Compact Signature Aggregation from Module-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 module 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 $\sigma_1, \ldots, \sigma_N$ for different messages $\mathsf{M}_1, \ldots, \mathsf{M}_N$ 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 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. Table 1 contains a comparison with prior work on

Table 1: Comparison of lattice-based aggregate signature schemes. The column $|\hat{\sigma}|$ indicates the size of the 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	$\lvert\hat{\sigma}\rvert$	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

aggregate signature schemes. Our scheme is obtained by a general approach using a 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 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 module short integer solution (MSIS) and module 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 a simplified variant of the commitment scheme by Esgin, Steinfeld, and Zhao [ESZ22].

2 Preliminaries

Notations. For a 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 $\sigma_{-1}(X) = X^{-1}$. For $\mathbf{x} = (x_1, \ldots, x_n) \in \{0, 1\}^n$, we denote by $\mathsf{Lift}_n(\mathbf{x}) \in \mathcal{R}_2^{\lceil n/d \rceil}$ the vector of polynomials with $(x_1, \ldots, x_n)$ as coefficients. Let $\mathsf{negl}(\lambda)$ be a negligible function, i.e. a function dominated by $O(\lambda^{-1})$ for all $n > 0$.

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*

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

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

Definition 2.3 (Unforgeability). *For* $\lambda \in \mathbb{N}$ *and an adversary* $\mathcal{A}$*, define the signature unforgeability game between* $\mathcal{A}$ *and a challenger* $\mathcal{C}$ *as follows:*

1. $\mathcal{C}$ *samples* $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$ *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*,* $\mathcal{C}$ *replies with* $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, \mathsf{M})$*.*
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* $\lambda \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathcal{A},\Pi_{\mathsf{Sig}}}(\lambda) := \Pr[b = 1] = \mathsf{negl}(\lambda)$$

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* $T \leq N$ *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* $T \leq N$ *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*

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

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 $T \le N$ and $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_i, \mathsf{M}_i, \sigma_i) = 1$ for all $i \in [T]$,

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

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). *For $\lambda, N \in \mathbb{N}$ and an adversary $\mathcal{A}$, define the unforgeability game for the aggregate signature between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:*

1. *$\mathcal{C}$ 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, $\mathcal{C}$ 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 $(\mathsf{vk}_1, \ldots, \mathsf{vk}_T)$, a tuple of messages $(\mathsf{M}_1, \ldots, \mathsf{M}_T)$ with $T \le N$, 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 $\mathsf{AggVerify}(\mathsf{pp}, (\mathsf{vk}_1, \ldots, \mathsf{vk}_T), (\mathsf{M}_1, \ldots, \mathsf{M}_T), \sigma^*) = 1$. Otherwise, the output is $b = 0$.*

An AggSig Π_{AggSig} is unforgeable if for all $\lambda, N \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$ and all polynomials $N = N(\lambda)$, it holds that

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathcal{A}, \Pi_{\mathsf{AggSig}}}(\lambda) \coloneqq \Pr[b = 1] = \mathsf{negl}(\lambda)$$

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 $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \{0,1\}^{S \times (x+w)}$ be matrices. We define the (binary) rank-1 constraint system (R1CS) relation $R^{\mathsf{R1CS}}_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ as follows:*

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

where $\circ$ denotes the component-wise multiplication. We also define the language of R1CS $\mathcal{L}^{\mathsf{R1CS}}_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ as follows:

$$\mathcal{L}^{\mathsf{R1CS}}_{(\mathbf{A},\mathbf{B},\mathbf{C})} \coloneqq \left\{ \mathbf{x} \in \{0,1\}^x \;\middle|\; \exists \mathbf{w} \in \{0,1\}^w \text{ such that } (\mathbf{x}, \mathbf{w}) \in R_{(\mathbf{A},\mathbf{B},\mathbf{C})} \right\}.$$

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

- $\mathsf{Setup}(1^\lambda, 1^N) \rightarrow \mathsf{crs}$*: 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*.*
- $\mathsf{Prove}(\mathsf{crs}, \vec{\mathbf{x}}, \vec{\mathbf{w}}) \rightarrow \pi$*: 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 witnesses $\vec{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_N) \in (\{0,1\}^w)^N$, the prove algorithm outputs a proof π.*
- $\mathsf{Verify}(\mathsf{crs}, \vec{\mathbf{x}}, \pi) \rightarrow b$*: 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). *A BARG Π_{BARG} for $R^{\mathsf{R1CS}}_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ 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 $(\mathbf{x}_i, \mathbf{w}_i) \in R^{\mathsf{R1CS}}_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ for all $i \in [N]$, it holds that*

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

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]). *A BARG Π_{BARG} for $R^{\mathsf{R1CS}}_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ is somewhere argument of knowledge if there exists a pair of PPT algorithms* $(\mathsf{TrapSetup}, \mathsf{Extract})$ *with the following properties:*

- $\mathsf{TrapSetup}(1^\lambda, 1^N, i^*) \rightarrow (\mathsf{crs}^*, \mathsf{td})$*: 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*.*
- $\mathsf{Extract}(\mathsf{td}, \vec{\mathbf{x}}, \pi) \rightarrow \mathbf{w}^*$*: 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 $\mathbf{w}^* \in \{0,1\}^w$. The extraction algorithm is deterministic.*

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

CRS indistinguishability: *For $\lambda, N \in \mathbb{N}$ and an adversary $\mathcal{A}$, define the CRS indistinguishability game for between $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:*

1. *$\mathcal{A}$ declares an index $i^* \in [N]$ and gives i^* to $\mathcal{C}$.*
2. *$\mathcal{C}$ samples $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, $\mathcal{C}$ gives $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N)$ to $\mathcal{A}$. If $b = 1$, $\mathcal{C}$ 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 $\lambda, N \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{crs}}_{\mathcal{A},\Pi_{\mathsf{BARG}}}(\lambda) := \Pr[b = b'] = 1/2 + \mathsf{negl}(\lambda)$$

in the above game.

Somewhere extractable in trapdoor mode: *For $\lambda, N \in \mathbb{N}$ and an adversary $\mathcal{A}$, define the somewhere extractable security game between $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:*

1. *$\mathcal{A}$ declares an index $i^* \in [N]$ and gives i^* to $\mathcal{C}$.*

2. *$\mathcal{C}$ 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*

$$\mathsf{Verify}(\mathsf{crs}^*, \vec{\mathbf{x}}, \pi) = 1 \textit{ and } (\mathbf{x}_{i^*}, \mathbf{w}^*) \notin R^{\mathsf{R1CS}}_{(\mathbf{A},\mathbf{B},\mathbf{C})}.$$

Otherwise, the output is $b = 0$.

A BARG Π_{BARG} *is somewhere extractable in trapdoor mode if for all* $\lambda, N \in \mathbb{N}$ *and all adversaries* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{se}}_{\mathcal{A},\Pi_{\mathsf{BARG}}}(\lambda) := \Pr[b = 1] = \mathsf{negl}(\lambda)$$

in the above game.

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

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 $R \subseteq \{0,1\}^x \times \{0,1\}^w$ 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 $\mathbf{x}$. The language $\mathcal{L}_R$ is defined as $\{\mathbf{x} \in \{0,1\}^x \mid \exists \mathbf{w} \in \{0,1\}^w \text{ such that } (\mathbf{x}, \mathbf{w}) \in R\}$. We relax the notion of soundness to hold only for a slightly wider relation $\tilde{R}$ (i.e., $R \subseteq \tilde{R}$) and the corresponding language $\mathcal{L}_{\tilde{R}} = \{\mathbf{x} \in \{0,1\}^x \mid \exists \mathbf{w} \in \{0,1\}^w \text{ such that } (\mathbf{x}, \mathbf{w}) \in \tilde{R}\}$.

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

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

Definition 2.13 (Interactive Arguments for NP). *A* $(2\mu + 1)$*-round public-coin interactive argument (ARG) for the relations* R *and* $\tilde{R}$ *such that* $R \subseteq \tilde{R}$ *in the common reference string (CRS) model is a tuple of* $\mu + 3$ *PPT algorithms* $\Pi_{\mathsf{ARG}} = (\mathsf{Setup}, \mathsf{P} = (\mathsf{P}_1, \ldots, \mathsf{P}_{\mu+1}), \mathsf{V})$ *and a family of the challenge spaces* $\{\mathcal{C}_i\}_{i \in [\mu]}$ *with the following properties:*

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

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$*: 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* $\mathsf{ch}_0 = \perp$*,* $\mathsf{st}_0 = (\mathbf{x}, \mathbf{w})$*,* $\mathbf{x}$ *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* $\vec{\mathsf{ch}} = (\mathsf{ch}_1, \ldots, \mathsf{ch}_\mu)$*, and responses* $\vec{\mathsf{res}} = (\mathsf{res}_1, \ldots, \mathsf{res}_{\mu+1})$*, the verification algorithm outputs a bit* $b \in \{0, 1\}$*.*

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

$$\Pr[\mathsf{V}(\mathsf{crs}, \mathbf{x}, (\mathsf{ch}_1, \ldots, \mathsf{ch}_\mu), (\mathsf{res}_1, \ldots, \mathsf{res}_{\mu+1})) = 1] = 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$*,* $\mathsf{res}_i \xleftarrow{\$} \mathsf{P}(\mathsf{crs}, \mathsf{ch}_{i-1}, \mathsf{st}_{i-1})$ *and* $\mathsf{ch}_i \xleftarrow{\$} \mathcal{C}_i$ *for* $i \in [\mu]$*,* $\mathsf{ch}_0 = \perp$*,* $\mathsf{st}_0 = (\mathbf{x}, \mathbf{w})$*, and the probability is taken over the randomness of all algorithms.*

Definition 2.15 (Soundness). *For* $\lambda \in \mathbb{N}$*,* $\mathbf{x} \in \{0, 1\}^x$*, and an adversary* P^**, define the soundness game for between* P^* *and a challenger* $\mathcal{C}$ *as follows:*

1. $\mathcal{C}$ *samples* $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(\lambda)$ *and gives* crs *to* $\mathcal{A}$*.*
2. *For* $i \in [\mu]$*:*

   2-1. $\mathcal{A}$ *outputs* res_i *and gives it to* $\mathcal{C}$*.*

   2-2. $\mathcal{C}$ *samples* $\mathsf{ch}_i \xleftarrow{\$} \mathcal{C}_i$ *and gives it to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs* $\mathsf{res}_{\mu+1}$ *and gives* $\mathcal{C}$*.*
4. *The output of the game* $b = 1$ *if* $\mathsf{V}(\mathsf{crs}, \mathbf{x}, (\mathsf{ch}_1, \ldots, \mathsf{ch}_\mu), (\mathsf{res}_1, \ldots, \mathsf{res}_{\mu+1})) = 1$*. Otherwise, the output is* $b = 0$*.*

An ARG Π_{ARG} *is sound if for all* $\lambda \in \mathbb{N}$*, all statements* $\mathbf{x}$ *such that* $\mathbf{x} \notin \mathcal{L}_{\tilde{R}}$*, and all PPT adversaries* P^**, it holds that*

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathcal{A}, \Pi_{\mathsf{ARG}}}(\lambda) \coloneqq \Pr[b = 1] = \mathsf{negl}(\lambda)$$

in the above game.

Definition 2.16 (Succinctness). *An ARG* Π_{ARG} *is succinct if there exists a fixed polynomial* $\mathsf{poly}(\cdot, \cdot, \cdot)$ *such that for all* $\lambda, w \in \mathbb{N}$*, the total size of responses satisfies* $\sum_{i=1}^{\mu+1} |\mathsf{res}_i| \leq \mathsf{poly}(\lambda, \log w)$*.*

2.4 Lattice Preliminaries

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

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

Definition 2.17 (MSIS Assumption). *Let* n, m, q *be positive integers and* β *be a positive real. The module 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* $\lambda \in \mathbb{N}$*:*

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

where $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{m \times n}$*.*

Definition 2.18 (MLWE Assumption). *Let n, n', m, q be positive integers and χ be a distribution over $\mathcal{R}_q$. The module 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 $\lambda \in \mathbb{N}$:*

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

where $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{m \times n}$, $\mathbf{Z} \xleftarrow{\$} \chi^{n' \times n}$, $\mathbf{E} \xleftarrow{\$} \chi^{n' \times m}$, and $\mathbf{B} \xleftarrow{\$} \mathcal{R}_q^{n' \times m}$.

3 Our BARG

In this section, we present our construction of a 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 $\mathcal{F} \coloneqq \{f^{(1)}, \dots, f^{(K)}\}$ of quadratic dot product functions $f : (\mathcal{R}_q^n)^r \to \mathcal{R}_q$ of the form

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

where $\phi_{i,j}, \nu \in \mathcal{R}_q$ and $\boldsymbol{\psi}_i \in \mathcal{R}_q^n$. Without loss of generality, we assume $\phi_{i,j} = \phi_{j,i}$. We now define the principal relation.

Definition 3.1 (Principal Relation). *Let $n, r \geq 1$ be integers, $\beta > 0$ be a real, and $\mathcal{F} \coloneqq \{f^{(1)}, \dots, f^{(K)}\}$ and $\hat{\mathcal{F}} \coloneqq \{\hat{f}^{(1)}, \dots, \hat{f}^{(\hat{K})}\}$ be two families of quadratic dot product functions. We define the principal relation*

$$R_\beta^{\mathsf{pr}} \coloneqq \left\{ ((\mathcal{F}, \hat{\mathcal{F}}), (\mathbf{s}_1, \dots, \mathbf{s}_r)) \;\middle|\; \begin{array}{l} \forall f \in \mathcal{F}, f(\mathbf{s}_1, \dots, \mathbf{s}_r) = \mathbf{0}, \\ \forall \hat{f} \in \hat{\mathcal{F}}, \mathsf{const}(\hat{f}(\mathbf{s}_1, \dots, \mathbf{s}_r)) = 0, \\ \sum_{i=1}^{r} \|\mathbf{s}_i\|_2^2 \leq \beta^2 \end{array} \right\}.$$

We also define the language of principal relation $\mathcal{L}_\beta^{\mathsf{pr}}$ as follows:

$$\mathcal{L}_\beta^{\mathsf{pr}} \coloneqq \left\{ (\mathcal{F}, \hat{\mathcal{F}}) \;\middle|\; \exists \mathbf{s}_1, \dots, \mathbf{s}_r \in \mathcal{R}_q^n \text{ such that } ((\mathcal{F}, \hat{\mathcal{F}}), (\mathbf{s}_1, \dots, \mathbf{s}_r)) \in R_\beta^{\mathsf{pr}} \right\}.$$

Recently, Beullens and Seilar [BS22] proposed a highly efficient ARG for principal relations. In the following lemma, we summarize their result.

Lemma 3.2 ([BS22]). *Let $\beta > 0$ be a real. There exists an ARG Π_{BS} for the principal relation R_β^{pr} in the CRS model with the following properties:*

- *Π_{BS} is complete.*
- *Assuming that the MSIS assumption holds, Π_{BS} is sound*[2] *for the wider relations $R^{\mathsf{pr}}_{\sqrt{128/30} \cdot \beta}$.*
- *Π_{BS} is succinct. More precisely, the total size of responses is $O(\log n + \log r) \cdot \mathsf{poly}(\lambda)$.*

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

3.2 Principal-Relation-Compatible Somewhere Extractable Commitment

We introduce *principal-relation-compatible* somewhere extractable commitment. For instance, the partially decryptable commitment scheme by Esgin et al. [ESZ22] is one specific instantiation.

Definition 3.3 (Principal-Relation-Compatible Somewhere Extractable Commitment). *A principal-relation-compatible somewhere extractable commitment (SECom) over $\mathcal{R}_q$ is a tuple of PPT algorithms $\Pi_{\mathsf{SECom}} = (\mathsf{KGen}, \mathsf{Com})$ with the following properties:*

- $\mathsf{KGen}(1^\lambda, 1^L, 1^\ell) \to \mathsf{ck}$*: On input of the security parameter λ, the input length L, and the locality parameter ℓ, the key generation algorithm outputs a commitment key* ck*.*
- $\mathsf{Com}(\mathsf{ck}, \mathbf{m}) \to \mathbf{c}$*: On input of the commitment key* ck *and a vector $\mathbf{m} \in \mathcal{R}_2^L$, the commit algorithm outputs a commitment $\mathbf{c} \in \mathcal{R}_q^n$. The commit algorithm is deterministic.*

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

Definition 3.4 (Principal-Relation-Compatibility). *Let $\lambda, L, \ell, n \in \mathbb{N}$, and $\mathcal{F}_{\mathsf{SECom}} = \{f_1, \ldots, f_n\}$ be a family of quadratic dot product functions. An SECom Π_{SECom} is $\mathcal{F}_{\mathsf{SECom}}$-principal-relation-compatible if for any $\mathsf{ck} \in \mathsf{KGen}(1^\lambda, 1^L, 1^\ell)$, $\mathbf{m} \in \mathcal{R}_2^L$, and $\mathbf{c} \in \mathsf{Com}(\mathsf{ck}, \mathbf{m})$, it holds that $c_i = f_i(\mathbf{m})$ for any $i \in [n]$, where $c_i \in \mathcal{R}_q$ is a i-th element of $\mathbf{c}$.*

Definition 3.5 (Somewhere Extractable). *An SECom Π_{SECom} is somewhere extractable if there exists a pair of PPT algorithms* $(\mathsf{TrapKGen}, \mathsf{Extract})$ *with the following properties:*

- $\mathsf{TrapKGen}(1^\lambda, 1^L, 1^\ell, i) \to (\mathsf{ck}^*, \mathsf{td})$*: On input of the security parameter λ, the input length L, the locality parameter ℓ, and an index $i \in [L]$, the trapdoor key generation algorithm outputs a commitment key* ck *and a trapdoor* td*.*
- $\mathsf{Extract}(\mathsf{td}, \mathbf{c}) \to \mathbf{m}$*: On input of the trapdoor* td *and a commitment $\mathbf{c}$, the extraction algorithm outputs a vector $\mathbf{m} \in \mathcal{R}_2^\ell$. The extraction algorithm is deterministic.*

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

Key indistinguishability: *For $\lambda, L, \ell \in \mathbb{N}$ and an adversary $\mathcal{A}$, define the key indistinguishability game for the SECom between $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:*

1. *$\mathcal{A}$ declares an interval $i \in [L]$ and gives it to $\mathcal{C}$.*
2. *The challenger samples $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, $\mathcal{C}$ gives $\mathsf{ck} \xleftarrow{\$} \mathsf{KGen}(1^\lambda, 1^L, 1^\ell)$ to $\mathcal{A}$. If $b = 1$, $\mathcal{C}$ samples $(\mathsf{ck}^*, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapKGen}(1^\lambda, 1^L, 1^\ell, i)$ and gives ck^* to $\mathcal{A}$.*
3. *At the end of the game, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

An SECom Π_{SECom} is key indistinguishable if for all PPT adversaries $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{key}}_{\mathcal{A},\Pi_{\mathsf{SECom}}}(\lambda) := \Pr[b = b'] = 1/2 + \mathsf{negl}(\lambda)$$

in the above game.

Somewhere extractable in trapdoor mode: *For any $\lambda, L, \ell \in \mathbb{N}$, any $\mathbf{m} \in \{0,1\}^L$, any $i \in [L]$, it holds that*

$$\Pr[\mathsf{Extract}(\mathsf{td}, \mathsf{Com}(\mathsf{ck}^*, \mathbf{m})) = (m_i, \ldots, m_{i+\ell})] = 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{ck}^*, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapKGen}(1^\lambda, 1^L, 1^\ell, i)$, $m_i \in \mathcal{R}_2$ is the i-th element of $\mathbf{m}$, and the probability is taken over the randomness of all algorithms.

Definition 3.6 (Succinctness). *An SECom Π_{SECom} is succinct if there exists a fixed polynomial* $\mathsf{poly}(\cdot,\cdot,\cdot)$ *such that for $\lambda, L, \ell \in \mathbb{N}$, the size of a commitment $\mathbf{c}$ satisfies $|\mathbf{c}| \leq \ell \cdot \mathsf{poly}(\lambda, \log L)$.*

In the following lemma, we show that a variant of the commitment scheme by Esgin et al. [ESZ22] is a principal-relation-compatible SECom scheme. The proof can be found in Appendix A.

Lemma 3.7 ([ESZ22, adapted]). *There exists an SECom Π_{ESZ} scheme with the following properties:*

- *Π_{ESZ} is principal-relation-compatible.*
- *Assuming that the MSIS and MLWE assumptions hold, Π_{ESZ} is somewhere extractable.*
- *Π_{ESZ} is succinct. More precisely, the size of a commitment is $\ell \cdot \log L \cdot \mathsf{poly}(\lambda)$.*

3.3 Construction: Our BARG

Building Blocks. Let $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \{0,1\}^{S\times(x+w)}$ be matrices. Our BARG $\Pi_{\mathsf{BARG}} = (\mathsf{Setup}_{\mathsf{BARG}}, \mathsf{Prove}, \mathsf{Verify})$ for $R^{\mathsf{R1CS}}_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ relies on the following building blocks.

- A SECom scheme $\Pi_{\mathsf{SECom}} = (\mathsf{KGen}, \mathsf{Com})$ over $\mathcal{R}_q$ that satisfies somewhere extractability, succinctness, and $\mathcal{F}_{\mathsf{SECom}}$-principal-relation-compatibility for a family of quadratic dot product functions $\mathcal{F}_{\mathsf{SECom}} = \{f_1, \ldots, f_n\}$.
- A $(2\mu+1)$-round public-coin ARG $\Pi_{\mathsf{pr}} = (\mathsf{Setup}_{\mathsf{pr}}, \mathsf{P} = (\mathsf{P}_1, \ldots, \mathsf{P}_{\mu+1}), \mathsf{V})$ with challenge spaces $\{\mathcal{C}_i\}_{i\in[\mu]}$ for the relations $R^{\mathsf{pr}}_{\sqrt{q}}$ and $R^{\mathsf{pr}}_{\sqrt{128/30}\cdot\sqrt{q}}$ that satisfies completeness, soundness, and succinctness.
- $\mu+1$ hash functions $\mathsf{H}, \mathsf{H}_1, \ldots, \mathsf{H}_\mu$ modeled as a random oracle in the security proof. $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\lambda\times 3NS}$ is a hash function that maps a crs-instances-commitment pair to the binary coefficients of the λ linear combinations. $\mathsf{H}_i : \{0,1\}^* \to \mathcal{C}_i$ for $i \in [\mu]$ are hash functions mapping into the respective challenge spaces $\mathcal{C}_i$.

Construction. Our BARG Π_{BARG} for $R^{\mathsf{R1CS}}_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ is described as follows:

- $\mathsf{Setup}_{\mathsf{BARG}}(1^\lambda, 1^N) \to \mathsf{crs}_{\mathsf{BARG}}$:
  1. Set $L \coloneqq 3\lceil NS/d\rceil + N(\lceil x/d\rceil + \lceil w/d\rceil)$ and $\ell \coloneqq \lceil w/d\rceil$.
  2. Sample $\mathsf{ck} \xleftarrow{\$} \mathsf{KGen}(1^\lambda, 1^L, 1^\ell)$ and $\mathsf{crs}_{\mathsf{pr}} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{pr}}(1^\lambda)$.
  3. Output $\mathsf{crs}_{\mathsf{BARG}} \coloneqq (\mathsf{ck}, \mathsf{crs}_{\mathsf{pr}})$.
- $\mathsf{Prove}(\mathsf{crs}_{\mathsf{BARG}}, \vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_N), \vec{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_N)) \to \pi$:
  1. Set

$$\mathbf{y}_{\mathsf{b}} \coloneqq \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{w}_1 \\ \vdots \\ \mathbf{x}_N \\ \mathbf{w}_N \end{pmatrix} \in \{0,1\}^{N(x+w)}, \qquad \mathbf{y}_{\mathsf{r}} \coloneqq \begin{pmatrix} \mathsf{Lift}_x(\mathbf{x}_1) \\ \mathsf{Lift}_w(\mathbf{w}_1) \\ \vdots \\ \mathsf{Lift}_x(\mathbf{x}_N) \\ \mathsf{Lift}_w(\mathbf{w}_N) \end{pmatrix} \in \mathcal{R}_2^{N(\lceil x/d\rceil + \lceil w/d\rceil)},$$

and

$$\begin{aligned} \mathbf{a}_{\mathsf{b}} &\coloneqq \mathbf{A}^{\otimes N}\mathbf{y}_{\mathsf{b}} \bmod 2 \in \{0,1\}^{NS}, & \mathbf{a}_{\mathsf{r}} &\coloneqq \mathsf{Lift}_{NS}(\mathbf{a}_{\mathsf{b}}) \in \mathcal{R}_2^{\lceil NS/d\rceil}, \\ \mathbf{b}_{\mathsf{b}} &\coloneqq \mathbf{B}^{\otimes N}\mathbf{y}_{\mathsf{b}} \bmod 2 \in \{0,1\}^{NS}, & \mathbf{b}_{\mathsf{r}} &\coloneqq \mathsf{Lift}_{NS}(\mathbf{b}_{\mathsf{b}}) \in \mathcal{R}_2^{\lceil NS/d\rceil}, \\ \mathbf{c}_{\mathsf{b}} &\coloneqq \mathbf{C}^{\otimes N}\mathbf{y}_{\mathsf{b}} \bmod 2 \in \{0,1\}^{NS}, & \mathbf{c}_{\mathsf{r}} &\coloneqq \mathsf{Lift}_{NS}(\mathbf{c}_{\mathsf{b}}) \in \mathcal{R}_2^{\lceil NS/d\rceil}, \end{aligned}$$

where

$$\mathbf{A}^{\otimes N} \coloneqq \mathbf{I}_N \otimes \mathbf{A} \in \{0,1\}^{NS \times N(x+w)},$$
$$\mathbf{B}^{\otimes N} \coloneqq \mathbf{I}_N \otimes \mathbf{B} \in \{0,1\}^{NS \times N(x+w)},$$
$$\mathbf{C}^{\otimes N} \coloneqq \mathbf{I}_N \otimes \mathbf{C} \in \{0,1\}^{NS \times N(x+w)}.$$

2. Compute $\mathbf{u} \coloneqq \mathsf{Com}(\mathsf{ck}, \mathbf{s}) \in \mathcal{R}_q^n$, where

$$\mathbf{s} \coloneqq \begin{pmatrix} \mathbf{a}_\mathsf{r} \\ \mathbf{b}_\mathsf{r} \\ \mathbf{c}_\mathsf{r} \\ \mathbf{y}_\mathsf{r} \end{pmatrix} \in \mathcal{R}_2^{3\lceil NS/d \rceil + N(\lceil x/d \rceil + \lceil w/d \rceil)} = \mathcal{R}_2^L.$$

3. Compute $\mathbf{P} \coloneqq \mathsf{H}(\mathsf{crs}_\mathsf{BARG}, \vec{\mathbf{x}}, \mathbf{u}) \in \{0,1\}^{\lambda \times 3NS}$ and set

$$\mathbf{D} \coloneqq \mathbf{P} \begin{pmatrix} \mathbf{A}^{\otimes N} \\ \mathbf{B}^{\otimes N} \\ \mathbf{C}^{\otimes N} \end{pmatrix} \bmod 2 \in \{0,1\}^{\lambda \times N(x+w)}, \qquad \mathbf{d} \coloneqq \mathbf{P} \begin{pmatrix} \mathbf{a}_\mathsf{b} \\ \mathbf{b}_\mathsf{b} \\ \mathbf{c}_\mathsf{b} \end{pmatrix} - \mathbf{D}\mathbf{y}_\mathsf{b} \bmod 2 \in \{0,1\}^{\lambda}.$$

4. Set $\tilde{\mathbf{a}}_\mathsf{r} \coloneqq \sigma_{-1}(\mathbf{a}_\mathsf{r})$, $\tilde{\mathbf{b}}_\mathsf{r} \coloneqq \sigma_{-1}(\mathbf{b}_\mathsf{r})$, $\tilde{\mathbf{c}}_\mathsf{r} \coloneqq \sigma_{-1}(\mathbf{c}_\mathsf{r})$, $\tilde{\mathbf{y}}_\mathsf{r} \coloneqq \sigma_{-1}(\mathbf{y}_\mathsf{r})$.
5. $\vec{\mathbf{r}} \coloneqq (\mathbf{a}_\mathsf{r}, \mathbf{b}_\mathsf{r}, \mathbf{c}_\mathsf{r}, \mathbf{y}_\mathsf{r}, \tilde{\mathbf{a}}_\mathsf{r}, \tilde{\mathbf{b}}_\mathsf{r}, \tilde{\mathbf{c}}_\mathsf{r}, \tilde{\mathbf{y}}_\mathsf{r})$ as witness vectors, we define the statement for the principal relations as follows:

$$\mathcal{F}_{(\mathsf{ck},\mathbf{u})} \coloneqq \{f_i(\mathbf{a}_\mathsf{r}, \mathbf{b}_\mathsf{r}, \mathbf{c}_\mathsf{r}, \mathbf{y}_\mathsf{r}) - u_i\}_{i \in [n]}, \tag{1}$$

$$\hat{\mathcal{F}}_{(\mathbf{P},\mathbf{D},\mathbf{d})} \coloneqq \left\{ \begin{array}{l} \tilde{\mathbf{a}}_\mathsf{r} = \sigma_{-1}(\mathbf{a}_\mathsf{r}),\ \tilde{\mathbf{b}}_\mathsf{r} = \sigma_{-1}(\mathbf{b}_\mathsf{r}), \\ \tilde{\mathbf{c}}_\mathsf{r} = \sigma_{-1}(\mathbf{c}_\mathsf{r}),\ \tilde{\mathbf{y}}_\mathsf{r} = \sigma_{-1}(\mathbf{y}_\mathsf{r}), \\ \mathbf{a}_\mathsf{r}^\top(\tilde{\mathbf{a}}_\mathsf{r} - \mathbf{1}_{\lceil NS/d \rceil}),\ \mathbf{b}_\mathsf{r}^\top(\tilde{\mathbf{b}}_\mathsf{r} - \mathbf{1}_{\lceil NS/d \rceil}), \\ \mathbf{c}_\mathsf{r}^\top(\tilde{\mathbf{c}}_\mathsf{r} - \mathbf{1}_{\lceil NS/d \rceil}),\ \mathbf{y}_\mathsf{r}^\top(\tilde{\mathbf{y}}_\mathsf{r} - \mathbf{1}_{N(\lceil x/d \rceil + \lceil w/d \rceil)}), \\ (\mathbf{a}_\mathsf{r} + \mathbf{b}_\mathsf{r} - 2\mathbf{c}_\mathsf{r})^\top(\tilde{\mathbf{a}}_\mathsf{r} + \tilde{\mathbf{b}}_\mathsf{r} - 2\tilde{\mathbf{c}}_\mathsf{r} - \mathbf{1}_{\lceil NS/d \rceil}) \end{array} \right\} \cup \left\{ \mathbf{P} \begin{pmatrix} \mathbf{a}_\mathsf{b} \\ \mathbf{b}_\mathsf{b} \\ \mathbf{c}_\mathsf{b} \end{pmatrix} - \mathbf{D}\mathbf{y}_\mathsf{b} - \mathbf{d} \right\}, \tag{2}$$

where u_i is the i-th element of $\mathbf{u}$ and $\mathbf{1}_k \in \mathcal{R}_2^k$ is a polynomial vector whose coefficients are all 1's.
6. To prove $((\mathcal{F}_{(\mathsf{ck},\mathbf{u})}, \hat{\mathcal{F}}_{(\mathbf{P},\mathbf{D},\mathbf{d})}), \vec{\mathbf{r}}) \in R^{\mathsf{pr}}_{\sqrt{q}}$,
   - set $\mathsf{ch}_0 \coloneqq \perp$ and $\mathsf{st}_0 \coloneqq (\vec{\mathbf{x}}, \vec{\mathbf{w}})$,
   - run $(\mathsf{st}_i, \mathsf{res}_i) \xleftarrow{\$} \mathsf{P}_i(\mathsf{crs}, \mathsf{st}_{i-1}, \mathsf{ch}_{i-1})$ and compute $\mathsf{ch}_i \coloneqq \mathsf{H}_i(\mathsf{crs}_\mathsf{BARG}, \vec{\mathbf{x}}, \mathbf{u}, \mathbf{d}, \mathsf{res}_1, \dots, \mathsf{res}_i)$ for $i \in [\mu]$, and
   - run $(\mathsf{st}_{\mu+1}, \mathsf{res}_{\mu+1}) \xleftarrow{\$} \mathsf{P}_{\mu+1}(\mathsf{crs}, \mathsf{st}_\mu, \mathsf{ch}_\mu)$.
7. Output $\pi \coloneqq (\mathbf{u}, \mathbf{d}, \pi_\mathsf{pr} = (\mathsf{res}_1, \dots, \mathsf{res}_{\mu+1}))$.

- $\mathsf{Verify}(\mathsf{crs}_\mathsf{BARG}, \vec{\mathbf{x}}, \pi) \to b$:
  1. Parse $\pi \eqqcolon (\mathbf{u}, \mathbf{d}, \pi_\mathsf{pr} = (\mathsf{res}_1, \dots, \mathsf{res}_{\mu+1}))$.
  2. If $\mathbf{d}$ has an odd element, then output 0.
  3. Compute $\mathbf{P} \coloneqq \mathsf{H}(\mathsf{crs}_\mathsf{BARG}, \vec{\mathbf{x}}, \mathbf{u}) \in \{0,1\}^{\lambda \times 3NS}$ and set

$$\mathbf{D} \coloneqq \mathbf{P} \begin{pmatrix} \mathbf{A}^{\otimes N} \\ \mathbf{B}^{\otimes N} \\ \mathbf{C}^{\otimes N} \end{pmatrix} \bmod 2 \in \{0,1\}^{\lambda \times N(x+w)}.$$

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 $\mathsf{ch}_i := \mathsf{H}_i(\mathsf{crs}_{\mathsf{BARG}}, \vec{\mathbf{x}}, \mathbf{u}, \mathbf{d}, \mathsf{res}_1, \dots, \mathsf{res}_i)$ for $i \in [\mu]$ and set $\vec{\mathsf{ch}} := (\mathsf{ch}_1, \dots, \mathsf{ch}_\mu)$.
6. To verify $(\mathcal{F}_{(\mathsf{ck},\mathbf{u})}, \hat{\mathcal{F}}_{(\mathbf{P},\mathbf{D},\mathbf{d})}) \in \mathcal{L}^{\mathsf{pr}}_{\sqrt{q}}$, set and output $b = \mathsf{V}(\mathsf{crs}_{\mathsf{pr}}, \vec{\mathbf{x}}, \vec{\mathsf{ch}}, \pi_{\mathsf{pr}})$.

The above Π_{BARG} has the following properties:

- If Π_{pr} is complete, then Π_{BARG} is complete.
- If Π_{SECom} is somewhere extractable and Π_{pr} is sound, Π_{BARG} is somewhere argument of knowledge in the random oracle model.
- If Π_{SECom} and Π_{pr} are succinct, then Π_{BARG} is also succinct.

3.4 Security Proof

Theorem 3.8. *The above* Π_{BARG} *has the following properties:*

- *If* Π_{pr} *is complete, then* Π_{BARG} *is complete.*
- *If* Π_{SECom} *is somewhere extractable and* Π_{pr} *is sound,* Π_{BARG} *is somewhere argument of knowledge in the random oracle model.*
- *If* Π_{SECom} *and* Π_{pr} *are succinct, then* Π_{BARG} *is also succinct.*

Proof. 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]. The main difference is the use of the principal-relation somewhere extractable commitment scheme instead of the Ajtai commitment scheme.

Completeness: We assume that $\vec{\mathbf{x}} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ and $\vec{\mathbf{w}} = (\mathbf{w}_1, \dots, \mathbf{w}_N)$ are valid instances-witnesses pairs. That is,

$$\mathbf{A}\begin{pmatrix}\mathbf{x}_i\\ \mathbf{w}_i\end{pmatrix} \circ \mathbf{B}\begin{pmatrix}\mathbf{x}_i\\ \mathbf{w}_i\end{pmatrix} = \mathbf{C}\begin{pmatrix}\mathbf{x}_i\\ \mathbf{w}_i\end{pmatrix},$$

for all $i \in [N]$. Hence, we have $\mathbf{A}^{\otimes N}\mathbf{y}_{\mathsf{b}} \circ \mathbf{B}^{\otimes N}\mathbf{y}_{\mathsf{b}} = \mathbf{C}^{\otimes N}\mathbf{y}_{\mathsf{b}}$. As described in [BS22, Section 6], $((\mathcal{F}_{\mathsf{ck},\mathbf{u}}, \hat{\mathcal{F}}_{\mathbf{P},\mathbf{D},\mathbf{d}}), \vec{\mathbf{r}}) \in R^{\mathsf{pr}}_{\sqrt{q}}$ implies

- $\mathbf{a}_{\mathsf{b}} = \mathbf{A}^{\otimes N}\mathbf{y}_{\mathsf{b}}$, $\mathbf{b}_{\mathsf{b}} = \mathbf{B}^{\otimes N}\mathbf{y}_{\mathsf{b}}$, $\mathbf{c}_{\mathsf{b}} = \mathbf{C}^{\otimes N}\mathbf{y}_{\mathsf{b}}$,
- the coefficients of $\mathbf{a}_{\mathsf{r}}, \mathbf{b}_{\mathsf{r}}, \mathbf{c}_{\mathsf{r}}, \mathbf{y}_{\mathsf{r}}$ are binary, and
- $\mathbf{a}_{\mathsf{b}} \circ \mathbf{b}_{\mathsf{b}} = \mathbf{c}_{\mathsf{b}}$.

Therefore, Π_{BARG} is complete if Π_{pr} is complete.

Somewhere argument of knowledge: 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. Set $L := 3\lceil NS/d\rceil + N(\lceil x/d\rceil + \lceil w/d\rceil)$, $\ell := \lceil w/d\rceil$, and $L' := 3\lceil NS/d\rceil + (i^*+1)\lceil x/d\rceil$.
  2. Sample $(\mathsf{ck}^*, \mathsf{td}_{\mathsf{SECom}}) \xleftarrow{\$} \mathsf{TrapKGen}_{\mathsf{SECom}}(1^\lambda, 1^L, 1^\ell, L' + i^*\lceil w/d\rceil)$ and $\mathsf{crs}_{\mathsf{pr}} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{pr}}(1^\lambda)$.
  3. Output $\mathsf{crs}^*_{\mathsf{BARG}} := (\mathsf{ck}^*, \mathsf{crs}_{\mathsf{pr}})$ and $\mathsf{td} := \mathsf{td}_{\mathsf{SECom}}$.
- $\mathsf{Extract}(\mathsf{td}, \vec{\mathbf{x}}, \pi) \to \mathbf{w}^*$:
  1. Parse $\pi =: (\mathbf{u}, \mathbf{d}, (\mathsf{res}_1, \dots, \mathsf{res}_{\mu+1}))$.
  2. Compute $\mathbf{w}^*_{\mathsf{r}} = \mathsf{Extract}_{\mathsf{SECom}}(\mathsf{td}, \mathbf{u}) \in \mathcal{R}_2^{\lceil w/d\rceil} = \mathcal{R}_2^\ell$.
  3. Output $\mathbf{w}^* = \mathsf{BD}_w(\mathbf{w}^*_{\mathsf{r}}) \in \{0,1\}^w$.

Here, the function BD_w performs bit decomposition of the coefficients of the $\mathbf{w}^*_{\mathsf{r}} \in \mathcal{R}_2^{\lceil w/d \rceil}$ and returns the resulting binary vector $\mathbf{w}^* \in \{0,1\}^w$.

Clearly, the CRS indistinguishability of Π_{BARG} directly follows the key indistinguishability of Π_{SECom}. Then, we show that Π_{BARG} is somewhere extractable in trapdoor mode.

Lemma 3.9 (Somewhere Extractable in Trapdoor Mode). *If Π_{SECom} is somewhere extractable in trapdoor mode and Π_{pr} is sound, then Π_{BARG} is somewhere extractable in trapdoor mode and the random oracle model.*

Proof Sketch of Lemma 3.9. Assume that there exists a PPT adversary $\mathcal{A}$ against the somewhere extractable security game for Π_{BARG}. Let i^* be the index declared by $\mathcal{A}$ at the beginning of the game and $(\mathsf{crs}^*_{\mathsf{BARG}}, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapSetup}(1^\lambda, 1^N, i^*)$. By construction $\mathsf{crs}^*_{\mathsf{BARG}} = (\mathsf{ck}^*, \mathsf{crs}_{\mathsf{pr}})$, where

$$(\mathsf{ck}^*, \mathsf{td}_{\mathsf{SECom}}) \xleftarrow{\$} \mathsf{TrapKGen}_{\mathsf{SECom}}(1^\lambda, 1^L, 1^\ell, L' + i^* \lceil w/d \rceil),$$
$$\mathsf{crs}_{\mathsf{pr}} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{pr}}(1^\lambda),$$

$L = 3\lceil NS/d \rceil + N(\lceil x/d \rceil + \lceil w/d \rceil)$, $\ell = \lceil w/d \rceil$, and $L' = 3\lceil NS/d \rceil + (i^*+1)\lceil x/d \rceil$. Let $\vec{\mathbf{x}} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ and $\pi = (\mathbf{u}, \mathbf{d}, \pi_{\mathsf{pr}} = (\mathsf{res}_1, \dots, \mathsf{res}_{\mu+1}))$ be statements and a proof outputted by $\mathcal{A}$. Suppose $\mathsf{Verify}(\mathsf{crs}_{\mathsf{BARG}}, \vec{\mathbf{x}}, \pi) = 1$. For the random oracle queries of $\mathcal{A}$, the challenger returns a uniformly randomly sampled element from the corresponding range.

By construction, we have

$$\mathsf{V}(\mathsf{crs}_{\mathsf{pr}}, \vec{\mathbf{x}}, \vec{\mathsf{ch}} = (\mathsf{ch}_1, \dots, \mathsf{ch}_\mu), \pi_{\mathsf{pr}}) = 1,$$

where $\mathsf{ch}_i = \mathsf{H}_i(\mathsf{crs}_{\mathsf{BARG}}, \vec{\mathbf{x}}, \mathbf{u}, \mathbf{d}, \mathsf{res}_1, \dots, \mathsf{res}_i)$ for $i \in [\mu]$. Then, from the soundness of Π_{pr}, there exists vectors $\vec{\mathbf{r}} = (\mathbf{a}_{\mathsf{r}}, \mathbf{b}_{\mathsf{r}}, \mathbf{c}_{\mathsf{r}}, \mathbf{y}_{\mathsf{r}}, \tilde{\mathbf{a}}_{\mathsf{r}}, \tilde{\mathbf{b}}_{\mathsf{r}}, \tilde{\mathbf{c}}_{\mathsf{r}}, \tilde{\mathbf{y}}_{\mathsf{r}})$ such that $(\mathcal{F}_{(\mathsf{ck}^*, \mathbf{u})}, \hat{\mathcal{F}}_{(\mathbf{P}, \mathbf{D}, \mathbf{d})}, \vec{\mathbf{r}}) \in R^{\mathsf{pr}}_{\sqrt{128/30} \cdot \sqrt{q}}$ with probability $1 - \mathsf{negl}(\lambda)$, where $\mathbf{P}$ and $\mathbf{D}$ are computed as in the construction using $\vec{\mathbf{x}}$ and $\mathbf{u}$. As described in [BS22, Section 6], it implies

- $\mathbf{u} = \mathsf{Com}(\mathsf{ck}^*, \mathbf{s})$, where $\mathbf{s} = (\mathbf{a}_{\mathsf{r}}^\top | \mathbf{b}_{\mathsf{r}}^\top | \mathbf{c}_{\mathsf{r}}^\top | \mathbf{y}_{\mathsf{r}}^\top)^\top \in= \mathcal{R}_2^L$ and
- $\mathbf{A}^{\otimes N} \mathbf{y}_b \circ \mathbf{B}^{\otimes N} \mathbf{y}_{\mathsf{b}} = \mathbf{C}^{\otimes N} \mathbf{y}_{\mathsf{b}}$, where $\mathbf{y}_{\mathsf{b}} \in \{0,1\}^{N(x+w)}$ is the coefficient vector of $\mathbf{y}_{\mathsf{r}}$.

Let $\mathbf{w}^* \leftarrow \mathsf{Extract}(\mathsf{td}, \vec{\mathbf{x}}, \pi)$. We claim that

$$\mathbf{A} \begin{pmatrix} \mathbf{x}_{i^*} \\ \mathbf{w}^* \end{pmatrix} \circ \mathbf{C} \begin{pmatrix} \mathbf{x}_{i^*} \\ \mathbf{w}^* \end{pmatrix} = \mathbf{C} \begin{pmatrix} \mathbf{x}_{i^*} \\ \mathbf{w}^* \end{pmatrix}.$$

To do this, we parse $\mathbf{y}_{\mathsf{b}} = (\mathbf{x}_1^\top | \mathbf{w}_1^\top | \cdots | \mathbf{x}_{i^*}^\top | \mathbf{w}_{i^*}^\top | \cdots | \mathbf{x}_N^\top | \mathbf{w}_N^\top)^\top$, where $\mathbf{w}_i \in \{0,1\}^w$ for all $i \in [N]$. From the above facts, we have

$$\mathbf{A} \begin{pmatrix} \mathbf{x}_i \\ \mathbf{w}_i \end{pmatrix} \circ \mathbf{C} \begin{pmatrix} \mathbf{x}_i \\ \mathbf{w}_i \end{pmatrix} = \mathbf{C} \begin{pmatrix} \mathbf{x}_i \\ \mathbf{w}_i \end{pmatrix}$$

for all $i \in [N]$. By somewhere extractability in trapdoor mode of Π_{SECom}, we have $\mathbf{w}^* = \mathbf{w}_{i^*}$. Therefore, Π_{BARG} is somewhere extractable in trapdoor mode. □

Succinctness: Finally, we show that the above BARG Π_{BARG} is succinct. The proof size of Π_{BARG} is bounded as follows:

$$\begin{aligned} |\pi| &= |\mathbf{u}| + |\mathbf{d}| + |\pi_{\mathsf{pr}}| \\ &\leq \underbrace{\ell \cdot \mathsf{poly}(\lambda, \log L)}_{|\mathbf{u}|} + \underbrace{\lambda}_{|\mathbf{d}|} + \underbrace{\mathsf{poly}(\lambda, \log 2dL)}_{|\pi_{\mathsf{pr}}|} \end{aligned}$$

$$\begin{aligned}&= \lceil w/d \rceil \cdot \mathsf{poly}(\lambda, \log N, \log S, \log x, \log w) + \lambda \\ &\quad + \mathsf{poly}(\lambda, \log N, \log S, \log x, \log w) \\ &\leq \mathsf{poly}(\lambda, \log N, \log S, \log x, w),\end{aligned}$$

where the second inequality follows from the succinctness of Π_{SECom} and Π_{pr} and $L = 3\lceil NS/d \rceil + N(\lceil x/d \rceil + \lceil w/d \rceil)$. Therefore, Π_{BARG} is succinct.

Putting it all together, we complete the proof of Theorem 3.8. □

If we instantiate Π_{SECom} with a simplified commitment scheme by Esgin et al. [ESZ22] and Π_{pr} with the ARG by Beullens and Seilar [BS22, Section 5.2], we immediately obtain the following corollary due to Lemmas 3.2 and 3.7.

Corollary 3.10. *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* $O(\log N) \cdot \mathsf{poly}(\lambda)$*.*

Remark 3.11 (Variable Number of Instances). Our BARG can be used to prove any $T \leq N$ instances (by ignoring components in the CRS). In this case, the proof size is unchanged.

4 Signature Aggregation from BARG

In this section, we first recall 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 the results of the instantiation in our BARG.

Building Blocks.

- A digital signature scheme $\Pi_{\mathsf{Sig}} = (\mathsf{KGen}_{\mathsf{Sig}}, \mathsf{Sign}_{\mathsf{Sig}}, \mathsf{Verify}_{\mathsf{Sig}})$ with message space $\mathcal{M}$ that satisfies correctness and unforgeability.
- A BARG $\Pi_{\mathsf{BARG}} = (\mathsf{Setup}_{\mathsf{BARG}}, \mathsf{Prove}_{\mathsf{BARG}}, \mathsf{Verify}_{\mathsf{BARG}})$ for R1CS that satisfies completeness, somewhere argument of knowledge, and succinctness and supports proving and verifying a variable number of instances.

Constructions. We can construct a bounded AggSig scheme $\Pi_{\mathsf{AggSig}} = (\mathsf{Setup}_{\mathsf{AggSig}}, \mathsf{KGen}_{\mathsf{AggSig}}, \mathsf{Sign}_{\mathsf{AggSig}}, \mathsf{Verify}_{\mathsf{AggSig}}, \mathsf{Aggregate}, \mathsf{AggVerify})$ 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 instance $(\mathbf{A}, \mathbf{B}, \mathbf{C}) \in (\{0,1\}^{S \times (x+w)})^3$. 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 $b = \mathsf{Verify}_{\mathsf{Sig}}(\mathsf{vk}, \mathsf{M}, \sigma)$.

- $\mathsf{Aggregate}(\mathsf{pp}, \{(\mathsf{vk}_i, \mathsf{M}_i, \sigma_i)\}_{i\in[T]}) \to \sigma_{\mathsf{agg}}$: On input of the public parameter $\mathsf{pp} = (1^\lambda, \mathsf{crs})$ 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.
- $\mathsf{AggVerify}(\mathsf{pp}, (\mathsf{vk}_1, \ldots, \mathsf{vk}_T), (\mathsf{M}_1, \ldots, \mathsf{M}_T), \sigma_{\mathsf{agg}}) \to b$: On input of the public parameter $\mathsf{pp} = (1^\lambda, \mathsf{crs})$, verification keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_T)$, messages $(\mathsf{M}_1, \ldots, \mathsf{M}_T) \in \mathcal{M}^T$, 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 $b = \mathsf{Verify}_{\mathsf{BARG}}(\mathsf{crs}, \vec{\mathbf{x}}, \sigma_{\mathsf{agg}})$.

Lemma 4.1 (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 (module-)lattice-based signature scheme (e.g., Dilithium [DKL+18]) and Π_{BARG} with our BARG in Section 3.3, we immediately obtain the result due to Corollary 3.10.

Corollary 4.2. *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 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

[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, 8, 12, 13, 14.)

[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 15.)

[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, 9, 10, 14.)

[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 14, 15.)

A Proof of Lemma 3.7

In this section, we prove Lemma 3.7. First, we provide the construction of the commitment scheme $\Pi_{\mathsf{ESZ}} = (\mathsf{KGen}, \mathsf{Com})$.

Construction.

- $\mathsf{KGen}(1^\lambda, 1^L, 1^\ell) \to \mathsf{ck}$:
  1. Set $n := n' + \ell$.
  2. Sample and output $\mathsf{ck} := \mathbf{U} \xleftarrow{\$} \mathcal{R}_q^{n \times L}$.
- $\mathsf{Com}(\mathsf{ck}, \mathbf{m} \in \mathcal{R}_2^L) \to \mathbf{c} := \mathbf{U}\mathbf{m} \in \mathcal{R}_q^n$:

Proof of Lemma 3.7. We show Π_{ESZ} is principal-relation compatible, somewhere extractable, and succinct.

Principal-Relation Compatibility: Since $\mathsf{Com}(\mathsf{ck}, \cdot)$ is a linear function, Π_{SECom} is clearly principal-relation compatible.

Somewhere Extractable: We start by defining the trapdoor key generation and extraction algorithms:

- $\mathsf{TrapKGen}(1^\lambda, 1^L, 1^\ell, i) \to (\mathsf{ck}^*, \mathsf{td})$:
  1. Sample $\mathbf{U}_1' \xleftarrow{\$} \mathcal{R}_q^{n' \times (i-1)}$, $\mathbf{U}_2' \xleftarrow{\$} \mathcal{R}_q^{n' \times \ell}$, and $\mathbf{U}_3' \xleftarrow{\$} \mathcal{R}_q^{n' \times (L-\ell-i+1)}$.
  2. Sample $\mathbf{Z} \xleftarrow{\$} \mathcal{R}_q^{\ell \times n'}$, $\mathbf{E}_1 \xleftarrow{\$} \mathcal{D}_\sigma^{\ell \times (i-1)}$, $\mathbf{E}_2 \xleftarrow{\$} \mathcal{D}_\sigma^{\ell \times \ell}$, and $\mathbf{E}_3 \xleftarrow{\$} \mathcal{D}_\sigma^{\ell \times (L-\ell-i+1)}$.
  3. Set
$$\mathbf{U}_1 := \begin{pmatrix} \mathbf{U}_1' \\ \mathbf{Z}\mathbf{U}_1' + \mathbf{E}_1 \end{pmatrix} \in \mathcal{R}_q^{(n'+\ell) \times (i-1)} = \mathcal{R}_q^{n \times (i-1)},$$
$$\mathbf{U}_2 := \begin{pmatrix} \mathbf{U}_2' \\ \mathbf{Z}\mathbf{U}_2' + \mathbf{E}_2 + \lfloor \frac{q}{2} \rfloor \cdot \mathbf{I}_\ell \end{pmatrix} \in \mathcal{R}_q^{n \times \ell},$$
$$\mathbf{U}_3 := \begin{pmatrix} \mathbf{U}_3' \\ \mathbf{Z}\mathbf{U}_3' + \mathbf{E}_3 \end{pmatrix} \in \mathcal{R}_q^{n \times (L-\ell-i+1)}.$$
  4. Set $\mathbf{U} := (\mathbf{U}_1 | \mathbf{U}_2 | \mathbf{U}_3) \in \mathcal{R}_q^{n \times L}$.
  5. Output $\mathsf{ck}^* := \mathbf{U}$ and $\mathsf{td} := \mathbf{Z}$.
- $\mathsf{Extract}(\mathsf{td}, \mathbf{c}) \to \mathbf{m}'$:
  1. Set $\mathbf{c}' := (-\mathbf{Z} | \mathbf{I}_\ell)\mathbf{c} \in \mathcal{R}_q^\ell$.
  2. Compute and output $\mathbf{m}' := \lfloor \frac{2}{q} \cdot \mathbf{c}' \rceil \in \mathcal{R}_2^\ell$.

  Here, $\lfloor \cdot \rceil$ is the rounding function.

The key indistinguishability directly follows the MLWE assumption, because

$$\mathbf{Z}(\mathbf{U}'_1|\mathbf{U}'_2|\mathbf{U}'_3) + (\mathbf{E}_1|\mathbf{E}_2|\mathbf{E}_3) \in \mathcal{R}_q^{\ell\times L}$$

is indistinguishable from the uniform random matrix over $\mathcal{R}_q^{\ell\times L}$ under the MLWE assumption.

Then, we show somewhere extractability in trapdoor mode. For any $\mathbf{m} = (m_1|\cdots|m_L)^\top \in \mathcal{R}_2^L$ and any $i \in [L]$, let $(\mathsf{ck}^*, \mathsf{td}) \xleftarrow{\$} \mathsf{TrapKGen}(1^\lambda, 1^L, 1^\ell, i)$ and $\mathbf{c} = \mathsf{Com}(\mathsf{ck}^*, \mathbf{m})$. Then, we have

$$\begin{aligned}\mathbf{c} &= \mathbf{U}\mathbf{m}\\ &= \mathbf{U}_1\mathbf{m}_1 + \mathbf{U}_2\mathbf{m}_2 + \mathbf{U}_3\mathbf{m}_3\\ &= \begin{pmatrix} \mathbf{U}'_1\mathbf{m}_1 + \mathbf{U}'_2\mathbf{m}_2 + \mathbf{U}'_3\mathbf{m}_3 \\ \mathbf{Z}(\mathbf{U}'_1\mathbf{m}_1 + \mathbf{U}'_2\mathbf{m}_2 + \mathbf{U}'_3\mathbf{m}_3) + (\mathbf{E}_1\mathbf{m}_1 + \mathbf{E}_2\mathbf{m}_2 + \mathbf{E}_3\mathbf{m}_3) + \lfloor \frac{q}{2} \rfloor \cdot \mathbf{m}_2 \end{pmatrix}\end{aligned}$$

where $\mathbf{U}$ is specified by the commitment key ck^* and

$$\mathbf{m}_1 = \begin{pmatrix} m_1 \\ \vdots \\ m_{i-1} \end{pmatrix}, \qquad \mathbf{m}_2 = \begin{pmatrix} m_i \\ \vdots \\ m_{i+\ell} \end{pmatrix}, \qquad \mathbf{m}_3 = \begin{pmatrix} m_{i+\ell+1} \\ \vdots \\ m_L \end{pmatrix}.$$

In the $\mathsf{Extract}$ algorithm, we compute

$$\mathbf{c}' = (-\mathbf{Z}|\mathbf{I}_\ell)\mathbf{c} = (\mathbf{E}_1\mathbf{m}_1 + \mathbf{E}_2\mathbf{m}_2 + \mathbf{E}_3\mathbf{m}_3) + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathbf{m}_2$$

and

$$\mathbf{m}' = \lfloor \frac{2}{q} \cdot \mathbf{c}' \rceil = \lfloor \frac{2}{q} \cdot (\mathbf{E}_1\mathbf{m}_1 + \mathbf{E}_2\mathbf{m}_2 + \mathbf{E}_3\mathbf{m}_3) + \frac{2}{q} \cdot \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathbf{m}_2 \rceil = \mathbf{m}_2,$$

where the last equality is derived by setting the parameters appropriately, e.g., $q/2 > O(\sigma d L)$. Hence, we have

$$\mathbf{m}' = \mathbf{m}_2 = \begin{pmatrix} m_i \\ \vdots \\ m_{i+\ell} \end{pmatrix}.$$

Succinctness: The size of the commitment $\mathbf{c}$ is bounded as follows:

$$|\mathbf{c}| = nd\log q = (n' + \ell) \cdot O(\lambda) \cdot O(\log\sigma + \log d + \log L) < \ell \cdot \mathsf{poly}(\lambda, \log L).$$

Therefore, Π_{SECom} is succinct.

Putting it all together, we complete the proof of Lemma 3.7. □

Contents