-Interactive Batch Arguments for NP

Here, we define non-interactive batch arguments for NP. First, we consider the NP-complete language of the binary rank-1 constraint system (R1CS). We now define the R1CS language.

Definition 2.8 (Binary Rank-1 Constraint System). *Let* $S, x, w \in \mathbb{N}$ *be positive integers and* $(\mathbf{A}, \mathbf{B}, \mathbf{C}) \in \left(\{0,1\}^{S\times(x+w)}\right)^3$ *be matrices. We define*

$$\mathcal{L}_{\mathsf{R1CS},(\mathbf{A},\mathbf{B},\mathbf{C})} := \left\{ \mathbf{x} \in \{0,1\}^x \middle| \exists \mathbf{w} \in \{0,1\}^w : \mathbf{A}\begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix} \circ \mathbf{B}\begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix} = \mathbf{C}\begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix} \right\}$$

to be the language of (binary) rank-1 constraint system (R1CS), where $\circ$ *denotes the component-wise multiplication. We also define the relation* $R_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ *as follows:*

$$R_{(\mathbf{A},\mathbf{B},\mathbf{C})} = \left\{ (\mathbf{x}, \mathbf{w}) \in \{0,1\}^x \times \{0,1\}^w \middle| \mathbf{A}\begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix} \circ \mathbf{B}\begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix} = \mathbf{C}\begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix} \right\}.$$

Definition 2.9 (Batch Arguments for R1CS). *Let* $(\mathbf{A}, \mathbf{B}, \mathbf{C}) \in \left(\{0,1\}^{S\times(x+w)}\right)^3$ *be matrices. A non-interactive batch argument (BARG) for an R1CS instance* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *is a tuple of PPT algorithms* $\Pi_{\mathsf{BARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *with the following properties:*

- $\mathsf{Setup}(1^\lambda, 1^N) \to \mathsf{crs}$: *On input of the security parameter* $\lambda \in \mathbb{N}$ *and the number of instances* $N \in \mathbb{N}$, *the setup algorithm outputs a common reference string* crs.
- $\mathsf{Prove}(\mathsf{crs}, \vec{\mathbf{x}}, \vec{\mathbf{w}}) \to \pi$: *On input of the common reference string* crs, *statements* $\vec{\mathbf{x}} = (\mathbf{x}_1, \dots, \mathbf{x}_N) \in (\{0,1\}^x)^N$, *and witnesses* $\vec{\mathbf{w}} = (\mathbf{w}_1, \dots, \mathbf{w}_N) \in (\{0,1\}^w)^N$, *the prove algorithm outputs a proof* π.

- $\mathsf{Verify}(\mathsf{crs}, \vec{\mathbf{x}}, \pi) \to b$: *On input of the common reference string* crs*, statements* $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in (\{0,1\}^x)^N$*, and a proof* π*, the verification algorithm outputs a bit* $b \in \{0,1\}$*.*

In addition, the above algorithms should have the following properties.

Definition 2.10 (Completeness). *Let* $(\mathbf{A}, \mathbf{B}, \mathbf{C}) \in \left(\{0,1\}^{S \times (x+w)}\right)^3$ *be matrices. A BARG* Π_{BARG} *for an R1CS instance* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *is complete if for all* $\lambda, N \in \mathbb{N}$*, all statements* $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in (\{0,1\}^x)^N$*, and all witnesses* $\vec{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_N) \in (\{0,1\}^w)^N$ *with* $(\mathbf{x}_i, \mathbf{w}_i) \in R_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ *for all* $i \in [N]$*, it holds that*

$$\Pr\left[\mathsf{Verify}\left(\mathsf{crs}, \vec{\mathbf{x}}, \mathsf{Prove}(\mathsf{crs}, \vec{\mathbf{x}}, \vec{\mathbf{w}})\right)\right] = 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N)$ *and the probability is taken over the randomness of all algorithms.*

Definition 2.11 (Somewhere Argument of Knowledge [CJJ22]). *Let* $(\mathbf{A}, \mathbf{B}, \mathbf{C}) \in \left(\{0,1\}^{S \times (x+w)}\right)^3$ *be matrices. A BARG* Π_{BARG} *for an R1CS instance* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *is somewhere argument of knowledge if there exists a pair of PPT algorithms* $(\mathsf{TrapSetup}, \mathsf{Extract})$ *with the following properties:*

- $\mathsf{TrapSetup}(1^\lambda, 1^N, i^*) \to (\mathsf{crs}^*, \mathsf{td})$*: On input of the security parameter* $\lambda \in \mathbb{N}$*, the number of instances* $N \in \mathbb{N}$*, and an index* $i^* \in [N]$*, the trapdoor setup algorithm outputs a common reference string* crs^* *and an extraction trapdoor* td*.*

- $\mathsf{Extract}(\mathsf{td}, \vec{\mathbf{x}}, \pi) \to \mathbf{w}^*$*: On input of the trapdoor* td*, statements* $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in (\{0,1\}^x)^N$*, and a proof* π*, the extraction algorithm outputs a witness* $\mathbf{w}^* \in \{0,1\}^w$*.*

We require $(\mathsf{TrapSetup}, \mathsf{Extract})$ *to satisfy the following two properties:*

- ***CRS indistinguishability:*** *Define the CRS indistinguishability game for the BARG between an adversary* $\mathcal{A}$ *and a challenger as follows:*

    1. $\mathcal{A}$ *declares an index* $i^* \in [N]$ *and gives* i^* *to the challenger.*
    2. *The challenger samples* $b \xleftarrow{\$} \{0,1\}$*. If* $b = 0$*, the challenger gives* $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N)$ *to* $\mathcal{A}$*. If* $b = 1$*, the challenger samples* $(\mathsf{crs}^*, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapSetup}(1^\lambda, 1^N, i^*)$ *and gives* crs^* *to* $\mathcal{A}$*.*
    3. *At the end of the game,* $\mathcal{A}$ *outputs a bit* $b' \in \{0,1\}$*.*

    A BARG Π_{BARG} *is CRS indistinguishable if for all PPT adversaries* $\mathcal{A}$*, it holds that* $\Pr[b = b'] = 1/2 + \mathsf{negl}(\lambda)$ *in the above game.*

- ***Somewhere extractable in trapdoor mode:*** *Define the somewhere extractable security game between an adversary* $\mathcal{A}$ *and a challenger as follows:*

    1. $\mathcal{A}$ *declares an index* $i^* \in [N]$ *and gives* i^* *to the challenger.*
    2. *The challenger samples* $(\mathsf{crs}^*, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapSetup}(1^\lambda, 1^N, i^*)$ *and gives* crs^* *to* $\mathcal{A}$*.*
    3. $\mathcal{A}$ *outputs statements* $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in (\{0,1\}^x)^N$ *and a proof* π*. Let* $\mathbf{w}^* \leftarrow \mathsf{Extract}(\mathsf{td}, \vec{\mathbf{x}}, \pi)$*.*
    4. *The output of the game is* $b = 1$ *if*

    $$\mathsf{Verify}(\mathsf{crs}^*, \vec{\mathbf{x}}, \pi) = 1 \text{ and } (\mathbf{x}_{i^*}, \mathbf{w}^*) \notin R_{(\mathbf{A},\mathbf{B},\mathbf{C})}.$$

    Otherwise, the output is $b = 0$*.*

    A BARG Π_{BARG} *is somewhere extractable in trapdoor mode if for all adversaries* $\mathcal{A}$ *and all polynomials* $S = S(\lambda)$*,* $N = N(\lambda)$*, it holds that* $\Pr[b = 1] = \mathsf{negl}(\lambda)$ *in the above game.*

Definition 2.12 (Succinctness). *A BARG* Π_{BARG} *is succinct if there exists a fixed polynomial* $\mathsf{poly}(\cdot,\cdot,\cdot)$ *such that for all* $\lambda, S, N \in \mathbb{N}$ *and all* crs *in the support of* $\mathsf{Setup}(1^\lambda, 1^N)$*, the size of proof* π *satisfies* $|\pi| \le \mathsf{poly}(\lambda, \log N, \log S)$[1]*.*

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

2.3 Interactive Arguments for NP

Here, we provide the definition of multi-round public-coin interactive arguments for NP in the common reference string model. An overview is depicted in Figure 1. We begin with a formal definition. Let

$$
\begin{array}{lcl}
\textbf{Prover: } \mathsf{P}(\mathsf{crs}, \mathbf{x}, \mathbf{w}) & & \textbf{Verifier: } \mathsf{V}(\mathsf{crs}, \mathbf{x}) \\
\hline
(\mathsf{st}_1, \mathsf{res}_1) \overset{\$}{\leftarrow} \mathsf{P}_1(\mathsf{crs}, \mathsf{ch}_0 = \bot, \mathsf{st}_0 = (\mathbf{x}, \mathbf{w})) & \xrightarrow{\qquad \mathsf{res}_1 \qquad} & \\
 & \xleftarrow{\qquad \mathsf{ch}_1 \qquad} & \mathsf{ch}_1 \overset{\$}{\leftarrow} \mathcal{C}_1 \\
 & \vdots & \\
(\mathsf{st}_\mu, \mathsf{res}_\mu) \overset{\$}{\leftarrow} \mathsf{P}_\mu(\mathsf{crs}, \mathsf{ch}_{\mu-1}, \mathsf{st}_{\mu-1}) & \xrightarrow{\qquad \mathsf{res}_\mu \qquad} & \\
 & \xleftarrow{\qquad \mathsf{ch}_\mu \qquad} & \mathsf{ch}_\mu \overset{\$}{\leftarrow} \mathcal{C}_\mu \\
(\mathsf{st}_{\mu+1}, \mathsf{res}_{\mu+1}) \overset{\$}{\leftarrow} \mathsf{P}_\mu(\mathsf{crs}, \mathsf{ch}_\mu, \mathsf{st}_\mu) & \xrightarrow{\qquad \mathsf{res}_{\mu+1} \qquad} & \\
 & & b \overset{\$}{\leftarrow} \mathsf{V}(\mathsf{crs}, \mathbf{x}, \vec{\mathsf{ch}}, \vec{\mathsf{res}})
\end{array}
$$

Figure 1: Multi-round public-coin interactive argument in the CRS model

$R \subseteq \{0,1\}^x \times \{0,1\}^w$ be a binary relation. We call $(\mathbf{x}, \mathbf{w}) \in R$ a statement-witness pair, i.e., $\mathbf{x} \in \{0,1\}^x$ is the statement and $\mathbf{w} \in \{0,1\}^w$ is a witness for $\mathbf{x}$. Let $\mathsf{Out}_\mathsf{V}(\mathsf{P}(a), \mathsf{V}(b))$ be the random variable corresponding to the output of V upon execution of the protocol between P with input a and V with input b.

Definition 2.13 (Multi-Round Public-Coin Interactive Arguments for NP in the Common Reference String Model). *Let R be an NP relation. A $(2\mu+1)$-round public-coin interactive argument (PCIA) for R in the common reference string (CRS) model is a tuple of $\mu+3$ PPT algorithms $\Pi_\mathsf{PCIA} = (\mathsf{Setup}, \mathsf{P} = (\mathsf{P}_1, \dots, \mathsf{P}_{\mu+1}), \mathsf{V})$ and a family of the challenge spaces $\{\mathcal{C}_i\}_{i \in [\mu]}$ with the following properties:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$*: On input of the security parameter $\lambda \in \mathbb{N}$, the setup algorithm outputs a common reference string* crs.
- $\mathsf{P}_i(\mathsf{crs}, \mathsf{ch}_{i-1}, \mathsf{st}_{i-1}) \to (\mathsf{res}_i, \mathsf{st}_i)$*: On input of the common reference string* crs*, a challenge* $\mathsf{ch}_{i-1} \in \mathcal{C}_{i-1}$*, and a state* st_{i-1}*, the prove algorithm outputs a response* res_i *and a new state* st_i*, where* $\mathsf{ch}_0 = \bot$, $\mathsf{st}_0 = (\mathbf{x}, \mathbf{w})$*, $\mathbf{x}$ is a statement, and $\mathbf{w}$ is a witness.*
- $\mathsf{V}(\mathsf{crs}, \mathbf{x}, \vec{\mathsf{ch}}, \vec{\mathsf{res}}) \to b$*: On input of the common reference string* crs*, a statements $\mathbf{x}$, challenges* $\vec{\mathsf{ch}} = (\mathsf{ch}_1, \dots, \mathsf{ch}_\mu)$*, and responses* $\vec{\mathsf{res}} = (\mathsf{res}_1, \dots, \mathsf{res}_\mu, \mathsf{res}_{\mu+1})$*, the verification algorithm outputs a bit $b \in \{0,1\}$.*

Definition 2.14 (Completeness). *A PCIA Π_PCIA for R is complete if for all $\lambda \in \mathbb{N}$ and all statement-witness pairs $(\mathbf{x}, \mathbf{w}) \in R$, it holds that*

$$\Pr[\mathsf{Out}_\mathsf{V}(\mathsf{P}(\mathsf{crs}, \mathbf{x}, \mathbf{w}), \mathsf{V}(\mathsf{crs}, \mathbf{x})) = 1] = 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{crs} \overset{\$}{\leftarrow} \mathsf{Setup}(1^\lambda)$ *and the probability is taken over the randomness of all algorithms.*

Definition 2.15 (Soundness). *A PCIA Π_PCIA for R is sound if for all $\lambda \in \mathbb{N}$, all statements $\mathbf{x}$ such that $(\mathbf{x}, \cdot) \notin R$, and all PPT adversaries* P^**, it holds that*

$$\Pr[\mathsf{Out}_\mathsf{V}(\mathsf{P}^*(\mathsf{crs}, \mathbf{x}), \mathsf{V}(\mathsf{crs}, \mathbf{x})) = 1] = \mathsf{negl}(\lambda),$$

where $\mathsf{crs} \overset{\$}{\leftarrow} \mathsf{Setup}(1^\lambda)$ *and the probability is taken over the randomness of all algorithms.*

2.4 Lattice Preliminaries

Gaussian Measures. For a positive real σ, let $\mathcal{D}_\sigma^d$ denote the discrete Gaussian distribution over $\mathbb{Z}^d$.

Assumptions. Here, we define well-known lattice assumptions.

Definition 2.16 (MSIS Assumption). *Let n, m, q be positive integers and β be a positive real. The modulo short integer solution (MSIS) assumption holds if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all $\lambda \in \mathbb{N}$:*

$$\Pr[\mathbf{A}\mathbf{u} = \mathbf{0} \wedge \|\mathbf{u}\| \leq \beta | \mathbf{u} \overset{\$}{\leftarrow} \mathcal{A}(1^\lambda, \mathbf{A})] = \mathsf{negl}(\lambda),$$

where $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{R}_q^{m \times n}$.

Definition 2.17 (MLWE Assumption). *Let n, m, q be positive integers and χ be a distribution over $\mathcal{R}_q$. The modulo learning with errors (MLWE) assumption holds if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all $\lambda \in \mathbb{N}$:*

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

where $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{R}_q^{m \times n}$, $\mathbf{s} \overset{\$}{\leftarrow} \chi^n$, $\mathbf{e} \overset{\$}{\leftarrow} \chi^m$, and $\mathbf{b} \overset{\$}{\leftarrow} \mathcal{R}_q^m$.

3 Our BARG

In this section, we present our BARG for NP. To describe this, we first summarize the results in [BS22].

3.1 Principal Relations

Here, we recall the definition of the principal relation, introduced in [BS22]. The 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 $\mathcal{F} \coloneqq \{f^{(1)}, \ldots, f^{(K)}\}$ of quadratic dot product functions $f : (\mathcal{R}_q^n)^r \to \mathcal{R}_q$ of the form

$$f(\mathbf{s}_1, \ldots, \mathbf{s}_r) = \sum_{i=1}^{r}\sum_{j=1}^{r} \phi_{i,j} \cdot \mathbf{s}_i^\top \mathbf{s}_j + \sum_{i=1}^{r} \boldsymbol{\psi}_i^\top \mathbf{s}_i - \nu,$$

where $\phi_{i,j}, \nu \in \mathcal{R}_q$ and $\boldsymbol{\psi}_i \in \mathcal{R}_q^n$. Without loss of generality, we assume $\phi_{i,j} = \phi_{j,i}$. Let $\mathcal{F} \coloneqq \{f^{(1)}, \ldots, f^{(K)}\}$ and $\hat{\mathcal{F}} \coloneqq \{\hat{f}^{(1)}, \ldots, \hat{f}^{(L)}\}$ be two families of quadratic dot product functions. Now, a witness consists of r vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r \in \mathcal{R}_q^n$ such that

$$f(\mathbf{s}_1, \ldots, \mathbf{s}_r) = 0 \text{ for all } f \in \mathcal{F}, \qquad \mathsf{const}(\hat{f}(\mathbf{s}_1, \ldots, \mathbf{s}_r)) = 0 \text{ for all } \hat{f} \in \hat{\mathcal{F}}, \qquad \sum_{i=1}^{r} \|\mathbf{s}_i\|_2^2 \leq \beta^2.$$

That is, a principal relation $\mathcal{R}_{\mathsf{pr}}$ is defined as

$$\mathcal{R}_{\mathsf{pr}} \coloneqq \left\{ ((\mathcal{F}, \hat{\mathcal{F}}, \beta), (\mathbf{s}_1, \ldots, \mathbf{s}_r)) \middle| f(\mathbf{s}_1, \ldots, \mathbf{s}_r) = \mathbf{0}\ \forall f \in \mathcal{F},\ \mathsf{const}(\hat{f}(\mathbf{s}_1, \ldots, \mathbf{s}_r)) = 0\ \forall \hat{f} \in \hat{\mathcal{F}},\ \sum_{i=1}^{r} \|\mathbf{s}_i\|_2^2 \leq \beta^2 \right\}.$$

Recently, Beullens and Seilar [BS22] proposed highly efficient multi-round PCIA for the principal relations. In the following lemma, we summarize their result.

Lemma 3.1 ([BS22]). *There exists a multi-round PCIA Π_{pr} for a principal relation in the CRS model with the following properties:*

- Π_{pr} *is complete.*
- *Assuming that the MSIS assumption holds,* Π_{pr} *is sound.*[2]
- *The total size of responses is* $O(\log n + \log r) \cdot \mathsf{poly}(\lambda)$.

3.2 Construction: Our BARG for R1CS

Let $\Pi_{\mathsf{pr}} = (\mathsf{Setup}_{\mathsf{pr}}, \mathsf{P} = (\mathsf{P}_1, \ldots, \mathsf{P}_{\mu+1}), \mathsf{V})$ be a PCIA for a principal relation and $(\mathbf{A}, \mathbf{B}, \mathbf{C}) \in \left(\{0,1\}^{S \times (x+w)}\right)^3$ be an R1CS instance. We consider random oracles $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\lambda \times 3NS}$ and $\mathsf{H}_i : \{0,1\}^* \to \mathcal{C}_i$ that map into the respective challenge spaces. Our BARG $\Pi_{\mathsf{BARG}} = (\mathsf{Setup}_{\mathsf{BARG}}, \mathsf{Prove}, \mathsf{Verify})$ for an R1CS instance $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is described as follows:

- $\mathsf{Setup}_{\mathsf{BARG}}(1^\lambda, 1^N) \to \mathsf{crs}_{\mathsf{BARG}}$:
  1. Sample $\mathbf{U}_a, \mathbf{U}_b, \mathbf{U}_c \xleftarrow{\$} \mathcal{R}_q^{n \times NS}$ and $\mathbf{U}_y \xleftarrow{\$} \mathcal{R}_q^{n \times N(x+w)}$ and set $\mathsf{ck} := (\mathbf{U}_a, \mathbf{U}_b, \mathbf{U}_c, \mathbf{U}_y)$.
  2. Sample $\mathsf{crs}_{\mathsf{pr}} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{pr}}(1^\lambda)$.
  3. Output $\mathsf{crs}_{\mathsf{BARG}} := (\mathsf{ck}, \mathsf{crs}_{\mathsf{pr}})$.
- $\mathsf{Prove}(\mathsf{crs}_{\mathsf{BARG}}, \vec{\mathbf{x}}, \vec{\mathbf{w}}) \to \pi$:
  1. Set
  $$\mathbf{y} := \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{w}_1 \\ \vdots \\ \mathbf{x}_N \\ \mathbf{w}_N \end{pmatrix} \in \{0,1\}^{N(x+y)}, \qquad \begin{aligned} \mathbf{a} &:= (\mathbf{I}_N \otimes \mathbf{A})\mathbf{y} \bmod 2 \in \{0,1\}^{NS}, \\ \mathbf{b} &:= (\mathbf{I}_N \otimes \mathbf{B})\mathbf{y} \bmod 2 \in \{0,1\}^{NS}, \\ \mathbf{c} &:= (\mathbf{I}_N \otimes \mathbf{C})\mathbf{y} \bmod 2 \in \{0,1\}^{NS}. \end{aligned}$$
  2. Set
  $$\mathbf{u} := \mathbf{U}_a\mathbf{a} + \mathbf{U}_b\mathbf{b} + \mathbf{U}_c\mathbf{c} + \mathbf{U}_y\mathbf{y} \in \mathcal{R}_q^n, \qquad \mathbf{P} := \mathsf{H}(\vec{\mathbf{x}}, \mathbf{u}) \in \{0,1\}^{\lambda \times 3NS}$$
  $$\mathbf{D} := \mathbf{P} \cdot \begin{pmatrix} \mathbf{I}_N \otimes \mathbf{A} \\ \mathbf{I}_N \otimes \mathbf{B} \\ \mathbf{I}_N \otimes \mathbf{C} \end{pmatrix} \bmod 2 \in \{0,1\}^{\lambda \times 3N(x+w)}, \qquad \mathbf{d} = \mathbf{P} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{pmatrix} - \mathbf{D}\mathbf{y} \bmod 2 \in \{0,1\}^{\lambda}.$$
  3. Set $\tilde{\mathbf{a}} := \sigma_{-1}(\mathbf{a})$, $\tilde{\mathbf{b}} := \sigma_{-1}(\mathbf{b})$, $\tilde{\mathbf{c}} := \sigma_{-1}(\mathbf{c})$, and $\tilde{\mathbf{y}} := \sigma_{-1}(\mathbf{y})$.
  4. $\mathbf{s} := (\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{y}, \tilde{\mathbf{a}}, \tilde{\mathbf{b}}, \tilde{\mathbf{c}}, \tilde{\mathbf{y}})$ as witness vectors, we define the statement for the principal relations as follows:
  $$\mathcal{F}_{(\mathsf{ck},\mathbf{u})} := \{\mathbf{U}_a\mathbf{a} + \mathbf{U}_b\mathbf{b} + \mathbf{U}_c\mathbf{c} + \mathbf{U}_y\mathbf{y} - \mathbf{u}\}, \tag{1}$$
  $$\hat{\mathcal{F}}_{(\mathbf{P},\mathbf{D},\mathbf{d})} := \left\{ \begin{array}{l} \tilde{\mathbf{a}} := \sigma_{-1}(\mathbf{a}),\ \tilde{\mathbf{b}} := \sigma_{-1}(\mathbf{b}), \\ \tilde{\mathbf{c}} := \sigma_{-1}(\mathbf{c}),\ \tilde{\mathbf{y}} := \sigma_{-1}(\mathbf{y}), \\ \mathbf{a}^\top(\tilde{\mathbf{a}} - \mathbf{1}_{NS}),\ \mathbf{b}^\top(\tilde{\mathbf{b}} - \mathbf{1}_{NS}), \\ \mathbf{c}^\top(\tilde{\mathbf{c}} - \mathbf{1}_{NS}),\ \mathbf{y}^\top(\tilde{\mathbf{y}} - \mathbf{1}_{N(x+w)}), \\ (\mathbf{a} + \mathbf{b} - 2\mathbf{c})^\top(\tilde{\mathbf{a}} + \tilde{\mathbf{b}} - 2\tilde{\mathbf{c}} - \mathbf{1}_{NS}) \end{array} \right\} \cup \left\{ \mathbf{P} \cdot \begin{pmatrix} \mathbf{I}_N \otimes \mathbf{A} \\ \mathbf{I}_N \otimes \mathbf{B} \\ \mathbf{I}_N \otimes \mathbf{C} \end{pmatrix} - \mathbf{D}\mathbf{y} \right\}. \tag{2}$$
  5. To prove $((\mathcal{F}_{(\mathsf{ck},\mathbf{u})}, \hat{\mathcal{F}}_{(\mathbf{P},\mathbf{D},\mathbf{d})}, \sqrt{q}), \mathbf{s}) \in R_{\mathsf{pr}}$, run $(\mathsf{st}_i, \mathsf{res}_i) \xleftarrow{\$} \mathsf{P}_i(\mathsf{crs}, \mathsf{st}_{i-1}, \mathsf{ch}_{i-1})$ and set $\mathsf{ch}_i := \mathsf{H}_i(\vec{\mathbf{x}}, \mathbf{u}, \mathbf{d}, \mathsf{res}_1, \ldots, \mathsf{res}_i)$ for $i \in [\mu]$, and $(\mathsf{st}_{\mu+1}, \mathsf{res}_{\mu+1}) \xleftarrow{\$} \mathsf{P}_{\mu+1}(\mathsf{crs}, \mathsf{st}_\mu, \mathsf{ch}_\mu)$.
  6. Output $\pi := (\mathbf{u}, \mathbf{d}, \pi_{\mathsf{pr}} = (\mathsf{res}_1, \ldots, \mathsf{res}_{\mu+1}))$.

[2]The protocol in [BS22] satisfies knowledge soundness, a stronger notion than the above soundness.

- $\mathsf{Verify}(\mathsf{crs}_{\mathsf{BARG}}, \vec{\mathbf{x}}, \pi) \to b$:
  1. Parse $\pi =: (\mathbf{u}, \mathbf{d}, \pi_{\mathsf{pr}} = (\mathsf{res}_1, \ldots, \mathsf{res}_{\mu+1}))$.
  2. If $\mathbf{d}$ has an odd element, then output 0.
  3. Set
  $$\mathbf{P} := \mathsf{H}(\vec{\mathbf{x}}, \mathbf{u}), \qquad \mathbf{D} := \mathbf{P} \cdot \begin{pmatrix} \mathbf{I}_N \otimes \mathbf{A} \\ \mathbf{I}_N \otimes \mathbf{B} \\ \mathbf{I}_N \otimes \mathbf{C} \end{pmatrix} \bmod 2 \in \{0,1\}^{\lambda \times 3N(x+w)}.$$
  4. Define $\mathcal{F}_{(\mathsf{ck},\mathbf{u})}$ and $\hat{\mathcal{F}}_{(\mathbf{P},\mathbf{D},\mathbf{d})}$ as Eqs. (1) and (2), respectively.
  5. Compute $\mathsf{ch}_i := \mathsf{H}_i(\vec{\mathbf{x}}, \mathsf{res}_1, \ldots, \mathsf{res}_i)$ for $i \in [\mu]$ and set $\vec{\mathsf{ch}} := (\mathsf{ch}_1, \ldots, \mathsf{ch}_\mu)$
  6. To verify $((\mathcal{F}_{(\mathsf{ck},\mathbf{u})}, \hat{\mathcal{F}}_{(\mathbf{P},\mathbf{D},\mathbf{d})}, \sqrt{q}), \mathbf{s}) \in R_{\mathsf{pr}}$, compute $b \leftarrow \mathsf{V}(\mathsf{crs}_{\mathsf{pr}}, \vec{\mathbf{x}}, \vec{\mathsf{ch}}, \pi_{\mathsf{pr}})$ and output b.

3.3 Security Proof

Theorem 3.2. *The above Π_{BARG} has the following properties:*

- *Π_{BARG} is complete.*
- *Assuming that the MLWE assumption holds and Π_{pr} is sound, Π_{BARG} is somewhere argument of knowledge in the random oracle model.*

Proof Sketch. The above Π_{BARG} is almost the same as the non-interactive variant (via Fiat-Shamir transformation) of the protocol for binary R1CS, presented in [BS22, Section 6]. Thus, the completeness of Π_{BARG} follows directly from the completeness their protocol, and we omit the proof.

Then, we provide the proof sketch that Π_{BARG} is somewhere argument of knowledge. To do this, we start by defining the trapdoor setup and extraction algorithms:

- $\mathsf{TrapSetup}(1^\lambda, 1^N, i^*) \to (\mathsf{crs}^*_{\mathsf{BARG}}, \mathsf{td})$:
  1. Sample $\mathbf{U}'_a, \mathbf{U}'_b, \mathbf{U}'_c \xleftarrow{\$} \mathcal{R}_q^{(n-1)\times NS}$, $\mathbf{U}'_{x,1}, \ldots, \mathbf{U}'_{x,N} \xleftarrow{\$} \mathcal{R}_q^{(n-1)\times x}$, and $\mathbf{U}'_{w,1}, \ldots, \mathbf{U}'_{w,N} \xleftarrow{\$} \mathcal{R}_q^{(n-1)\times w}$.
  2. Sample $\mathbf{z} \xleftarrow{\$} \mathcal{R}_q^{n-1}$, $\mathbf{e}_a, \mathbf{e}_b, \mathbf{e}_c \xleftarrow{\$} \mathcal{D}_\sigma^{NS}$, $\mathbf{e}_{x,1}, \ldots, \mathbf{e}_{x,N} \xleftarrow{\$} \mathcal{D}_\sigma^{x}$, and $\mathbf{e}_{w,1}, \ldots, \mathbf{e}_{w,N} \xleftarrow{\$} \mathcal{D}_\sigma^{w}$.
  3. Set
  $$\mathbf{U}_a := \begin{pmatrix} \mathbf{U}'_a \\ \mathbf{z}^\top \mathbf{U}'_a + \mathbf{e}_a^\top \end{pmatrix} \in \mathcal{R}_q^{n\times NS}, \qquad \mathbf{U}_{x,i} := \begin{pmatrix} \mathbf{U}'_{x,i} \\ \mathbf{z}^\top \mathbf{U}'_{x,i} + \mathbf{e}_{x,i}^\top \end{pmatrix} \in \mathcal{R}_q^{n\times x} \text{ for } i \in [N],$$
  $$\mathbf{U}_b := \begin{pmatrix} \mathbf{U}'_b \\ \mathbf{z}^\top \mathbf{U}'_b + \mathbf{e}_b^\top \end{pmatrix} \in \mathcal{R}_q^{n\times NS}, \qquad \mathbf{U}_{w,i} := \begin{pmatrix} \mathbf{U}'_{w,i} \\ \mathbf{z}^\top \mathbf{U}'_{w,i^*} + \mathbf{e}_{w,i}^\top \end{pmatrix} \in \mathcal{R}_q^{n\times w} \text{ for } i \in [N]\setminus\{i^*\},$$
  $$\mathbf{U}_c := \begin{pmatrix} \mathbf{U}'_c \\ \mathbf{z}^\top \mathbf{U}'_c + \mathbf{e}_c^\top \end{pmatrix} \in \mathcal{R}_q^{n\times NS}, \qquad \mathbf{U}_{w,i^*} := \begin{pmatrix} \mathbf{U}'_{w,i^*} \\ \mathbf{z}^\top \mathbf{U}'_{w,i^*} + \mathbf{e}_{w,i}^\top + t\cdot \mathbf{g}^\top \end{pmatrix} \in \mathcal{R}_q^{n\times w},$$
  where
  $$\mathbf{g}^\top := \begin{pmatrix} 2^0 X^0 & \cdots & 2^{\tau-1}X^0 & 2^0 X^1 & \cdots & 2^{\tau-1}X_1 & \cdots & 2^0 X^{d'} & \cdots & 2^{\ell-1}X^{d'} \end{pmatrix} \in \mathcal{R}_q^{1\times w},$$
  $\tau := \lceil w/d \rceil$, $d' := \lfloor w/\tau \rfloor$ (note that $d' \le d$), and $\ell := w \bmod \tau$.
  4. Set
  $$\mathbf{U}_y := \begin{pmatrix} \mathbf{U}_{x,1} \\ \mathbf{U}_{w,1} \\ \vdots \\ \mathbf{U}_{x,N} \\ \mathbf{U}_{w,N} \end{pmatrix} \in \mathcal{R}_q^{n\times N(x+w)}$$
  and $\mathsf{ck} := (\mathbf{U}_a, \mathbf{U}_b, \mathbf{U}_c, \mathbf{U}_y)$.

5. Sample $\mathsf{crs_{pr}} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$.
6. Output $\mathsf{crs}^*_{\mathsf{BARG}} := (\mathsf{ck}, \mathsf{crs_{pr}})$ and $\mathsf{td} := \mathbf{z}$.

- $\mathsf{Extract}(\mathsf{td}, \vec{\mathbf{x}}, \pi) \to \mathbf{w}^*$:
  1. Parse $\pi =: (\mathbf{u}, \mathbf{d}, \pi_{\mathsf{pr}} = (\mathsf{res}_1, \dots, \mathsf{res}_{\mu+1}))$.
  2. Set $C' := \begin{pmatrix}\mathbf{z}^\top & -1\end{pmatrix} \mathbf{u} \in \mathcal{R}_q$.
  3. Set $\bar{t} := \lfloor q/t \rfloor$ and $C'' := \mathsf{Rnd}_{\bar{t}}(C')$.
  4. Set $\omega := (\bar{t})^{-1} \cdot C'' \in \mathcal{R}$ and output $\mathbf{w}^* := \mathsf{BD}_{\tau,w}(\omega) \in \{0,1\}^w$.

  Here, the function $\mathsf{Rnd}_{\bar{t}}$ rounds each coefficient of C' to the nearest integer multiple of $\bar{t}$, and the function $\mathsf{BD}_{\tau,w}$ performs bit decomposition of the coefficients of the $\omega = \omega_0 + \omega_1 X + \cdots + \omega_{d-1} X^{d-1}$ and returns the resulting binary vector $\mathbf{w}^* = (\mathbf{w}_0, \dots, \mathbf{w}_{w-1}) \in \{0,1\}^w$. Namely, for $j \in 0, 1, \dots, w-1$, it sets $\mathbf{w}_j$ to the k-th bit of the coefficient $\omega_{\lfloor j/\tau \rfloor}$, where $k := j \bmod \tau$.

Roughly speaking, ck outputted by $\mathsf{TrapSetup}$ are a trapdoored commitment key produced by CAddTd in [ESZ22, Section 5.3], and $\mathsf{Extract}$ algorithm is the almost same as the CDecGR algorithm in [ESZ22, Section 5.3].

Clearly, the distribution of ck sampled by $\mathsf{TrapSetup}(1^\lambda, 1^N, i^*)$ is indistinguishable from the distribution of ck sampled by $\mathsf{Setup}(1^\lambda, 1^N)$. This implies Π_{BARG} is CRS indistinguishable. Next, we briefly describe that if Π_{pr} is sound then Π_{BARG} is somewhere extractable in the trapdoor mode. Let $\mathcal{A}$ be an adversary in the somewhere extractable security game for Π_{BARG} and $\vec{\mathbf{x}} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ and $\pi = (\mathbf{u}, \mathbf{d}, \pi_{\mathsf{pr}} = (\mathsf{res}_1, \dots, \mathsf{res}_{\mu+1}))$ be statements and a proof outputted by $\mathcal{A}$. From the construction, if $\mathsf{Verify}(\mathsf{crs}^*_{\mathsf{BARG}}, \vec{\pi}) = 1$ then $\mathsf{V}(\mathsf{crs_{pr}}, \vec{\mathbf{x}}, \vec{\mathsf{ch}}, \pi_{\mathsf{pr}}) = 1$, where $(\mathsf{crs}^*_{\mathsf{BARG}}, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapSetup}(1^\lambda, 1^N, i^*)$, i^* is the index declared by $\mathcal{A}$, and $\vec{\mathsf{ch}} = (\mathsf{ch}_1, \dots, \mathsf{ch}_\mu)$ is produced as in the above construction. By the soundness of Π_{pr} (and the reduction from R1CS to principal relation, described in [BS22]), this implies there exists $\vec{\mathbf{w}} = (\mathbf{w}_1, \dots, \mathbf{w}_N) \in (\{0,1\}^w)^N$ such that

$$\mathbf{u} = \begin{pmatrix}\mathbf{U}_a & \mathbf{U}_b & \mathbf{U}_c & \mathbf{U}_y\end{pmatrix} \begin{pmatrix}\mathbf{I}_N \otimes \mathbf{A} \\ \mathbf{I}_N \otimes \mathbf{B} \\ \mathbf{I}_N \otimes \mathbf{C} \\ \mathbf{I}_{N(x+w)}\end{pmatrix} \begin{pmatrix}\mathbf{x}_1 \\ \mathbf{w}_1 \\ \vdots \\ \mathbf{x}_N \\ \mathbf{w}_N\end{pmatrix},$$

$$\mathbf{A}\begin{pmatrix}\mathbf{x}_i \\ \mathbf{w}_i\end{pmatrix} \circ \mathbf{B}\begin{pmatrix}\mathbf{x}_i \\ \mathbf{w}_i\end{pmatrix} = \mathbf{C}\begin{pmatrix}\mathbf{x}_i \\ \mathbf{w}_i\end{pmatrix} \quad \text{for } i \in [N].$$

Now, since $\mathsf{ck} = (\mathbf{U}_a, \mathbf{U}_b, \mathbf{U}_c, \mathbf{U}_y)$ is produced from $\mathsf{TrapSetup}$, we have

$$\begin{pmatrix}\mathbf{z}^\top & -1\end{pmatrix} \mathbf{u} \approx t \cdot \mathbf{g}^\top \mathbf{w}_{i^*}.$$

Then, from the construction of $\mathsf{Extract}$ (and the functionality of CDecGR in [ESZ22]), the output of $\mathsf{Extract}(\mathsf{td}, \vec{\mathbf{x}}, \pi)$ is $\mathbf{w}_{i^*}$ as an extracted witness $\mathbf{w}^*$. Therefore, Π_{BARG} is somewhere extractable in the trapdoor mode.

□

If we instantiate Π_{pr} with the protocol of Beullens and Seilar [BS22, Section 5.2], we immediately obtain the following corollary due to Lemma 3.1.

Corollary 3.3. *There exists a BARG Π_{BARG} for R1CS with the following properties:*

- Π_{BARG} *is complete.*
- *Assuming that the MSIS and MLWE assumptions hold, Π_{BARG} is somewhere argument of knowledge in the random oracle model.*
- Π_{BARG} *is succinct. More precisely, the proof size of Π_{BARG} is $O(\log N + \log S) \cdot \mathsf{poly}(\lambda)$*

4 Signature Aggregation from BARG

In this section, we first recall that how to construct an aggregate signature scheme from a BARG for NP and a standard signature scheme, by Waters and Wu [WW22]. Then, we show

Construction 4.1 (Aggregate Signature from BARG [WW22, Construction 7.3]). Let $\Pi_{\mathsf{Sig}} = (\mathsf{KGen}_{\mathsf{Sig}}, \mathsf{Sign}_{\mathsf{Sig}}, \mathsf{Verify}_{\mathsf{Sig}})$ be a digital signature scheme, and $\Pi_{\mathsf{BARG}} = (\mathsf{Setup}_{\mathsf{BARG}}, \mathsf{Prove}_{\mathsf{BARG}}, \mathsf{Verify}_{\mathsf{BARG}})$ be a BARG for NP. We can construct a bounded aggregate signature scheme $\Pi_{\mathsf{AggSig}} = (\mathsf{Setup}_{\mathsf{AggSig}}, \mathsf{KGen}_{\mathsf{AggSig}}, \mathsf{Sign}_{\mathsf{AggSig}}, \mathsf{Verify}_{\mathsf{AggSig}}, \mathsf{Aggregate}_{\mathsf{AggSig}}, \mathsf{AggVerify}_{\mathsf{AggSig}})$ as follows:

- $\mathsf{Setup}_{\mathsf{AggSig}}(1^\lambda, 1^N) \to \mathsf{pp}$: On input of the security parameter λ and the aggregation bound N. Convert $\mathsf{Verify}_{\mathsf{Sig}}$ (as a circuit) to an R1CS $(\mathbf{A}, \mathbf{B}, \mathbf{C}) \in (\{0,1\}^{S \times (x+w)})^3$. Sample $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{BARG}}(1^\lambda, 1^N)$ and output a public parameter $\mathsf{pp} = (1^\lambda, \mathsf{crs})$.
- $\mathsf{KGen}_{\mathsf{AggSig}}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$: On input of the public parameter $\mathsf{pp} = (1^\lambda, \mathsf{crs})$, output $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{KGen}_{\mathsf{Sig}}(1^\lambda)$.
- $\mathsf{Sign}_{\mathsf{AggSig}}(\mathsf{pp}, \mathsf{sk}, \mathsf{M}) \to \sigma$: On input of the public parameter $\mathsf{pp} = (1^\lambda, \mathsf{crs})$, the signing key sk, and the message $\mathsf{M} \in \mathcal{M}$, output $\sigma \xleftarrow{\$} \mathsf{Sign}_{\mathsf{Sig}}(\mathsf{sk}, \mathsf{M})$.
- $\mathsf{Verify}_{\mathsf{AggSig}}(\mathsf{pp}, \mathsf{vk}, \mathsf{M}, \sigma) \to b$: On input of the public parameter $\mathsf{pp} = (1^\lambda, \mathsf{crs})$, the verification key vk, the message $\mathsf{M} \in \mathcal{M}$, and the signature σ, output $\mathsf{Verify}_{\mathsf{Sig}}(\mathsf{vk}, \mathsf{M}, \sigma)$.
- $\mathsf{Aggregate}_{\mathsf{AggSig}}(\mathsf{pp}, \{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i \in [T]}) \to \sigma_{\mathsf{agg}}$: On input of the public parameter $\mathsf{pp} = (1^\lambda, \mathsf{crs})$ and a collection of tuples $\{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i \in [T]}$. Convert $\{(\mathsf{vk}_i, \mathsf{M}_i)\}_{i \in [T]}$ to R1CS statements $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_T) \in (\{0,1\}^x)^T$ and $\{\sigma_i\}_{i \in [T]}$ to R1CS witnesses $\vec{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_T) \in (\{0,1\}^w)^T$. The aggregation algorithm computes
$$\pi \xleftarrow{\$} \mathsf{Prove}_{\mathsf{BARG}}(\mathsf{crs}, \vec{\mathbf{x}}, \vec{\mathbf{w}})$$
and output $\sigma_{\mathsf{agg}} := \pi$ as an aggregated signature.
- $\mathsf{AggVerify}_{\mathsf{AggSig}}(\mathsf{pp}, (\mathsf{vk}_1, \ldots, \mathsf{vk}_T), (\mathsf{M}_1, \ldots, \mathsf{M}_T), \sigma_{\mathsf{agg}}) \to b$: On input of the public parameter $\mathsf{pp} = (1^\lambda, \mathsf{crs})$, verification keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_T)$, messages $(\mathsf{M}_1, \ldots, \mathsf{M}_T) \in \mathcal{M}$, and an aggregated signature σ_{agg}. Convert $\{(\mathsf{vk}_i, \mathsf{M}_i)\}_{i \in [T]}$ to R1CS statements $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_T) \in (\{0,1\}^x)^T$. The aggregate verification algorithm outputs
$$\mathsf{Verify}_{\mathsf{BARG}}(\mathsf{crs}, \vec{\mathbf{x}}, \sigma_{\mathsf{agg}}).$$

Lemma 4.2 (Aggregate Signature from Batch Argument [WW22, Thorems 7.4, 7.5, and 7.6]).

- *If Π_{Sig} is correct and Π_{BARG} is complete, then Π_{AggSig} is correct.*
- *If Π_{BARG} is a somewhere argument of knowledge and Π_{Sig} is unforgeable, then Π_{AggSig} is unforgeable.*
- *If Π_{BARG} is succinct, then Π_{AggSig} is efficient.*

If we instantiate Π_{Sig} with a modulo-lattice-based signature scheme (e.g., Dilithium [DKL+18]) and Π_{BARG} with our BARG in Section 3.2, we immediately obtain the result due to Corollary 3.3.

Corollary 4.3. *There exists an aggregate signature scheme Π_{AggSig} with the following properties:*

- *Π_{AggSig} is correct.*
- *Assuming that the MSIS and MLWE assumptions hold, Π_{AggSig} is unforgeable in the random oracle model.*

- Π_{AggSig} *is efficient. More precisely, the aggregate signature size of* Π_{AggSig} *is* $O(\log N) \cdot \mathsf{poly}(\lambda)$

Acknowledgment. This research was (in part) conducted under a construct 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.

References

[ACL$^+$22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In Yevgeniy Dodis and Thomas Shrimpton, editors, CRYPTO 2022, Part II, volume 13508 of LNCS, pages 102–132. Springer, Heidelberg, August 2022. (Cited on page 2.)

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, EUROCRYPT 2003, volume 2656 of LNCS, pages 416–432. Springer, Heidelberg, May 2003. (Cited on page 1, 3.)

[BGOY07] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, ACM CCS 2007, pages 276–285. ACM Press, October 2007. (Cited on page 1.)

[BGR12] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. Sequential aggregate signatures with lazy verification from trapdoor permutations - (extended abstract). In Xiaoyun Wang and Kazue Sako, editors, ASIACRYPT 2012, volume 7658 of LNCS, pages 644–662. Springer, Heidelberg, December 2012. (Cited on page 1.)

[BK20] Dan Boneh and Sam Kim. One-time and interactive aggregate signatures from lattices. preprint, 2020. (Cited on page 1, 2.)

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, ICALP 2007, volume 4596 of LNCS, pages 411–422. Springer, Heidelberg, July 2007. (Cited on page 1.)

[BR21] Katharina Boudgoust and Adeline Roux-Langlois. Compressed linear aggregate signatures based on module lattices. Cryptology ePrint Archive, Report 2021/263, 2021. `https://eprint.iacr.org/2021/263`. (Cited on page 1, 2.)

[BS22] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. Cryptology ePrint Archive, Report 2022/1341, 2022. `https://eprint.iacr.org/2022/1341`. (Cited on page 2, 7, 8, 9, 10.)

[BT23] Katharina Boudgoust and Akira Takahashi. Sequential half-aggregation of lattice-based signatures. Cryptology ePrint Archive, 2023. (Cited on page 1, 2.)

[CJJ22] Arka Rai Choudhuri, Abhihsek Jain, and Zhengzhong Jin. Snargs for $\mathcal{P}$ from lwe. In 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS), pages 68–79. IEEE, 2022. (Cited on page 2, 5.)

[DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-np and applications. In 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS), pages 1057–1068. IEEE, 2022. (Cited on page 2.)

[DHSS20] Yarkın Doröz, Jeffrey Hoffstein, Joseph H. Silverman, and Berk Sunar. 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, 2.)

[DKL$^+$18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 238–268, 2018. (Cited on page 11.)

[ESZ22] Muhammed F Esgin, Ron Steinfeld, and Raymond K Zhao. Efficient verifiable partially-decryptable commitments from lattices and applications. In IACR International Conference on Public-Key Cryptography, pages 317–348. Springer, 2022. (Cited on page 2, 10.)

[FLS12] Marc Fischlin, Anja Lehmann, and Dominique Schröder. History-free sequential aggregate signatures. In Ivan Visconti and Roberto De Prisco, editors, SCN 12, volume 7485 of LNCS, pages 113–130. Springer, Heidelberg, September 2012. (Cited on page 1.)

[GOR18] Craig Gentry, Adam O'Neill, and Leonid Reyzin. A unified framework for trapdoor-permutation-based sequential aggregate signatures. In Michel Abdalla and Ricardo Dahab, editors, PKC 2018, Part II, volume 10770 of LNCS, pages 34–57. Springer, Heidelberg, March 2018. (Cited on page 1.)

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, EUROCRYPT 2004, volume 3027 of LNCS, pages 74–90. Springer, Heidelberg, May 2004. (Cited on page 1.)

[LOS$^+$06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, EUROCRYPT 2006, volume 4004 of LNCS, pages 465–485. Springer, Heidelberg, May / June 2006. (Cited on page 1.)

[Nev08] Gregory Neven. Efficient sequential aggregate signed data. In Nigel P. Smart, editor, EUROCRYPT 2008, volume 4965 of LNCS, pages 52–69. Springer, Heidelberg, April 2008. (Cited on page 1.)

[WW22] Brent Waters and David J. Wu. Batch arguments for sfNP and more from standard bilinear group assumptions. In Yevgeniy Dodis and Thomas Shrimpton, editors, CRYPTO 2022, Part II, volume 13508 of LNCS, pages 433–463. Springer, Heidelberg, August 2022. (Cited on page 11.)