EPiC Series in Computing

Volume 111, 2026, Pages 79–94

Proceedings of The 21st Asia Joint Conference on Information Security

Ring Signatures Based on Subfield Bilinear Collision Problem via MPC-in-the-Head*

Hiroaki Anada[1], Masayuki Fukumitsu[2], and Shingo Hasegawa[3]

[1] Meiji Gakuin University, Yokohama, Japan
hiroaki.anada@mi.meijigakuin.ac.jp
[2] University of Nagasaki, Nagasaki, Japan
fukumitsu@sun.ac.jp
[3] Fukushima University, Fukushima, Japan
hasegawa@sss.fukushima-u.ac.jp

Abstract

We propose a ring signature scheme whose unforgeability is based on the assumption that the subfield bilinear collision (SBC) problem is hard. The SBC problem was introduced at CRYPTO 2024, which is related to a bilinear form over a finite field with a symmetric solution in its subfield. It is considered to be hard even against the power of quantum computers, and hence our ring signature scheme is expected to be a post-quantum one. The size of our ring signature is linear to the size of a ring, but shorter than a naïve approach that uses the OR-proof technique. Concretely, when the size of a ring is 32, 128-bit quantum security is attained with the signature length being about 68 KB.

1 Introduction

Ring signatures [1, 19] are an extension of digital signatures [5, 9, 18], where a signer generates a signature on a message with respect to a set of public keys called a ring including its own. In the special case that the ring consists of only its own public key, a ring signature coincides with an ordinary digital signature. There are two requirements for a ring signature scheme. One is unforgeability of a ring signature, and the other is anonymity, where the latter means a property that an actual signer cannot be identified in the ring. One of the recognized definitions of a ring signature scheme is due to the work of Bender-Katz-Morselli [1]. As for concrete constructions, several computational assumptions have been introduced in the corresponding mathematical structures.

In this paper, extending the signature scheme of Huth and Joux [11], we propose a ring signature scheme, which is unforgeable based on the subfield bilinear collision (SBC) problem. The SBC problem is concerned with a finite field and its subfield, which is informally stated as follows.

*This work was supported by JSPS KAKENHI Grant Numbers JP23K11105 and JP23K11106.

Suppose that two bilinear forms $\vec{u} \cdot \vec{x}$ and $\vec{v} \cdot \vec{y}$ over a fixed finite field $\mathbb{F}_{q^k}$ of order a prime-power q^k are given, where $\vec{u}$, $\vec{v}$, $\vec{x}$ and $\vec{y}$ are vectors having n entries and '$\cdot$' means inner product. Fix the values of the entries of $\vec{u}, \vec{v} \in (\mathbb{F}_{q^k})^n$. An instance of the SBC problem is to find a solution $\vec{x}, \vec{y}$ s.t.

$$(\vec{u} \cdot \vec{x})(\vec{v} \cdot \vec{y}) = (\vec{u} \cdot \vec{y})(\vec{v} \cdot \vec{x}),\ \vec{x}, \vec{y} \in (\mathbb{F}_q)^n$$
(i.e. the entries should be in the subfield $\mathbb{F}_q$).

According to the analysis of Huth-Joux [11], there are no polynomial-time algorithms including quantum ones solving the SBC problem, so far. Thus, the SBC problem has the possibility of being computationally hard even against the power of quantum computers.

1.1 Our Contribution

In this paper, we propose a Σ-protocol [2] that is an extension of the original Σ-protocol of Huth-Joux [11]. By applying the Fiat-Shamir transformation [6], we obtain a non-interactive zero-knowledge (NIZK) proof system in the random oracle model, and then we use the NIZK proof system to construct our ring signature scheme. More precisely, the Σ-protocol of Huth-Joux [11] is in the paradigm of "MPC-in-the-Head" that was initiated by the seminal work of Ishai-Kushilevitz-Ostrovsky-Sahai [13, 14], and we will also work in it. One might think that the OR-proof technique [4] is applicable to the original Σ-protocol to obtain a ring signature scheme. Actually, the asymptotic behavior of the length of a signature of our ring signature is linear in the ring size l, which is the same as that of the (naïve) generic approach. However, though the first term of a ring signature would be the same, the second term behaves better in our case (See Section 6.1). As a result, the signature length of our ring signature is about 68 KB when the size l of a ring is 32 and the other parameters are considered to attain 128-bit quantum security.

There are two ideas in the construction of our ring signature scheme. One is to take the product of the formulas that appear in the construction of the normal signature scheme of Huth-Joux [11]. Here the product is over all the entities of the public keys in the ring. Due to the feature of each formula that is a polynomial, the degree is one when a witness of the statement is applied to the formula, and the degree is two otherwise. Then the prover can generate an OR-proof in the sense that at least one entity in the ring knows a witness. The other idea is to generate a large GGM tree that merges all the trees each of which corresponds to an entity in the ring. Due to the logarithmic structure of the tree, the bit-length of the above second term behaves better than the generic approach (See Section 6.1).

2 Preliminaries

In this section, we prepare for notations and needed notions. $\mathbb{N}$ denotes the set of all natural numbers, and $[n]$ denotes the set $\{1, \ldots, n\} \subset \mathbb{N}$. $\mathbb{F}$ denotes a general finite field and $\mathbb{F}_q$ denotes the finite field having q (a prime-power) elements. $\lambda \in \mathbb{N}$ denotes the security parameter. pp denotes a set of public parameters, whose elements are accessible and referred to publicly. We often omit the notation pp. For a finite set S, $x \in_U S$ denotes a uniform random sampling, and $|S|$ denotes the cardinality of S. For an element a of S, $|a|$ denotes the bit-length of a. For a finite set S and a natural number n, $\vec{x} \in_U S^n$ denotes a sampling of a vector $\vec{x}$ whose entries are uniformly random and pairwise independent. Moreover, for a string r, $x \overset{r}{\in} S$ denotes a generation of a pseudorandom element in S using r as the seed. $x, y \overset{r}{\in} S$ denotes a generation

of pseudorandom elements using r as the seed in a *stateful* manner; i.e. *reproducibility* of x and y in this order is assured. For strings s_1 and s_2, a predicate $(s_1 \stackrel{?}{=} s_2)$ returns a boolean 1 if $s_1 = s_2$, and otherwise, 0. "Probabilistic polynomial-time" and "deterministic polynomial-time" are abbreviated as PPT and DPT, respectively. A function F of λ is said to be negligible in λ if, for any given polynomial poly, there exists a sufficiently large number $\lambda_0 \in \mathbb{N}$ s.t. $\lambda > \lambda_0$ implies $|F(\lambda)| < 1/|\mathrm{poly}(\lambda)|$. A function P whose values are in the interval $[0, 1]$ is said to be overwhelming if $1 - P(\lambda)$ is negligible in λ.

2.1 Ring Signatures [1]

Let l be a natural number bounded by a fixed polynomial in λ. A scheme RS of ring signatures consists of three PPT algorithms, KG, Sign and Vrfy: $\mathsf{RS} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Vrfy})$.
• $\mathsf{KG}(1^\lambda) \to (pk, sk)$. This PPT algorithm takes as input the security parameter λ, and returns a public key pk and the corresponding secret key sk.
• $\mathsf{Sign}(R, s, sk_s, m) \to \sigma$. This PPT algorithm takes as input a ring $R = (pk_1, \ldots, pk_l)$, an identifier s of a signer, a secret key sk_s whose corresponding public key $pk_s \in R$, and a message m, and returns a ring signature σ.
• $\mathsf{Vrfy}(R, m, \sigma) \to d$. This DPT algorithm takes as input a ring R, a message m and a ring signature σ, and returns a boolean decision d.
Correctness. We consider the following experimental algorithm for the scheme RS, where $\mathbf{A}$ denotes any given algorithm.

$\mathsf{Expr}^{\mathrm{corr}}_{\mathsf{RS},\mathbf{A}}(1^\lambda)$
$l \leftarrow \mathbf{A}(1^\lambda)$, [For $h \in [l]$: $(pk_h, sk_h) \leftarrow \mathsf{KG}(1^\lambda)$]; $R := (pk_h)_{h\in[l]}$
$(s, m) \leftarrow \mathbf{A}(R)$, If $pk_s \notin R$ then return 0 else $\sigma \leftarrow \mathsf{Sign}(R, s, sk_s, m)$
If $\mathsf{Vrfy}(R, m, \sigma) = 0$ then return 1 else return 0

We set the advantage of $\mathbf{A}$ as $\mathbf{Adv}^{\mathrm{corr}}_{\mathsf{RS},\mathbf{A}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{Expr}^{\mathrm{corr}}_{\mathsf{RS},\mathbf{A}}(1^\lambda) = 1]$. If for any given (possibly unbounded) algorithm $\mathbf{A}$ the advantage $\mathbf{Adv}^{\mathrm{corr}}_{\mathsf{RS},\mathbf{A}}(\lambda)$ is 0, then we say that RS has correctness.
Unforgeability. In accordance with "Unforgeability w.r.t. insider corruption" ("Definition 8" in [1]), we consider the following experimental algorithm for the scheme RS, where $\mathbf{A}$ denotes any given algorithm.

$\mathsf{Expr}^{\mathrm{uf}}_{\mathsf{RS},\mathbf{A}}(1^\lambda)$
$l \leftarrow \mathbf{A}(1^\lambda)$, [For $h \in [l]$: $(pk_h, sk_h) \leftarrow \mathsf{KG}(1^\lambda)$]; $R := (pk_h)_{h\in[l]}$
$(R^*, m^*, \sigma^*) \leftarrow \mathbf{A}^{\mathsf{SignO},\mathsf{CrptO}}(R)$
If $\mathsf{Vrfy}(R^*, m^*, \sigma^*) = 1$ and $R^* \subset R \backslash \mathrm{Crpt}$ and $\mathbf{A}$ never queried to $\mathsf{SignO}(R^*, \star, m^*)$
then return 1 else return 0

Here SignO is the signing oracle which, on input (R, s, m) as a query, replies a ring signature σ output by $\mathsf{Sign}(R, s, sk_s, m)$ when $s \in R$, and otherwise, $\perp$. $\star$ means that any identifier of a signer having a public key can be applied at the position. CrptO is the corrupt oracle which, on input s as a query, replies a secret key sk_s. We denote by Crpt the set of public keys $\{pk_s\}$ whose corresponding secret keys $\{sk_s\}$ is corrupted.

We set the advantage of $\mathbf{A}$ as $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{RS},\mathbf{A}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{Expr}^{\mathrm{uf}}_{\mathsf{RS},\mathbf{A}}(1^\lambda) = 1]$. If for any given PPT algorithm $\mathbf{A}$ the advantage $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{RS},\mathbf{A}}(\lambda)$ is negligible in λ, then we say that RS has unforgeability.

Anonymity. We consider the following experimental algorithm for the scheme RS, where $\mathbf{A}$ denotes any given algorithm.

$\mathsf{Expr}^{\mathrm{anon}}_{\mathsf{RS},\mathbf{A}}(1^\lambda)$
$l \leftarrow \mathbf{A}(1^\lambda)$, [For $h \in [l]$: $(pk_h, sk_h) \leftarrow \mathsf{KG}(1^\lambda)$]; $S := (pk_h)_{h\in[l]}$
$((R, s_0, s_1, m), \mathrm{St}) \leftarrow \mathbf{A}^{\mathsf{SignO}}((pk_h, sk_h)_{h\in[l]})$
If $\{pk_{s_0}, pk_{s_1}\} \subset S$: then $R' := R \cup \{pk_{s_0}, pk_{s_1}\}$, $b \in_U \{0,1\}$, $\sigma \leftarrow \mathsf{Sign}(R', s_b, sk_{s_b}, m)$
else return 0
$b' \leftarrow \mathbf{A}(\mathrm{St}, \sigma)$; if $b = b'$ then return 1 else return 0

We set the advantage of $\mathbf{A}$ as $\mathbf{Adv}^{\mathrm{anon}}_{\mathsf{RS},\mathbf{A}}(\lambda) \stackrel{\mathrm{def}}{=} |\Pr[\mathsf{Expr}^{\mathrm{anon}}_{\mathsf{RS},\mathbf{A}}(1^\lambda) = 1] - (1/2)|$. If for any given PPT algorithm $\mathbf{A}$ the advantage $\mathbf{Adv}^{\mathrm{anon}}_{\mathsf{RS},\mathbf{A}}(\lambda)$ is negligible in λ, we say that RS has anonymity.

2.2 Additive Secret Sharing ([15] etc.)

Secret sharing in this paper means the N-out-of-N additive secret sharing over a fixed finite field $\mathbb{F}$. That is, suppose that $x \in \mathbb{F}$ is any given secret. Then we generate the set of shares, denoted by $[\![x]\!] \in (\mathbb{F})^N$, of x by a PPT algorithm Share_N, as follows.

$\mathsf{Share}_N(x)$:
[For $i \in [N-1]$: $x^{[\![i]\!]} \in_U \mathbb{F}$]; $x^{[\![N]\!]} := x - \sum_{i\in[N-1]} x^{[\![i]\!]}$
Return $[\![x]\!] := (x^{[\![i]\!]})_{i\in[N]}$

We call each $x^{[\![i]\!]}$ a share of x. Here we have a linear constraint $x = \sum_{i\in[N]} x^{[\![i]\!]}$.

In addition, we use the following the N-out-of-N additive offset secret sharing by using Share_{N+1}. That is, suppose that $x \in \mathbb{F}$ is any given secret. Then we generate $(\delta_x, [\![x]\!]) \in (\mathbb{F})^{N+1}$ by a PPT algorithm $\mathsf{OffShare}_N$, as follows.

$\mathsf{OffShare}_N(x)$:
$([\![x]\!], \delta_x) \leftarrow_R \mathsf{Share}_{N+1}(x)$; [For $i \in [N]$: $R_x^{[\![i]\!]} := x^{[\![i]\!]}$]; $[\![R_x]\!] := (R_x^{[\![i]\!]})_{i\in[N]}$
Return $(\delta_x; [\![R_x]\!])$

Here we have a linear constraint $\delta_x = x - \sum_{i\in[N]} R_x^{[\![i]\!]}$. The purpose of the offset secret sharing is to make every share $x^{[\![i]\!]}$ be independent and uniformly random for $i = 1, \ldots, N$. On the other hand, δ_x is called an offset, which is given to all parties.

We extend the input and output of Share to a natural vector-form as: When an input to Share is a vector $\vec{x} = (x_i)_{i\in[N]} \in (\mathbb{F})^n$, Share is applied to every entry and to obtain an output of a share-vector $(\mathsf{Share}(x_i))_{i\in[N]}$ which is denoted by $\mathsf{Share}(\vec{x})$.

2.3 MPC-in-the-Head Σ-Protocol [13, 14]

We tailor the description below for our case. Let N be a number that is bounded by a polynomial in λ. Suppose that an MPC protocol (see for example [3, 7]) in which N parties $P_1, \ldots, P_N$ compute the output of a function f of a secret input x. The output is 1 or 0 according to "accept" or "reject", respectively. Each party P_i, $i \in [N]$ is given a share $x^{[\![i]\!]}$ of x. Let

View$_i$, $i \in [N]$ denote the view of each party P_i, $i \in [N]$, respectively. We require that any set of views of $N-1$ parties do not reveal any information about x, which is called $(N-1)$-privacy. This type of MPC protocol can be transformed into a Σ-protocol ([2]), between a prover P and a verifier V, of an honest-verifier zero-knowledge proof of knowledge of the secret x for which $f(x) = 1$. Informally,

1. P generates a set of random N shares $[\![x]\!]$ of x.
2. P emulates ("in the head") all N parties of the MPC protocol by providing $x^{[\![i]\!]}$ to P_i, $i \in [N]$.
3. P sends commitments to the views of parties to V.
4. V chooses $\hat{i} \in_U [N]$ and sends it to P.
5. P sends all the views and commitment keys except $\hat{i}$, to V.
6. V checks except $\hat{i}$ that (1) Validity of all the commitments using the keys; (2) Each party P_i actually returns 1 (accept); (3) Incoming and outgoing messages included in the views are all consistent. If the three checks pass, then V returns 1, and otherwise, 0.

If the MPC protocol is $(N-1)$-private (see [14] for the definition), then the above Σ-protocol does not leak any information about the secret x. Since the $(N-1)$ opened views are selected uniformly at random, a cheating prover can manipulate at most one party that will not be opened with probability $1/N$. Thus, the Σ-protocol has soundness error $1/N$.

Along the design strategy in the previous work by Huth and Joux [11], we will deviate the above original protocol so that it is suitable to the SBC problem.

2.4 Puncturable Pseudorandom Functions [11]

We tailor the description below for our case. That is, the domain is $[N]$ where N is bounded by a polynomial in λ. For $\lambda \in \mathbb{N}$, let $\mathcal{RF}^\lambda = \{f : [N] \to \{0,1\}^\lambda\}$ be the set of all maps. Let $\mathcal{PRF}^\lambda = \{\mathsf{PRF}_k^\lambda \mid k \in \{0,1\}^\lambda\}$ be a set of maps, where PRF_k^λ brings, using a key $k \in \{0,1\}^\lambda$, a number in $[N]$ to a λ-bit string. When $\mathcal{PRF}^\lambda$ satisfies the following two requirements, $\mathcal{PRF}^\lambda$ is called a family of pseudorandom functions (PRFs). (Efficiency) For any k, PRF_k^λ is computable in polynomial-time in λ for any input. (PR: Pseudorandomness) For any given PPT algorithm $\mathbf{A}$, it holds that $\mathbf{Adv}^{\mathrm{rand}}_{\mathsf{PRF},\mathbf{A}}(\lambda) := |\Pr[\mathbf{A}^{\mathsf{PRF}_k^\lambda(\cdot)}(1^\lambda) = 1 \mid k \in_U \{0,1\}^\lambda] - \Pr[\mathbf{A}^{f(\cdot)}(1^\lambda) = 1 \mid f \in_U \mathcal{RF}^\lambda]|$ is negligible in λ. Further, consider the following property. (Puncturableness) For each k and index $\hat{i}$ there exists a string called a punctured key $k_{\hat{i}}$ and a DPT algorithm F s.t. for each $j \in [N]\backslash\{\hat{i}\}$ $\mathsf{F}(k_{\hat{i}}, j) = \mathsf{PRF}_k^\lambda(j)$; and the punctured key $k_{\hat{i}}$ does not leak any information about $\mathsf{PRF}_k^\lambda(\hat{i})$.

A PRF and also a puncturable PRF can be constructed by using a binary tree called a GGM tree [8]. In our case the depth of the tree is $\lceil \log_2(N) \rceil$. The root node is labeled with the key k. The remaining nodes are labeled by applying a pseudorandom number generator (PRG) that takes as input a λ-bit string and returns a 2λ-bit string. That is, PRG is applied to the label of a node to obtain a 2λ-bit string, and then the former λ-bit is labeled to the left child, while the latter λ-bit is labeled to the right child. Then, for an input number $i \in [N]$, the binary representation of i indicates the left child or the right sequentially from MSB in the top down manner from the root node to a leaf. The sequence of the labels on the path is the value of $\mathsf{PRF}_k^\lambda(i)$. To reveal all the labels on the N leaves *except* one label on a leaf that corresponds to $\hat{i}$, the idea is to reveal all the labels of the *siblings* of the nodes that are on the path of $\hat{i}$.

A useful feature of this construction is that all the leaf-labels except $\hat{i}$ can be reconstructed by storing $\lceil \log_2(N) \rceil$ labels instead of $(N-1)$ labels. For the later sections, we denote by PncPRF_k the puncturable PRF of this GGM construction (the superscript $^\lambda$ is omitted).

2.5 Hash Functions [20]

For $\lambda \in \mathbb{N}$, let $\mathcal{H} = \{\mathsf{H}^\lambda : \{0,1\}^* \to \{0,1\}^\lambda\}$ be a set of functions, where H^λ maps a string in $\{0,1\}^*$ to a λ-bit string. Then $\mathcal{H} = \{\mathsf{H}^\lambda\}_{\lambda\in\mathbb{N}}$ is a function family. When $\mathcal{H}$ satisfies the following three requirements, $\mathcal{H}$ is called a family of hash functions. (Efficiency) H^λ is computable in polynomial-time in λ for any input x. (HF1: Preimage resistance) For any given PPT algorithm $\mathbf{A}$, $\mathbf{Adv}^{\mathrm{preim}}_{\mathsf{H},\mathbf{A}}(\lambda) := \Pr[\mathsf{H}^\lambda(x) = y \mid y \in_U \{0,1\}^\lambda;\ x \leftarrow \mathbf{A}(y)]$ is negligible in λ. (HF2: Collision resistance) For any given PPT algorithm $\mathbf{A}$, $\mathbf{Adv}^{\mathrm{coll}}_{\mathsf{H},\mathbf{A}}(\lambda) := \Pr[x_1 \neq x_2 \wedge \mathsf{H}^\lambda(x_1) = \mathsf{H}^\lambda(x_2) \mid (x_1,x_2) \leftarrow \mathbf{A}]$ is negligible in λ. Hereafter we omit the superscript $^\lambda$ of H^λ, as H.

3 On Subfield Bilinear Collision Problem

In this section, in accordance with the previous work [11] that introduced the subfield bilinear collision problem (the SBC problem, for short), we explain the formulation of the problem. Then we discuss the hardness of the SBC problem on which we will rely.

3.1 Problem Instance, Solution and Their Generation

According to the setting of the previous work [11], let $q = p^e$ be a prime-power, k be a natural number. Hereafter, q, k and n are the parameters of our problem. Then an instance of our problem is stated as follows.

Problem Instance. Two vectors $\vec{u}, \vec{v} \in (\mathbb{F}_{q^k})^n$ s.t. $\vec{u}$ and $\vec{v}$ are linearly independent over $\mathbb{F}_q$.

Solution. Two vectors $\vec{x}, \vec{y} \in (\mathbb{F}_q)^n$ s.t. $\vec{x}$ and $\vec{y}$ are of the form $\vec{x} := (\vec{x'}, 1, 0), \vec{y} := (\vec{y'}, 0, 1)$ and satisfy the following equation:

$$(\vec{u}\cdot\vec{x})(\vec{v}\cdot\vec{y}) = (\vec{u}\cdot\vec{y})(\vec{v}\cdot\vec{x}). \tag{1}$$

According to the previous work [11], we call the set of the above instances as the normalized subfield bilinear collision problem, or simply, the SBC problem. Here an instance and its solution are generated in the following way.

Generating Solution and Instance ([11]).

1. Sample $\vec{x'}, \vec{y'} \in_U (\mathbb{F}_q)^{n-2}$ and set $\vec{x} := (\vec{x'}, 1, 0), \vec{y} := (\vec{y'}, 0, 1)$.
2. Sample entries of $\vec{u}, \vec{v}$ as $u_1, \ldots, u_n, v_1, \ldots, v_{n-1} \in_U \mathbb{F}_{q^k}$.
3. Put the entry v_n as

$$v_n := \frac{(\sum_{i=1}^{n-1} v_i x_i)(\vec{u}\cdot\vec{y}) - (\vec{u}\cdot\vec{x})(\sum_{i=1}^{n-1} v_i y_i)}{y_n(\vec{u}\cdot\vec{x})}. \tag{2}$$

When the denominator of (2) is 0, sample u_{n-1} again. Also, when $\vec{u}$ and $\vec{v}$ are linearly dependent over $\mathbb{F}_q$, sample u_{n-1} again.

3.2 Hardness of Subfield Bilinear Collision Problem

When an instance of the SBC problem is in a special case, it is captured as an instance of the discrete logarithm problem [11]. Therefore, if the Shor's algorithm [21] is applied, then it would be vulnerable. As for the general case, an algebraic attack using the Gröbner basis algorithm can be applied to find a solution of an instance of the SBC problem [11]. To escape the special case and to be hard against the algebraic attack, a parameter setting is recommended [11]; that is, n should be approximately a half of k ($n \approx \frac{k}{2}$). Also, from the view point of exhaustive search using the Grover's algorithm [10], we set the security parameter λ to be a half of the bit-length of q^k: $\lambda \stackrel{\text{def}}{=} \frac{1}{2}|q^k|$. For example, (q, n, k) can be $(2, 130, 257)$ for the 128-bit ($\approx \frac{1}{2}|q^k|$) security against quantum computers.

4 Our Σ-Protocol

In this section, we describe a Σ-protocol that is used to construct our ring signature scheme in the next section. Our protocol is an extension of the Σ-protocol of Huth-Joux [11] to a Σ-protocol that fits to ring signatures, which is in the design principle of MPC-in-the-Head ([13, 14]).

4.1 Details of Our Protocol

The definition of our Σ-protocol between a prover P and a verifier V, which are PPT algorithms, is in accordance with that of the previous work of Cramer [2]. The parameters q, k, n are supposed to be public and fixed. Let l be a natural number that is the size of a ring R (i.e. $l = |R|$), which is bounded by a polynomial in λ. A statement and its witness which our Σ-protocol will prove are the following ones.

Statement. $[(\vec{u}_1, \vec{v}_1)$ or $\ldots$ or $(\vec{u}_l, \vec{v}_l)]$, where $(\vec{u}_h, \vec{v}_h) \in (\mathbb{F}_{q^k})^n \times (\mathbb{F}_{q^k})^n, h = 1, \ldots, l$ is an instance of the SBC problem (see (1)).

Witness. For some $s \in [l]$, a solution $(\vec{x}_s, \vec{y}_s) \in (\mathbb{F}_q)^n \times (\mathbb{F}_q)^n$ of an instance $(\vec{u}_s, \vec{v}_s)$.

Thus, our interactive protocol is for the following relation R.

$$\mathcal{R} = \{(\ [(\vec{u}_1, \vec{v}_1) \text{ or } \ldots \text{ or } (\vec{u}_l, \vec{v}_l)],\ (\vec{x}_s, \vec{y}_s)\ ) \mid \exists s \in [l] \text{ s.t. } (\vec{x}_s, \vec{y}_s) \text{ is a solution of } (\vec{u}_s, \vec{v}_s)\}. \tag{3}$$

Following the custom of notation, the relation $\mathcal{R}$ can be a boolean-valued function.

$$\mathcal{R}([(\vec{u}_1, \vec{v}_1) \text{ or } \ldots \text{ or } (\vec{u}_l, \vec{v}_l)],\ (\vec{x}_s, \vec{y}_s)) = \begin{cases} 1 \text{ if } ([(\vec{u}_1, \vec{v}_1) \text{ or } \ldots \text{ or } (\vec{u}_l, \vec{v}_l)],\ (\vec{x}_s, \vec{y}_s)) \in \mathcal{R} \\ 0 \text{ otherwise} \end{cases}$$

- $\mathsf{P}([(\vec{u}_1, \vec{v}_1) \text{ or } \ldots \text{ or } (\vec{u}_l, \vec{v}_l)],\ (\vec{x}_s, \vec{y}_s)) \to (\mathrm{msg}_1, \mathrm{St}_\mathsf{P})$.

The prover P takes as input a statement $[(\vec{u}_1, \vec{v}_1)$ or $\ldots$ or $(\vec{u}_l, \vec{v}_l)]$ and its witness $(\vec{x}_s, \vec{y}_s)$ in the relation $\mathcal{R}$. Since there are no witnesses for each $h \in [l]$ except s, P samples $\vec{x}_h, \vec{y}_h$ independently and uniformly at random from the subfield.

$$\text{For } h \in [l], h \neq s: \ \vec{x}_h, \vec{y}_h \in_U (\mathbb{F}_q)^n. \tag{4}$$

First of all, P samples a λ-bit key k of our puncturable pseudorandom function $\mathsf{PncPRF}.(\cdot)$ uniformly at random.

$$k \in_U \{0,1\}^\lambda. \tag{5}$$

Later, a punctured key $k_{\hat{i}}$ where $\hat{i}$ is chosen by V works for reproducibility except the index $\hat{i}$. Then for each $i \in [N]$, P computes a seed $r_i \leftarrow \mathsf{PncPRF}_k(i)$. Now P generates the input of each party P_i as follows.

$$\begin{aligned}&\text{For } i \in [N]:\\ &\quad \text{For } j = 0, \ldots, 2l-1: \ R_{A_j}^{[\![i]\!]} \xleftarrow{r_i} \mathbb{F}_{q^k} &(6)\\ &\quad \text{For } h \in [l]: \ \alpha_{h,1}^{[\![i]\!]}, \alpha_{h,2}^{[\![i]\!]}, \beta_{h,1}^{[\![i]\!]}, \beta_{h,2}^{[\![i]\!]} \xleftarrow{r_i} \mathbb{F}_{q^k}, \vec{R}_{x_h}^{[\![i]\!]}, \ \vec{R}_{y_h}^{[\![i]\!]} \xleftarrow{r_i} (\mathbb{F}_q)^n &(7)\end{aligned}$$

Then P executes the following computation, which yields an equivalent distribution to that obtained by executing the additive secret sharing Share.

$$\begin{aligned}&\text{For } h \in [l]:\\ &\alpha_{h,1} := \sum_{i\in[N]} \alpha_{h,1}^{[\![i]\!]}, \ \alpha_{h,2} := \sum_{i\in[N]} \alpha_{h,2}^{[\![i]\!]}, &(8)\\ &\beta_{h,1} := \sum_{i\in[N]} \beta_{h,1}^{[\![i]\!]}, \ \beta_{h,2} := \sum_{i\in[N]} \beta_{h,2}^{[\![i]\!]}. &(9)\end{aligned}$$

Then P generates a formula $F^{\vec{x}_h,\vec{y}_h,\vec{u}_h,\vec{v}_h}_{\alpha_{h,1},\alpha_{h,2},\beta_{h,1},\beta_{h,2}}(t)$ of a variable t, as follows.

$$\begin{aligned}F^{\vec{x}_h,\vec{y}_h,\vec{u}_h,\vec{v}_h}_{\alpha_{h,1},\alpha_{h,2},\beta_{h,1},\beta_{h,2}}(t) \stackrel{\text{def}}{=} &(\alpha_{h,1} + t(\vec{u}_h \cdot \vec{x}_h))(\beta_{h,1} + t(\vec{v}_h \cdot \vec{y}_h))\\ &-(\alpha_{h,2} + t(\vec{v}_h \cdot \vec{x}_h))(\beta_{h,2} + t(\vec{u}_h \cdot \vec{y}_h)). \qquad (10)\end{aligned}$$

Then P expands the right-hand side of the formula (10) about t and obtains

$$\begin{aligned}F^{\vec{x}_h,\vec{y}_h,\vec{u}_h,\vec{v}_h}_{\alpha_{h,1},\alpha_{h,2},\beta_{h,1},\beta_{h,2}}(t) = &\alpha_{h,1}\beta_{h,1} - \alpha_{h,2}\beta_{h,2}\\ &+\big(\alpha_{h,1}(\vec{v}_h \cdot \vec{y}_h) + \beta_{h,1}(\vec{u}_h \cdot \vec{x}_h) - \alpha_{h,2}(\vec{u}_h \cdot \vec{y}_h) - \beta_{h,2}(\vec{v}_h \cdot \vec{x}_h)\big)t\\ &+\big((\vec{u}_h \cdot \vec{x}_h)(\vec{v}_h \cdot \vec{y}_h) - (\vec{v}_h \cdot \vec{x}_h)(\vec{u}_h \cdot \vec{y}_h)\big)t^2. \qquad (11)\end{aligned}$$

Then P generates the product of the formulas (11) for $h \in [l]$ and names it $F(t)$.

$$F(t) \stackrel{\text{def}}{=} \prod_{h\in[l]} F^{\vec{x}_h,\vec{y}_h,\vec{u}_h,\vec{v}_h}_{\alpha_{h,1},\alpha_{h,2},\beta_{h,1},\beta_{h,2}}(t). \qquad (12)$$

P expands the right-hand side of the formula (12) about t and obtains

$$F(t) = A_0 + A_1 t + A_2 t^2 + \cdots + A_{2l} t^{2l}, \ A_i \in \mathbb{F}_{q^k}, i \in \{0, \ldots, 2l\}. \qquad (13)$$

Here, since a witness in the relation (3) is applied to (12), the coefficient of the term of degree 2 of (11) is actually 0 for $h = s$. Therefore, the coefficient of the term of degree $2l$ vanishes. Thus, we can write $F(t)$ as;

$$F(t) = A_0 + A_1 t + A_2 t^2 + \cdots + A_{2l-1} t^{2l-1}. \qquad (14)$$

Then P executes the following computation, which yields an equivalent distribution to that obtained by executing the additive offset secret sharing $\mathsf{OffShare}$.

$$\begin{aligned}&\text{For } j = 0 \text{ to } 2l-1: \ \delta_{A_j} := A_j - \sum_{i\in[N]} R_{A_j}^{[\![i]\!]} &(15)\\ &\text{For } h \in [l]: \ \vec{\delta}_{x_h} := \vec{x}_h - \sum_{i\in[N]} \vec{R}_{x_h}^{[\![i]\!]}, \ \vec{\delta}_{y_h} := \vec{y}_h - \sum_{i\in[N]} \vec{R}_{y_h}^{[\![i]\!]} &(16)\end{aligned}$$

Now P proceeds according to the design principle of MPC-in-the-Head [13, 14]. Below we describe a multi-party computation (MPC) which the prover P emulates (in its head). We denote the parties of our MPC by $P_1, \ldots, P_N$. The functionality of our MPC is to compute a boolean value of the function

$$\mathcal{R}([(\vec{u}_1, \vec{v}_1) \text{ or } \ \ldots \ \text{ or } (\vec{u}_l, \vec{v}_l)],\ (\vec{x}_s, \vec{y}_s)) \tag{17}$$

The outline of the procedure executed by each party P_i consists of the following four steps.

1. Each P_i generates a randomness $t_0^{[\![i]\!]} \in \mathbb{F}_{q^k}$ and broadcasts it.
2. Each P_i evaluates the value of the formula $F(t)$ of (12) at $t_0 := \sum_{i\in[N]} t_0^{[\![i]\!]}$.
3. Each P_i evaluates the value of the right-hand side of the formula $F(t)$ of (14) at t_0.
4. Each P_i returns 1 if the both evaluated values are equal, and otherwise, 0.

The details of the procedure executed by each party P_i are as follows, from the view point of emulation by the prover P of our Σ-protocol. First, P gives the following Input_i to each party $P_i,\ i \in [N]$ of MPC (emulated in its head).

$$\mathrm{Input}_i := \Big((\vec{R}_{x_h}^{[\![i]\!]}, \vec{R}_{y_h}^{[\![i]\!]}, \alpha_{h,1}^{[\![i]\!]}, \alpha_{h,2}^{[\![i]\!]}, \beta_{h,1}^{[\![i]\!]}, \beta_{h,2}^{[\![i]\!]})_{h\in[l]}, (R_{A_j}^{[\![i]\!]})_{j=0}^{2l-1}, \tag{18}$$

$$(\vec{\delta}_{x_h}, \vec{\delta}_{y_h})_{h\in[l]}, (\delta_{A_j})_{j=0}^{2l-1} \Big). \tag{19}$$

Here (18) is the private input to each party P_i, while (19) is the common input to all the parties $P_1, \ldots, P_N$. Given Input_i, each party $P_i,\ i \in [N]$ (in P's head), generates a hash value $t_0^{[\![i]\!]} \in \mathbb{F}_{q^k}$ by using the hash function H_1, as follows.

$$t_0^{[\![i]\!]} \leftarrow \mathsf{H}_1\Big(i,\ \big(\vec{R}_{x_h}^{[\![i]\!]}, \vec{R}_{y_h}^{[\![i]\!]}, \alpha_{h,1}^{[\![i]\!]}, \alpha_{h,2}^{[\![i]\!]}, \beta_{h,1}^{[\![i]\!]}, \beta_{h,2}^{[\![i]\!]}\big)_{h\in[l]},\ (R_{A_j}^{[\![i]\!]})_{j=0}^{2l-1},$$
$$\big(\vec{\delta}_{x_h}, \vec{\delta}_{y_h}\big)_{h\in[l]},\ (\delta_{A_j})_{j=0}^{2l-1}\ \Big) \tag{20}$$

Each party $P_i,\ i \in [N]$ broadcasts $t_0^{[\![i]\!]}$ to every other party. Then each party $P_i,\ i \in [N]$ generates the value t_0 as the sum of all $t_0^{[\![i]\!]}$.

$$t_0 := \sum_{i\in[N]} t_0^{[\![i]\!]} \tag{21}$$

Then each party $P_i,\ i \in [N]$ computes the following values.

For $h \in [l]$:

$$(\alpha_{h,1} + t_0(\vec{u}_h \cdot \vec{x}_h))^{[\![i]\!]} := \alpha_{h,1}^{[\![i]\!]} + t_0(\vec{u}_h \cdot \vec{R}_{x_h}^{[\![i]\!]}), \tag{22}$$

$$(\alpha_{h,2} + t_0(\vec{v}_h \cdot \vec{x}_h))^{[\![i]\!]} := \alpha_{h,2}^{[\![i]\!]} + t_0(\vec{v}_h \cdot \vec{R}_{x_h}^{[\![i]\!]}), \tag{23}$$

$$(\beta_{h,1} + t_0(\vec{v}_h \cdot \vec{y}_h))^{[\![i]\!]} := \beta_{h,1}^{[\![i]\!]} + t_0(\vec{v}_h \cdot \vec{R}_{y_h}^{[\![i]\!]}), \tag{24}$$

$$(\beta_{h,2} + t_0(\vec{u}_h \cdot \vec{y}_h))^{[\![i]\!]} := \beta_{h,2}^{[\![i]\!]} + t_0(\vec{u}_h \cdot \vec{R}_{y_h}^{[\![i]\!]}), \tag{25}$$

$$(\sum_{j=0}^{2l-1} A_j t_0^j)^{[\![i]\!]} := \sum_{j=0}^{2l-1} R_{A_j}^{[\![i]\!]} t_0^j. \tag{26}$$

We note here that the notation of the left-hand sides are justified because addition, multiplication by a constant and adding a constant can be appropriately well-defined for additive secret sharing. (For more details, see Section 2.3 of [11].)

Each party $P_i,\ i \in [N]$ broadcasts (22), (23), (24), (25) and (26) to every other party. Then each party $P_i,\ i \in [N]$ generates the following values as the sum of all the received values.

For $h \in [l]$:

$$\alpha_{h,1} + t_0(\vec{u}_h \cdot \vec{x}_h) := \sum_{i\in[N]} (\alpha_{h,1} + t_0(\vec{u}_h \cdot \vec{x}_h))^{[\![i]\!]} + t_0 u_{h,n-1} + t_0(\vec{u}_h \cdot \vec{\delta}_{x_h}), \tag{27}$$

$$\alpha_{h,2} + t_0(\vec{v}_h \cdot \vec{x}_h) := \sum_{i\in[N]} (\alpha_{h,2} + t_0(\vec{v}_h \cdot \vec{x}_h))^{[\![i]\!]} + t_0 v_{h,n-1} + t_0(\vec{v}_h \cdot \vec{\delta}_{x_h}), \tag{28}$$

$$\beta_{h,1} + t_0(\vec{v}_h \cdot \vec{y}_h) := \sum_{i\in[N]} (\beta_{h,1} + t_0(\vec{v}_h \cdot \vec{y}_h))^{[\![i]\!]} + t_0 v_{h,n} + t_0(\vec{v}_h \cdot \vec{\delta}_{y_h}), \tag{29}$$

$$\beta_{h,2} + t_0(\vec{u}_h \cdot \vec{y}_h) := \sum_{i\in[N]} (\beta_{h,2} + t_0(\vec{u}_h \cdot \vec{y}_h))^{[\![i]\!]} + t_0 u_{h,n} + t_0(\vec{u}_h \cdot \vec{\delta}_{y_h}), \tag{30}$$

$$\sum_{j=0}^{2l-1} A_j t_0^j := \Big(\sum_{i\in[N]} \Big(\sum_{j=0}^{2l-1} A_j t_0^j \Big)^{[\![i]\!]} \Big) + \sum_{j=0}^{2l-1} \delta_{A_j} t_0^j. \tag{31}$$

Again, the notation of the left-hand sides are justified because addition, multiplication by a constant and adding a constant can be appropriately well-defined for additive secret sharing ([11]).

To evaluate the right-hand side of the formula (12) at $t = t_0$, each party $P_i,\ i \in [N]$ executes the following computation.

For $h \in [l]$:

$$\begin{aligned} F^{\vec{x}_h,\vec{y}_h,\vec{u}_h,\vec{v}_h}_{\alpha_{h,1},\alpha_{h,2},\beta_{h,1},\beta_{h,2}}(t_0) &:= (\alpha_{h,1} + t_0(\vec{u}_h \cdot \vec{x}_h))(\beta_{h,1} + t_0(\vec{v}_h \cdot \vec{y}_h)) \\ &\quad - (\alpha_{h,2} + t_0(\vec{v}_h \cdot \vec{x}_h))(\beta_{h,2} + t_0(\vec{u}_h \cdot \vec{y}_h)), \end{aligned} \tag{32}$$

$$F(t_0) := \prod_{h\in[l]} F^{\vec{x}_h,\vec{y}_h,\vec{u}_h,\vec{v}_h}_{\alpha_{h,1},\alpha_{h,2},\beta_{h,1},\beta_{h,2}}(t_0). \tag{33}$$

Each party $P_i,\ i \in [N]$ returns 1 if (31) and (33) are equal, and otherwise, 0:

$$\sum_{j=0}^{2l-1} A_j t_0^j \stackrel{?}{=} F(t_0). \tag{34}$$

The view of each party $P_i,\ i \in [N]$ in the above MPC is the following View_i.

$$\begin{aligned} \mathrm{View}_i = \Big(& t_0^{[\![i]\!]}, \big((\alpha_{h,1} + t_0(\vec{u}_h \cdot \vec{x}_h))^{[\![i]\!]},\ (\alpha_{h,2} + t_0(\vec{v}_h \cdot \vec{x}_h))^{[\![i]\!]}, \\ & (\beta_{h,1} + t_0(\vec{v}_h \cdot \vec{y}_h))^{[\![i]\!]},\ (\beta_{h,2} + t_0(\vec{u}_h \cdot \vec{y}_h))^{[\![i]\!]}\big)_{h\in[l]},\ \Big(\sum_{j=0}^{2l-1} A_j t_0^j\Big)^{[\![i]\!]} \Big). \end{aligned} \tag{35}$$

Then, following the MPC-in-the-Head principle, the prover P can compute commitments to the views, respectively. But here, as a variation due to the previous work [11], P computes a commitment to the views as the hash value of all the views, by using the hash function H_2.

$$C \leftarrow \mathsf{H}_2(\mathrm{View}_1, \ldots, \mathrm{View}_N). \tag{36}$$

Then P outputs the following first message msg_1 and its inner state St_P and sends msg_1 to the verifier V.

$$\mathrm{msg}_1 := \Big(C, \big(\alpha_{h,1} + t_0(\vec{u}_h \cdot \vec{x}_h),\ \alpha_{h,2} + t_0(\vec{v}_h \cdot \vec{x}_h),\ \beta_{h,1} + t_0(\vec{v}_h \cdot \vec{y}_h),\ \beta_{h,2} + t_0(\vec{u}_h \cdot \vec{y}_h)\big)_{h\in[l]},$$
$$\sum_{j=0}^{2l-1} A_j t_0^j,\ t_0,\ \big(\vec{\delta}_{x_h}, \vec{\delta}_{y_h}\big)_{h\in[l]},\ \big(\delta_{A_j}\big)_{j=0}^{2l-1}\Big). \tag{37}$$

• $\mathsf{V}([(\vec{u}_1, \vec{v}_1)$ or $\ldots$ or $(\vec{u}_l, \vec{v}_l)], \mathrm{msg}_1) \to \mathrm{msg}_2$.
The verifier V takes as input the statement $[(\vec{u}_1, \vec{v}_1)$ or $\ldots$ or $(\vec{u}_l, \vec{v}_l)]$ and the received first message msg_1. V chooses an index $\hat{i} \in [N]$ uniformly at random(, which means "Open all the committed values except this index"). Then V outputs the following second message msg_2, and sends msg_2 to the prover P.

$$\mathrm{msg}_2 := \hat{i}. \tag{38}$$

• $\mathsf{P}(\mathrm{St}_\mathsf{P}, \mathrm{msg}_2) \to \mathrm{msg}_3$.
P takes as input the inner state St_P and the received second message msg_2. Then P sets the punctured key $k_{\hat{i}}$ as msg_3 and sends msg_3 to the verifier V.

$$\mathrm{msg}_3 := k_{\hat{i}}. \tag{39}$$

• $\mathsf{V}([(\vec{u}_1, \vec{v}_1)$ or $\ldots$ or $(\vec{u}_l, \vec{v}_l)], \mathrm{msg}_1, \mathrm{msg}_2, \mathrm{msg}_3) \to d \in \{0, 1\}$.
V takes as input the statement, msg_1, msg_2 and the received third message msg_3. By using the punctured key $k_{\hat{i}}$, V reconstructs all the View_i for $i \neq \hat{i}$. As for $\hat{i}$ V computes $\mathrm{View}_{\hat{i}}$ by using the values that are contained in the first message msg_1, as follows.

For $h \in [l]$:

$$t_0^{[\![\hat{i}]\!]} := t_0 - \sum_{i\neq\hat{i}} t_0^{[\![i]\!]}, \tag{40}$$

$$(\alpha_{h,1} + t_0(\vec{u}_h \cdot \vec{x}_h))^{[\![\hat{i}]\!]} := \alpha_{h,1} + t_0(\vec{u}_h \cdot \vec{x}_h) - t_0 u_{h,n-1} - t_0(\vec{u}_h \cdot \vec{\delta}_{x_h}) - \sum_{i\neq\hat{i}} (\alpha_{h,1} + t_0(\vec{u}_h \cdot \vec{x}_h))^{[\![i]\!]}, \tag{41}$$

$$(\alpha_{h,2} + t_0(\vec{v}_h \cdot \vec{x}_h))^{[\![\hat{i}]\!]} := \alpha_{h,2} + t_0(\vec{v}_h \cdot \vec{x}_h) - t_0 v_{h,n-1} - t_0(\vec{v}_h \cdot \vec{\delta}_{x_h}) - \sum_{i\neq\hat{i}} (\alpha_{h,2} + t_0(\vec{v}_h \cdot \vec{x}_h))^{[\![i]\!]}, \tag{42}$$

$$(\beta_{h,1} + t_0(\vec{v}_h \cdot \vec{y}_h))^{[\![\hat{i}]\!]} := \beta_{h,1} + t_0(\vec{v}_h \cdot \vec{y}_h) - t_0 v_{h,n} - t_0(\vec{v}_h \cdot \vec{\delta}_{y_h}) - \sum_{i\neq\hat{i}} (\beta_{h,1} + t_0(\vec{v}_h \cdot \vec{y}_h))^{[\![i]\!]}, \tag{43}$$

$$(\beta_{h,2} + t_0(\vec{u}_h \cdot \vec{y}_h))^{[\![\hat{i}]\!]} := \beta_{h,2} + t_0(\vec{u}_h \cdot \vec{y}_h) - t_0 u_{h,n} - t_0(\vec{u}_h \cdot \vec{\delta}_{y_h}) - \sum_{i\neq\hat{i}} (\beta_{h,2} + t_0(\vec{u}_h \cdot \vec{y}_h))^{[\![i]\!]}, \tag{44}$$

$$(\sum_{j=0}^{2l-1} A_j t_0^j)^{[\![\hat{i}]\!]} := \sum_{j=0}^{2l-1} A_j t_0^j - \sum_{i\neq\hat{i}} \big((\sum_{j=0}^{2l-1} A_j t_0^j)^{[\![i]\!]}\big). \tag{45}$$

V computes the hash value of $\mathrm{View}_1, \ldots, \mathrm{View}_N$ by using H_2.

$$C' \leftarrow \mathsf{H}_2(\mathrm{View}_1, \ldots, \mathrm{View}_N) \tag{46}$$

Finally, V compares the hash value C contained in msg_1 with C'. If they are equal, then return 1, and otherwise, 0.

$$\text{Return } d := [C \stackrel{?}{=} C']. \tag{47}$$

4.2 Security of Our Protocol

The following lemmas and propositions are validated basically in the same way as those on the normal signature scheme [11].

Lemma 1. *Our MPC protocol is correct with overwhelming probability.*

Lemma 2. *Our MPC protocol has $(N-1)$-privacy.*

Proposition 1. *Our Σ-protocol has perfect completeness.*

Proposition 2. *Our Σ-protocol has special soundness in the random oracle model, with knowledge error $p_{k\text{-}err}$, where*

$$p_{k\text{-}err} = \frac{1}{N} + \frac{2l}{q^k}. \tag{48}$$

Proposition 3. *Our Σ-protocol has statistical honest-verifier zero-knowledge in the random oracle model.*

5 Our Ring Signatures

In this section, we describe our ring signature scheme that uses the non-interactive zero-knowledge proof system obtained by applying the Fiat-Shamir transformation to the Σ-protocol in Section 4.

5.1 Description of Algorithms

First, we introduce a repetition number τ and add it in the public parameter pp, which indicates that the prover P is repeatedly executed for τ times to make the knowledge error (48) negligible in λ. We note that τ should be lower bounded by a polynomial in λ. Our ring signature scheme RS which consists of $(\mathsf{KG}, \mathsf{Sign}, \mathsf{Vrfy})$ is described as follows.

• $\mathsf{KG}(1^\lambda) \to (pk, sk)$.

KG takes as input the security parameter 1^λ. Then KG reads from the public parameters pp the values q, k and n. KG generates a problem instance and its solution following the procedures in Section 4 and sets them as the public key pk and the secret key sk, respectively.

$$pk := (\vec{u}, \vec{v}) \in (\mathbb{F}_{q^k})^n \times (\mathbb{F}_{q^k})^n, \tag{49}$$

$$sk := (\vec{x}, \vec{y}) \in (\mathbb{F}_q)^n \times (\mathbb{F}_q)^n. \tag{50}$$

KG returns (pk, sk).

• $\mathsf{Sign}(R, s, sk_s, m) \to \sigma$.
Sign takes as input a ring $R = (pk_1, \ldots, pk_l)$, an identifier of a signer s, a secret key sk_s and a message m. By using the public key $pk_h = (\vec{u}_h, \vec{v}_h),\ h = 1, \ldots, l$ and the secret key $sk_s = (\vec{x}_s, \vec{y}_s)$, Sign sets a statement and its witness as follows.

$$\text{Statement} : [(\vec{u}_1, \vec{v}_1) \text{ or } \ldots \text{ or } (\vec{u}_l, \vec{v}_l)], \tag{51}$$

$$\text{Witness} : (\vec{x}_s, \vec{y}_s). \tag{52}$$

Sign inputs the statement and its witness to the prover P in Section 4, and obtains the first message msg_1^1 and the inner state $\mathrm{St}_{\mathsf{P}}^1$. Sign repeats this procedure τ times invoking P to obtain

$$\mathrm{msg}_1 := (\mathrm{msg}_1^1, \ldots, \mathrm{msg}_1^\tau), \tag{53}$$

$$\mathrm{St}_{\mathsf{P}}^1, \ldots, \mathrm{St}_{\mathsf{P}}^\tau. \tag{54}$$

Then, according to the Fiat-Shamir transformation [6], Sign applies the hash function $\mathsf{G} : \{0,1\}^* \to [N]$, to the concatenation as

$$\text{For } \kappa \in [\tau]:$$
$$\hat{i^\kappa} \leftarrow \mathsf{G}([(\vec{u}_1, \vec{v}_1) \text{ or } \ldots \text{ or } (\vec{u}_l, \vec{v}_l)] \parallel \kappa \parallel \mathrm{msg}_1 \parallel m). \tag{55}$$

(We note here that, in the security proofs of the scheme RS, the hash function G will be a random oracle.) Sign sets the hash values $(\hat{i^\kappa})_{\kappa \in [\tau]}$ as the second message as

$$\text{For } \kappa \in [\tau] :\ \mathrm{msg}_2^\kappa := \hat{i^\kappa}, \tag{56}$$

$$\mathrm{msg}_2 := (\mathrm{msg}_2^1, \ldots, \mathrm{msg}_2^\tau). \tag{57}$$

Then, Sign inputs the inner state $\mathrm{St}_{\mathsf{P}}^1$ and the challenge message msg_2^1 to the P, and obtains the third message msg_3^1. Sign repeats this procedure τ times to obtain

$$\mathrm{msg}_3 := (\mathrm{msg}_3^1, \ldots, \mathrm{msg}_3^\tau), \tag{58}$$

Now Sign sets

$$\sigma := (\mathrm{msg}_1, \mathrm{msg}_3). \tag{59}$$

Sign returns σ.
• $\mathsf{Vrfy}(R, m, \sigma) \to d$.
Vrfy takes as input a ring $R = (pk_1, \ldots, pk_l)$, a message m and a signature σ. By using the public key $pk_h = (\vec{u}_h, \vec{v}_h),\ h = 1, \ldots, l$, Vrfy sets a statement as follows.

$$\text{Statement} : [(\vec{u}_1, \vec{v}_1) \text{ or } \ldots \text{ or } (\vec{u}_l, \vec{v}_l)]. \tag{60}$$

Then Vrfy parses the signature as $\sigma = (\mathrm{msg}_1, \mathrm{msg}_3)$. Then, Vrfy obtains the hash value $\hat{i}$ as

$$\text{For } \kappa \in [\tau]:$$
$$\hat{i^\kappa} \leftarrow \mathsf{G}([(\vec{u}_1, \vec{v}_1) \text{ or } \ldots \text{ or } (\vec{u}_l, \vec{v}_l)] \parallel \kappa \parallel \mathrm{msg}_1 \parallel m). \tag{61}$$

Then Vrfy parses $\mathrm{msg}_1 = (\mathrm{msg}_1^1, \ldots, \mathrm{msg}_1^\tau)$ and $\mathrm{msg}_3 = (\mathrm{msg}_3^1, \ldots, \mathrm{msg}_3^\tau)$. Then, Vrfy inputs the statement, msg_1^1, $\mathrm{msg}_2^1 := \hat{i^1}$ and msg_3^1 to V, and obtains a boolean decision d^1. Vrfy repeats this procedure τ times to obtain boolean values $d_1, \ldots, d_\tau$. If all the values are 1, then Vrfy returns the decision $d = 1$, and otherwise, 0.

5.2 Security Properties of Our Ring Signatures

The following theorems are validated basically in the same way as those on the normal signature scheme [11].

Theorem 1 (Correctness)**.** *Our RS has correctness. More precisely, for any given possibly unbounded algorithm* $\mathbf{A}$*,* $\mathbf{Adv}^{corr}_{RS,\mathbf{A}}(\lambda) = 0$*.*

Theorem 2 (Unforgeability)**.** *Our RS has unforgeability in the random oracle model if the assumption that the SBC problem is hard holds. More precisely, for any given* PPT *algorithm* $\mathbf{A}$*,* $\mathbf{Adv}^{uf}_{RS,\mathbf{A}}(\lambda)$ *is negligible in* λ*, where* $\lambda = |q^k|$*,* τ *is lower bounded by a polynomial in* λ*,* q_G *is the maximum number of hash queries issued by* $\mathbf{A}(1^\lambda)$ *to the random oracle of* G*,* l_{max} *is the maximum number of possible* l *output by* $\mathbf{A}(1^\lambda)$*, and* p_{solve} *is the probability that for some* PPT *algorithm* S_{SBC} *to solve the given instance.*

$$\mathbf{Adv}^{uf}_{RS,\mathbf{A}}(\lambda) \leq (1+q_G)\Big(\frac{1}{N} + \frac{2l_{max}}{q^k} + \sqrt{l_{max}\cdot p_{solve}}\Big)^\tau. \tag{62}$$

Theorem 3 (Anonymity)**.** *Our RS has anonymity in the random oracle model. More precisely, for any given possibly unbounded algorithm* $\mathbf{A}$*,* $\mathbf{Adv}^{anon}_{RS,\mathbf{A}}(\lambda)$ *is negligible in* λ*.*

6 Signature Length and Comparison

In this section, we briefly evaluate the length of a signature of our RS from the viewpoints of asymptotic behavior and expected concrete length.

6.1 Expected Signature Length

Our ring signature σ consists of the two components, msg_1 (53) and msg_3 (58). We see that, if the security parameter λ is fixed, the length of a signature of our RS is asymptotically $O(l\log(N))$, where l is the size of an ad hoc ring R and N is the number of parties to execute our MPC protocol in the prover P's head. Note here that a naïve construction of a ring signature scheme is possible by applying the so-called OR-proof technique [4] to the original Σ-protocol [11]. Actually, the *asymptotic* behavior of the length of a signature of our RS would be the same as that of the generic approach [4] (i.e. $O(l\log(N))$).

However, the other component, that is, msg_3 which would also appear in the generic approach, behaves differently in our case. More precisely, the length of msg_3 is $O(\log(lN))$, which is asymptotically shorter than the case of msg_1. In contrast, in the generic-approach construction, the length of the corresponding component remains to be $O(l\log(N))$.

As for the concrete length of a signature, we can consult the previous work of Huth-Joux [11] (Section 6) because in our case it is roughly linear in the size l. When $(q,n,k) = (2,130,257)$ and $(N,\tau) = (256,16)$ and hence $\lambda = 128 \approx 257/2$ (i.e. 128-bit quantum security), the size of a normal signature [11] is about 6KB. (We note that it [11] applies the hypercube-technique and the correlated GGM tree, which we omit the details.) Therefore, when $l = 32$ for example, a concrete length of a ring signature of our RS would be expected as $l \cdot 6 = 32 \cdot 6 = 192$ KB, but considering the better behavior of msg_3, i.e., $O(\log(lN))$, the length is decreased by $(l\cdot\log(N) - \log(lN))\cdot 2\lambda\cdot\tau = (32\cdot 8 - 13)\cdot 256\cdot 16 = 995328$ bit that is about 124 KB, which yields about $|\sigma| \approx 68$ KB.

Table 1: Length Comparison of a Ring Signature.

Scheme	Asymptotic	Concrete: $l = 32$
Katz et al. [16]	$O(N \log(l))$	≈ 200 KB
Generic OR-proof [4]	$O(l \log(N))$	≈ 192 KB
Our RS	$O(l \log(N))$ $(\mathrm{msg}_3 : O(\log(lN)))$	≈ 68 KB

6.2 Length Comparison

Table 1 shows a comparison of a ring signature of our RS with the state-of-the-art ring signature scheme by Katz, Kolesnikov and Wang (KKW) [16] (Table 4 in Section 4). From the view point of asymptotic behavior, the comparison with KKW [16] would be case-by-case because in our case it is $O(l \log(N))$ while in the case of KKW [16] it is $O(N \log(l))$. As for the above concrete parameter setting that achieves 128-bit quantum security, we observe that the length of a ring signature of our RS (68 KB) is shorter than that (≈ 200 KB) of KKW [16]. We also note that the length of a ring signature of our RS is less than half of that of the above (naïve) generic approach (192 KB). We remark that ring signatures of the KKW [16] behave better than our RS when l increases, due to the $\log(l)$ factor.

7 Conclusion

In this paper, we proposed a ring signature scheme RS that is an extension of a (normal) signature scheme of Huth-Joux [11]. Our RS has unforgeability in the random oracle model under the assumption that the SBC problem is hard. Also, our RS has anonymity in the random oracle model. The size of a ring signature of our RS is linear to the size l of a ring R, but it is shorter than that obtained by the (naïve) generic approach that uses the generic OR-proof technique [4].

One of our future directions should be a further modification for shorter signature-length by specializing to the 2-party case of "VOLE-in-the-Head" [12] instead of the (generic) MPC-in-the-Head. Another point should be the recent remark by Kosuge and Xagawa [17] on the security proof of unforgeability in the case of correlated GGM tree.

References

[1] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *J. Cryptol.*, 22(1):114–138, 2009.

[2] Ronald Cramer. *Modular Designs of Secure, yet Practical Cyptographic Protocols*. PhD thesis, University of Amsterdam, Amsterdam, the Netherlands, 1996.

[3] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[4] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, pages 174–187, 1994.

[5] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

[6] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[7] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[8] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 464–479. IEEE Computer Society, 1984.

[9] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[10] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.

[11] Janik Huth and Antoine Joux. MPC in the head using the subfield bilinear collision problem. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 39–70. Springer, 2024.

[12] Janik Huth and Antoine Joux. Vole-in-the-head signatures from subfield bilinear collisions. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology – ASIACRYPT 2025*, pages 3–34, Singapore, 2026. Springer Nature Singapore.

[13] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 21–30, 2007.

[14] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[15] Daniel Kales. Secret sharing. https://www.iaik.tugraz.at/wp-content/uploads/teaching/mfc/secret_sharing.pdf. Accessed: 2024-12-14.

[16] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 525–537. ACM, 2018.

[17] Haruhisa Kosuge and Keita Xagawa. New security proofs of mpc-in-the-head signatures in the quantum random oracle model. *IACR Cryptol. ePrint Arch.*, page 1999, 2025.

[18] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[19] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.

[20] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, pages 371–388, 2004.

[21] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.*, 41(2):303–332, 1999.