New Code-Based Privacy-Preserving Cryptographic Constructions

Khoa Nguyen, Hanh Tang, Huaxiong Wang, and Neng Zeng

School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore

Abstract. Code-based cryptography has a long history but did suffer from periods of slow development. The field has recently attracted a lot of attention as one of the major branches of post-quantum cryptography. However, its subfield of privacy-preserving cryptographic constructions is still rather underdeveloped, e.g., important building blocks such as zero-knowledge range proofs and set membership proofs, and even proofs of knowledge of a hash preimage, have not been known under code-based assumptions. Moreover, almost no substantial technical development has been introduced in the last several years.

This work introduces several new code-based privacy-preserving cryptographic constructions that considerably advance the state-of-the-art in code-based cryptography. Specifically, we present 3 major contributions, each of which potentially yields various other applications. Our first contribution is a code-based statistically hiding and computationally binding commitment scheme with companion zero-knowledge (ZK) argument of knowledge of a valid opening that can be easily extended to prove that the committed bits satisfy other relations. Our second contribution is the first code-based zero-knowledge range argument for committed values, with communication cost logarithmic in the size of the range. A special feature of our range argument is that, while previous works on range proofs/arguments (in all branches of cryptography) only address ranges of *non-negative integers*, our protocol can handle *signed fractional numbers*, and hence, can potentially find a larger scope of applications. Our third contribution is the first code-based Merkle-tree accumulator supported by ZK argument of membership, which has been known to enable various interesting applications. In particular, it allows us to obtain the first code-based ring signatures and group signatures with logarithmic signature sizes.

1 Introduction

Code-based cryptography, pioneered by McEliece [55] in 1978, is the study of cryptosystems based on conjectured hard problems from coding theory and is one of the oldest branches of public-key cryptography. The field did suffer from periods of relatively slow development, but recent years have witnessed its resurgence. On the one hand, solutions to important theoretical problems such as constructing identity-based encryption [37,16] and obtaining worst-case hardness for

Learning Parity with Noise (LPN) [17] have been introduced. On the other hand, plausible algorithms for practical applications are being recognized by the community: with 7 PKE/KEM from codes accepted into the second round of the NIST Post-Quantum Cryptography Standardization process, the field stands together with lattice-based cryptography [2] as the two most promising candidates for post-quantum cryptography. Nevertheless, many interesting questions are still left open in the scope of code-based cryptography.

A prominent subfield of cryptography research is the designs of advanced schemes aiming to protect both privacy and security of users, namely, privacy-preserving cryptographic constructions. The major tools for building those constructions are zero-knowledge (ZK) proof [40] and argument [38,19] systems that allow to prove the truth of a statement while revealing no additional information. Almost all known zero-knowledge proof/argument systems used in code-based cryptography follow Stern's framework [67], in which the main technical idea is to employ random permutations to prove some specific properties of binary secret vectors, e.g., the secret vectors have a fixed Hamming weight. Variants of Stern's protocol have been employed to design a few privacy-preserving constructions, e.g., proofs of plaintext knowledge for code-based encryption [59], linear-size ring signatures [56,30,57,18], linear-size and sublinear-size group signatures [35,3], proofs of valid openings for LPN-based computationally hiding commitments [46] and proofs for general relations [46]. However, this line of research is still rather underdeveloped, since many important building blocks for privacy-preserving code-based cryptography are still missing, ranging from very basic ones like proof of knowledge of a hash preimage to advanced ones such as range proofs and set membership proofs. Even more worrisome is the slow progress in the field, with almost no substantial technical development being introduced in the last 6 years. This unsatisfactory state-of-affairs motivates our work.

Our Results. In this work, we introduce several new privacy-preserving constructions that we believe will considerably advance the state-of-the-art in code-based cryptography. Specifically, we provide 3 main contributions, each of which potentially yields various other applications.

First, we put forward a code-based statistically hiding and computationally binding commitment scheme with companion zero-knowledge argument of knowledge (ZKAoK) of a valid opening. The commitment scheme is based on a family of collision-resistant hash functions introduced by Augot, Finiasz and Sendrier (AFS) [6,7], similar variants of which was recently studied in [4,68,17]. The design of the scheme is quite standard, in which we plug in a randomness with sufficient min-entropy and make use of the left-over hash lemma [39]. Our non-trivial contribution here is a companion ZK argument system that makes the commitment scheme much more useful for privacy-preserving protocols. In many advanced protocols, one typically works with different sub-protocols that share a common secret, and a commitment supported by ZK proofs/arguments can greatly help in bridging these layers. In the code-based setting, such a commitment was proposed in [46], but it relies on the hardness of the LPN problem and

operates in the computationally-hiding setting. In our setting, to base security on a variant of the Syndrome Decoding problem, the committed message has to be non-linearly encoded into a low-weight vector of larger dimension before being hashed. This makes proving knowledge of a valid opening quite challenging, since one has to prove that the non-linear encoding process is done correctly. We overcome this problem by employing a specific permuting technique that works in the framework of Stern's protocol [67] and that enables us to keep fine-grained control on how each bit of the message behaves in the encoding process. The fact that we can "keep track" of the secret committed bits indeed makes our protocol composable with other privacy-preserving protocols, where we can additionally prove that these bits satisfy other relations. In particular, it paves the way for our next 2 contributions.

Second, we provide zero-knowledge range arguments for *signed fractional numbers* committed via our code-based commitment. For $\ell > 0$ and $f \geq 0$, we consider fractional numbers X represented as $x_\ell x_{\ell-1} \dots x_0 \bullet x_{-1} x_{-2} \dots x_{-f}$, where x_ℓ is the sign bit, $x_{\ell-1}, \dots, x_0$ are the integer bits, and $x_{-1}, \dots, x_{-f}$ are the fractional bits. Our techniques allow to prove in zero-knowledge that a committed number X satisfies inequalities $X \leq Y$ or $X < Y$, where Y is another signed fractional number (that could be publicly given or be committed). These techniques directly yield range arguments addressing both public and hidden ranges with communication cost logarithmic in the sizes of the ranges. This not only solves an open problem in code-based cryptography but also brings a novel feature to the topic of range proofs in general. Range proofs, introduced by Brickell *et al.* [20], serve as building blocks in various applications, including anonymous credentials [25], auctions [53], e-voting [43] and anonymous e-cash [23]. Efficient constructions [22,52,44,27,29,41,51,34] have been proposed in almost all major branches of cryptography, but up to our knowledge, they only address non-negative integers. Negative numbers do often appear in our daily life in the forms of financial loss, bad reputation, medical data, etc., and it would be desirable to be able to handle them in a privacy-preserving manner. Moreover, these data values could be stored as fractional numbers, e.g., bank account balances, GPAs and tax records, and hence, a protocol addressing them directly in such forms would potentially be interesting. This inspires our investigation of range arguments for signed fractional numbers.

Our third contribution is the first code-based accumulator [10] supported by ZK arguments of valid accumulated values, which directly imply ZK arguments of set membership. Accumulators are essential building blocks in numerous authentication mechanisms, including ring and group signatures [32,50,31], anonymous credentials [26,24,1], e-cash [5], and authenticated data structures [64,63]. Accumulators with companion ZK proofs have been proposed from number-theoretic assumptions [10,61], lattice assumptions [50] and from symmetric-key primitives [31,15], but have not been known in the scope of code-based cryptography. Our construction fills in this gap and opens up a wide range of applications that have not been achieved from code-based assumptions. Our design resembles Libert et al.'s approach for lattices [50], which relies on Merkle hash trees [58] and

ZKAoK of a tree path from the root to a secret leaf. However, unlike the lattice setting where smallness (and computational hardness) can be defined with respect to various metrics and the output of each hashing can be easily decomposed into binary to serve as the input of the next step, the binary linear code setting with Hamming metric makes the problem more challenging. At each step, we have to encode the hash output to a small-weight vector (with respect to its dimension) before going to the next step, and prove that the whole recursive process is done correctly. Fortunately, this difficulty can be overcome with our ZK techniques. As applications, we put forward 2 prominent anonymity-oriented constructions: ring signature [66] and group signature [28].

Our ring signature scheme is the first one from code-based assumptions that achieves signature size logarithmic in the cardinality of the ring. Previous constructions [56,30,57,18] all suffer from linear-size signatures. Designing logarithmic-size ring signatures is generally a hard problem, which usually requires a powerful supporting technique, which is - in this case - an accumulator that enables logarithmic-size arguments of set membership.

Our group signature scheme is also the first one that produces logarithmic-size signatures in the scope of code-based cryptography. Compared with previous works [35,3], our scheme not only has shorter signatures (for large groups), but also achieve the stronger notion of CCA-anonymity. The scheme is also appealing in the sense that it is the first time in all branches of cryptography a CCA-anonymous group signature scheme is achieved before a standard-model signature compatible with ZK proofs is known. (The latter is traditionally considered to be a necessary ingredient for building the former in a generic manner.)

Our Techniques. Let us first discuss our basic techniques for proving in zero-knowledge the knowledge of a preimage of a hash, computed via the AFS hash function $\mathcal{H}_{\text{afs}} : \{0,1\}^k \to \{0,1\}^n$. Let $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times 2^c \cdot k/c}$, for some constant c dividing k. Let RE: $\{0,1\}^k \to \{0,1\}^{2^c \cdot k/c}$ be an encoding function that maps $\mathbf{x}$ to $\mathsf{RE}(\mathbf{x})$, defined as follows. First, write $\mathbf{x}$ block-wise as $\mathbf{x} = (\mathbf{x}_1 \| \dots \| \mathbf{x}_{k/c})$, where $\mathbf{x}_j = (x_{j,1}, \dots, x_{j,c})^\top$ for $j \in [k/c]$. Denote by $\Delta_{2^c}(\mathbf{x}_j)$ the binary vector of dimension 2^c and Hamming weight 1 whose sole 1 entry is at the t_j-th position, for $t_j = \Sigma_{i=1}^{c} 2^{c-i} \cdot x_{j,i} \in [0, 2^c - 1]$. Then $\mathsf{RE}(\mathbf{x})$ is defined to be $(\Delta_{2^c}(\mathbf{x}_1) \| \dots \| \Delta_{2^c}(\mathbf{x}_{k/c}))$, and the hash output is set as $\mathbf{u} = \mathbf{B} \cdot \mathsf{RE}(\mathbf{x})$. Given $(\mathbf{B}, \mathbf{u})$, to prove that we know $\mathbf{x}$ such that $\mathcal{H}_{\text{afs}}(\mathbf{x}) = \mathbf{u}$, we have to demonstrate that the encoding $\mathsf{RE}(\cdot)$ is done correctly for $\mathbf{x}$. To this end, we introduce the following permuting technique.

For every vector $\mathbf{s} = (s_1, \dots, s_c) \in \{0,1\}^c$, define the permutation $E_{\mathbf{s}}$ that transforms vector $\mathbf{x} = (x_{0,0,\dots,0}, \dots, x_{i_1,\dots,i_c}, \dots, x_{1,1,\dots,1}) \in \{0,1\}^{2^c}$ into vector $E_{\mathbf{s}}(\mathbf{x}) = (x'_{0,0,\dots,0}, \dots, x'_{1,1,\dots,1})$, where for each $(i_1, \dots, i_c) \in \{0,1\}^c$, we have $x_{i_1,\dots,i_c} = x'_{i_1 \oplus s_1, \dots, i_c \oplus s_c}$.

Note that, for any $\mathbf{s}, \mathbf{v} \in \{0,1\}^c$, we have:

$$\mathbf{x} = \Delta_{2^c}(\mathbf{v}) \iff E_{\mathbf{s}}(\mathbf{x}) = \Delta_{2^c}(\mathbf{v} \oplus \mathbf{s}). \tag{1}$$

For $\mathbf{t} = (\mathbf{t}_1 \| \dots \| \mathbf{t}_{k/c}) \in \{0,1\}^k$ consisting of k/c blocks of length c, define the permutation $E'_{\mathbf{t}}$ that transforms vector $\mathbf{y} = (\mathbf{y}_1 \| \dots \| \mathbf{y}_{k/c}) \in \{0,1\}^{2^c \cdot k/c}$

consisting of k/c blocks of length 2^c into vector of the following form $E'_{\mathbf{t}}(\mathbf{y}) = \big(E_{\mathbf{t}_1}(\mathbf{y}_1) \| \ldots \| E_{\mathbf{t}_{n/c}}(\mathbf{y}_{n/c})\big)$. Note that, for any $\mathbf{t}, \mathbf{x} \in \{0,1\}^k$, we have:

$$\mathbf{y} = \mathsf{RE}(\mathbf{x}) \iff E'_{\mathbf{t}}(\mathbf{y}) = \mathsf{RE}(\mathbf{x} \oplus \mathbf{t}). \tag{2}$$

In the framework of Stern's protocol [67], the equivalence in (2) enables us to prove that $\mathbf{y}$ is the correct encoding of $\mathbf{x}$, as follows. The prover samples uniformly random $\mathbf{t}$ and demonstrates to the verifier that the right-hand side of (2) holds. The verifier is thus convinced that its left-hand side also holds, while learning no additional information about $\mathbf{x}$, thanks to the "one-time pad" $\mathbf{t}$. Moreover, this permuting technique allows us to keep control over the bits of $\mathbf{x}$, in order to prove that they satisfy other relations. To this end, it suffices to use the same "one-time pad" at other appearances of $\mathbf{x}$.

Our discussed above technique then readily extends to handle the case when there are two inputs to the hash function, i.e., $\mathcal{H}_{\text{afs}} : \{0,1\}^\ell \times \{0,1\}^k \to \{0,1\}^n$. If we set k sufficiently large so that the leftover hash lemma [39] applies, then we obtain a statistically hiding commitment scheme that is supported by our ZK technique. On the other hand, if we set $\ell = k = n$, then we get a function that compresses two child-inputs of n bits to one parent-output, which is then can be used to build a Merkle hash tree. Then, by combining the ZK techniques from [50] and our techniques for proving correctness of re-encoding at each step in a tree path, we get a Merkle-tree-accumulator supported by logarithmic-size zero-knowledge arguments.

To build a ring signature, we add one more level of secret under every leaf in the tree, so that each leaf corresponds to a user's public key, and define the signing process as the process of proving knowledge of an extended path from beneath a leaf up to the root of the tree. Furthermore, as in [50], by adding a CCA2-secure encryption layer supported by zero-knowledge arguments of plaintext knowledge, we can build a secure group signature. To this end, we employ the randomized McEliece scheme [62] and make it CCA2-secure in the random oracle model via the Naor-Yung transformation [60]. Both our ring and group signatures feature logarithmic-size signatures, thanks to the tree structure.

Let us next discuss our techniques for handling inequalities among signed fractional numbers, which lead to our range arguments. Comparing signed fractional numbers in zero-knowledge is highly non-trivial, due to 2 main reasons. First, unlike for non-negative numbers, the order of (binary) signed numbers is not lexicographical, e.g., for $(\ell, f) = (5, 2)$, the number $110110 \bullet 11$ is lexicographically larger than $000011 \bullet 00$, yet its decimal value is -9.25, which is smaller than than the decimal value 3 of the latter. Thus, it is counterintuitive when we compare them in zero-knowledge. Second, the approach of proving $X \leq Y$ via demonstrating the existence of $Z \geq 0$ such that $X + Z = Y$ (as used in [51]) is not easily applicable here, due to the problem of overflows. For instance, the binary addition (with carries) of $011110 \bullet 11$ and $000011 \bullet 00$ would yield $100001 \bullet 11$, which is translated into an incorrect expression $30.75 + 3 = -30.25$. Therefore, we have to carefully address the complications caused by the signed bits and to ensure that overflows do not occur.

Our idea is to derive necessary and sufficient conditions for $X \leq Y$, in a way such that these conditions can be correctly and efficiently proved in ZK. Let $(x_\ell, \ldots, x_0, x_{-1}, \ldots, x_{-f})$, $(y_\ell, \ldots, y_0, y_{-1}, \ldots, y_{-f})$ be the bits representing X and Y, respectively. We observe and then formally prove that $X \leq Y$ if and only if there exist bits $z_\ell, z_{\ell-1}, \ldots, z_0, z_{-1}, \ldots, z_{-f}$, $c_{\ell+1}, c_\ell, c_{\ell-1}, \ldots, c_0, c_{-1}, \ldots, c_{-f+1}$ satisfying

$$\begin{cases} c_{-f+1} = x_{-f} \cdot z_{-f} \\ c_i = x_{i-1} \cdot z_{i-1} \ \oplus \ y_{i-1} \cdot c_{i-1} \ \oplus \ c_{i-1}, \quad \forall i \in [-f+2, \ell+1] \\ y_{-f} = x_{-f} \ \oplus \ z_{-f} \\ y_i = x_i \ \oplus \ z_i \oplus c_i, \quad \forall i \in [-f+1, \ell] \\ y_\ell = x_\ell \ \oplus \ c_{\ell+1}. \end{cases}$$

This simple-yet-vital result allows us to reduce the inequality relations among signed fractional numbers to simple relations among bits, which can be effectively addressed using existing techniques [50,49] for Stern's protocol.

Organization. The rest of the paper is organized as follows. In Section 2, we recall the background on code-based hash functions, ZK arguments and previous Stern-like techniques. Our commitment scheme together with our techniques for proving knowledge of code-based hash preimages and committed values are described in Section 3. In Section 4, we present our treatment of signed fractional numbers and construct ZK range arguments for committed signed fractional numbers. Our accumulator and its supporting ZK argument of membership are given in Section 5. Applications of to ring signatures and group signatures are then discussed in Section 6. The descriptions and analyses of our ring and group signature schemes are deferred to Appendix C and Appendix D, respectively.

2 Background

2.1 Code-Based Collision-Resistant Hash Functions

This section recalls the family of code-based hash functions proposed by Augot, Finiasz and Sendrier (AFS) [6,7], which is based on the hardness of the 2-Regular Null Syndrome Decoding (2-RNSD) problem. We note that the more recent proposals of code-based hash functions [4,68,17], although relying on different assumptions, are syntactically similar to the AFS family at a high level. Working with the AFS family allows us to derive practical parameters, based on the analyses of [14,13]. Let us begin by introducing some supporting notations.

Notations. We identify $\mathbb{Z}_2$ as the set $\{0, 1\}$. The set $\{a, \ldots, b\}$ is denoted by $[a, b]$. We often write $[b]$ when $a = 1$. Let $\oplus$ denote the bit-wise addition operation modulo 2. If S is a finite set, then $x \xleftarrow{\$} S$ means that x is chosen uniformly at random from S. Throughout this paper, all vectors are column vectors. When concatenating vectors $\mathbf{x} \in \mathbb{Z}_2^m$ and $\mathbf{y} \in \mathbb{Z}_2^k$, for simplicity, we use the notation $(\mathbf{x} \| \mathbf{y}) \in \mathbb{Z}_2^{m+k}$ instead of $(\mathbf{x}^\top \| \mathbf{y}^\top)^\top$. Denote $\mathsf{B}(n, \omega)$ to be the set of all binary

vectors of length n with weight ω and the symmetric group of all permutations of n elements to be S_n.

For $c \in \mathbb{Z}^+$ and for k divisible by c, define the following.

- $\mathsf{Regular}(k, c)$: the set of all vectors $\mathbf{w} = (\mathbf{w}_1 \| \ldots \| \mathbf{w}_{k/c}) \in \{0,1\}^{2^c \cdot k/c}$ consisting of k/c blocks, each of which is an element of $\mathsf{B}^1_{2^c}$. Here $\mathsf{B}^1_{2^c}$ is the set that contains all the elements in $\{0,1\}^{2^c}$ with Hamming weight 1. If $\mathbf{w} \in \mathsf{Regular}(k, c)$ for some k, c, then we call $\mathbf{w}$ a *regular word*.
- RE: $\{0,1\}^k \to \{0,1\}^{2^c \cdot k/c}$, a regular encoding function that maps $\mathbf{x}$ to $\mathsf{RE}(\mathbf{x})$, defined as follows. Denote $\mathbf{x} = (\mathbf{x}_1 \| \ldots \| \mathbf{x}_{k/c})$, where $\mathbf{x}_j = (x_{j,1}, \ldots, x_{j,c})^\top$ for $j \in [k/c]$. Then compute $t_j = \Sigma_{i=1}^{c} 2^{c-i} \cdot x_{j,i}$. Denote by $\Delta_{2^c}(\mathbf{x}_j)$ the element in $\mathsf{B}^1_{2^c}$ whose sole 1 entry is at the t_j-th position for $t_j \in [0, 2^c - 1]$. $\mathsf{RE}(\mathbf{x})$ is then defined to be $(\Delta_{2^c}(\mathbf{x}_1) \| \ldots \| \Delta_{2^c}(\mathbf{x}_{k/c}))$. One can check that $\mathsf{RE}(\mathbf{x}) \in \mathsf{Regular}(k, c)$.
- $\mathsf{2\text{-}Regular}(k, c)$: the set of all vectors $\mathbf{x} \in \{0,1\}^{2^c \cdot k/c}$, such that there exist regular words $\mathbf{v}, \mathbf{w} \in \mathsf{Regular}(k, c)$ satisfying $\mathbf{x} = \mathbf{v} \oplus \mathbf{w}$. Note that, $\mathbf{x} \in \mathsf{2\text{-}Regular}(k, c)$ if and only if $\mathbf{x}$ can be written as the concatenation of k/c blocks of length 2^c, each of which has Hamming weight 0 or 2. If $\mathbf{x} \in \mathsf{2\text{-}Regular}(k, c)$ for some k, c, then we call $\mathbf{x}$ a 2-*regular word*.

The 2-RNSD problem. Introduced by Augot, Finiasz and Sendrier [6,7], the 2-RNSD problem asks to find low-weight 2-regular codewords in random binary linear codes. This problem is closely related to the Small Codeword Problem [54] and binary Shortest Vector Problem [4], with an additional and strong constraint that the solution codeword must be 2-regular.

Definition 1. *The* $\mathsf{2\text{-}RNSD}_{n,k,c}$ *problem, parameterized by integers* n, k, c*, is as follows. Given a uniformly random matrix* $\mathbf{B} \in \mathbb{Z}_2^{n \times m}$*, where* $m = 2^c \cdot k/c$*, find a non-zero vector* $\mathbf{z} \in \mathsf{2\text{-}Regular}(k, c) \subseteq \{0,1\}^m$ *such that* $\mathbf{B} \cdot \mathbf{z} = \mathbf{0}$.

The problem is shown to be NP-complete in the worst case [6]. In practice, for appropriate choices of n, k, c, the best known algorithms require exponential times in the security parameter. See [13] for a comprehensive discussion of known attacks and parameter settings.

The AFS hash functions. Let λ be the security parameter. The AFS family of hash functions $\mathcal{H}_{\text{afs}}$ maps $\{0,1\}^k$ to $\{0,1\}^n$, where $n, k = \Omega(\lambda)$ and $k > n$. Each function in the family is associated with a matrix $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times 2^c \cdot k/c}$, for some properly chosen constant c dividing k. To compute the hash value of $\mathbf{x} \in \{0,1\}^k$, one encodes it to the corresponding regular word $\mathsf{RE}(\mathbf{x}) \in \{0,1\}^{2^c \cdot k/c}$ and outputs $\mathbf{B} \cdot \mathsf{RE}(\mathbf{x})$.

The above hash functions are collision-resistant assuming the hardnes of the $\mathsf{2\text{-}RNSD}_{n,k,c}$ problem. Suppose that the adversary can produce distinct $\mathbf{x}_0, \mathbf{x}_1$ such that $\mathbf{B} \cdot \mathsf{RE}(\mathbf{x}_0) = \mathbf{B} \cdot \mathsf{RE}(\mathbf{x}_1)$. Let $\mathbf{z} = \mathsf{RE}(\mathbf{x}_0) \oplus \mathsf{RE}(\mathbf{x}_1) \neq \mathbf{0}$ then we have $\mathbf{z} \in \mathsf{2\text{-}Regular}(k, c)$ and $\mathbf{B} \cdot \mathbf{z} = \mathbf{0}$. In other words, $\mathbf{z}$ is a solution to the $\mathsf{2\text{-}RNSD}_{n,k,c}$ problem associated with matrix $\mathbf{B}$.

In this work, we rely on the above hash function family to develop two tools for privacy-preserving code-based cryptography: (i) computationally binding and statistically hiding commitments supporting by ZK arguments of knowledge of valid openings; (ii) Cryptographic accumulators supporting by ZK arguments of accumulated values.

2.2 Zero-Knowledge Argument Systems and Stern-like Protocols

We will work with statistical zero-knowledge argument systems, namely, interactive protocols where the zero-knowledge property holds against *any* cheating verifier, while the soundness property only holds against *computationally bounded* cheating provers. More formally, let the set of statements-witnesses $\mathrm{R} = \{(y, w)\} \in \{0,1\}^* \times \{0,1\}^*$ be an NP relation. A two-party game $\langle \mathcal{P}, \mathcal{V} \rangle$ is called an interactive argument system for the relation R with soundness error e if the following conditions hold:

- $\mathsf{Completeness}$. If $(y, w) \in \mathrm{R}$ then $\Pr\big[\langle \mathcal{P}(y, w), \mathcal{V}(y) \rangle = 1\big] = 1$.
- $\mathsf{Soundness}$. If $(y, w) \notin \mathrm{R}$, then $\forall$ PPT $\widehat{\mathcal{P}}$: $\Pr[\langle \widehat{\mathcal{P}}(y, w), \mathcal{V}(y) \rangle = 1] \leq e$.

An argument system is called statistical zero-knowledge if there exists a PPT simulator $\mathcal{S}(y)$ having oracle access to any $\widehat{\mathcal{V}}(y)$ and producing a simulated transcript that is statistically close to the one of the real interaction between $\mathcal{P}(y, w)$ and $\widehat{\mathcal{V}}(y)$. A related notion is argument of knowledge, which requires the witness-extended emulation property. For protocols consisting of 3 moves (*i.e.*, commitment-challenge-response), witness-extended emulation is implied by *special soundness* [42], where the latter assumes that there exists a PPT extractor which takes as input a set of valid transcripts w.r.t. all possible values of the "challenge" to the same "commitment", and outputs w' such that $(y, w') \in \mathrm{R}$.

Stern-like protocols. The statistical zero-knowledge arguments of knowledge presented in this work are Stern-like [67] protocols. In particular, they are Σ-protocols in the generalized sense defined in [46,12] (where 3 valid transcripts are needed for extraction, instead of just 2). The basic protocol consists of 3 moves: commitment, challenge, response. If we employ our first explicit construction of statistically hiding string commitment from a code-based assumption in the first move, then one obtains a statistical zero-knowledge argument of knowledge (ZKAoK) with perfect completeness, constant soundness error 2/3. In many applications, the protocol is repeated a sufficient number of times to make the soundness error negligibly small. For instance, to achieve soundness error 2^{-80}, it suffices to repeat the basic protocol 137 times.

An abstraction of Stern's protocols. We recall an abstraction, adapted from [48], which captures the sufficient conditions to run a Stern-like protocol. Looking ahead, this abstraction will be helpful for us in presenting our ZK argument systems: we will reduce the relations we need to prove to instances of the abstract protocol, using our specific techniques. We recall an abstraction proposed in [48]. Let K, L be positive integers, where $L \geq K$, and let VALID be a

subset of $\{0,1\}^L$. Suppose that $\mathcal{S}$ is a finite set such that one can associate every $\phi \in \mathcal{S}$ with a permutation Γ_ϕ of L elements, satisfying the following conditions:

$$\begin{cases} \mathbf{w} \in \mathsf{VALID} \iff \Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}, \\ \text{If } \mathbf{w} \in \mathsf{VALID} \text{ and } \phi \text{ is uniform in } \mathcal{S}, \text{ then } \Gamma_\phi(\mathbf{w}) \text{ is uniform in } \mathsf{VALID}. \end{cases} \tag{3}$$

We aim to construct a statistical ZKAoK for the following abstract relation:

$$\mathrm{R}_{\mathrm{abstract}} = \big\{(\mathbf{M}, \mathbf{v}), \mathbf{w} \in \mathbb{Z}_2^{K \times L} \times \mathbb{Z}_2^K \times \mathsf{VALID} : \mathbf{M} \cdot \mathbf{w} = \mathbf{v}\big\}.$$

The conditions in (3) play a crucial role in proving in ZK that $\mathbf{w} \in \mathsf{VALID}$: To do so, the prover samples $\phi \xleftarrow{\$} \mathcal{S}$ and lets the verifier check that $\Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}$, while the latter cannot learn any additional information about $\mathbf{w}$ thanks to the randomness of ϕ. Furthermore, to prove in ZK that the linear equation holds, the prover samples a masking vector $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, and convinces the verifier instead that $\mathbf{M} \cdot (\mathbf{w} \oplus \mathbf{r}_w) = \mathbf{M} \cdot \mathbf{r}_w \oplus \mathbf{v}$.

The interaction between prover $\mathcal{P}$ and verifier $\mathcal{V}$ is described in Figure 1. The protocol employs a statistically hiding and computationally binding string commitment scheme COM.

1. **Commitment:** Prover samples $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, $\phi \xleftarrow{\$} \mathcal{S}$ and randomness ρ_1, ρ_2, ρ_3 for COM. Then he sends $\mathsf{CMT} = \big(C_1, C_2, C_3\big)$ to the verifier, where

$$C_1 = \mathsf{COM}(\phi, \mathbf{M} \cdot \mathbf{r}_w; \rho_1), \quad C_2 = \mathsf{COM}(\Gamma_\phi(\mathbf{r}_w); \rho_2),$$
$$C_3 = \mathsf{COM}(\Gamma_\phi(\mathbf{w} \oplus \mathbf{r}_w); \rho_3).$$

2. **Challenge:** The verifier sends a challenge $Ch \xleftarrow{\$} \{1,2,3\}$ to the prover.
3. **Response:** Depending on Ch, the prover sends RSP computed as follows:
   - $Ch = 1$: Let $\mathbf{t}_w = \Gamma_\phi(\mathbf{w})$, $\mathbf{t}_r = \Gamma_\phi(\mathbf{r}_w)$, and RSP $= (\mathbf{t}_w, \mathbf{t}_r, \rho_2, \rho_3)$.
   - $Ch = 2$: Let $\phi_2 = \phi$, $\mathbf{w}_2 = \mathbf{w} \oplus \mathbf{r}_w$, and RSP $= (\phi_2, \mathbf{w}_2, \rho_1, \rho_3)$.
   - $Ch = 3$: Let $\phi_3 = \phi$, $\mathbf{w}_3 = \mathbf{r}_w$, and RSP $= (\phi_3, \mathbf{w}_3, \rho_1, \rho_2)$.

Verification: Receiving RSP, the verifier proceeds as follows:

- $Ch = 1$: Check that $\mathbf{t}_w \in \mathsf{VALID}$, $C_2 = \mathsf{COM}(\mathbf{t}_r; \rho_2)$, $C_3 = \mathsf{COM}(\mathbf{t}_w \oplus \mathbf{t}_r; \rho_3)$.
- $Ch = 2$: Check that $C_1 = \mathsf{COM}(\phi_2, \mathbf{M} \cdot \mathbf{w}_2 \oplus \mathbf{v}; \rho_1)$, $C_3 = \mathsf{COM}(\Gamma_{\phi_2}(\mathbf{w}_2); \rho_3)$.
- $Ch = 3$: Check that $C_1 = \mathsf{COM}(\phi_3, \mathbf{M} \cdot \mathbf{w}_3; \rho_1)$, $C_2 = \mathsf{COM}(\Gamma_{\phi_3}(\mathbf{w}_3); \rho_2)$.

In each case, the verifier outputs 1 if and only if all the conditions hold.

Fig. 1: Stern-like ZKAoK for the relation $\mathrm{R}_{\mathrm{abstract}}$.

The properties of the protocols are summarized in Theorem1, whose proof is given in Appendix A.

Theorem 1 ([48]). *Assume that* COM *is a statistically hiding and computationally binding string commitment scheme. Then, the protocol in Figure 1 is a statistical* ZKAoK *with perfect completeness, soundness error* $2/3$*, and communication cost* $\mathcal{O}(L)$*. In particular:*

- *There exists a polynomial-time simulator that, on input* $(\mathbf{M}, \mathbf{v})$*, outputs an accepted transcript statistically close to that produced by the real prover.*
- *There exists a polynomial-time knowledge extractor that, on input a commitment* CMT *and* 3 *valid responses* $(\mathrm{RSP}_1, \mathrm{RSP}_2, \mathrm{RSP}_3)$ *to all* 3 *possible values of the challenge* Ch*, outputs* $\mathbf{w}' \in \mathsf{VALID}$ *such that* $\mathbf{M} \cdot \mathbf{w}' = \mathbf{v}$*.*

2.3 Previous Extending-then-Permuting Techniques

We next recall the extending-then-permuting techniques for proving in Stern's framework the knowledge of a single secret bit x and a product of 2 secret bits $x_1 \cdot x_2$, presented in [50] and [49], respectively.

Let $\oplus$ denote the bit-wise addition operation modulo 2. For any bit $b \in \{0,1\}$, denote by $\overline{b}$ the bit $b = b \oplus 1$. Note that, for any $b, c \in \{0,1\}$, we have $\overline{b \oplus c} = b \oplus c \oplus 1 = \overline{b} \oplus c$. For any bit b, let $\mathsf{enc}(b) = (\overline{b}, b) \in \{0,1\}^2$.

For any bit $c \in \{0,1\}$, define F_c as the permutation that transforms integer vector $\mathbf{v} = (v_0, v_1) \in \mathbb{Z}^2$ into vector $F_c(\mathbf{v}) = (v_c, v_{\overline{c}})$. Namely, if $c = 0$ then F_c keeps the arrangement the coordinates of $\mathbf{v}$; or swaps them if $c = 1$. Note that:

$$\mathbf{v} = \mathsf{enc}(b) \iff F_c(\mathbf{v}) = \mathsf{enc}(b \oplus c). \tag{4}$$

The authors of [50] showed that the equivalence (4) is helpful for proving knowledge of a secret bit x that may appear in several correlated linear equations. To this end, one extends x to $\mathsf{enc}(x) \in \{0,1\}^2$, and permutes the latter using F_c, where c is a uniformly random bit. Then one demonstrates to the verifier that the permuted vector is $\mathsf{enc}(x \oplus c)$, which implies that the original vector $\mathsf{enc}(x)$ is well-formed - which in turn implies knowledge of some bit x. Meanwhile, the bit c acts as a "one-time pad" that completely hides x.

In [49], Libert et al. proposed a method for proving the well-formedness of the product of two secret bits x_1, x_2, based on the following technique.

- For any two bits b_1, b_2, define the vector

$$\mathsf{ext}(b_1, b_2) = (\ \overline{b}_1 \cdot \overline{b}_2,\ \overline{b}_1 \cdot b_2,\ b_1 \cdot \overline{b}_2,\ b_1 \cdot b_2\ ) \in \{0,1\}^4,$$

  that is an extension of the bit product $b_1 \cdot b_2$.
- For any two bits $c_1, c_2 \in \{0,1\}$, define T_{c_1,c_2} as the permutation that transforms vector $\mathbf{v} = (v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}) \in \mathbb{Z}^4$ into vector

$$T_{c_1,c_2}(\mathbf{v}) = \left(v_{c_1,c_2}, v_{c_1,\overline{c}_2}, v_{\overline{c}_1,c_2}, v_{\overline{c}_1,\overline{c}_2}\right) \in \mathbb{Z}^4.$$

Then, the following equivalence holds. For any bits b_1, b_2, c_1, c_2 and any vector $\mathbf{v} = (v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}) \in \mathbb{Z}^4$,

$$\mathbf{v} = \mathsf{ext}(b_1, b_2) \iff T_{c_1,c_2}(\mathbf{v}) = \mathsf{ext}(b_1 \oplus c_1, b_2 \oplus c_2). \tag{5}$$

As a result, to prove that a bit has the form $x_1 \cdot x_2$, one can extend it to vector $\mathsf{ext}(x_1, x_2)$, then permute the latter using T_{c_1,c_2}, where c_1, c_2 are uniformly random bits. One then demonstrates to the verifier that the permuted vector is $\mathsf{ext}(x_1 \oplus c_1, x_2 \oplus c_2)$. This convinces the verifier that the original vector, i.e., $\mathsf{ext}(x_1, x_2)$, is well-formed, while learning no additional information about x_1 and x_2, thanks to the randomness of c_1 and c_2.

3 Code-Based Statistically Hiding Commitments With Companion Zero-Knowledge Protocols

In Section 3.1, we develop the AFS hash function to obtain a code-based computationally binding and statistically hiding commitment scheme. Then, in Section 3.2, we build up our techniques for proving in zero-knowledge the correct encoding of binary strings into regular words. Relying on these techniques, we put forward in Section 3.3 a ZKAoK of a valid opening for the given commitment scheme. This building block will further be used in other advances constructions, which will be presented later in the paper.

3.1 Our Construction

Given security parameter λ, choose $n = \mathcal{O}(\lambda)$, $k \geq n + 2\lambda + \mathcal{O}(1)$. Let the message space be $\mathcal{M} = \{0,1\}^L$, and let c be a constant dividing L and k. Let $m_0 = 2^c \cdot L/c$, $m_1 = 2^c \cdot k/c$ and $m = m_0 + m_1$. Our scheme works as follows.

- KGen: Sample $\mathbf{B}_0 \xleftarrow{\$} \mathbb{Z}_2^{n \times m_0}$, $\mathbf{B}_1 \xleftarrow{\$} \mathbb{Z}_2^{n \times m_1}$, and output commitment key $pk = \mathbf{B} = [\mathbf{B}_0 \mid \mathbf{B}_1] \in \mathbb{Z}_2^{n \times m}$.
- Com: On input a message $\mathbf{x} \in \{0,1\}^L$ and commitment key pk, sample randomness $\mathbf{s} \xleftarrow{\$} \{0,1\}^k$, compute $\mathbf{c} = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}) \in \mathbb{Z}_2^n$, and output commitment $\mathbf{c}$ together with opening $\mathbf{s}$. Here, $\mathsf{RE}(\cdot)$ is the regular encoding function from Section 2.1.
- Open: On input commitment key pk, a commitment $\mathbf{c} \in \mathbb{Z}_2^n$, a message $\mathbf{x} \in \{0,1\}^L$ and an opening $\mathbf{s} \in \{0,1\}^k$, output 1 if $\mathbf{c} = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s})$, or 0 otherwise.

One can check that the proposed scheme is correct. Let us now prove the computationally binding and statistically hiding properties.

Lemma 1. *The scheme is computationally binding, assuming the hardness of the 2-$\mathsf{RNSD}_{n,L+k,c}$ problem.*

Proof. Suppose that the adversary outputs $\mathbf{c}, \mathbf{x}, \mathbf{x}', \mathbf{s}, \mathbf{s}'$ such that $\mathbf{x} \neq \mathbf{x}'$ and

$$\mathbf{c} = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}) = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}') \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}').$$

Let $\mathbf{z} = \mathsf{RE}(\mathbf{x}) \oplus \mathsf{RE}(\mathbf{x}') \neq \mathbf{0}$ and $\mathbf{y} = \mathsf{RE}(\mathbf{s}) \oplus \mathsf{RE}(\mathbf{s}')$. Then $\mathbf{B}_0 \cdot \mathbf{z} + \mathbf{B}_1 \cdot \mathbf{y} = \mathbf{0}$. Next, let $\mathbf{t} = (\mathbf{z} \| \mathbf{y})$ then we have $\mathbf{t} \neq \mathbf{0}$, $\mathbf{t} \in \text{2-}\mathsf{Regular}(L + k, c)$ and $\mathbf{B} \cdot \mathbf{t} = \mathbf{0}$. In other words, $\mathbf{t}$ is a solution to the 2-$\mathsf{RNSD}_{n,L+k,c}$ problem associated with uniformly random matrix $\mathbf{B}$. □

The statistically hiding property of the scheme is based on the following leftover hash lemma.

Lemma 2 (Leftover hash lemma, adapted from [39]). *Let D be a distribution over $\{0,1\}^t$ with min-entropy k. For any $\epsilon > 0$ and $n \leq k - 2\log(1/\epsilon) - \mathcal{O}(1)$, the statistical distance between the joint distribution of $(\mathbf{B}, \mathbf{B} \cdot \mathbf{t})$, where $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times t}$ and $\mathbf{t} \in \{0,1\}^t$ is drawn from distribution D, and the uniform distribution over $\mathbb{Z}_2^{n \times t} \times \mathbb{Z}_2^n$ is at most ϵ.*

If $\mathbf{t}$ is uniformly random over $\{0,1\}^k$, then the distribution of $\mathsf{RE}(\mathbf{t})$ over $\{0,1\}^{m_1}$ has min-entropy exactly k. Since $k \geq n + 2\lambda + \mathcal{O}(1)$, the distribution of $\mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s})$ is at most $2^{-\lambda}$-far from the uniform distribution over $\mathbb{Z}_2^n$. Then, for any $\mathbf{x} \in \{0,1\}^L$, the distribution of $\mathbf{c} = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s})$ is statistically close to uniform over $\mathbb{Z}_2^n$. As a result, the scheme is statistically hiding.

Remark 1. As for the lattice-based commitment scheme from [47], we can extend the message space of our scheme to $\{0,1\}^L$ for arbitrary $L = \mathsf{poly}(\lambda)$ using the Merkle-Damgard technique together with the AFS hash function.

3.2 Techniques for Handling Well-Formed Regular Words

In our ZKAoK of a valid opening for the given commitment scheme, which will be presented in Section 3.3, as well as in all subsequent argument systems of this paper, we need a special mechanism allowing to prove the correctness of the (non-linear) encoding process from $\mathbf{v} \in \{0,1\}^m$, for some $m \in \mathbb{Z}^+$, to regular word $\mathbf{y} = \mathsf{RE}(\mathbf{v}) \in \mathsf{Regular}(m, c)$, where c divides m. To this end, we introduce the following notations and techniques.

- Let $c \in \mathbb{Z}^+$. For every $\mathbf{s} = (s_1, \ldots, s_c) \in \{0,1\}^c$, define the permutation $E_{\mathbf{s}}$ that transforms vector $\mathbf{x} = (x_{0,0,\ldots,0}, \ldots, x_{i_1,\ldots,i_c}, \ldots, x_{1,1,\ldots,1}) \in \{0,1\}^{2^c}$ into vector $E_{\mathbf{s}}(\mathbf{x}) = (x'_{0,0,\ldots,0}, \ldots, x'_{1,1,\ldots,1})$, where for each $(i_1, \ldots, i_c) \in \{0,1\}^c$, we have $x_{i_1,\ldots,i_c} = x'_{i_1 \oplus s_1, \ldots, i_c \oplus s_c}$.

  Note that, for any $\mathbf{s}, \mathbf{v} \in \{0,1\}^c$, we have:

$$\mathbf{x} = \Delta_{2^c}(\mathbf{v}) \iff E_{\mathbf{s}}(\mathbf{x}) = \Delta_{2^c}(\mathbf{v} \oplus \mathbf{s}). \tag{6}$$

- For $\mathbf{t} = (\mathbf{t}_1 \| \ldots \| \mathbf{t}_{n/c}) \in \{0,1\}^n$ consisting of n/c blocks of length c, define the permutation $E'_{\mathbf{t}}$ that transforms vector $\mathbf{y} = (\mathbf{y}_1 \| \ldots \| \mathbf{y}_{n/c}) \in \{0,1\}^{2^c \cdot n/c}$ consisting of n/c blocks of length 2^c into vector of the following form $E'_{\mathbf{t}}(\mathbf{y}) = \big(E_{\mathbf{t}_1}(\mathbf{y}_1) \| \ldots \| E_{\mathbf{t}_{n/c}}(\mathbf{y}_{n/c})\big)$. Note that, for any $\mathbf{t}, \mathbf{v} \in \{0,1\}^n$, we have:

$$\mathbf{y} = \mathsf{RE}(\mathbf{v}) \iff E'_{\mathbf{t}}(\mathbf{y}) = \mathsf{RE}(\mathbf{v} \oplus \mathbf{t}). \tag{7}$$

The equivalence in (7) enables us to prove that $\mathbf{y}$ is the correct encoding of $\mathbf{v}$, as follows. The prover samples uniformly random $\mathbf{t}$ and demonstrates to the verifier that the right-hand side of (7) holds. The verifier is thus convinced that its left-hand side also holds, while learning no additional information about $\mathbf{v}$, thanks to the "one-time pad" $\mathbf{t}$.

3.3 ZKAoK of a Valid Opening

We now describe a ZKAoK of a valid opening for the commitment scheme from Section 3.1. Specifically, we consider the relation R_{com}, defined as:

$$R_{\mathsf{com}} = \big\{\big((\mathbf{B} = [\mathbf{B}_0 \mid \mathbf{B}_1], \mathbf{c}), \mathbf{x}, \mathbf{s}\big) : \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}) + \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}) = \mathbf{c}\big\}.$$

The protocol is realized based on the permuting technique of Section 3.2. Let $\mathbf{z} = (\mathbf{x} \parallel \mathbf{s}) \in \{0,1\}^{L+k}$ and $\mathbf{w}_{\mathsf{com}} = \mathsf{RE}(\mathbf{z}) \in \mathsf{Regular}(L+k, c) \subset \{0,1\}^{2^c \cdot (L+k)/c}$. Then the equation $\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}) + \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}) = \mathbf{c}$ can be written as $\mathbf{B} \cdot \mathbf{w}_{\mathsf{com}} = \mathbf{c}$.

Next, define the sets $\mathsf{VALID}_{\mathsf{com}} = \mathsf{Regular}(L+k, c)$ and $\mathcal{S} = \{0,1\}^{L+k}$. For $\mathbf{t} = (\mathbf{t}_1 \| \ldots \| \mathbf{t}_{(L+k)/c}) \in \mathcal{S}$ consisting of $(L+k)/c$ blocks of length c, it follows from (7) that we have

$$\mathbf{w}_{\mathsf{com}} = \mathsf{RE}(\mathbf{z}) \iff E'_{\mathbf{t}}(\mathbf{w}_{\mathsf{com}}) = \mathsf{RE}(\mathbf{z} \oplus \mathbf{t}). \tag{8}$$

Moreover, if $\mathbf{t}$ is chosen uniformly at random in $\mathcal{S}$, then $E'_{\mathbf{t}}(\mathbf{w}_{\mathsf{com}})$ is uniformly random in $\mathsf{VALID}_{\mathsf{com}}$. In other words, the conditions of (3) hold, and relation R_{com} can be reduced to an instance of R_{abstract} in Section 2.2. As a result, we can run the interactive protocol in Figure 1 with public input $(\mathbf{B}, \mathbf{c})$ and prover's witness $\mathbf{w}_{\mathsf{com}}$, and obtain a ZKAoK for R_{com}.

4 Range Arguments for Signed Fractional Numbers

In this section, we present our techniques for obtaining zero-knowledge arguments that *signed fractional numbers*, committed via the code-based commitment scheme of Section 3, belong to a (hidden or given) range. We first describe our method for handling signed fractional numbers and establish the crucial theoretical foundations of our range arguments in Section 4.1. Next, in Section 4.2, we present our protocol for proving in zero-knowledge that two committed signed fractional numbers X, Y satisfies the inequality $X \leq Y$. In Section 4.3, we then discuss how to handily derive various variants of range arguments, based on the results of Section 4.1 and 4.2.

NOTATIONS. For $a, b \in \mathbb{Z}$ and $c \in \mathbb{Q}$, we let $[a, b]$ denote the set of all integers between a and b (inclusive), and let $c \cdot [a, b]$ denote the set $\{c \cdot x \mid x \in [a, b]\}$.

4.1 A Treatment of Signed Fractional Numbers

We will work with signed fractional numbers represented in fixed-point binary format. For integers $\ell > 0, f \geq 0$, define the set

$$\begin{aligned}\mathbf{Q}\langle \ell \bullet f \rangle &= 2^{-f} \cdot [-2^{\ell+f}, 2^{\ell+f} - 1] \\ &= \Big\{ -2^{\ell} \cdot x_\ell + \sum_{i=-f}^{\ell-1} 2^i \cdot x_i \mid (x_\ell, x_{\ell-1}, \ldots, x_0, x_{-1}, \ldots, x_{-f}) \in \{0,1\}^{1+\ell+f} \Big\}.\end{aligned}$$

Each element $X \in \mathbf{Q}\langle \ell \bullet f \rangle$ can be represented as $x_\ell x_{\ell-1} \dots x_0 \bullet x_{-1} x_{-2} \dots x_{-f}$, where x_ℓ is the sign bit, $x_{\ell-1}, \dots, x_0$ are the integer bits, and $x_{-1}, \dots, x_{-f}$ are the fractional bits.

The binary vector $(x_\ell, x_{\ell-1}, \dots, x_0, x_{-1}, \dots, x_{-f}) \in \{0,1\}^{1+\ell+f}$ representing X is denoted as $\mathsf{sbin}_{\ell,f}(A)$. For notational convenience, we write $A = \mathsf{sbin}^{-1}_{\ell,f}(\mathbf{a})$ if $\mathbf{a} = \mathsf{sbin}_{\ell,f}(A)$. In this way, we have $\mathbf{Q}\langle \ell \bullet f \rangle = \{\mathsf{sbin}^{-1}_{\ell,f}(\mathbf{a}) \mid \mathbf{a} \in \{0,1\}^{1+\ell+f}\}$.

We aim to prove in zero-knowledge inequality relations among elements of $\mathbf{Q}\langle \ell \bullet f \rangle$, e.g., to prove that $X \leq Y$ for secret/committed X, Y. As we have discussed in Section 1, handling inequalities over $\mathbf{Q}\langle \ell \bullet f \rangle$ is highly non-trivial, due to 2 main reasons: the existence of the signed bit and the problem of overflows.

Our idea is to derive necessary and sufficient conditions for $X \leq Y$, with $X, Y \in \mathbf{Q}\langle \ell \bullet f \rangle$, in a way such that these conditions can be correctly and efficiently proved in zero-knowledge. Theorem 2 captures this idea via the existence of $2(\ell + f + 1)$ extra bits that are related to the bits representing X and Y via $2(\ell + f) + 3$ simple equations modulo 2. This result lays the vital foundation for the argument system we will construct in Section 4.2.

Theorem 2. *Let $X, Y \in \mathbf{Q}\langle \ell \bullet f \rangle$ and let $\mathsf{sbin}_{\ell,f}(X) = (x_\ell, \dots, x_0, x_{-1}, \dots, x_{-f})$, $\mathsf{sbin}_{\ell,f}(Y) = (y_\ell, \dots, y_0, y_{-1}, \dots, y_{-f})$. Then, $X \leq Y$ if and only if there exist bits $z_\ell, z_{\ell-1}, \dots, z_0, z_{-1}, \dots, z_{-f}$, $c_{\ell+1}, c_\ell, c_{\ell-1}, \dots, c_0, c_{-1}, \dots, c_{-f+1}$ satisfying*

$$\begin{cases} c_{-f+1} = x_{-f} \cdot z_{-f} \\ c_i = x_{i-1} \cdot z_{i-1} \ \oplus \ y_{i-1} \cdot c_{i-1} \ \oplus \ c_{i-1}, \quad \forall i \in [-f+2, \ell+1] \\ y_{-f} = x_{-f} \ \oplus \ z_{-f} \\ y_i = x_i \ \oplus \ z_i \oplus c_i, \quad \forall i \in [-f+1, \ell] \\ y_\ell = x_\ell \ \oplus \ c_{\ell+1}. \end{cases} \tag{9}$$

Before proving Theorem 2, we first introduce a few notations, definitions and a technical lemma.

Additions. To avoid the problem of overflows, we will treat elements of $\mathbf{Q}\langle \ell \bullet f \rangle$ as elements of $\mathbf{Q}\langle (\ell+2) \bullet f \rangle$. If $X \in \mathbf{Q}\langle \ell \bullet f \rangle$ with $\mathsf{sbin}_{\ell,f}(X) = (x_\ell, x_{\ell-1}, \dots, x_{-f})$ then we have

$$A = -2^\ell \cdot x_\ell + \sum_{i=-f}^{\ell-1} 2^i \cdot x_i = (-2^{\ell+2} + 2^{\ell+1} + 2^\ell) \cdot x_\ell + \sum_{i=-f}^{\ell-1} 2^i \cdot x_i,$$

and thus $\mathsf{sbin}_{\ell+2,f}(X) = (x_\ell, x_\ell, x_\ell, x_{\ell-1}, \dots, x_{-f}) \in \{0,1\}^{3+\ell+f}$.

Now, let $X, Z \in \mathbf{Q}\langle (\ell+2) \bullet f \rangle$. The addition of X and Z when executed in a conventional computer is indeed the addition of two fractional binary numbers whose decimal encodings are equal to X and Z. Such addition is formally defined as follows.

Definition 2 (Signed Fractional Additions in Binary). *Let X, Z be elements of $\mathbf{Q}\langle (\ell+2) \bullet f \rangle$. The sum $\mathsf{sbin}_{\ell+2,f}(X) \ \boxplus_{\ell+2,f} \ \mathsf{sbin}_{\ell+2,f}(Z)$ is a vector*

$\mathbf{y} = (y_{\ell+2}, y_{\ell+1}, \ldots, y_{-f})$ *associated with a vector* $\mathbf{c} = (c_{\ell+2}, c_{\ell+1}, \ldots, c_{-f+1})$ *such that*

$$\begin{cases} c_{-f+1} = x_{-f} \cdot z_{-f} \\ c_i = x_{i-1} \cdot z_{i-1} \ \oplus \ y_{i-1} \cdot c_{i-1} \ \oplus \ c_{i-1}, \quad \forall i \in [-f+2, \ell+2] \\ y_{-f} = x_{-f} \ \oplus \ z_{-f} \\ y_i = x_i \ \oplus \ z_i \ \oplus \ c_i, \quad \forall i \in [-f+1, \ell+2]. \end{cases}$$

The above definition is similar to computing sum of two's complement integers up to a scaled factor [65, Section 3.2]. Intuitively, Definition 2 can be viewed as a binary addition of two binary numbers $\mathbf{x}, \mathbf{y}$ while sequence $\mathbf{c}$ plays a role as a sequence of carries computed in each step (the last carry-out bit is discarded).

	$c_{\ell+2}$	$c_{\ell+1}$	$\ldots$	c_1	c_0	$\ldots$	c_{-f+1}	0
	$x_{\ell+2}$	$x_{\ell+1}$	$\ldots$	x_1	x_0	$\ldots$	x_{-f+1}	x_{-f}
$+$	$z_{\ell+2}$	$z_{\ell+1}$	$\ldots$	z_1	z_0	$\ldots$	z_{-f+1}	z_{-f}
	$y_{\ell+2}$	$y_{\ell+1}$	$\ldots$	y_1	y_0	$\ldots$	y_{-f+1}	y_{-f}

It is clear that $y_{-f} = x_{-f} \ \oplus \ z_{-f}$ and, $\forall i \in [-f+1, \ell+2]$, $y_i = x_i \ \oplus z_i \ \oplus c_i$. Regarding computing carries, $c_i = (x_{i-1} \cdot z_{i-1}) \oplus (c_{i-1} \cdot (x_{i-1} \oplus z_{i-1}))$, $\forall i \in [-f+1, \ell+2]$ and $c_{-f} = 0$. It is easy to verify that, $\forall i \in [-f+1, \ell+2]$, $c_i = x_{i-1} \cdot z_{i-1} \ \oplus \ y_{i-1} \cdot c_{i-1} \ \oplus \ c_{i-1}$.

Overflows. Note that $\mathbf{Q}\langle(\ell+2) \bullet f\rangle = 2^{-f} \cdot [-2^{2+\ell+f}, 2^{2+\ell+f} - 1]$. For $X, Y \in \mathbf{Q}\langle(\ell+2) \bullet f\rangle$, where $X+Y$ falls out of the range $2^{-f} \cdot [-2^{2+\ell+f}, 2^{2+\ell+f} - 1]$, i.e., $X + Y < -2^{2+\ell}$ or $X + Y > 2^{2+\ell} - 2^{-f}$, the addition would yield an overflow. This phenomenon is formally defined as follows.

Definition 3 (Overflows). *Let* $X, Y \in \mathbf{Q}\langle(\ell+2) \bullet f\rangle$ *and let* $\mathbf{x} = \mathsf{sbin}_{\ell+2,f}(X)$, $\mathbf{y} = \mathsf{sbin}_{\ell+2,f}(Y)$. *The signed fractional addition* $\mathbf{x} \ \boxplus_{\ell+2,f} \ \mathbf{y}$ *is called to cause an overflow (with respect to* $\ell + 2$ *and* f*) if and only if* $X + Y < -2^{2+\ell}$ *or* $X + Y > 2^{2+\ell} - 2^{-f}$.

The following lemma implies that, if we are given $X, Y \in \mathbf{Q}\langle \ell \bullet f\rangle$ but we compute their sum over $\mathbf{Q}\langle(\ell+2) \bullet f\rangle$, then we can avoid overflows, and hence, can reliably capture the inequality $X \leq Y$ via addition.

Lemma 3. *Let* $X, Y \in \mathbf{Q}\langle \ell \bullet f\rangle \subset \mathbf{Q}\langle(\ell+2) \bullet f\rangle$ *. Then* $X \leq Y$ *if and only if* $Z = Y - X \in 2^{-f} \cdot [0, 2^{1+\ell+f} - 1] \subset \mathbf{Q}\langle(\ell+2) \bullet f\rangle$ *and* $\mathsf{sbin}_{\ell+2,f}(X) \boxplus_{\ell+2,f} \mathsf{sbin}_{\ell+2,f}(Z)$ *does not cause an overflow. As a corollary,* $\mathsf{sbin}_{\ell+2,f}(X) \ \boxplus_{\ell+2,f} \ \mathsf{sbin}_{\ell+2,f}(Z) = \mathsf{sbin}_{\ell+2,f}(Y)$ *and* $\mathsf{sbin}_{\ell+2,f}(Z) = (0, 0, z_\ell, z_{\ell-1}, \ldots, z_{-f})$.

Proof. Assume that $X \leq Y$ and let $Z = Y - X$ (over $\mathbb{Q}$). Since we have $X, Y \in 2^{-f} \cdot [-2^{\ell+f}, 2^{\ell+f} - 1]$, it follows that

$$Z \in 2^{-f} \cdot [0, 2^{1+\ell+f} - 1] \subset 2^{-f} \cdot [-2^{\ell+2+f}, 2^{\ell+2+f} - 1] = \mathbf{Q}\langle(\ell+2) \bullet f\rangle.$$

Furthermore, $X + Z \in 2^{-f} \cdot [-2^{\ell+f}, 2^{1+\ell+f} + 2^{\ell+f} - 2]$. Hence, we have $-2^{\ell+2} < X + Z < 2^{\ell+2} - 2^{-f}$. As a result, the addition $\mathsf{sbin}_{\ell+2,f}(X) \ \boxplus_{\ell+2,f} \ \mathsf{sbin}_{\ell+2,f}(Z)$ does not cause an overflow and produces the correct result $\mathsf{sbin}_{\ell+2,f}(Y)$.

For the reverse direction, if $\mathsf{sbin}_{\ell+2,f}(X) \boxplus_{\ell+2,f} \mathsf{sbin}_{\ell+2,f}(Z)$ does not cause an overflow, then $\mathsf{sbin}_{\ell+2,f}(X) \boxplus_{\ell+2,f} \mathsf{sbin}_{\ell+2,f}(Z) = \mathsf{sbin}_{\ell+2,f}(Y)$ and hence $X + Z = Y$. Moreover, since $Z \in 2^{-f} \cdot [0, 2^{1+\ell+f} - 1]$, we have $Z \geq 0$. It then follows that $X \leq Y$.

To see that the first two bits of $\mathsf{sbin}_{\ell+2,f}(Z)$ are 0, let $Z' := 2^f \cdot Z$. Then Z' is a non-negative integer in the range $[0, 2^{1+\ell+f} - 1] \subset [-2^{2+\ell+f}, 2^{2+\ell+f} - 1]$ which needs exactly $1 + \ell + f$ bits to store in place of $3 + \ell + f$ bits. Therefore, $\mathsf{sbin}_{\ell+2,f}(Z') = (0, 0, z'_{\ell+f}, z'_{\ell+f-1}, \ldots, z'_0)$ and thus $\mathsf{sbin}_{\ell+2,f}(Z) = (0, 0, z_\ell, z_{\ell-1}, \ldots, z_{-f})$. □

We are now ready to prove Theorem 2.

Proof of Theorem 2. We first assume that $X, Y \in \mathbf{Q}\langle \ell \bullet f \rangle$ and $X \leq Y$. Let $\mathbf{x} = \mathsf{sbin}_{\ell+2,f}(X) = (x_\ell, x_\ell, x_\ell, x_{\ell-1}, \ldots, x_{-f})$ and $\mathbf{y} = \mathsf{sbin}_{\ell+2,f}(Y) = (y_\ell, y_\ell, y_\ell, y_{\ell-1}, \ldots, y_{-f})$. By Lemma 3, $Z = Y - X \in \mathbf{Q}\langle (\ell + 2) \bullet f \rangle$ such that $\mathsf{sbin}_{\ell+2,f}(Z) = (0, 0, z_\ell, \ldots, z_{-f})$ and $\mathsf{sbin}_{\ell+2,f}(X) \boxplus_{\ell+2,f} \mathsf{sbin}_{\ell+2,f}(Z) = \mathsf{sbin}_{\ell+2,f}(Y)$ where the signed fractional addition does not cause an overflow. Let $\mathbf{c}$ be the sequence of carries as in Definition 2. It follows that

$$\begin{cases} c_{-f+1} = x_{-f} \cdot z_{-f} \\ c_i = x_{i-1} \cdot z_{i-1} \ \oplus \ y_{i-1} \cdot c_{i-1} \ \oplus \ c_{i-1}, \quad \forall i \in [-f+2, \ell+1] \\ c_{\ell+2} = y_\ell \cdot c_{\ell+1} \ \oplus \ c_{\ell+1} \\ y_{-f} = x_{-f} \ \oplus \ z_{-f} \\ y_i = x_i \ \oplus \ z_i \ \oplus \ c_i, \quad \forall i \in [-f+1, \ell] \\ y_\ell = x_\ell \ \oplus \ c_{\ell+i}, \quad \forall i \in [1, 2]. \end{cases}$$

We can deduce that $c_{\ell+1} = y_\ell \oplus x_\ell = c_{\ell+2}$. Hence, we obtain the system of equations in (9).

We now prove the reverse direction. If there exists bits $z_\ell, z_{\ell-1}, \ldots, z_{-f}, c_{\ell+1}, c_\ell, \ldots, c_{-f+1}$ satisfying (9), then we can construct the following vectors: $\mathbf{x} = (x'_{\ell+2}, x'_{\ell+1}, \ldots, x'_{-f}) = (x_\ell, x_\ell, x_\ell, x_{\ell-1}, \ldots, x_{-f}), \mathbf{y} = (y'_{\ell+2}, y'_{\ell+1}, \ldots, y'_{-f}) = (y_\ell, y_\ell, y_\ell, y_{\ell-1}, \ldots, y_{-f}), \mathbf{z} = (z'_{\ell+2}, z'_{\ell+1}, \ldots, z'_{-f}) = (0, 0, z_\ell, z_{\ell-1}, \ldots, z_{-f})$, and $\mathbf{c} = (c'_{\ell+2}, c'_{\ell+1}, \ldots, c'_{-f+1}) = (c_{\ell+1}, c_{\ell+1}, c_\ell, c_{\ell-1}, \ldots, c_{-f+1})$. From the assumption, we deduce that

$$\begin{cases} c'_{-f+1} = x'_{-f} \cdot z'_{-f} \\ c'_i = x'_{i-1} \cdot z'_{i-1} \ \oplus \ y'_{i-1} \cdot c'_{i-1} \ \oplus \ c'_{i-1}, \quad \forall i \in [-f+2, \ell+1] \\ y'_{-f} = x'_{-f} \ \oplus \ z'_{-f} \\ y'_i = x'_i \ \oplus \ z'_i \ \oplus \ c'_i, \quad \forall i \in [-f+1, \ell+2]. \end{cases}$$

It remains to show that $c'_{\ell+2} = x'_{\ell+1} \cdot z'_{\ell+1} \ \oplus \ y'_{\ell+1} \cdot c'_{\ell+1} \ \oplus \ c'_{\ell+1}$. We have

$$\begin{aligned} y'_{\ell+1} \cdot c'_{\ell+1} \oplus c'_{\ell+1} &= y_\ell \cdot c_{\ell+1} \oplus c_{\ell+1} \\ &= y_\ell \cdot (x_\ell \cdot z_\ell \oplus y_\ell \cdot c_\ell \oplus c_\ell) \oplus c_{\ell+1} = y_\ell \cdot x_\ell \cdot z_\ell \oplus c_{\ell+1}. \end{aligned}$$

We claim that $y_\ell \cdot x_\ell \cdot z_\ell = 0$. To prove this claim, we assume by contradiction that $y_\ell \cdot x_\ell \cdot z_\ell = 1$. This is equivalent to $x_\ell = z_\ell = y_\ell = 1$. Hence $c'_\ell = 1$

because $y'_\ell = x'_\ell \oplus z'_\ell \oplus c'_\ell$. This also implies that $x'_{\ell+1} = 1$ and $c'_{\ell+1} = 1$ because $c'_{\ell+1} = x'_\ell \cdot z'_\ell \oplus y'_\ell \cdot c'_\ell \oplus c'_\ell$. Since $y'_{\ell+1} = x'_{\ell+1} \oplus z'_{\ell+1} \oplus c'_{\ell+1}$ and $z'_{\ell+1} = 0$, we have $y'_{\ell+1} = 0 \neq y'_\ell$, which is a contradiction. Therefore, the claim follows. Thus $y'_{\ell+1} \cdot c'_{\ell+1} \oplus c'_{\ell+1} = c_{\ell+1}$ and, since $z'_{\ell+1} = 0$, we deduce that $x'_{\ell+1} \cdot z'_{\ell+1} \oplus y'_{\ell+1} \cdot c'_{\ell+1} \oplus c'_{\ell+1} = c_{\ell+1} = c'_{\ell+2}$.

By Definition 2, the above facts imply that $\mathbf{x} \boxplus_{\ell+2,f} \mathbf{z} = \mathbf{y}$. It is clear that $X = \mathsf{sbin}^{-1}_{\ell+2,f}(\mathbf{x})$ and $Y = \mathsf{sbin}^{-1}_{\ell+2,f}(\mathbf{y}) \in \mathbf{Q}\langle \ell \bullet f \rangle$. Let $Z := \mathsf{sbin}^{-1}_{\ell+2,f}(\mathbf{z}) \in 2^{-f} \cdot [0, 2^{1+\ell+f} - 1]$. By Definition 3, the addition $\mathbf{x} \boxplus_{\ell+2,f} \mathbf{z}$ does not cause an overflow. Therefore, $X + Z = Y$, and since $Z \geq 0$, we obtain that $X \leq Y$. This completes the proof. □

We also obtain necessary and sufficient conditions for strict inequalities of elements in $\mathbf{Q}\langle \ell \bullet f \rangle$, which allow us to handle those of the form $X < Y$ in zero-knowledge. This result is stated in Theorem 3, whose proof follows the same lines as that of Theorem 2, and is thus omitted.

Theorem 3. *Let $X, Y \in \mathbf{Q}\langle \ell \bullet f \rangle$ and let $\mathsf{sbin}_{\ell,f}(X) = (x_\ell, \ldots, x_0, x_{-1}, \ldots, x_{-f})$, $\mathsf{sbin}_{\ell,f}(Y) = (y_\ell, \ldots, y_0, y_{-1}, \ldots, y_{-f})$. Then, $X < Y$ if and only if there exist bits $z_\ell, z_{\ell-1}, \ldots, z_0, z_{-1}, \ldots, z_{-f}$, $c_{\ell+1}, c_\ell, c_{\ell-1}, \ldots, c_0, c_{-1}, \ldots, c_{-f+1}$ satisfying*

$$\begin{cases} c_{-f+1} = x_{-f} \cdot z_{-f} \ \oplus \ y_{-f} \ \oplus \ 1 \\ c_i = x_{i-1} \cdot z_{i-1} \ \oplus \ y_{i-1} \cdot c_{i-1} \ \oplus \ c_{i-1}, \quad \forall i \in [-f+2, \ell+1] \\ y_{-f} = x_{-f} \ \oplus \ z_{-f} \ \oplus \ 1 \\ y_i = x_i \ \oplus \ z_i \ \oplus \ c_i, \quad \forall i \in [-f+1, \ell] \\ y_\ell = x_\ell \ \oplus \ c_{\ell+1}. \end{cases}$$

4.2 Proving Inequalities Between Committed Elements of $\mathbf{Q}\langle \ell \bullet f \rangle$

Let $\ell > 0, f \geq 0$ be integers, and let $L = 1 + \ell + f$. Consider the code-based commitment scheme of Section 3 with parameters n, k, c, m_0, m_1, m and L and commitment key $\mathbf{B} = [\mathbf{B}_0 \mid \mathbf{B}_1] \in \mathbb{Z}_2^{n \times m}$.

Let $X, Y \in \mathbf{Q}\langle \ell \bullet f \rangle$, whose binary representations

$$\mathbf{x} = \mathsf{sbin}_{\ell,f}(X) = (x_\ell, x_{\ell-1}, \ldots, x_0, x_{-1}, \ldots, x_{-f}) \in \{0,1\}^L,$$
$$\mathbf{y} = \mathsf{sbin}_{\ell,f}(Y) = (y_\ell, y_{\ell-1}, \ldots, y_0, y_{-1}, \ldots, y_{-f}) \in \{0,1\}^L.$$

are committed as

$$\mathbf{e}_x = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}_x) \in \mathbb{Z}_2^n, \quad \mathbf{e}_y = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{y}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}_y) \in \mathbb{Z}_2^n,$$

respectively. Our goal is to design an argument system allowing the prover to convince the verifier in zero-knowledge that the vectors $\mathbf{x}, \mathbf{y}$ committed in $\mathbf{e}_x, \mathbf{e}_y$ satisfy $\mathsf{sbin}^{-1}_{\ell,f}(\mathbf{x}) \leq \mathsf{sbin}^{-1}_{\ell,f}(\mathbf{y})$, i.e., they represent numbers $X, Y \in \mathbf{Q}\langle \ell \bullet f \rangle$ such that $X \leq Y$. Formally, we will build a ZKAoK for the relation R_{ineq} defined as follows.

$$R_{\mathsf{ineq}} = \big\{\big((\mathbf{B} = [\mathbf{B}_0 \mid \mathbf{B}_1], \mathbf{e}_x, \mathbf{e}_y), \mathbf{x}, \mathbf{y}, \mathbf{s}_x, \mathbf{s}_y\big) : \mathsf{sbin}^{-1}_{\ell,f}(\mathbf{x}) \leq \mathsf{sbin}^{-1}_{\ell,f}(\mathbf{y}) \ \wedge \\ \mathbf{e}_x = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}_x) \ \wedge \ \mathbf{e}_y = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{y}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}_y)\big\}.$$

To prove in zero-knowledge that the inequality $\mathsf{sbin}^{-1}_{\ell,f}(\mathbf{x}) \leq \mathsf{sbin}^{-1}_{\ell,f}(\mathbf{y})$ holds, we rely on the results established in Section 4.1. Specifically, based on Theorem 2, we can equivalently prove the existence of bits $z_\ell, z_{\ell-1}, \ldots, z_0, z_{-1}, \ldots, z_{-f}, c_{\ell+1}, c_\ell, c_{\ell-1}, \ldots, c_0, c_{-1}, \ldots, c_{-f+1}$ satisfying the following $2(\ell+f)+3 = 2L+1$ equations modulo 2:

$$\begin{cases} c_{-f+1} \ \oplus \ x_{-f} \cdot z_{-f} = 0, \\ c_i \ \oplus \ x_{i-1} \cdot z_{i-1} \ \oplus \ y_{i-1} \cdot c_{i-1} \ \oplus \ c_{i-1} = 0, \ \ \forall i \in [-f+2, \ell+1] \\ y_{-f} \ \oplus \ x_{-f} \ \oplus \ z_{-f} = 0, \\ y_i \ \oplus \ x_i \ \oplus \ z_i \ \oplus \ c_i = 0, \ \ \forall i \in [-f+1, \ell] \\ y_\ell \ \oplus \ x_\ell \ \oplus \ c_{\ell+1} = 0. \end{cases} \tag{10}$$

Now, to handle (10) in zero-knowledge, we can use the extending-then-permuting techniques of Section 2.3. To this end, we first perform the following extensions for each $i \in [-f, \ell]$

$$x_i \mapsto \mathbf{x}_i = \mathsf{enc}(x_i), \ y_i \mapsto \mathbf{y}_i = \mathsf{enc}(y_i), \ z_i \mapsto \mathbf{z}_i = \mathsf{enc}(z_i), \ c_{i+1} \mapsto \mathsf{enc}(c_{i+1}),$$

as well as the following extensions

$$\forall i \in [-f+1, \ell+1] : \ x_{i-1} \cdot z_{i-1} \mapsto \mathbf{t}_{i-1} = \mathsf{ext}(x_{i-1}, z_{i-1}); \\ \forall i \in [-f+2, \ell+1] : \ y_{i-1} \cdot c_{i-1} \mapsto \mathbf{g}_{i-1} = \mathsf{ext}(y_{i-1}, c_{i-1}).$$

Let $\mathbf{M}_2 = (0,1) \in \mathbb{Z}_2^{1\times 2}$ and $\mathbf{M}_4 = (0,0,0,1) \in \mathbb{Z}_2^{1\times 4}$. Then (10) can be rewritten as

$$\begin{cases} \mathbf{M}_2 \cdot \mathbf{c}_{-f+1} \ \oplus \ \mathbf{M}_4 \cdot \mathbf{t}_{-f} = 0, \\ \mathbf{M}_2 \cdot \mathbf{c}_i \ \oplus \ \mathbf{M}_4 \cdot \mathbf{t}_{i-1} \ \oplus \ \mathbf{M}_4 \cdot \mathbf{g}_{i-1} \ \oplus \ \mathbf{M}_2 \cdot \mathbf{c}_{i-1} = 0, \ \ \forall i \in [-f+2, \ell+1] \\ \mathbf{M}_2 \cdot \mathbf{y}_{-f} \ \oplus \ \mathbf{M}_2 \cdot \mathbf{x}_{-f} \ \oplus \ \mathbf{M}_2 \cdot \mathbf{z}_{-f} = 0, \\ \mathbf{M}_2 \cdot \mathbf{y}_i \ \oplus \ \mathbf{M}_2 \cdot \mathbf{x}_i \ \oplus \ \mathbf{M}_2 \cdot \mathbf{z}_i \ \oplus \ \mathbf{M}_2 \cdot \mathbf{c}_i = 0, \ \ \forall i \in [-f+1, \ell] \\ \mathbf{M}_2 \cdot \mathbf{y}_\ell \ \oplus \ \mathbf{M}_2 \cdot \mathbf{x}_\ell \ \oplus \ \mathbf{M}_2 \cdot \mathbf{c}_{\ell+1} = 0, \end{cases}$$

which then can be combined via linear algebra into one equation of the form $\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{0}$, where $\mathbf{M}_0 \in \mathbb{Z}_2^{(2L+1)\times 16L}$ is a public matrix, and $\mathbf{w}_0 \in \{0,1\}^{16L}$ has the form:

$$\mathbf{w}_0 = \ \big(\mathbf{x}_\ell \parallel \ \ldots \ \parallel \mathbf{x}_{-f} \parallel \mathbf{y}_\ell \parallel \ \ldots \ \parallel \mathbf{y}_{-f} \parallel \mathbf{z}_\ell \parallel \ \ldots \ \parallel \mathbf{z}_{-f} \parallel \mathbf{c}_{\ell+1} \parallel \ \ldots \ \parallel \mathbf{c}_{-f+1} \parallel \\ \mathbf{t}_\ell \parallel \ \ldots \ \parallel \mathbf{t}_{-f} \parallel \mathbf{g}_{\ell+1} \parallel \ \ldots \ \parallel \mathbf{g}_{-f+1}\big). \tag{11}$$

Next, by combining equation $\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{0}$ with the two equations underlying the commitments $\mathbf{e}_x, \mathbf{e}_y$, we can obtain a unified equation of the form $\mathbf{M} \cdot \mathbf{w} =$

$\mathbf{v}$, where $\mathbf{M} \in \mathbb{Z}_2^{(2L+2n+1)\times(16L+2m)}$ and $\mathbf{v} \in \mathbb{Z}_2^{2L+2n+1}$ are public, and $\mathbf{w} \in \{0,1\}^{16L+2m}$ has the form

$$\mathbf{w} = \big(\ \mathbf{w}_0 \parallel \mathsf{RE}(\mathbf{x}) \parallel \mathsf{RE}(\mathbf{y}) \parallel \mathsf{RE}(\mathbf{s}_x) \parallel \mathsf{RE}(\mathbf{s}_y)\ \big). \tag{12}$$

At this point, we have translated our task into proving knowledge of a well-formed vector $\mathbf{w} \in \{0,1\}^{16L+2m}$ satisfying equation $\mathbf{M} \cdot \mathbf{w} = \mathbf{v}$. We next will reduce the latter task to an instance of the abstraction of Stern's protocol in Section 2.2. To this end, we will specify the sets $\mathsf{VALID}_{\mathsf{ineq}}, \mathcal{S}$ and permutations $\{\Gamma_\phi : \phi \in \mathcal{S}\}$ that meet the requirements in (3).

Define $\mathsf{VALID}_{\mathsf{ineq}}$ as the set of all vectors $\mathbf{w} \in \{0,1\}^{16L+2m}$ that has the form (12), where $\mathbf{s}_x, \mathbf{s}_y \in \{0,1\}^k$ and

- $\mathbf{x} = (x_\ell, \ldots, x_0, \ldots, x_{-f}) \in \{0,1\}^L$, $\mathbf{y} = (y_\ell, \ldots, y_0, \ldots, y_{-f}) \in \{0,1\}^L$;
- $\mathbf{w}_0$ has the form (11), where: for each $i \in [-f, \ell]$, there exist bits z_i, c_{i+1} satisfying
  - (i) For each $i \in [-f, \ell]$, one has $\mathbf{x}_i = \mathsf{enc}(x_i)$, $\mathsf{y}_i = \mathsf{enc}(y_i)$, $\mathbf{z}_i = \mathsf{enc}(z_i)$, and $\mathbf{c}_{i+1} = \mathsf{enc}(c_{i+1})$;
  - (ii) For each $i \in [-f+1, \ell+1]$, one has $\mathbf{t}_{i-1} = \mathsf{ext}(x_{i-1}, z_{i-1})$;
  - (iii) For each $i \in [-f+2, \ell+1]$, one has $\mathbf{g}_{i-1} = \mathsf{ext}(y_{i-1}, c_{i-1})$.

It can be observed that the vector $\mathbf{w}$ we obtained from the above transformations is an element of this tailored set $\mathsf{VALID}_{\mathsf{ineq}}$. Next, we will employ the permuting techniques from Section 2.3 and Section 3.2 to handle the special constraint of $\mathbf{w}$.

Define $\mathcal{S}$ as the set $\{0,1\}^{4L+2k}$ and for each element $\phi \in \mathcal{S}$ of the form

$$\phi = \big(b_{x,\ell}, \ldots, b_{x,-f}, b_{y,\ell}, \ldots, b_{y,-f}, b_{z,\ell}, \ldots, b_{z,-f}, b_{c,\ell+1}, \ldots, b_{c,-f+1}, \mathbf{b}_{s,x}, \mathbf{b}_{s,y}\big),$$

where $\mathbf{b}_{s,x}, \mathbf{b}_{s,y} \in \{0,1\}^k$, define the permutation Γ_ϕ that, when applying to vector $\mathbf{t} \in \mathbb{Z}^{16L+2m}$, whose blocks are denoted as

$$\begin{aligned}\big(\ &\mathbf{x}_\ell \parallel \ldots \parallel \mathbf{x}_{-f} \parallel \mathbf{y}_\ell \parallel \ldots \parallel \mathbf{y}_{-f} \parallel \mathbf{z}_\ell \parallel \ldots \parallel \mathbf{z}_{-f} \parallel \mathbf{c}_{\ell+1} \parallel \ldots \parallel \mathbf{c}_{-f+1} \parallel \\ &\mathbf{t}_\ell \parallel \ldots \parallel \mathbf{t}_{-f} \parallel \mathbf{g}_{\ell+1} \parallel \ldots \parallel \mathbf{g}_{-f+1} \parallel \mathsf{RE}(\mathbf{x}) \parallel \mathsf{RE}(\mathbf{y}) \parallel \mathsf{RE}(\mathbf{s}_x) \parallel \mathsf{RE}(\mathbf{s}_y)\ \big),\end{aligned}$$

it transforms $\mathbf{t}$ as follows:

- $\forall i \in [-f, \ell]$: $\mathbf{x}_i \mapsto F_{b_{x,i}}(\mathbf{x}_i)$; $\mathbf{y}_i \mapsto F_{b_{y,i}}(\mathbf{y}_i)$; $\mathbf{z}_i \mapsto F_{b_{z,i}}(\mathbf{z}_i)$; $\mathbf{c}_{i+1} \mapsto F_{b_{c,i+1}}(\mathbf{c}_{i+1})$
- $\forall i \in [-f+1, \ell+1]$: $\mathbf{t}_{i-1} \mapsto T_{b_{x,i-1}, b_{z,i-1}}(\mathbf{t}_{i-1})$
- $\forall i \in [-f+2, \ell+1]$: $\mathbf{g}_{i-1} \mapsto T_{b_{y,i-1}, b_{c,i-1}}(\mathbf{g}_{i-1})$
- $\mathsf{RE}(\mathbf{x}) \mapsto E'_{\mathbf{b}_x}(\mathsf{RE}(\mathbf{x}))$, where $\mathbf{b}_x = (b_{x,\ell}, \ldots, b_{x,-f})$.
- $\mathsf{RE}(\mathbf{y}) \mapsto E'_{\mathbf{b}_y}(\mathsf{RE}(\mathbf{y}))$, where $\mathbf{b}_y = (b_{y,\ell}, \ldots, b_{y,-f})$.
- $\mathsf{RE}(\mathbf{s}_x) \mapsto E'_{\mathbf{b}_{s,x}}(\mathsf{RE}(\mathbf{s}_x))$, $\mathsf{RE}(\mathbf{s}_y) \mapsto E'_{\mathbf{b}_{s,y}}(\mathsf{RE}(\mathbf{s}_y))$.

Based on the equivalences observed in (4), (5) and (7), one can verify that the requirements in (3) are satisfied. In other words, we have reduced the considered relation R_{ineq} to an instance of the relation R_{abstract} in Section 2.2.

The interactive protocol. Given the above preparations, our protocol now goes as follows.

– The public input consists of matrix $\mathbf{M}$ and vector $\mathbf{v}$, which are constructed from the original public input, as discussed above.
– The prover's witness consists of vector $\mathbf{w} \in \mathsf{VALID}_{\mathsf{ineq}}$, which is built from the initial secret input, as described above.

The prover and the verifier then interact as in Figure 1. The protocol employs the statistically hiding and computationally binding string commitment scheme from Section 3 to obtain the desired statistical ZKAoK. As a corollary of Theorem 1, we obtain the following theorem, which summarized the properties our protocol for inequality relation between committed signed fractional numbers.

Theorem 4. *There exists a statistical zero-knowledge argument of knowledge for the relation R_{ineq} with perfect completeness, soundness error $2/3$ and communication cost $\mathcal{O}(L+m) = \mathcal{O}(\ell + f)$.*

For simulation, we simply invoke the simulator of Theorem 1. For extraction, we first run the knowledge extractor of Theorem 1 to obtain $\mathbf{w}' \in \mathsf{VALID}_{\mathsf{ineq}}$ such that $\mathbf{M} \cdot \mathbf{w}' = \mathbf{v}$. Then, by backtracking the transformations presented above, we can obtain $\mathbf{x}', \mathbf{y}', \mathbf{s}'_x, \mathbf{s}'_y$ such that $\mathsf{sbin}^{-1}_{\ell,f}(\mathbf{x}') \leq \mathsf{sbin}^{-1}_{\ell,f}(\mathbf{y}')$ and

$$\mathbf{e}_x = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}') \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}'_x) \ \wedge \ \mathbf{e}_y = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{y}') \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{s}'_y).$$

In particular, let $X' = \mathsf{sbin}^{-1}_{\ell,f}(\mathbf{x}') \in \mathbf{Q}\langle \ell \bullet f \rangle$ and $Y' = \mathsf{sbin}^{-1}_{\ell,f}(\mathbf{Y}') \in \mathbf{Q}\langle \ell \bullet f \rangle$, then we have $X' \leq Y'$.

4.3 Range Arguments

We now discuss how to use our results in Section 4.1 and Section 4.2 to derive various variants of range arguments for signed fractional numbers. Depending on the application contexts, different range types could be considered. Let us name a few of them.

1. **Hidden ranges with non-strict inequalities.** This requires to prove that $T \leq X \leq Y$, where T, X, Y are all committed. Such a range argument can be easily obtained by running two instances of our protocol of Section 4.2.
2. **Hidden ranges with strict inequalities.** In this setting, T, X, Y are hidden and the range relations are defined as $T < X < Y$, or $T \leq X < Y$. Here, a zero-knowledge argument of strict inequality is required. Such a protocol can be obtained by results of Theorem 3 and the techniques used in Section 4.2.
3. **Public ranges**: This type of range arguments is the easiest one, where A, B are publicly given and one proves that $A \leq X \leq B$ or $A < X < B$, for a committed number X. In fact, inequality $X \leq B$ can be handled using a simplified version of the protocol in Section 4.2, where the bits representing B are not required to be kept secret. Meanwhile, strict inequality $X < B$ can simply be interpreted as $X \leq B'$ for public $B' = B - 2^{-f}$.

In all cases considered above, the size of range arguments remains $\mathcal{O}(\ell + f)$, i.e., it is logarithmic in the size of the range.

5 Code-Based Accumulators and Logarithmic-Size Zero-Knowledge Arguments of Set Membership

In this section, we provide a code-based accumulator scheme supported by zero-knowledge argument of knowledge of an accumulated value. Such an argument system is essentially an argument of set membership, in which the prover convinces the verifier in zero-knowledge that his data item (e.g., his public key or his pseudonym) belongs to a given set, and is a highly desirable building block in various privacy-preserving applications. Our accumulator relies on a Merkle hash tree that is built from a suitable variant of the AFS hash function. To design a supporting zero-knowledge protocol for the hash tree, we first use the techniques for handling regular words from Section 3.2 to prove knowledge of hash preimages and images in the path from a leaf to the tree root, and then adapt Libert et al.'s method [50] to hide the bits determining steps in the path. As the tree depth is logarithmic in its size, we obtain an argument system for set membership with size logarithmic in the cardinality of the set.

In Section 5.1, we first recall the definitions and security requirement for cryptographic accumulators. Then, in Section 5.2, we modify the AFS hash function to support hashing with two inputs, upon which we build our Merkle-tree accumulator in Section 5.3. In Section 5.4, we describe our zero-knowledge argument system.

5.1 Cryptographic Accumulators

We recall the definitions and security requirement for accumulators.

Definition 4. *An accumulator scheme is a tuple of polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Accu}, \mathsf{WitGen}, \mathsf{Verify})$*:*

- $\mathsf{Setup}(1^\lambda)$ *Given a security parameter* 1^λ*, outputs the public parameter* pp*.*
- $\mathsf{Accu}_{pp}(R)$ *Take as input a set* R *with* n *data values as* $R = \{\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}\}$*, outputs an accumulator value* $\mathbf{u}$*.*
- $\mathsf{WitGen}_{pp}(R, \mathbf{d})$ *Take as input the set* R *and a value* $\mathbf{d}$*, outputs a witness* w *such that* $\mathbf{d}$ *is accumulated in* $\mathsf{TAcc}(R)$*, otherwise returns* $\perp$ *if* $\mathbf{d} \notin R$*.*
- $\mathsf{Verify}_{pp}(\mathbf{u}, (\mathbf{d}, w))$ *This deterministic algorithm takes as inputs the accumulator value* $\mathbf{u}$ *and* $(\mathbf{d}, w)$*, outputs* 1 *if* $(\mathbf{d}, w)$ *is valid for the accumulator* $\mathbf{u}$*, otherwise returns* 0 *if invalid.*

Correctness. For all $pp \leftarrow \mathsf{Setup}(n)$, the following holds:

$$\mathsf{Verify}_{pp}\big(\mathsf{Accu}_{pp}(R), \mathbf{d}, \mathsf{WitGen}_{pp}(R, \mathbf{d})\big) = 1 \text{ for } \mathbf{d} \in R.$$

An accumulator is secure, as defined in [8,26], if it is infeasible to output a valid witness for a value $\mathbf{d}^*$ that is not chosen from the data value set.

Definition 5. A*n accumulator scheme is secure if for all PPT adversaries* $\mathcal{A}$*:*

$$\Pr\big[pp \leftarrow \mathsf{Setup}(\lambda); (R, \mathbf{d}^*, w^*) \leftarrow \mathcal{A}(pp) : \\ \mathbf{d}^* \notin R \wedge \mathsf{Verify}_{pp}(\mathsf{Accu}_{pp}(R), \mathbf{d}^*, w^*) = 1\big] = \mathsf{negl}(\lambda).$$

5.2 Hashing with Two Inputs

We aim to build a Merkle-tree accumulator based on the AFS family of hash functions. Since in Merkle trees, every internal node is the hash of its two children nodes, we slightly modify the AFS hash functions so that the function takes two inputs instead of just one.

Definition 6. *Let $m = 2 \cdot 2^c \cdot n/c$. The function family $\mathcal{H}$ mapping $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}^n$ is defined as $\mathcal{H} = \{h_{\mathbf{B}} \mid \mathbf{B} \in \mathbb{Z}_2^{n \times m}\}$, where for $\mathbf{B} = [\mathbf{B}_0|\mathbf{B}_1]$ with $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{Z}_2^{n \times m/2}$, and for any $(\mathbf{u}_0, \mathbf{u}_1) \in \{0,1\}^n \times \{0,1\}^n$, we have:*

$$h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{u}_0) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{u}_1) \in \{0,1\}^n.$$

Lemma 4. *The function family $\mathcal{H}$, defined in Definition 6 is collision-resistant, assuming the hardness of the $\mathsf{2\text{-}RNSD}_{n,2n,c}$ problem.*

Proof. Given $\mathbf{B} = [\mathbf{B}_0|\mathbf{B}_1] \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$, if one can find two distinct pairs $(\mathbf{u}_0, \mathbf{u}_1) \in \left(\{0,1\}^n\right)^2$ and $(\mathbf{v}_0, \mathbf{v}_1) \in \left(\{0,1\}^n\right)^2$ such that $h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1) = h_{\mathbf{B}}(\mathbf{v}_0, \mathbf{v}_1)$, then one can obtain a non-zero vector $\mathbf{z} = \begin{pmatrix} \mathsf{RE}(\mathbf{u}_0) \oplus \mathsf{RE}(\mathbf{v}_0) \\ \mathsf{RE}(\mathbf{u}_1) \oplus \mathsf{RE}(\mathbf{v}_1) \end{pmatrix} \in \mathsf{2\text{-}Regular}(2n, c)$ such that

$$\mathbf{B} \cdot \mathbf{z} = h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1) \oplus h_{\mathbf{B}}(\mathbf{v}_0, \mathbf{v}_1) = \mathbf{0}.$$

In other words, $\mathbf{z}$ is a valid solution to the $\mathsf{2\text{-}RNSD}_{n,2n,c}$ problem associated with matrix $\mathbf{B}$. □

5.3 Code-Based Merkle-tree Accumulator

We now describe our Merkle-tree accumulator based on the code-based hash functions $\mathcal{H}$ in Definition 6. The construction is adapted from the blueprint by Libert et al. [50].

$\mathsf{Setup}(\lambda)$. Given $n = \mathcal{O}(\lambda)$, $c = \mathcal{O}(1)$ and $m = 2 \cdot 2^c \cdot n/c$. Sample $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$, and output the public parameter $pp = \mathbf{B}$.

$\mathsf{Accu}_{\mathbf{B}}(R = \{\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}\} \subseteq (\{0,1\}^n)^N)$. Let the binary representation of j be $(j_1, \ldots, j_\ell) \in \{0,1\}^\ell$, re-write $\mathbf{d}_j$ as $\mathbf{u}_{j_1,\ldots,j_\ell}$. Build a binary tree with $N = 2^\ell$ leaves $\mathbf{u}_{0,0,\ldots,0}, \ldots, \mathbf{u}_{1,1,\ldots,1}$ in the following way:

1. At depth $i \in [1, \ell - 1]$, for the nodes $\mathbf{u}_{a_1,\ldots,a_i,0} \in \{0,1\}^n$ and $\mathbf{u}_{a_1,\ldots,a_i,1} \in \{0,1\}^n$, compute $h_{\mathbf{B}}(\mathbf{u}_{a_1,\ldots,a_i,0}, \mathbf{u}_{a_1,\ldots,a_i,1})$ and define it to be $\mathbf{u}_{a_1,\ldots,a_i}$ for all $(a_1, \ldots, a_i) \in \{0,1\}^i$.
2. At depth 0, for the nodes $\mathbf{u}_0 \in \{0,1\}^n$ and $\mathbf{u}_1 \in \{0,1\}^n$, compute $h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1)$ and define it to be the root value $\mathbf{u}$.

Output the accumulated value $\mathbf{u}$.

$\mathsf{WitGen}_{\mathbf{B}}(R, \mathbf{d})$. If $\mathbf{d} \notin R$, the algorithm outputs $\perp$. Otherwise, it outputs the witness w for $\mathbf{d}$ as follows.

1. Set $\mathbf{d} = \mathbf{d}_j$ for some $j \in [0, N-1]$. Re-write $\mathbf{d}_j$ as $\mathbf{u}_{j_1,\ldots,j_\ell}$ using the binary representation of the index j.
2. Consider the path from $\mathbf{u}_{j_1,\ldots,j_\ell}$ to the root $\mathbf{u}$, the witness w then consists of the binary representation $(j_1, \ldots, j_\ell)$ for j as well as all the sibling nodes of the path. Specifically,

$$w = \big((j_1, \ldots, j_\ell), (\mathbf{u}_{j_1,\ldots,j_{\ell-1},\bar{j_\ell}}, \ldots, \mathbf{u}_{j_1,\bar{j_2}}, \mathbf{u}_{\bar{j_1}})\big) \in \{0,1\}^\ell \times \big(\{0,1\}^n\big)^\ell,$$

$\mathsf{Verify}_{\mathbf{B}}\big(\mathbf{u}, \mathbf{d}, w\big)$. Let w be of the following form:

$$w = \big((j_1, \ldots, j_\ell), (\mathbf{w}_\ell, \ldots, \mathbf{w}_1)\big) \in \{0,1\}^\ell \times \big(\{0,1\}^n\big)^\ell.$$

This algorithm then computes $\mathbf{v}_\ell, \ldots, \mathbf{v}_0$. Let $\mathbf{v}_\ell = \mathbf{d}$ and

$$\forall i \in \{\ell-1, \ldots, 1, 0\} : \mathbf{v}_i = \begin{cases} h_{\mathbf{B}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), \text{ if } j_{i+1} = 0; \\ h_{\mathbf{B}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), \text{ if } j_{i+1} = 1. \end{cases}$$

Output 1 if $\mathbf{v}_0 = \mathbf{u}$ or 0 otherwise.

The correctness of the above Merkle-tree accumulator scheme follows immediately from the construction. Its security is based on the collision resistance of the hash function family $\mathcal{H}$: if an adversary can break the security of the accumulator, then one can find a solution to the $\mathsf{2\text{-}RNSD}_{n,2n,c}$ problem.

Theorem 5. *Assume that the* $\mathsf{2\text{-}RNSD}_{n,2n,c}$ *problem is hard, then the given accumulator scheme is secure.*

Proof. Assume that there exists a PPT adversary $\mathcal{B}$ who breaks the security of the given accumulator scheme with non-negligible probability. By Definition 5, $\mathcal{B}$ first receives a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_2^{n \times m}$ generated by $\mathsf{Setup}(1^\lambda)$, and then outputs $(R = (\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}), \mathbf{d}^*, w^*)$ such that $\mathbf{d}^* \notin R$ and $\mathsf{Verify}_{\mathbf{B}}(\mathbf{u}^*, \mathbf{d}^*, w^*) = 1$, where $\mathbf{u}^* = \mathsf{Accu}_{\mathbf{B}}(R)$.

Let $w^* = ((j_1^*, \ldots, j_\ell^*), (\mathbf{w}_\ell^*, \ldots, \mathbf{w}_1^*))$ and $(j_1^*, \ldots, j_\ell^*)$ be the binary representation of some index $j^* \in [0, N-1]$, then we can find a path that starts from $\mathbf{d}_{j^*}$ to the accumulated value $\mathbf{u}$: $\mathbf{u}_{j_1^*,\ldots,j_\ell^*} = \mathbf{d}_{j^*}, \mathbf{u}_{j_1^*,\ldots,j_{\ell-1}^*}, \ldots, \mathbf{u}_{j_1^*}, \mathbf{u}^*$.

On the other hand, the algorithm $\mathsf{Verify}_{\mathbf{B}}(\mathbf{u}^*, \mathbf{d}^*, w^*)$ can compute another path: $\mathbf{k}_\ell^* = \mathbf{d}^*, \mathbf{k}_{\ell-1}^*, \ldots, \mathbf{k}_1^*, \mathbf{k}_0^* = \mathbf{u}^*$.

Since $\mathbf{d}^* \notin R$, then $\mathbf{d}^* \neq \mathbf{d}_{j^*}$. Comparing two paths, $\mathbf{d}_{j^*}, \mathbf{u}_{j_1^*,\ldots,j_{\ell-1}^*}, \ldots, \mathbf{u}_{j_1^*}, \mathbf{u}^*$ and $\mathbf{d}^*, \mathbf{k}_{\ell-1}^*, \ldots, \mathbf{k}_1^*, \mathbf{u}^*$, we can find the smallest integer $i \in [\ell]$, such that $\mathbf{k}_i^* \neq \mathbf{u}_{j_1^*,\ldots,j_i^*}$. So we obtain two distinct solutions to form a collision solution to the hash function $h_{\mathbf{B}}$ at the parent node of $\mathbf{u}_{j_1^*,\ldots,j_i^*}$. □

5.4 Logarithmic-Size Arguments of Set Membership

In this section, we describe a statistical zero-knowledge argument that allows prover $\mathcal{P}$ to convince verifier $\mathcal{V}$ in zero-knowledge that $\mathcal{P}$ knows a value that was correctly accumulated into the root of the above code-based Merkle tree. Our

protocol directly implies a logarithmic-size argument of set membership, with respect to the set of all data items accumulated into the tree root.

Given a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_2^{n\times m}$ and the accumulated value $\mathbf{u} \in \{0,1\}^n$ as input, the goal of $\mathcal{P}$ is to convince $\mathcal{V}$ that it possesses a value $\mathbf{d}$ and a valid witness w. We define the associated relation $\mathrm{R}_{\mathrm{acc}}$ as follows:

$$\mathrm{R}_{\mathrm{acc}} = \Big\{ \big((\mathbf{B},\mathbf{u}) \in \mathbb{Z}_2^{n\times m} \times \{0,1\}^n; \mathbf{d} \in \{0,1\}^n, w \in \{0,1\}^\ell \times (\{0,1\}^n)^\ell\big) : \\ \mathsf{Verify}_{\mathbf{B}}\big(\mathbf{u},\mathbf{d},w\big) = 1 \Big\}.$$

Before constructing a ZKAoK for the above relation, we first present several additional permuting techniques, which are developed from [50] and from our permuting technique presented Section 3.2.

- For vector $\mathbf{b} = (b_1,\ldots,b_n) \in \{0,1\}^n$, where $n \in \mathbb{Z}^+$, denote by $\mathsf{Encode}(\mathbf{b})$ the vector $(\bar{b}_1, b_1, \ldots, \bar{b}_n, b_n) \in \{0,1\}^{2n}$.
- Let $\mathbf{I}_n^* \in \mathbb{Z}_2^{n\times 2n}$ be an extension of the identity matrix $\mathbf{I}_n$, obtained by inserting a zero-column $\mathbf{0}^n$ right before each of the columns of $\mathbf{I}_n$. Note that if $\mathbf{b} \in \{0,1\}^n$, then $\mathbf{b} = \mathbf{I}_n^* \cdot \mathsf{Encode}(\mathbf{b})$.
- For $\mathbf{t} = (t_1,\ldots,t_n)^\top \in \{0,1\}^n$, define the permutation $F'_{\mathbf{t}}$ that transforms vector $\mathbf{w} = (w_{1,0}, w_{1,1}, \ldots, w_{n,0}, w_{n,1})^\top \in \{0,1\}^{2n}$ into:

  $$F'_{\mathbf{t}}(\mathbf{w}) = (w_{1,t_1}, w_{1,\bar{t}_1}, \ldots, w_{n,t_n}, w_{n,\bar{t}_n})^\top.$$

  Note that, for any $\mathbf{t},\mathbf{v} \in \{0,1\}^n$, we have:

  $$\mathbf{w} = \mathsf{Encode}(\mathbf{b}) \iff F'_{\mathbf{t}}(\mathbf{w}) = \mathsf{Encode}(\mathbf{b} \oplus \mathbf{t}). \tag{13}$$

- For $b \in \{0,1\}$ and $\mathbf{v} \in \{0,1\}^{m/2}$, we denote $\mathsf{Ext}(b,\mathbf{v})$ as $\begin{pmatrix} \bar{b}\cdot\mathbf{v} \\ b\cdot\mathbf{v}\end{pmatrix}$.
- For $e \in \{0,1\}$, for $\mathbf{t} \in \{0,1\}^n$, define the permutation $\Psi_{e,\mathbf{t}}$ that acts on $\mathbf{z} = \begin{pmatrix}\mathbf{z}_0 \\ \mathbf{z}_1\end{pmatrix} \in \{0,1\}^m$, where $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^{m/2}$, as follows. It transforms $\mathbf{z}$ to $\Psi_{e,\mathbf{t}}(\mathbf{z}) = \begin{pmatrix} E'_{\mathbf{t}}(\mathbf{z}_e) \\ E'_{\mathbf{t}}(\mathbf{z}_{\bar{e}})\end{pmatrix}$. Namely, it rearranges the blocks of $\mathbf{z}$ according to e and permutes each block using $E'_{\mathbf{t}}$.
- For any $b,e \in \{0,1\}$ and $\mathbf{v},\mathbf{w},\mathbf{t} \in \{0,1\}^n$, it follows from (7) that the following equivalences hold:

  $$\begin{cases} \mathbf{z} = \mathsf{Ext}(b, \mathsf{RE}(\mathbf{v})) \iff \Psi_{e,\mathbf{t}}(\mathbf{z}) = \mathsf{Ext}(b\oplus e, \mathsf{RE}(\mathbf{v}\oplus\mathbf{t})) \\ \mathbf{y} = \mathsf{Ext}(\bar{b}, \mathsf{RE}(\mathbf{w})) \iff \Psi_{\bar{e},\mathbf{t}}(\mathbf{y}) = \mathsf{Ext}(\bar{b}\oplus\bar{e}, \mathsf{RE}(\mathbf{w}\oplus\mathbf{t})). \end{cases} \tag{14}$$

Now let us examine the equations associated with the relation R_{acc}. Note that algorithm Verify computes the path $\mathbf{v}_\ell = \mathbf{d}, \mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_1, \mathbf{v}_0 = \mathbf{u}$, where $\mathbf{v}_i$ for $i \in \{\ell-1,\ldots,1,0\}$ is computed as follows:

$$\mathbf{v}_i = \begin{cases} h_{\mathbf{B}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), \text{ if } j_{i+1} = 0; \\ h_{\mathbf{B}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), \text{ if } j_{i+1} = 1. \end{cases} \tag{15}$$

We translate equation (15) into the following equivalent form.

$$\begin{aligned}\mathbf{v}_i &= \bar{j}_{i+1} \cdot h_{\mathbf{B}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}) \oplus j_{i+1} \cdot h_{\mathbf{B}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}) \\ &= \bar{j}_{i+1} \cdot \big(\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{v}_{i+1}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{w}_{i+1})\big) \oplus j_{i+1} \cdot \big(\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{w}_{i+1}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{v}_{i+1})\big) \\ &= \mathbf{B} \cdot \begin{pmatrix} \bar{j}_{i+1} \cdot \mathsf{RE}(\mathbf{v}_{i+1}) \\ j_{i+1} \cdot \mathsf{RE}(\mathbf{v}_{i+1}) \end{pmatrix} \oplus \mathbf{B} \cdot \begin{pmatrix} j_{i+1} \cdot \mathsf{RE}(\mathbf{w}_{i+1}) \\ \bar{j}_{i+1} \cdot \mathsf{RE}(\mathbf{w}_{i+1}) \end{pmatrix} \\ &= \mathbf{B} \cdot \mathsf{Ext}(j_{i+1}, \mathsf{RE}(\mathbf{v}_{i+1})) \oplus \mathbf{B} \cdot \mathsf{Ext}(\bar{j}_{i+1}, \mathsf{RE}(\mathbf{w}_{i+1}))\end{aligned}$$

Therefore, to obtain a ZKAoK for $\mathrm{R}_{\mathrm{acc}}$, it is necessary and sufficient to construct an argument system in which $\mathcal{P}$ convinces $\mathcal{V}$ in zero-knowledge that $\mathcal{P}$ knows $j_1, \ldots, j_\ell \in \{0,1\}^\ell$ and $\mathbf{v}_1, \ldots, \mathbf{v}_\ell, \mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \{0,1\}^n$ satisfying

$$\begin{cases} \mathbf{B} \cdot \mathsf{Ext}(j_1, \mathsf{RE}(\mathbf{v}_1)) \oplus \mathbf{B} \cdot \mathsf{Ext}(\bar{j}_1, \mathsf{RE}(\mathbf{w}_1)) = \mathbf{u}; \\ \mathbf{B} \cdot \mathsf{Ext}(j_2, \mathsf{RE}(\mathbf{v}_2)) \oplus \mathbf{B} \cdot \mathsf{Ext}(\bar{j}_2, \mathsf{RE}(\mathbf{w}_2)) \oplus \mathbf{v}_1 = \mathbf{0}. \\ \qquad\qquad\qquad\qquad \cdots\cdots\cdots \\ \mathbf{B} \cdot \mathsf{Ext}(j_\ell, \mathsf{RE}(\mathbf{v}_\ell)) \oplus \mathbf{B} \cdot \mathsf{Ext}(\bar{j}_\ell, \mathsf{RE}(\mathbf{w}_\ell)) \oplus \mathbf{v}_{\ell-1} = \mathbf{0}. \end{cases} \tag{16}$$

Next, we apply the function Encode defined above to vectors $\mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_1$. Let $\mathbf{x}_i = \mathsf{Encode}(\mathbf{v}_i) \in \{0,1\}^{2n}$ for $i \in [\ell - 1]$. Then we have $\mathbf{v}_i = \mathbf{I}_n^* \cdot \mathbf{x}_i$ for $i \in [\ell - 1]$. For ease of notation, for $i \in [\ell]$, denote

$$\begin{cases} \mathbf{y}_i = \mathsf{Ext}(j_i, \mathsf{RE}(\mathbf{v}_i)) \in \{0,1\}^m \\ \mathbf{z}_i = \mathsf{Ext}(\bar{j}_i, \mathsf{RE}(\mathbf{w}_i)) \in \{0,1\}^m \end{cases} \tag{17}$$

Therefore, the equations in (16) is equivalent to the following.

$$\begin{cases} \mathbf{B} \cdot \mathbf{y}_1 \oplus \mathbf{B} \cdot \mathbf{z}_1 = \mathbf{u}; \\ \mathbf{B} \cdot \mathbf{y}_2 \oplus \mathbf{B} \cdot \mathbf{z}_2 \oplus \mathbf{I}_n^* \cdot \mathbf{x}_1 = \mathbf{0}. \\ \qquad\qquad \cdots\cdots\cdots \\ \mathbf{B} \cdot \mathbf{y}_\ell \oplus \mathbf{B} \cdot \mathbf{z}_\ell \oplus \mathbf{I}_n^* \cdot \mathbf{x}_{\ell-1} = \mathbf{0}. \end{cases} \tag{18}$$

Now, using linear algebra, we can transform the equations in (18) into a unifying equation of the form $\mathbf{M}_A \cdot \mathbf{w}_A = \mathbf{v}_A$, where $\mathbf{M}_A \in \mathbb{Z}_2^{\ell n \times L}$, $\mathbf{v}_A \in \mathbb{Z}_2^{\ell n}$ are public and $\mathbf{w}_A \in \{0,1\}^L$ is secret with $L = 2\ell m + 2(\ell - 1)n$ and

$$\mathbf{w}_A = \big(\mathbf{y}_1 \parallel \ldots \parallel \mathbf{y}_\ell \parallel \mathbf{z}_1 \parallel \ldots \parallel \mathbf{z}_\ell \parallel \mathbf{x}_1 \parallel \ldots \parallel \mathbf{x}_{\ell-1} \big) \tag{19}$$

At this point, let us specify the set VALID_A containing our secret vector $\mathbf{w}_A$, the set $\mathcal{S}_A$ and permutations $\{\Gamma_\phi : \phi \in \mathcal{S}_A\}$ such that the conditions in (3) hold. Let VALID_A be the set of all $\mathbf{w}'_A = \big(\mathbf{y}'_1 \| \ldots \| \mathbf{y}'_\ell \| \mathbf{z}'_1 \| \ldots \| \mathbf{z}'_\ell \| \mathbf{x}'_1 \| \ldots \| \mathbf{x}'_{\ell-1}\big) \in \{0,1\}^L$ satisfying the following conditions:

– For $i \in [\ell]$, there exists $\mathbf{v}'_i, \mathbf{w}'_i \in \{0,1\}^n$, $j'_i \in \{0,1\}$ such that

$$\mathbf{y}'_i = \mathsf{Ext}(j'_i, \mathsf{RE}(\mathbf{v}'_i)) \in \{0,1\}^m, \quad \text{and} \quad \mathbf{z}'_i = \mathsf{Ext}(\bar{j}'_i, \mathsf{RE}(\mathbf{w}'_i)) \in \{0,1\}^m.$$

– For $i \in [\ell - 1]$, $\mathbf{x}'_i = \mathsf{Encode}(\mathbf{v}'_i) \in \{0,1\}^{2n}$.

Let $\mathcal{S}_A = \left(\{0,1\}^n\right)^\ell \times \left(\{0,1\}^n\right)^\ell \times \{0,1\}^\ell$. Then, for each element

$$\phi = \left(\mathbf{b}_1 \parallel \ \ldots \ \parallel \mathbf{b}_\ell \parallel \mathbf{e}_1 \parallel \ \ldots \ \parallel \mathbf{e}_\ell \parallel g_1 \parallel \ \ldots \ \parallel g_\ell \right) \in \mathcal{S}_A,$$

define the permutation Γ_ϕ that transforms

$$\mathbf{w}^*_A = \left(\mathbf{y}^*_1 \| \ldots \| \mathbf{y}^*_\ell \| \mathbf{z}^*_1 \| \ldots \| \mathbf{z}^*_\ell \| \mathbf{x}^*_1 \| \ldots \| \mathbf{x}^*_{\ell-1}\right) \in \{0,1\}^L$$

with $\mathbf{y}^*_i, \mathbf{z}^*_i \in \{0,1\}^m$ for $i \in [\ell]$ and $\mathbf{x}^*_i \in \{0,1\}^{2n}$ for $i \in [\ell - 1]$ into

$$\Gamma_\phi(\mathbf{w}^*_A) = \big(\Psi_{g_1,\mathbf{b}_1}(\mathbf{y}^*_1) \| \ldots \| \Psi_{g_\ell,\mathbf{b}_\ell}(\mathbf{y}^*_\ell) \| \Psi_{\bar{g}_1,\mathbf{e}_1}(\mathbf{z}^*_1) \| \ldots \| \Psi_{\bar{g}_\ell,\mathbf{e}_\ell}(\mathbf{z}^*_\ell) \| F'_{\mathbf{b}_1}(\mathbf{x}^*_1) \| \ldots \| F'_{\mathbf{b}_{\ell-1}}(\mathbf{x}^*_{\ell-1}) \big).$$

Based on the equivalences observed in (13) and (14), it can be checked that the conditions in (3) are satisfied. We thus have reduced the considered relation into an instance of $\mathrm{R}_{\mathrm{abstract}}$.

The interactive protocol. Given the above preparations, our protocol goes as follows.

– The public input consists of matrix $\mathbf{M}_A$ and vector $\mathbf{v}_A$, which are constructed from the original public input, as discussed above.
– The prover's witness consists of vector $\mathbf{w}_A \in \mathsf{VALID}_A$, which is built from the initial secret input, as described above.

The prover and the verifier then interact as in Figure 1. The protocol utilizes the statistically hiding and computationally binding string commitment scheme from Section 3 to obtain the desired statistical ZKAoK. The protocol has communication cost $\mathcal{O}(L) = \ell \cdot \mathcal{O}(m + n) = \mathcal{O}(\log N)$ bits.

6 Applications to Ring and Group Signatures

Our Merkle-tree accumulator together with its supporting zero-knowledge argument of set membership do enable a wide range of applications in code-based anonymity-oriented cryptographic protocols. In particular, these building blocks pave the way for the designs of logarithmic-size ring signatures and group signatures from code-based assumptions.

Ring signatures are arguably the most natural applications of accumulators, due to their decentralized setting and the observation that the ring signing procedure does capture a proof of ownership of a secret key corresponding to one of the public keys in the given ring. In our instantiation, the secret $\mathbf{x}$ of each user is an AFS hash preimage, while its image $\mathbf{d} = \mathbf{B} \cdot \mathsf{RE}(\mathbf{x})$ serves as the user's public key. To issue a signature with respect to a ring $R = \{\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}\}$ containing his public key, the user builds a Merkle tree on top of R, and proves knowledge of an extended path of hash preimages from his own secret key to the leaf corresponding to his public key, and then, from there to the tree root. This

can be done by extending the ZKAoK from Section 5.4 to handle one more layer of hashing. The obtained interactive zero-knowledge protocol is then repeated a sufficient number of times to achieve negligibly small soundness error, and then converted to a ring signature in the random oracle model via the Fiat-Shamir transformation [36]. The scheme is statistically anonymous and is unforgeable thanks to the security of the AFS hash function. Details are provided in Appendix C.

Building group signatures from accumulators is somewhat less intuitive. In fact, accumulators have been mainly used in group signatures for handling revocations. Libert et al. [50], however, showed that one in fact can design fully-anonymous group signatures from a Merkle-tree-based ring signature and a CCA2-secure encryption scheme where the latter admits a zero-knowledge argument of plaintext knowledge that is compatible with the supporting ZKAoK of the former. Since we have already obtained the ring signature block, to adapt the blueprint of [50], it remains to seek a suitable CCA2-secure encryption scheme and make them work together. To this end, we employ the Naor-Yung double encryption technique [60] to a randomized variant of the McEliece encryption scheme [55], suggested in [62]. The resulting CCA2-secure encryption mechanism is used to encrypt the identity of the signer - which is defined to be the $\log N$ bits determining the path from the tree leaf corresponding to the signer to the tree root. To complete the picture, we develop a Stern-like zero-knowledge layer for proving that such CCA2 ciphertexts are well-formed, which works smoothly with the zero-knowledge underlying the ring signature. Details of our construction are given in Appendix D.

Acknowledgements. We thank Duong Hieu Phan, Benoît Libert, Nicolas Sendrier and Ayoub Otmani and the anonymous reviewers of ASIACRYPT 2019 for their comments and suggestions. The research is supported by the Singapore Ministry of Education under Research Grant MOE2016-T2-2-014(S). Khoa Nguyen is also supported by the Gopalakrishnan – NTU Presidential Postdoctoral Fellowship 2018.

References

1. T. Acar and L. Nguyen. Revocation for delegatable anonymous credentials. In *PKC 2011*, volume 6571 of *LNCS*, pages 423–440. Springer, 2011.
2. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC 1996*, pages 99–108. ACM, 1996.
3. Q. Alamélou, O. Blazy, S. Cauchie, and P. Gaborit. A code-based group signature scheme. *Des. Codes Cryptography*, 82(1-2):469–493, 2017.
4. B. Applebaum, N. Haramaty, Y. Ishai, E. Kushilevitz, and V. Vaikuntanathan. Low-complexity cryptographic hash functions. In *ITCS 2017*, volume 67 of *LIPIcs*, pages 7:1–7:31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
5. M. H. Au, Q. Wu, W. Susilo, and Y. Mu. Compact e-cash from bounded accumulator. In *CT-RSA 2007*, volume 4377 of *LNCS*, pages 178–195. Springer, 2007.
6. D. Augot, M. Finiasz, and N. Sendrier. A fast provably secure cryptographic hash function. *IACR Cryptology ePrint Archive*, 2003:230, 2003.

7. D. Augot, M. Finiasz, and N. Sendrier. A family of fast syndrome based cryptographic hash functions. In *Mycrypt 2005*, volume 3715 of *LNCS*, pages 64–83. Springer, 2005.
8. N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 480–494. Springer, 1997.
9. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.
10. J. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital sinatures. In *EUROCRYPT 1993*, volume 765 of *LNCS*, pages 274–285. Springer, 1993.
11. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *J. Cryptology*, 22(1):114–138, 2009.
12. F. Benhamouda, J. Camenisch, S. Krenn, V. Lyubashevsky, and G. Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT 2014*, pages 551–572, 2014.
13. D. J. Bernstein, T. Lange, C. Peters, and P. Schwabe. Faster 2-regular information-set decoding. In *IWCC 2011*, volume 6639 of *LNCS*, pages 81–98. Springer, 2011.
14. D. J. Bernstein, T. Lange, C. Peters, and P. Schwabe. Really fast syndrome-based hashing. In *AFRICACRYPT 2011*, volume 6737 of *LNCS*, pages 134–152. Springer, 2011.
15. D. Boneh, S. Eskandarian, and B. Fisch. Post-quantum group signatures from symmetric primitives. *IACR Cryptology ePrint Archive*, 2018:261, 2018.
16. Z. Brakerski, A. Lombardi, G. Segev, and V. Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *EUROCRYPT 2018*, volume 10820 of *LNCS*, pages 535–564. Springer, 2018.
17. Z. Brakerski, V. Lyubashevsky, V. Vaikuntanathan, and D. Wichs. Worst-case hardness for LPN and cryptographic hashing via code smoothing. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:56, 2018.
18. P. Branco and P. Mateus. A code-based linkable ring signature scheme. In *ProvSec 2018*, volume 11192 of *LNCS*, pages 203–219. Springer, 2018.
19. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.
20. E. Brickell, D. Chaum, I. Damgård, and J. van de Graaf. Gradual and verifiable release of a secret. In *Crypto*. Springer, 1988.
21. E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design validations for discrete logarithm based signature schemes. In *PKC 2000*, volume 1751 of *LNCS*, pages 276–292. Springer, 2000.
22. J. Camenisch, R. Chaabouni, and a. shelat. Efficient protocols for set membership and range proofs. In *Asiacrypt*, 2008.
23. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.
24. J. Camenisch, M. Kohlweiss, and C. Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC 2009*, volume 5443 of *LNCS*, pages 481–500. Springer, 2009.
25. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.

26. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.
27. R. Chaabouni, H. Lipmaa, and B. Zhang. A non-interactive range proof with constant communication. In *Financial Cryptography*, 2012.
28. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.
29. G. Couteau, T. Peters, and D. Pointcheval. Removing the strong RSA assumption from arguments over the integers. In *Eurocrypt*, 2017.
30. L. Dallot and D. Vergnaud. Provably secure code-based threshold ring signatures. In *IMACC 2009*, volume 5921 of *LNCS*, pages 222–235. Springer, 2009.
31. D. Derler, S. Ramacher, and D. Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *PQCrypto 2018*, volume 10786 of *LNCS*, pages 419–440. Springer, 2018.
32. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 609–626. Springer, 2004.
33. N. Döttling. *Cryptography based on the hardness of decoding*. PhD thesis, Karlsruhe Institute of Technology, 2014. Available at `https://crypto.iti.kit.edu/fileadmin/User/Doettling/thesis.pdf`.
34. M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. Cryptology ePrint Archive, Report 2019/445, 2019. To appear at CRYPTO 2019.
35. M. F. Ezerman, H. T. Lee, S. Ling, K. Nguyen, and H. Wang. A provably secure group signature scheme from code-based assumptions. In *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 260–285. Springer, 2015.
36. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.
37. P. Gaborit, A. Hauteville, D. H. Phan, and J. Tillich. Identity-based encryption from codes with rank metric. In *CRYPTO 2017*, volume 10403 of *LNCS*, pages 194–224. Springer, 2017.
38. O. Goldreich, S. Micali, and A. Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185. Springer, 1986.
39. S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240. Tsinghua University Press, 2010.
40. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
41. A. Gonzalez and C. Ràfols. New techniques for non-interactive shuffle and range arguments. In *ACNS*, 2017.
42. J. Groth. Evaluating security of voting schemes in the universal composability framework. In *ACNS 2004*, volume 3089 of *LNCS*, pages 46–60. Springer, 2004.
43. J. Groth. Non-interactive zero-knowledge arguments for voting. In *ACNS*, 2005.
44. J. Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *Asiacrypt*, 2011.
45. J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 253–280. Springer, 2015.

46. A. Jain, S. Krenn, K. Pietrzak, and A. Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 663–680. Springer, 2012.
47. A. Kawachi, K. Tanaka, and K. Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 372–389. Springer, 2008.
48. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *ASIACRYPT 2016*, pages 373–403, 2016.
49. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *ASIACRYPT 2016*, pages 101–131, 2016.
50. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *EUROCRYPT 2016*, pages 1–31, 2016.
51. B. Libert, S. Ling, K. Nguyen, and H. Wang. Lattice-based zero-knowledge arguments for integer relations. In *CRYPTO 2018*, volume 10992 of *LNCS*, pages 700–732. Springer, 2018.
52. H. Lipmaa. On Diophantine complexity and statistical zero-knowledge arguments. In *Asiacrypt*, 2003.
53. H. Lipmaa, N. Asokan, and V. Niemi. Secure vickrey auctions without threshold trust. In *Financial Cryptography*, 2002.
54. V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. *J. Cryptology*, 31(3):774–797, 2018.
55. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, 1978.
56. C. A. Melchor, P.-L. Cayrel, and P. Gaborit. A new efficient threshold ring signature scheme based on coding theory. In *PQCrypto*, volume 5299 of *LNCS*, pages 1–16. Springer, 2008.
57. C. A. Melchor, P.-L. Cayrel, P. Gaborit, and F. Laguillaumie. A new efficient threshold ring signature scheme based on coding theory. *IEEE Trans. on Inf. Theory*, 57(7):4833–4842, 2011.
58. R. C. Merkle. A certified digital signature. In *CRYPTO 1989*, volume 435 of *LNCS*, pages 218–238. Springer, 1989.
59. K. Morozov and T. Takagi. Zero-Knowledge Protocols for the McEliece Encryption. In *ACISP*, pages 180–193. Springer, 2012.
60. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC 1990*, pages 427–437. ACM, 1990.
61. L. Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292. Springer, 2005.
62. R. Nojima, H. Imai, K. Kobara, and K. Morozov. Semantic security for the McEliece cryptosystem without random oracles. *Des. Codes Cryptography*, 49(1-3):289–305, 2008.
63. C. Papamanthou, E. Shi, R. Tamassia, and K. Yi. Streaming authenticated data structures. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 353–370. Springer, 2013.
64. C. Papamanthou, R. Tamassia, and N. Triandopoulos. Authenticated hash tables. In *ACM-CCS 2008*, pages 437–448. ACM, 2008.
65. D. A. Patterson and J. L. Hennessy. *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*. Morgan Kaufmann Publishers Inc., 5th edition, 2013.

66. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, 2001.
67. J. Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
68. Y. Yu, J. Zhang, J. Weng, C. Guo, and X. Li. Collision resistant hashing from learning parity with noise. *IACR Cryptology ePrint Archive*, 2017:1260, 2017.

A Proof of Theorem 1

Proof. If an honest prover follows the protocol, then he always gets accepted by the verifier. Thus, the protocol has perfect completeness. It is also easy to see that the communication cost is bounded by $\mathcal{O}(L)$.

We now prove that the protocol is a statistical zero-knowledge argument of knowledge.

Zero-Knowledge Property. We construct a PPT simulator SIM interacting with a (possibly dishonest) verifier $\widehat{\mathcal{V}}$, such that, given only the public input, SIM outputs with probability negligibly close to $2/3$ a simulated transcript that is statistically close to the one produced by the honest prover in the real interaction.

The simulator first chooses a random $\overline{Ch} \in \{1, 2, 3\}$ as a prediction of the challenge value that $\widehat{\mathcal{V}}$ will *not* choose.

Case $\overline{Ch} = 1$: Using basic linear algebra over $\mathbb{Z}_2$, SIM computes a vector $\mathbf{w}' \in \mathbb{Z}_2^L$ such that $\mathbf{M} \cdot \mathbf{w}' = \mathbf{v}$. Next, it samples $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, $\phi \xleftarrow{\$} \mathcal{S}$, and randomness ρ_1, ρ_2, ρ_3 for COM. Then, it sends the commitment $\mathsf{CMT} = \big(C'_1, C'_2, C'_3\big)$ to $\widehat{\mathcal{V}}$, where

$$C'_1 = \mathsf{COM}(\phi, \mathbf{M} \cdot \mathbf{r}_w; \rho_1),$$
$$C'_2 = \mathsf{COM}(\Gamma_\phi(\mathbf{r}_w); \rho_2), \quad C'_3 = \mathsf{COM}(\Gamma_\phi(\mathbf{w}' \oplus \mathbf{r}_w); \rho_3).$$

Receiving a challenge Ch from $\widehat{\mathcal{V}}$, the simulator responds as follows:

- If $Ch = 1$: Output $\perp$ and abort.
- If $Ch = 2$: Send $\mathrm{RSP} = \big(\phi, \mathbf{w}' \oplus \mathbf{r}_w, \rho_1, \rho_3\big)$.
- If $Ch = 3$: Send $\mathrm{RSP} = \big(\phi, \mathbf{r}_w, \rho_1, \rho_2\big)$.

Case $\overline{Ch} = 2$: SIM samples $\mathbf{w}' \xleftarrow{\$} \mathsf{VALID}$, $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, $\phi \xleftarrow{\$} \mathcal{S}$, and randomness ρ_1, ρ_2, ρ_3 for COM. Then it sends the commitment $\mathsf{CMT} = \big(C'_1, C'_2, C'_3\big)$ to $\widehat{\mathcal{V}}$, where

$$C'_1 = \mathsf{COM}(\phi, \mathbf{M} \cdot \mathbf{r}_w; \rho_1),$$
$$C'_2 = \mathsf{COM}(\Gamma_\phi(\mathbf{r}_w); \rho_2), \quad C'_3 = \mathsf{COM}(\Gamma_\phi(\mathbf{w}' \oplus \mathbf{r}_w); \rho_3).$$

Receiving a challenge Ch from $\widehat{\mathcal{V}}$, the simulator responds as follows:

- If $Ch = 1$: Send $\mathrm{RSP} = \big(\Gamma_\phi(\mathbf{w}'), \Gamma_\phi(\mathbf{r}_w), \rho_2, \rho_3\big)$.
- If $Ch = 2$: Output $\perp$ and abort.

– If $Ch = 3$: Send RSP $= \big(\phi, \mathbf{r}_w, \rho_1, \rho_2\big)$.

Case $\overline{Ch} = 3$: SIM samples $\mathbf{w}' \xleftarrow{\$} \mathsf{VALID}$, $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, $\phi \xleftarrow{\$} \mathcal{S}$, and randomness ρ_1, ρ_2, ρ_3 for COM. Then it sends the commitment $\mathsf{CMT} = \big(C_1', C_2', C_3'\big)$ to $\widehat{\mathcal{V}}$, where $C_2' = \mathsf{COM}(\mathit{\Gamma}_\phi(\mathbf{r}_w); \rho_2)$, $C_3' = \mathsf{COM}(\mathit{\Gamma}_\phi(\mathbf{w}' \oplus \mathbf{r}_w); \rho_3)$ as in the previous two cases, while

$$C_1' = \mathsf{COM}(\phi, \mathbf{M} \cdot (\mathbf{w}' \oplus \mathbf{r}_w) \oplus \mathbf{v}; \rho_1).$$

Receiving a challenge Ch from $\widehat{\mathcal{V}}$, it responds as follows:

– If $Ch = 1$: Send RSP computed as in the case $(\overline{Ch} = 2, Ch = 1)$.
– If $Ch = 2$: Send RSP computed as in the case $(\overline{Ch} = 1, Ch = 2)$.
– If $Ch = 3$: Output $\perp$ and abort.

We observe that, in every case we have considered above, since COM is statistically hiding, the distribution of the commitment CMT and the distribution of the challenge Ch from $\widehat{\mathcal{V}}$ are statistically close to those in the real interaction. Hence, the probability that the simulator outputs $\perp$ is negligibly close to $1/3$. Moreover, one can check that whenever the simulator does not halt, it will provide an accepted transcript, the distribution of which is statistically close to that of the prover in the real interaction. In other words, we have constructed a simulator that can successfully impersonate the honest prover with probability negligibly close to $2/3$.

Argument of Knowledge. Suppose that $\mathrm{RSP}_1 = (\mathbf{t}_w, \mathbf{t}_r, \rho_2, \rho_3)$, $\mathrm{RSP}_2 = (\phi_2, \mathbf{w}_2, \rho_1, \rho_3)$, $\mathrm{RSP}_3 = (\phi_3, \mathbf{w}_3, \rho_1, \rho_2)$ are 3 valid responses to the same commitment $\mathsf{CMT} = (C_1, C_2, C_3)$, with respect to all 3 possible values of the challenge. The validity of these responses implies that:

$$\begin{cases} \mathbf{t}_w \in \mathsf{VALID}; \\ C_1 = \mathsf{COM}(\phi_2, \mathbf{M} \cdot \mathbf{w}_2 \oplus \mathbf{v}; \rho_1) = \mathsf{COM}(\phi_3, \mathbf{M} \cdot \mathbf{w}_3; \rho_1); \\ C_2 = \mathsf{COM}(\mathbf{t}_r; \rho_2) = \mathsf{COM}(\mathit{\Gamma}_{\phi_3}(\mathbf{w}_3); \rho_2); \\ C_3 = \mathsf{COM}(\mathbf{t}_w \oplus \mathbf{t}_r; \rho_3) = \mathsf{COM}(\mathit{\Gamma}_{\phi_2}(\mathbf{w}_2); \rho_3). \end{cases}$$

Since COM is computationally binding, we can deduce that

$$\begin{cases} \mathbf{t}_w \in \mathsf{VALID};\ \phi_2 = \phi_3;\ \mathbf{t}_r = \mathit{\Gamma}_{\phi_3}(\mathbf{w}_3);\ \mathbf{t}_w \oplus \mathbf{t}_r = \mathit{\Gamma}_{\phi_2}(\mathbf{w}_2); \\ \mathbf{M} \cdot \mathbf{w}_2 \oplus \mathbf{v} = \mathbf{M} \cdot \mathbf{w}_3. \end{cases} \tag{20}$$

Since $\mathbf{t}_w \in \mathsf{VALID}$, if we let $\mathbf{w}' = [\mathit{\Gamma}_{\phi_2}]^{-1}(\mathbf{t}_w)$, then $\mathbf{w}' \in \mathsf{VALID}$. Furthermore, we have

$$\mathit{\Gamma}_{\phi_2}(\mathbf{w}') \oplus \mathit{\Gamma}_{\phi_2}(\mathbf{w}_3) = \mathit{\Gamma}_{\phi_2}(\mathbf{w}_2) \bmod 2,$$

which implies that $\mathbf{w}' \oplus \mathbf{w}_3 = \mathbf{w}_2$, and that $\mathbf{M} \cdot \mathbf{w}' \oplus \mathbf{M} \cdot \mathbf{w}_3 = \mathbf{M} \cdot \mathbf{w}_2$. As a result, we have $\mathbf{M} \cdot \mathbf{w}' = \mathbf{v}$. This concludes the proof. □

Remark 2. In many concrete instances of the protocol, the vector $\mathbf{t}_w = \mathit{\Gamma}_\phi(\mathbf{w})$ sent by the prover in the case $Ch = 1$ is fully determined by $d_0 \ll d$ bits. Then, the prover can save the communication cost by sending only the d_0 bits that are sufficient for the verifier to determine $\mathbf{t}_w$.

B Statistically Hiding and Computationally Binding Commitments

In this section, we recall the standard definitions of statistically hiding and computationally binding commitments.

B.1 Definitions

We recall the definition, statistically hiding and computationally binding properties for commitment schemes.

Definition 7. *A commitment scheme with message space $\mathcal{M}$ is a triple of algorithms* $(\mathsf{KGen}, \mathsf{Com}, \mathsf{Open})$ *for generating a commitment key, committing to a message and opening a commitment, respectively.*

- KGen*: On input* 1^λ*, it outputs a public commitment key* pk*.*
- Com*: On input a message* $\mathbf{x} \in \mathcal{M}$ *and commitment key* pk*, and it outputs a commitment/opening pair* $(\mathbf{c}, \mathbf{s})$*.*
- Open*: On input commitment key* pk*, a commitment* $\mathbf{c}$*, a message* $\mathbf{x}$ *and an opening* $\mathbf{s}$*, and this algorithm outputs* 1 *or* 0*.*

Correctness requires that Open evaluates to 1 whenever the inputs were computed by an honest party, namely:

$$\Pr[\mathsf{Open}(pk, \mathbf{c}, \mathbf{x}, \mathbf{s}) = 1 : \ pk \leftarrow \mathsf{KGen}(1^\lambda); \mathbf{x} \in \mathcal{M}, (\mathbf{c}, \mathbf{s}) \leftarrow \mathsf{Com}(pk, \mathbf{x})] = 1.$$

Computationally binding property. This property requires that it is infeasible for any PPT adversary $\mathcal{A}$ to output a commitment $\mathbf{c}$, two distinct messages $\mathbf{x}$, $\mathbf{x}'$ and openings $\mathbf{s}, \mathbf{s}'$ such that $\mathsf{Open}(pk, \mathbf{c}, \mathbf{x}, \mathbf{s}) = \mathsf{Open}(pk, \mathbf{c}, \mathbf{x}', \mathbf{s}') = 1$.

Statistically hiding property. This property requires that, given $pk \leftarrow \mathsf{KGen}(1^\lambda)$, for any $\mathbf{x}, \mathbf{x}' \in \mathcal{M}$, the distributions of $\mathsf{Com}(pk, \mathbf{x})$ and $\mathsf{Com}(pk, \mathbf{x}')$ are statistically close.

C Code-Based Logarithmic-Size Ring Signatures

In this section, we present a code-based ring signature scheme [66] with signature size logarithmic in the cardinality of the ring. The construction employs our accumulator and its associated ZKAoK, given in Section 5.3, as the building blocks. First, we recall the definitions of ring signatures in Section C.1. Our code-based instantiation is then described in Section C.2, while its supporting zero-knowledge argument system is presented in Section C.3. Finally, Section C.4 provides the analysis of the scheme.

C.1 Definitions and Security Requirements of Ring Signatures

Now we recall the standard definition and security requirements for ring signatures, as put forward in [11,45].

Definition 8. *A ring signature scheme consists of a tuple of polynomial-time algorithms* $(\mathsf{RSetup}, \mathsf{RKgen}, \mathsf{RSign}, \mathsf{RVerify})$.

$\mathsf{RSetup}(1^\lambda)$**:** *On input the security parameter* 1^λ*, this algorithm outputs the public parameter pp, which are available to all users.*

$\mathsf{RKgen}(pp)$**:** *On input public parameter, it generates a pair of public key and the corresponding secret signing key* (pk, sk).

$\mathsf{RSign}_{pp}(sk, M, R)$**:** *Take public parameter, secret key sk, a message* $M \in \{0,1\}^*$ *and* $R = (pk_0, \ldots, pk_{N-1})$ *as inputs, it outputs a signature* Σ *on the message* M *with respect to the ring* R*. Here,* (pk, sk) *is a valid key pair output by* $\mathsf{RKgen}(pp)$ *and* $pk \in R$.

$\mathsf{RVerify}_{pp}(M, R, \Sigma)$**:** *This deterministic algorithm verifies a purported ring signature* Σ *on the message* M *with respect to the ring of public keys* R*, it outputs* 1 *if the signature is valid or* 0 *otherwise.*

The correctness requirement says that signatures generated by honest users can always be accepted as valid ones. This is formalized as follows.

Definition 9 (Correctness). A ring signature $(\mathsf{RSetup}, \mathsf{RKgen}, \mathsf{RSign}, \mathsf{RVerify})$ is correct if for any $pp \leftarrow \mathsf{RSetup}(1^\lambda)$, any $(pk, sk) \leftarrow \mathsf{RKgen}(pp)$, any R such that $pk \in R$, any $M \in \{0,1\}^*$, we have $\mathsf{RVerify}_{pp}\big(M, R, \mathsf{RSign}_{pp}(sk, M, R)\big) = 1$.

A ring signature scheme is said to be secure if it satisfies unforgeability with respect to insider corruption, and statistical anonymity.

A ring signature is unforgeable with respect to insider corruption if it is infeasible to forge a ring signature without controlling one of the ring members. It is said to be statistically anonymous if signatures generated by two adversarially chosen keys are statistically indistinguishable. The formal definitions are given below.

Definition 10 (Unforgeability w.r.t. insider corruption). *A ring signature scheme* $(\mathsf{RSetup}, \mathsf{RKgen}, \mathsf{RSign}, \mathsf{RVerify})$ *is unforgeable w.r.t. insider corruption if for all PPT adversaries* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *who is given access to the following oracles succeeds is negligible:*

- PKGen *on the* j*-th query runs* $(pk_j, sk_j) \leftarrow \mathsf{RKgen}(pp)$ *and returns* pk_j *to* $\mathcal{A}$.
- $\mathcal{A}$ *is given access to* $\mathsf{Sign}(j, M, R)$*, which returns* $\mathsf{RSign}_{pp}(sk_j, M, R)$ *conditioned on the fact that* (pk_j, sk_j) *has been generated by* PKGen *and* $pk_j \in R$.
- $\mathcal{A}$ *is given access to a corrupted oracle* $\mathsf{Corrupt}(j)$*, which returns* sk_j*, provided that* (pk_j, sk_j) *has been generated by* PKGen.
- $\mathcal{A}$ *outputs* $(M^\star, R^\star, \Sigma^\star)$ *such that* $\mathsf{Sign}(\cdot, M^\star, R^\star)$ *has not been queried. Moreover,* $R^\star$ *is non-empty and only contains public keys* pk_j *generated by* PKGen *for which* j *has not been corrupted.*

the probability is

$$\Pr[pp \leftarrow \mathsf{RSetup}(1^\lambda); (M^\star, R^\star, \Sigma^\star) \leftarrow \mathcal{A}^{\mathsf{PKGen},\mathsf{Sign},\mathsf{Corrupt}}(pp) : \mathsf{RVerify}_{pp}(M^\star, R^\star, \Sigma^\star) = 1] \in \mathsf{negl}(\lambda),$$

Definition 11 (Statistical anonymity). A ring signature scheme provides statistical anonymity if, for any (possibly unbounded) adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathsf{RSetup}(1^\lambda); (M^\star, j_0, j_1, R^\star) \leftarrow \mathcal{A}^{\mathsf{RKgen}(pp)}(pp) \\ b \xleftarrow{\$} \{0,1\}; \Sigma^* \leftarrow \mathsf{RSign}_{pp}(sk_{j_b}, M^\star, R^\star) \end{array} : \mathcal{A}(\Sigma^\star) = b\right] = 1/2 + \mathsf{negl}(\lambda),$$

where $pk_{j_0}, pk_{j_1} \in R^\star$.

C.2 Description of Our Ring Signature

The scheme works as follows.

$\mathsf{RSetup}(1^\lambda)$: Let λ be the security parameter, choose $n = \mathcal{O}(\lambda)$, $c = \mathcal{O}(1)$ and $m = 2 \cdot 2^c \cdot n/c$. This algorithm outputs an uniformly random matrix $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$, and returns $pp = \mathbf{B}$.

$\mathsf{RKgen}(pp = \mathbf{B})$: On input $pp = \mathbf{B}$, choose an uniformly random vector $\mathbf{x} = \begin{pmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \end{pmatrix} \xleftarrow{\$} \{0,1\}^{2n}$, generates $\mathbf{d} = h_{\mathbf{B}}(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}) \in \{0,1\}^n$, and outputs $(pk, sk) = (\mathbf{d}, \mathbf{x})$.

$\mathsf{RSign}_{pp}(sk, M, R)$: On input pp, sk, a message $M \in \{0,1\}^*$ and a ring $R = (\mathbf{d}_0, \ldots, \mathbf{d}_{N-1})$, where $\mathbf{d}_i \in \{0,1\}^{nk}$ for every $i \in [0, N-1]$, and $sk = \mathbf{x} = \begin{pmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \end{pmatrix} \in \{0,1\}^{2n}$ such that $\mathbf{d} = h_{\mathbf{B}}(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}) \in R$, outputs a ring signature Σ on $M \in \{0,1\}^*$ with respect to the ring as follows:

1. First, On input the ring R, $\mathsf{TAcc}_{\mathbf{B}}(R)$ is able to build a Merkle tree based on the ring to obtain the accumulated value $\mathbf{u} \in \{0,1\}^n$.
2. On input ring R and $\mathbf{d}$, we aim to get the witnesses of $\mathbf{d}$ with which to find a path from $\mathbf{d}$ to the accumulated value $\mathbf{u}$ in the Merkle tree. Run algorithm $\mathsf{TWitness}_{\mathbf{B}}(R, \mathbf{d})$, it outputs the witness
$$w = \big((j_1, \ldots, j_\ell) \in \{0,1\}^\ell, (\mathbf{w}_\ell, \ldots, \mathbf{w}_1) \in (\{0,1\}^n)^\ell\big)$$
3. Generate a $\mathsf{NIZKAoK}$ Π_{ring} to show the possession of a valid pair $(pk, sk) = (\mathbf{d}, \mathbf{x})$ such that $\mathbf{d}$ is correctly accumulated in $\mathbf{u}$ in the Merkle tree. The details of the protocol is given in Section C.3. The interactive protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligibly small soundness error and then made non-interactive via the Fiat-Shamir heuristic as $\Pi_{\mathsf{ring}} = (\{\mathsf{CMT}_i\}_{i=1}^\kappa, \mathrm{CH}, \{\mathrm{RSP}\}_{i=1}^\kappa)$, where
$$\mathrm{CH} = \mathcal{H}_{\mathsf{FS}}\big(M, \{\mathsf{CMT}_i\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}, R\big) \in \{1,2,3\}^\kappa.$$

4. Outputs $\Sigma = \Pi_{\mathsf{ring}}$.

$\mathsf{RVerify}_{pp}(M, R, \Sigma)$: On input $pp = \mathbf{B}$, a message M, a ring $R = (\mathbf{d}_0, \ldots, \mathbf{d}_{N-1})$, and a signature Σ, the algorithm outputs 0/1 as follows:

1. Build a Merkle tree based on the ring R, then the algorithm $\mathsf{TAcc}_{\mathbf{B}}(R)$ computes the root $\mathbf{u}$ of the tree.
2. Parse Σ as $\Sigma = (\{\mathsf{CMT}_i\}_{i=1}^{\kappa}, (Ch_1, \ldots, Ch_\kappa), \{\mathrm{RSP}\}_{i=1}^{\kappa})$. Return 0 if $(Ch_1, \ldots, Ch_\kappa) \neq \mathcal{H}_{\mathsf{FS}}\big(M, \{\mathsf{CMT}_i\}_{i=1}^{\kappa}, \mathbf{B}, \mathbf{u}, R\big)$. For each $i = 1$ to κ, run the verification phase of the protocol from Section C.3 with public input $(\mathbf{B}, \mathbf{u})$ to check the validity of RSP_i with respect to CMT_i and Ch_i. If any of the conditions does not hold, then return 0. Otherwise, return 1.

C.3 The Underlying Zero-Knowledge Argument System

In this section, we describe the statistical ZKAoK that is invoked by the signer when producing ring signatures. The protocol is an extension of the one for the accumulator from Section 5.4. Here, the prover not only convinces the verifier that a secret value $\mathbf{d}$ is properly accumulated to the root of the tree, but it also shows that he knows $\mathbf{x} = \begin{pmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \end{pmatrix} \in \{0,1\}^n \times \{0,1\}^n$ satisfying

$$\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{x}^{(1)}) = \mathbf{d}. \tag{21}$$

The associated relation $\mathrm{R}_{\mathrm{ring}}$ is as follows:

$$\mathrm{R}_{\mathrm{ring}} = \Big\{ \big((\mathbf{B}, \mathbf{u});\ (\mathbf{d}, w, \mathbf{x}) \big) : \mathsf{Verify}_{\mathbf{B}}\big(\mathbf{u}, \mathbf{d}, w\big) = 1 \ \wedge \ \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{x}^{(1)}) = \mathbf{d} \Big\}.$$

Since the transformation layer for the accumulator has been established in Section 5.4, we only need to consider the new equation (21). Recall that $\mathbf{v}_\ell = \mathbf{d}$, define $\mathbf{x}_\ell = \mathsf{Encode}(\mathbf{v}_\ell) \in \{0,1\}^{2n}$, then we have $\mathbf{v}_\ell = \mathbf{I}_n^* \cdot \mathbf{x}_\ell$. Hence, the equation (21) is equivalent to the following:

$$\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{x}^{(1)}) \oplus \mathbf{I}_n^* \cdot \mathbf{x}_\ell = \mathbf{0}. \tag{22}$$

Up to this point, one can check that it suffices for $\mathcal{P}$ to convince $\mathcal{V}$ that he knows $j_1, \ldots, j_\ell \in \{0,1\}^\ell$, $\mathbf{v}_1, \ldots, \mathbf{v}_\ell, \mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \{0,1\}^n$, and $\mathbf{x}^{(0)}, \mathbf{x}^{(1)} \in \{0,1\}^n$ satisfying

$$\begin{cases} \mathbf{B} \cdot \mathbf{y}_1 \oplus \mathbf{B} \cdot \mathbf{z}_1 = \mathbf{u} \\ \mathbf{B} \cdot \mathbf{y}_2 \oplus \mathbf{B} \cdot \mathbf{z}_2 \oplus \mathbf{I}_n^* \cdot \mathbf{x}_1 = \mathbf{0} \\ \qquad \cdots\cdots\cdots \\ \mathbf{B} \cdot \mathbf{y}_\ell \oplus \mathbf{B} \cdot \mathbf{z}_\ell \oplus \mathbf{I}_n^* \cdot \mathbf{x}_{\ell-1} = \mathbf{0} \\ \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{x}^{(1)}) \oplus \mathbf{I}_n^* \cdot \mathbf{x}_\ell = \mathbf{0}, \end{cases}$$

where

$$\begin{cases} \mathbf{y}_i = \mathsf{Ext}(j_i, \mathsf{RE}(\mathbf{v}_i)) \in \{0,1\}^m, i \in [\ell] \\ \mathbf{z}_i = \mathsf{Ext}(\bar{j}_i, \mathsf{RE}(\mathbf{w}_i)) \in \{0,1\}^m, i \in [\ell] \\ \mathbf{x}_i = \mathsf{Encode}(\mathbf{v}_i) \in \{0,1\}^{2n}, i \in [\ell]. \end{cases} \tag{23}$$

By suitably concatenating and extending the matrices and vectors from public input, we can obtain public matrix $\mathbf{M}_R$, public vector $\mathbf{v}_R$ such that $\mathbf{M}_R \cdot \mathbf{w}_R = \mathbf{v}_R$ with secret vector $\mathbf{w}_R \in \{0,1\}^{L_R}$ with $L_R = (2\ell+1)m + 2\ell n$ and

$$\mathbf{w}_R = (\mathbf{y}_1 \| \ldots \| \mathbf{y}_\ell \| \mathbf{z}_1 \| \ldots \| \mathbf{z}_\ell \| \mathbf{x}_1 \| \ldots \| \mathbf{x}_\ell \| \mathsf{RE}(\mathbf{x}^{(0)}) \| \mathsf{RE}(\mathbf{x}^{(1)})) \tag{24}$$

With the desired unifying equation, we now define the set VALID_R that contains our secret vector $\mathbf{w}_R$, the set $\mathcal{S}_R$ and permutations $\{\Gamma_{\phi_R} : \phi_R \in \mathcal{S}_R\}$ such that the conditions in (3) hold. Let VALID_R contain all vectors of the form

$$\mathbf{w}'_R = \left(\mathbf{y}'_1 \| \ldots \| \mathbf{y}'_\ell \| \mathbf{z}'_1 \| \ldots \| \mathbf{z}'_\ell \| \mathbf{x}'_1 \| \ldots \| \mathbf{x}'_\ell \| \mathbf{u}'_0 \| \mathbf{u}'_1 \right) \in \{0,1\}^{L_R} \tag{25}$$

satisfying the following conditions:

- For $i \in [\ell]$, there exists $\mathbf{v}'_i, \mathbf{w}'_i \in \{0,1\}^n$, $j'_i \in \{0,1\}$ such that

$$\mathbf{y}'_i = \mathsf{Ext}(j'_i, \mathsf{RE}(\mathbf{v}'_i)) \in \{0,1\}^m, \quad \text{and} \quad \mathbf{z}'_i = \mathsf{Ext}(\bar{j}'_i, \mathsf{RE}(\mathbf{w}'_i)) \in \{0,1\}^m.$$

- For $i \in [\ell]$, $\mathbf{x}'_i = \mathsf{Encode}(\mathbf{v}'_i) \in \{0,1\}^{2n}$.
- For $i \in \{0,1\}$, there exists $\mathbf{x}^{(i)\prime} \in \{0,1\}^n$ such that $\mathbf{u}'_i = \mathsf{RE}(\mathbf{x}^{(i)\prime}) \in \{0,1\}^{m/2}$.

Let $\mathcal{S}_R$ be of the following form:

$$\mathcal{S}_R = \left(\{0,1\}^n\right)^\ell \times \left(\{0,1\}^n\right)^\ell \times \{0,1\}^\ell \times \left(\{0,1\}^n\right)^2.$$

Then, for each $\phi_R = \left(\mathbf{b}_1, \ldots, \mathbf{b}_\ell, \mathbf{e}_1, \ldots, \mathbf{e}_\ell, g_1, \ldots, g_\ell, \mathbf{p}_0, \mathbf{p}_1\right) \in \mathcal{S}_R$, define the permutation Γ_{ϕ_R} that transforms

$$\mathbf{w}^*_R = \left(\mathbf{y}^*_1 \| \ldots \| \mathbf{y}^*_\ell \| \mathbf{z}^*_1 \| \ldots \| \mathbf{z}^*_\ell \| \mathbf{x}^*_1 \| \ldots \| \mathbf{x}^*_\ell \| \mathbf{u}^*_0 \| \mathbf{u}^*_1 \right) \in \{0,1\}^{L_R} \tag{26}$$

with $\mathbf{y}^*_i, \mathbf{z}^*_i \in \{0,1\}^m, \mathbf{x}^*_i \in \{0,1\}^{2n}$ for $i \in [\ell]$ and $\mathbf{u}^*_i \in \{0,1\}^{m/2}$ for $i \in \mathbb{Z}_2$ to

$$\begin{aligned} \Gamma_{\phi_R}(\mathbf{w}^*_R) = \big(& \Psi_{g_1,\mathbf{b}_1}(\mathbf{y}^*_1) \| \ldots \| \Psi_{g_\ell,\mathbf{b}_\ell}(\mathbf{y}^*_\ell) \| \Psi_{\bar{g}_1,\mathbf{e}_1}(\mathbf{z}^*_1) \| \ldots \| \Psi_{\bar{g}_\ell,\mathbf{e}_\ell}(\mathbf{z}^*_\ell) \| \\ & F'_{\mathbf{b}_1}(\mathbf{x}^*_1) \| \ldots \| F'_{\mathbf{b}_\ell}(\mathbf{x}^*_\ell) \| E'_{\mathbf{p}_0}(\mathbf{u}^*_0) \| E'_{\mathbf{p}_1}(\mathbf{u}^*_1) \big). \end{aligned}$$

Based on the equivalences observed in (14), (13) and (7), it can be checked that the conditions in (3) are satisfied. We thus have reduced the considered relation into an instance of $\mathrm{R}_{\mathrm{abstract}}$ and the desired statistical ZKAoK protocol can be obtained by running the protocol in Figure 1. The protocol has communication cost $\mathcal{O}(L_R) = \ell \cdot \mathcal{O}(m+n)$ bits.

C.4 Analysis of the Scheme

We summarize the properties of the given ring signature scheme in the following theorem.

Theorem 6. *The ring signature scheme described in Section C.2 is correct, and produces signatures of bit-size $\mathcal{O}(n \cdot \log N)$. The scheme is unforgeable w.r.t. insider corruption based on the hardness of the $\mathsf{2\text{-}RNSD}_{n,2n,c}$ problem, and it is statistically anonymous.*

The proof of unforgeability employs the following lemma.

Lemma 5. *Let $h : \{0,1\}^{2n} \to \{0,1\}^n$ be a hash function. Then for $\mathbf{x} \xleftarrow{\$} \{0,1\}^{2n}$, the probability that there exists $\mathbf{x}' \in \{0,1\}^{2n}$ such that $h(\mathbf{x}') = h(\mathbf{x})$ is at least $1 - 2^{-n}$.*

Proof. There are in total 2^{2n} elements $\mathbf{x} \in \{0,1\}^{2n}$. Among them, there exist at most $2^n - 1$ elements that do not have $\mathbf{x}' \in \{0,1\}^{2n}$ such that $h(\mathbf{x}') = h(\mathbf{x})$. Thus, the probability that a uniformly random element $\mathbf{x}$ has a corresponding $\mathbf{x}'$ such that $h(\mathbf{x}') = h(\mathbf{x})$ is at least $\frac{2^{2n}-(2^n-1)}{2^{2n}} > 1 - 2^{-n}$.

With $m = 2n$ and $\mathbf{x} \xleftarrow{\$} \{0,1\}^m$, there exists $\mathbf{x}' \in \{0,1\}^m \setminus \{\mathbf{x}\}$ such that $\mathbf{B} \cdot \mathbf{x} = \mathbf{B} \cdot \mathbf{x}'$ with overwhelming probability $1 - 2^{-n}$.

Theorem 7. *The scheme provides unforgeability w.r.t. insider corruption in the random oracle model if the $\mathsf{2\text{-}RNSD}_{n,2n,c}$ problem is hard.*

Proof. We prove unforgeability w.r.t. insider corruption by contraposition. Suppose that an adversary $\mathcal{A}$ succeeds with non-negligible advantage ϵ in breaking unforgeability property, then we are able to construct a PPT algorithm $\mathcal{B}$ that either breaks the security of the accumulator, or breaks the computational soundness of the zero-knowledge protocol, or directly solves an $\mathsf{2\text{-}RNSD}_{n,n+k,c}$ problem with non-negligible probability.

Towards this goal, $\mathcal{B}$ first defines the public parameter $pp = \mathbf{A}$. When $\mathcal{A}$ makes queries to the PKGen oracle, $\mathcal{B}$ chooses a uniformly random vector $\mathbf{x} = \begin{pmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(0)} \end{pmatrix} \in \{0,1\}^{2n}$ and computes $\mathbf{d} = h_{\mathbf{A}}(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$. It then provides $\mathcal{A}$ with public key $pk = \mathbf{d}$ while keeping the secret key $sk = \mathbf{x}$. For the $\mathsf{Corrupt}(j)$ oracle and $\mathsf{Sign}(j, M, R)$ oracle, $\mathcal{B}$ is able to answer all the queries made by $\mathcal{A}$ since $\mathcal{B}$ possesses all the secret keys. When $\mathcal{A}$ halts and outputs $(M^\star, R^\star, \Sigma^\star)$. Consider the case that $\mathcal{A}$ succeeds in breaking the unforgeability w.r.t. insider corruption. It then follows that $\Sigma^\star$ is a valid signature on message $M^\star$, $\mathsf{Sign}(\cdot, M^\star, R^\star)$ has not been queried, and $R^\star$ contains all the public keys pk_j generated by PKGen in which j has not been corrupted. Denote $R^\star = (pk_{i_1}, \ldots, pk_{i_{|R^\star|}})$ and rewrite it as a set of binary vectors $(\mathbf{d}_0, \ldots, \mathbf{d}_{|R^\star|-1})$. Let $\Sigma^\star = \Pi^\star_{\mathsf{ring}} = (\{\mathsf{CMT}^\star_i\}_{i=1}^\kappa, \mathrm{CH}^\star, \{\mathsf{RSP}^\star\}_{i=1}^\kappa)$, where $\mathsf{CH}^\star = \mathcal{H}_{\mathsf{FS}}\big(M^\star, \{\mathsf{CMT}^\star_i\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}, R^\star\big)$ and $\mathsf{RSP}^\star$ is a valid response w.r.t. $\mathsf{CMT}^\star_i$ and $\mathsf{CH}^\star$. Note that the oracle $\mathcal{H}_{\mathsf{FS}}(\cdot)$ outputs uniformly random elements in $\{1,2,3\}^\kappa$ and the same answer is output when a hash query is made more than once.

We claim that $\mathcal{A}$ had queried $\big(M^\star, \{\mathsf{CMT}_i^\star\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}^\star, R^\star\big)$ to the hash oracle $\mathcal{H}_{\mathsf{FS}}$ with overwhelming probability, where $\mathbf{u}^\star = \mathsf{TAcc}_{\mathbf{A}}(R^\star)$. Otherwise, the probability of guessing the value of $\mathsf{CH}^\star = \mathcal{H}_{\mathsf{FS}}\big(M^\star, \{\mathsf{CMT}_i^\star\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}^\star, R^\star\big)$ correctly would be at most $3^{-\kappa}$, which is negligible. Therefore, with probability at least $\epsilon' := \epsilon - 3^{-\kappa}$, $\mathcal{A}$ had queried the hash oracle $\mathcal{H}_{\mathsf{FS}}$ and we denote $t^\star \in \{1, \ldots, Q_H\}$ as the index of this specific query, where Q_H is the total number of hash queries made by $\mathcal{A}$.

Algorithm $\mathcal{B}$ then runs at most $32 \cdot Q_H/(\epsilon - 3^{-\kappa})$ extra executions of the adversary $\mathcal{A}$. In each new run, all queries receive exactly the same answers as in the original run until the point of $t^\star$-th query to the hash oracle. From the $t^\star$-th query on, $\mathcal{B}$ replies $\mathcal{A}$ with uniformly random and independent values for each new run. As a result, the input of $t^\star$-th query is the same as in the initial run while the output is uniformly random and independent from the original run. By the Forking Lemma of Brickell et al. [21], with probability at least $1/2$, $\mathcal{B}$ can obtain a 3-fork involving the tuple $\big(M^\star, \{\mathsf{CMT}_i^\star\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}^\star, R^\star\big)$ with pairwise distinct hash values $\mathsf{CH}_{t^\star}^{(1)}, \mathsf{CH}_{t^\star}^{(2)}, \mathsf{CH}_{t^\star}^{(3)} \in \{1,2,3\}^\kappa$ and corresponding valid responses $\mathsf{RSP}_{t^\star}^{(1)}, \mathsf{RSP}_{t^\star}^{(2)}, \mathsf{RSP}_{t^\star}^{(3)}$. With probability $1-(7/9)^\kappa$, the results of [21] imply that there exists some $j \in \{1, \ldots, \kappa\}$ such that $\{\mathsf{CH}_{t^\star,j}^{(1)}, \mathsf{CH}_{t^\star,j}^{(2)}, \mathsf{CH}_{t^\star,j}^{(3)}\} = \{1,2,3\}$.

From 3 valid responses $(\mathsf{RSP}_{t^\star,j}^{(1)}, \mathsf{RSP}_{t^\star,j}^{(2)}, \mathsf{RSP}_{t^\star,j}^{(3)})$ w.r.t. the same commitment $\mathsf{CMT}_j^\star$ and challenges $1,2,3$, Theorem 1 ensures that $\mathcal{B}$ is able to extract witnesses $(\mathbf{x}^\star, \mathbf{d}^\star, w^\star)$, where $\mathbf{x}^\star = \begin{pmatrix} \mathbf{x}^{\star(0)} \\ \mathbf{x}^{\star(1)} \end{pmatrix}$, $w^\star = \big((j_1^\star, \ldots, j_\ell^\star), (\mathbf{w}_\ell^\star, \ldots, \mathbf{w}_1^\star)\big)$ such that $(j_1^\star, \ldots, j_\ell^\star) \in \{0,1\}^\ell$ is the binary expansion of some index $j^\star \in \{0, \ldots, |R^\star| - 1\}$ and

$$h_{\mathbf{A}}(\mathbf{x}^{\star(0)}, \mathbf{x}^{\star(1)}) = \mathbf{d}^\star, \tag{27}$$

$$\mathsf{TVerify}_{\mathbf{A}}\big(\mathbf{u}^\star, \mathbf{d}^\star, w^\star\big) = 1. \tag{28}$$

Since $\mathcal{A}$ wins the game, either we have (i) $\mathbf{d}^\star \notin R^\star = (\mathbf{d}_0, \ldots, \mathbf{d}_{|R^\star|-1})$ or (ii) $\mathbf{d}^\star \in R^\star = (\mathbf{d}_0, \ldots, \mathbf{d}_{|R^\star|-1})$ and $\mathbf{d}^\star = \mathbf{d}_{j^\star} = pk_{j^\star}$.

Case (i) implies that, $\mathcal{B}$ can use $(\mathbf{d}^\star, R^\star, \mathbf{u}^\star)$ to break the security of the accumulator from equation (28).

Case (ii) implies that, the extracted witnesses $(\mathbf{d}_{j^\star}, \mathbf{x}^\star)$ satisfy equation (27) by the soundness of the argument system. When $\mathcal{A}$ queried the PKGen oracle, $\mathcal{B}$ chose $sk_{j^\star} = \mathbf{x}_{j^\star} = \begin{pmatrix} \mathbf{x}_{j^\star}^{(0)} \\ \mathbf{x}_{j^\star}^{(1)} \end{pmatrix} \in \{0,1\}^{2n}$ satisfying

$$\mathbf{d}_{j^\star} = \mathbf{A}_0 \cdot \mathsf{RE}(\mathbf{x}_{j^\star}^{(0)}) \oplus \mathbf{A}_1 \cdot \mathsf{RE}(\mathbf{x}_{j^\star}^{(1)}). \tag{29}$$

Since $R^\star$ contains only uncorrupted public keys, we have that $\mathbf{x}_{j^\star} \neq \mathbf{x}^\star$ with probability at least $1/2$ according to Lemma 5. Furthermore, the statistical Witness Indistinguishability (WI) of the argument system implies that only a negligible amount of information regarding which witness among $\mathbf{x}^\star$ and $\mathbf{x}_{j^\star}$ is leaked. Combining equation (27) and equation (29), we obtain $\mathbf{A}_0 \cdot (\mathsf{RE}(\mathbf{x}_{j^\star}^{(0)}) \oplus$

$\mathsf{RE}(\mathbf{x}^{\star(0)})) \oplus \mathbf{A}_1 \cdot (\mathsf{RE}(\mathbf{x}_{j^\star}^{(1)}) \oplus \mathsf{RE}(\mathbf{x}^{\star(1)})) = \mathbf{0}$, which yields a valid 2-$\mathsf{RNSD}_{n,2n,c}$ solution $\mathbf{w} = \mathsf{RE}(\mathbf{x}_{j^\star}) \oplus \mathsf{RE}(\mathbf{x}^\star)$.

Therefore, with probability at least $1/2 \cdot (\epsilon - 3^{-\kappa}) \cdot (1 - (7/9)^\kappa) \cdot 1/2$, which is non-negligible, $\mathcal{B}$ solves an instance of the 2-$\mathsf{RNSD}_{n,2n,c}$ problem. □

Theorem 8. *The scheme provides statistical anonymity in the random oracle model.*

The proof of the above theorem relies on the statistical witness indistinguishability of the underlying argument system. The proof is straightforward and omitted.

D Code-Based Logarithmic-Size Group Signatures

This section presents our construction of logarithmic-size group signature from code-based assumptions. The scheme is based on the ring signature of Section C, where an encryption layer is introduced to enable the tracing capability. To this end, the signer is constrained to encrypt his identity, which is the essentially the bits determining the path from his respective leaf in the Merkle tree to the tree root. To enable anonymity in the strongest sense, we apply the Naor-Yung double encryption technique [60] to a randomized variant of the McEliece encryption scheme [55], suggested in [62], and obtain a CCA2-secure encryption mechanism.

We first recall the definitions and security requirements of group signatures in Section D.1 and the randomized McEliece encryption scheme is Section D.2. Our group signature construction is then described in Section D.3. Its supporting zero-knowledge argument system is presented in Section D.4 and its security proofs are provided in Section D.5.

D.1 Definitions and Security Requirements of Group Signatures

In this section, we give the syntax and the security model of the static group signatures [9]. A static group signature means that the group size is determined when the group is setup and no other new members can be added in, the group members can issue signatures on behalf of the group anonymously and the opener is able to trace back the actual signer.

Definition 12. *A group signature scheme consists of the following* 4 *polynomial-time algorithms* ($\mathsf{GKgen}, \mathsf{GSign}, \mathsf{GVerify}, \mathsf{GOpen}$)*:*

- *GKgen: The probabilistic group key generation algorithm GKgen takes input* $1^\lambda, 1^N$*, where* λ *is the security parameter and* $N \in \mathbb{N}$ *is the number of group users, outputs the group public key* gpk*, the group manager's secret key* gmsk *and* $\mathsf{gsk}[k]$ *the secret signing key for each group user of index* $k \in \{0, \ldots, N-1\}$*, that is, returns a triple* ($\mathsf{gpk}, \mathsf{gmsk}, \mathbf{gsk}$).
- *GSign: This probabilistic group signing algorithm on inputs the group public key* gpk*, a secret signing key* $\mathsf{gsk}[k]$ *for some signer* $k \in \{0, \ldots, N-1\}$*, and a message* M*, outputs a group signature* Σ *on* M*.*

– *GVerify: This deterministic group verify algorithm on inputs* gpk*,* M*, a signature* Σ*, and outputs either* 1 *if the signature is valid or* 0 *otherwise.*
– *GOpen: On inputs* gpk*,* gmsk*, a message* M*, a signature* Σ*, and this deterministic opening algorithm provides an index* k *when it succeeds or* $\perp$ *otherwise.*

We now recall the requirements for group signature scheme: correctness, *full anonymity* and *traceability*.
Correctness requires that signatures generated by honest group members are always deemed valid and the opener can always correctly identify the originator of any signature. Formally, for all $\lambda, N \in \mathbb{N}$, all $(\mathsf{gpk}, \mathsf{gmsk}, \mathbf{gsk})$ generated by $\mathsf{GKgen}(1^\lambda, 1^N)$, for any $k \in [0, N-1]$, and any message $M \in \{0,1\}^*$, the following conditions hold:

$$\mathsf{GVerify}\big(\mathsf{gpk}, M, \mathsf{GSign}(\mathsf{gpk}, \mathsf{gsk}[k], M)\big) = 1 \text{ and}$$
$$\mathsf{GOpen}\big(\mathsf{gpk}, \mathsf{gmsk}, M, \mathsf{GSign}(\mathsf{gsk}[k], M)\big) = k.$$

Full anonymity requires that it is infeasible for any PPT adversary to distinguish which of two signers of its choice signed a targeted message even if the adversary is accessible to all group user's secret keys, can choose that message and can query to the opening oracle for any signature except the challenged one. The requirement is modelled in the first experiment in Figure 2. The advantage of the adversary $\mathcal{A}$ against full anonymity denoted as $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{GS},\mathcal{A}}(\lambda, N)$ is defined as follows:

$$\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{GS},\mathcal{A}}(\lambda, N) = \left|\Pr[\mathbf{Exp}^{\mathsf{anon\text{-}1}}_{\mathcal{GS},\mathcal{A}}(\lambda, N) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon\text{-}0}}_{\mathcal{GS},\mathcal{A}}(\lambda, N) = 1]\right|.$$

A group signature is *fully anonymous* if for any PPT adversary $\mathcal{A}$, advantage $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{GS},\mathcal{A}}(\lambda, N)$ is negligible in the security parameter λ.
Full traceability demands that it is infeasible for any PPT adversary to output a valid group signature that either fails in the opening algorithm or that was traced to a user who is not in the coalition set even if the adversary could corrupt the group manager. The requirement is modeled in the second experiment in Figure 2. The advantage of the adversary $\mathcal{A}$ against full traceability is defined as

$$\mathbf{Succ}^{\mathsf{trace}}_{\mathcal{GS},\mathcal{A}}(\lambda, N) = \Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{GS},\mathcal{A}}(\lambda, N) = 1].$$

A group signature scheme is *fully traceable* if for any PPT adversary $\mathcal{A}$, the probability $\mathbf{Succ}^{\mathsf{trace}}_{\mathcal{GS},\mathcal{A}}(\lambda, N)$ is negligible in λ.

D.2 The Randomized McEliece Encryption Scheme

Now we recall a randomized variant of the McEliece [55] encryption scheme as suggested in [62], which is CPA-secure. The scheme is summarized as follows. It consists of the following algorithms $\mathsf{ME.Setup}$, $\mathsf{ME.KeyGen}$, $\mathsf{ME.Enc}$, and $\mathsf{ME.Dec}$.

– $\mathsf{ME.Setup}(1^\lambda)$: Let $n_\mathsf{e} = n_\mathsf{e}(\lambda), k_\mathsf{e} = k_\mathsf{e}(\lambda), t_\mathsf{e} = t_\mathsf{e}(\lambda)$ be the parameters for a binary $[n_\mathsf{e}, k_\mathsf{e}, 2t_\mathsf{e}+1]$ Goppa code. Choose k_1, $k_2 \in \mathbb{Z}$ such that $k_\mathsf{e} = k_1 + k_2$. Let $\mathbb{Z}_2^{k_2}$ be the plaintext space.

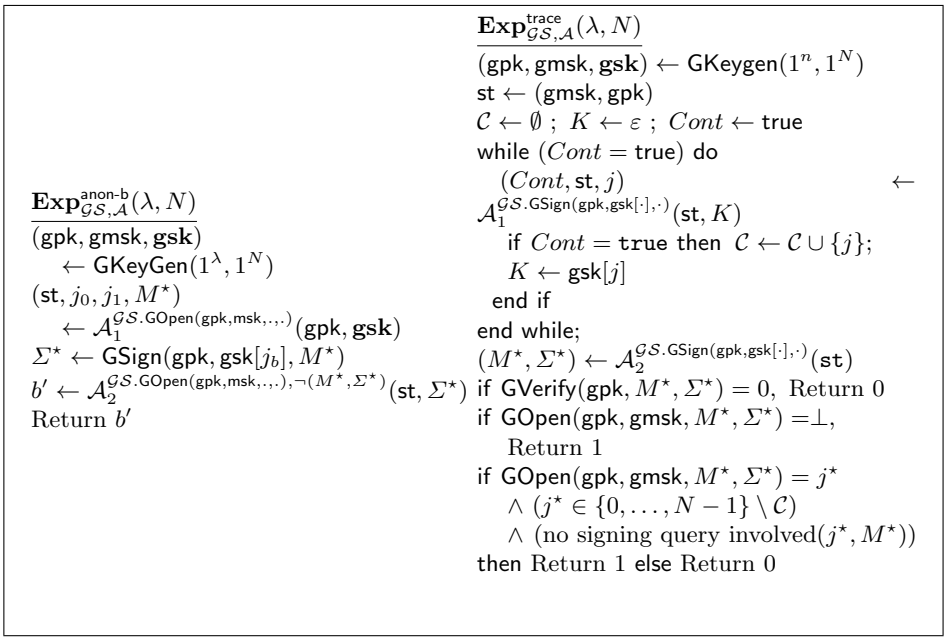

Fig. 2: Experiments for the definitions of anonymity and full traceability

- $\mathsf{ME.KeyGen}(n_\mathsf{e}, k_\mathsf{e}, t_\mathsf{e})$: This algorithm outputs the encryption key and decryption key for the randomized McEliece encryption scheme. It works as follows:
  1. Choose a generator matrix $\mathbf{G}' \in \mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}}$ of a randomly selected $[n_\mathsf{e}, k_\mathsf{e}, 2t_\mathsf{e}+1]$ Goppa code. Let $\mathbf{S} \in \mathbb{Z}_2^{k_\mathsf{e} \times k_\mathsf{e}}$ be a random invertible matrix and $\mathbf{P} \in \mathbb{Z}_2^{n_\mathsf{e} \times n_\mathsf{e}}$ be a random permutation matrix, then compute $\mathbf{G} = \mathbf{PG'S} \in \mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}}$. Output encryption key $\mathsf{pk_{ME}} = \mathbf{G}$ and decryption key $\mathsf{sk_{ME}} = (\mathbf{S}, \mathbf{G}', \mathbf{P})$.
- $\mathsf{ME.Enc}(\mathsf{pk_{ME}}, \mathbf{m})$: On input a message $\mathbf{m} \in \mathbb{Z}_2^{k_2}$ and $\mathsf{pk_{ME}}$, sample $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_2^{k_1}$ and $\mathbf{e} \xleftarrow{\$} \mathsf{B}(n_\mathsf{e}, t_\mathsf{e})$, and then output the ciphertext $\mathbf{c} = \mathbf{G} \cdot \begin{pmatrix} \mathbf{u} \\ \mathbf{m} \end{pmatrix} \oplus \mathbf{e} \in \mathbb{Z}_2^{n_\mathsf{e}}$.
- $\mathsf{ME.Dec}(\mathsf{sk_{ME}}, \mathbf{c})$: On input the ciphertext $\mathbf{c}$ and decryption key $\mathsf{sk_{ME}}$, it works as follows:
  1. Multiply $\mathbf{P}^{-1}$ to the left of the ciphertext $\mathbf{c}$, then apply an error-correcting algorithm. Obtain $\mathbf{m}'' = \mathit{Decode}_{\mathbf{G}'}(\mathbf{c} \cdot \mathbf{P}^{-1})$ where Decode is an error-correcting algorithm with respect to $\mathbf{G}'$. Returns $\perp$ if Decode fails.
  2. Multiply $\mathbf{S}^{-1}$ to the right of the ciphertext $\mathbf{m}''$, then $\mathbf{m}' = \mathbf{S}^{-1} \cdot \mathbf{m}''$, parse $\mathbf{m}' = \begin{pmatrix} \mathbf{u} \\ \mathbf{m} \end{pmatrix}$, where $\mathbf{u} \in \mathbb{Z}_2^{k_1}$ and $\mathbf{m} \in \mathbb{Z}_2^{k_2}$, and return $\mathbf{m}$.

Assuming the hardness of the Decisional McEliece problem DMcE and the Decisional Learning Parity with (fixed-weight) Noise problem DLPN [62,33], we have the CPA-security of the randomized McEliece scheme in the standard model. The definitions of the two problems are given below.

Definition 13. *The* $\mathsf{DMcE}(n_\mathsf{e}, k_\mathsf{e}, t_\mathsf{e})$ *problem asks to determine whether a matrix* $\mathbf{G} \in \mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}}$ *is uniformly chosen from* $\mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}}$ *or is generated by the algorithm* $\mathsf{ME.KeyGen}(n_\mathsf{e}, k_\mathsf{e}, t_\mathsf{e})$ *described above.*

When $n_\mathsf{e} = n_\mathsf{e}(\lambda), k_\mathsf{e} = k_\mathsf{e}(\lambda), t_\mathsf{e} = t_\mathsf{e}(\lambda)$, we say that the $\mathsf{DMcE}(n_\mathsf{e}, k_\mathsf{e}, t_\mathsf{e})$ problem is hard, if the success probability of any PPT distinguisher is at most $1/2 + \mathsf{negl}(\lambda)$.

Definition 14. *The* $\mathsf{DLPN}(k_\mathsf{e}, n_\mathsf{e}, \mathsf{B}(n_\mathsf{e}, t_\mathsf{e}))$ *problem asks to determine whether a pair* $(\mathbf{A}, \mathbf{s}) \in \mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}} \times \mathbb{Z}_2^n$ *is uniformly chosen from* $\mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}} \times \mathbb{Z}_2^n$ *or is obtained by choosing* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}}$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_2^k$, $\mathbf{e} \xleftarrow{\$} \mathsf{B}(n_\mathsf{e}, t_\mathsf{e})$ *and outputting* $(\mathbf{A}, \mathbf{A} \cdot \mathbf{u} \oplus \mathbf{e})$.

When $k_\mathsf{e} = k_\mathsf{e}(\lambda), n_\mathsf{e} = n_\mathsf{e}(\lambda), t_\mathsf{e} = t_\mathsf{e}(\lambda)$, we say that the $\mathsf{DLPN}(k_\mathsf{e}, n_\mathsf{e}, \mathsf{B}(n_\mathsf{e}, t_\mathsf{e}))$ problem is hard, if the success probability of any PPT distinguisher is at most $1/2 + \mathsf{negl}(\lambda)$.

D.3 Our Logarithmic-Size Group Signature Scheme

GKeygen$(1^\lambda, 1^N)$: On input the parameters $1^\lambda, 1^N$, the algorithm samples a uniformly random matrix $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$. Then it performs as follows to get the group public key, group manager's secret key and secret signing key for each group user.

1. For each $j \in [0, N-1]$, sample a random binary vector $\mathbf{x}_j = \begin{pmatrix} \mathbf{x}_j^{(0)} \\ \mathbf{x}_j^{(1)} \end{pmatrix} \xleftarrow{\$} \{0,1\}^{2n}$ and compute $\mathbf{d}_j = h_\mathbf{B}(\mathbf{x}_j^{(0)}, \mathbf{x}_j^{(1)}) \in \{0,1\}^n$. $\{\mathbf{d}_j\}_{j=0}^{N-1}$ should be pairwise distinct, otherwise restart the process. Then define the set $R = (\mathbf{d}_0, \ldots, \mathbf{d}_{N-1})$.
2. Run algorithm $\mathsf{TAcc}_\mathbf{B}(R)$ to build the Merkle tree based on R and the hash function $h_\mathbf{B}$, and obtain the accumulated value $\mathbf{u} \in \{0,1\}^n$.
3. Let $\ell = \lceil \log N \rceil$. For each $j \in [0, N-1]$, let $(j_1, \ldots, j_\ell)$ be the binary representation of j. Run algorithm $\mathsf{TWitness}_\mathbf{B}(R, \mathbf{d}_j)$ to outputs a witness $w^{(j)}$ of $\mathbf{d}_j$ such that $\mathbf{d}_j$ is correctly accumulated in $\mathbf{u}$.

$$w^{(j)} = \big((j_1, \ldots, j_\ell) \in \{0,1\}^\ell, (\mathbf{w}_\ell^{(j)}, \ldots, \mathbf{w}_1^{(j)}) \in (\{0,1\}^n)^\ell\big)$$

   Then define $\mathsf{gsk}[j] = (\mathbf{x}_j, \mathbf{d}_j, w^{(j)})$.
4. Run $\mathsf{ME.KeyGen}(n_\mathsf{e}, k_\mathsf{e}, t_\mathsf{e})$ twice to obtain two key-pairs $(\mathsf{pk_{ME}}^{(1)} = \mathbf{G}_1 \in \mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}}, \mathsf{sk_{ME}}^{(1)})$ and $(\mathsf{pk_{ME}}^{(2)} = \mathbf{G}_2 \in \mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}}, \mathsf{sk_{ME}}^{(2)})$.
5. Output

$$\mathsf{gpk} := \{\mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2\}\,;\ \ \mathsf{gmsk} := \mathsf{sk_{ME}}^{(1)};\ \ \mathbf{gsk} := (\mathsf{gsk}[0], \ldots, \mathsf{gsk}[N-1]).$$

GSign$(\mathsf{gpk}, \mathsf{gsk}[j], M)$: On input $M \in \{0,1\}^*$ and the user's secret signing key $\mathsf{gsk}[j] = (\mathbf{x}_j, \mathbf{d}_j, w^{(j)})$, where $w^{(j)} = \big((j_1, \ldots, j_\ell), (\mathbf{w}_\ell^{(j)}, \ldots, \mathbf{w}_1^{(j)})\big)$, the user performs as follows :

1. Encrypt the identity $\mathsf{bin}(j) = (j_1, \ldots, j_\ell) \in \{0,1\}^\ell$ twice using the randomized McEliece encryption scheme. More precisely, for each $i \in \{1,2\}$, sample $\mathbf{r}_i \xleftarrow{\$} \mathbb{Z}_2^{k_\mathsf{e}-\ell}$, $\mathbf{e}_i \xleftarrow{\$} \mathsf{B}(n_\mathsf{e}, t_\mathsf{e})$ and compute

$$\mathbf{c}_i = \mathbf{G}_i \cdot \begin{pmatrix} \mathbf{r}_i \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_i \in \mathbb{Z}_2^{n_\mathsf{e}}.$$

2. Generate a $\mathsf{NIZKAoK}$ Π_{group} to show the possession of a valid tuple $\tau = \big(\mathbf{x}_j, \mathbf{d}_j, w^{(j)}, \mathbf{r}_1, \mathbf{e}_1, \mathbf{r}_2, \mathbf{e}_2\big)$, where

$$\mathbf{x}_j = \begin{pmatrix} \mathbf{x}_j^{(0)} \\ \mathbf{x}_j^{(1)} \end{pmatrix} \text{ and } w^{(j)} = \big((j_1, \ldots, j_\ell), (\mathbf{w}_\ell^{(j)}, \ldots, \mathbf{w}_1^{(j)})\big),$$

such that:

(a) $h_{\mathbf{B}}(\mathbf{x}_j^{(0)}, \mathbf{x}_j^{(1)}) = \mathbf{d}_j$ and $\mathsf{TVerify}_{\mathbf{B}}\big(\mathbf{u}, \mathbf{d}_j, w^{(j)}\big) = 1$.

(b) $\mathbf{c}_1$ and $\mathbf{c}_2$ are both correct encryptions of $\mathsf{bin}(j) = (j_1, \ldots, j_\ell)^\top$ with randomness $(\mathbf{r}_1, \mathbf{e}_1) \in \mathbb{Z}_2^{k_\mathsf{e}-\ell} \times \mathsf{B}(n_\mathsf{e}, t_\mathsf{e})$ and $(\mathbf{r}_2, \mathbf{e}_2) \in \mathbb{Z}_2^{k_\mathsf{e}-\ell} \times \mathsf{B}(n_\mathsf{e}, t_\mathsf{e})$, respectively.

This is done by running the interactive argument system of Section D.4 on public input $(\mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1, \mathbf{c}_2)$ and prover's witness τ defined above. The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as $\Pi_{\mathsf{group}} = (\{\mathsf{CMT}_i\}_{i=1}^\kappa, \mathrm{CH}, \{\mathrm{RSP}\}_{i=1}^\kappa)$, where

$$\mathrm{CH} = \mathcal{H}_{\mathsf{FS}}\big(M, \{\mathsf{CMT}_i\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1, \mathbf{c}_2\big) \in \{1,2,3\}^\kappa.$$

3. Outputs the group signature $\Sigma = (\Pi_{\mathsf{group}}, \mathbf{c}_1, \mathbf{c}_2)$.

GVerify$(\mathsf{gpk}, M, \Sigma)$: On input gpk, M, Σ, the verification algorithm proceeds as follows:

1. Parse Σ as $\Sigma = \big(\{\mathsf{CMT}_i\}_{i=1}^\kappa, (Ch_1, \ldots, Ch_\kappa), \{\mathrm{RSP}\}_{i=1}^\kappa, \mathbf{c}_1, \mathbf{c}_2\big)$.
   If $(Ch_1, \ldots, Ch_\kappa) \neq \mathcal{H}_{\mathsf{FS}}\big(M, \{\mathsf{CMT}_i\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1, \mathbf{c}_2\big)$, then return 0.
2. For each $i = 1$ to κ, run the verification phase of the protocol in Section D.4 with public input $(\mathbf{A}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1, \mathbf{c}_2)$ to check the validity of RSP_i w.r.t. CMT_i and Ch_i. If any of the conditions does not hold, then return 0.
3. Return 1.

GOpen$(\mathsf{gpk}, \mathsf{gmsk}, \Sigma, M)$**:** On input $\mathsf{gmsk} = \mathsf{sk}_{\mathsf{ME}}{}^{(1)}$ and a group signature $\Sigma = (\Pi_{\mathsf{group}}, \mathbf{c}_1, \mathbf{c}_2)$ on message M, this algorithm proceeds as follows:

1. Run $\mathsf{ME.Dec}(\mathsf{sk}_{\mathsf{ME}}{}^{(1)}, \mathbf{c}_1)$ to decrypt $\mathbf{c}_1$. If decryption fails, then return $\perp$. Otherwise, let $\mathbf{p} = (j'_1, \ldots, j'_\ell)^\top \in \{0,1\}^\ell$ be the result of decryption.
2. Outputs index $j \in [0, N-1]$ that has binary representation $(j'_1, \ldots, j'_\ell)$.

D.4 The Supporting Zero-Knowledge Argument System

In this section, we present the zero-knowledge protocol that is exploited by signers when generating signatures in Section D.3. This protocol is extended from the one in Section C.3, for which an encryption layer is added. Specifically, the prover additionally proves possession of secret values $\mathbf{r}_1, \mathbf{r}_2 \in \{0,1\}^{k_e-\ell}$, $\mathbf{e}_1, \mathbf{e}_2 \in \mathsf{B}(n_e, t_e)$ such that

$$\mathbf{c}_1 = \mathbf{G}_1 \cdot \begin{pmatrix} \mathbf{r}_1 \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_1, \quad \text{and} \quad \mathbf{c}_2 = \mathbf{G}_2 \cdot \begin{pmatrix} \mathbf{r}_2 \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_2, \tag{30}$$

where $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{Z}_2^{n_e \times k_e}$ and $\mathbf{c}_1, \mathbf{c}_2 \in \{0,1\}^{n_e}$. The associated relation $\mathrm{R}_{\text{group}}$ is given below.

$$\begin{aligned} \mathrm{R}_{\text{group}} = \Big\{ \big((\mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1, \mathbf{c}_2);\ (\mathbf{d}, w, \mathbf{x}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{e}_1, \mathbf{e}_2) \big) : \\ \mathsf{Verify}_{\mathbf{B}}\big(\mathbf{u}, \mathbf{d}, w\big) = 1 \ \wedge \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{x}^{(1)}) = \mathbf{d} \ \wedge \\ \mathbf{c}_1 = \mathbf{G}_1 \cdot \begin{pmatrix} \mathbf{r}_1 \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_1 \ \wedge \ \mathbf{c}_2 = \mathbf{G}_2 \cdot \begin{pmatrix} \mathbf{r}_2 \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_2 \Big\}. \end{aligned}$$

We first recall the permuting technique to prove in ZK the knowledge of $\mathbf{e} \in \mathsf{B}(n_e, t_e)$ suggested by Stern [67]. To this end, the prover samples a uniformly random permutation $\sigma \in \mathsf{S}_{n_e}$ and shows the verifier that $\sigma(\mathbf{e}) \in \mathsf{B}(n_e, t_e)$. Due to the following equivalence

$$\mathbf{e} \in \mathsf{B}(n_e, t_e) \iff \sigma(\mathbf{e}) \in \mathsf{B}(n_e, t_e), \tag{31}$$

the verifier should be convinced that $\mathbf{e} \in \mathsf{B}(n_e, t_e)$. Furthermore, $\sigma(\mathbf{e})$ is uniform in $\mathsf{B}(n_e, t_e)$ since σ is uniform in S_{n_e}.

Now let us look at relation $\mathrm{R}_{\text{group}}$. As the transformation for the ring signature layer has been demonstrated in Section C.3, we consider the newly appeared relations in (30).

Denote $\mathsf{bin}(j) = (j_1, \ldots, j_\ell)^\top \in \{0,1\}^\ell$, $\mathbf{f} = \mathsf{Encode}(\mathsf{bin}(j)) \in \{0,1\}^{2\ell}$, and $\mathbf{h}_i = \mathsf{Encode}(\mathbf{r}_i) \in \{0,1\}^{2k_e-2\ell}$ for $i \in \{1,2\}$. Let $\mathbf{G}_i^* \in \mathbb{Z}_2^{n_e \times (k_e+\ell)}$ be the matrix obtained by adding a zero-column $\mathbf{0}^{n_e}$ right before each of the columns of $\mathbf{G}_i$, for $i \in \{1,2\}$. It is then verifiable that equations in (30) is equivalent to

$$\mathbf{c}_1 = \mathbf{G}_1^* \cdot \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{f} \end{pmatrix} \oplus \mathbf{e}_1, \quad \text{and} \quad \mathbf{c}_2 = \mathbf{G}_2^* \cdot \begin{pmatrix} \mathbf{h}_2 \\ \mathbf{f} \end{pmatrix} \oplus \mathbf{e}_2. \tag{32}$$

Let $\mathbf{w}_g = (\mathbf{f} \| \mathbf{h}_1 \| \mathbf{h}_2 \| \mathbf{e}_1 \| \mathbf{e}_2) \in \{0,1\}^{2n_e+4k_e-2\ell}$ be the secret vectors appeared in (32). Before proceeding further, let us recall that for the ring signature layer, we obtain $\mathbf{M}_R \cdot \mathbf{w}_R = \mathbf{v}_R$, where $\mathbf{w}_R$ is of the form (24). Combing the ring signature layer and the new encryption layer, we are able to obtain a unifying equation $\mathbf{M}_G \cdot \mathbf{w}_G = \mathbf{v}_G$, where $\mathbf{w}_G = (\mathbf{w}_R \| \mathbf{w}_g) \in \{0,1\}^{L_G}$ with $L_G = L_R + 2n_e + 4k_e - 2\ell$.

Up to this point, we have transformed the considered statement into a unified form. We are now ready to define the set VALID_G that consists of our secret vector

$\mathbf{w}_G$, the set $\mathcal{S}_G$ and the associated permutation $\{\Gamma_{\phi_G} : \phi_G \in \mathcal{S}_G\}$ such that the conditions in (3) hold.

Let VALID_G be the set of all vectors of the form $\mathbf{w}'_G = (\mathbf{w}'_R \| \mathbf{f}' \| \mathbf{h}'_1 \| \mathbf{h}'_2 \| \mathbf{e}'_1 \| \mathbf{e}'_2) \in \{0,1\}^{L_G}$, satisfying the following conditions:

- $\mathbf{w}'_R$ satisfies the conditions specified in (25).
- $\mathbf{f}' = \mathsf{Encode}(\mathsf{bin}(j'))$ with $\mathsf{bin}(j') = (j'_1, \ldots, j'_\ell)^\top$.
- There exists $\mathbf{r}'_i \in \{0,1\}^{k_e - \ell}$ such that $\mathbf{h}'_i = \mathsf{Encode}(\mathbf{r}'_i)$ for $i \in \{1,2\}$.
- $\mathbf{e}'_1, \mathbf{e}'_2 \in \mathsf{B}(n_e, t_e)$.

Define the set $\mathcal{S}_G = \mathcal{S}_R \times \left(\{0,1\}^{k_e-\ell}\right)^2 \times \left(\mathsf{S}_{n_e}\right)^2$, where $\mathcal{S}_R$ is defined in Section C.3. Then for each $\phi_G = (\phi_R, \mathbf{a}_1, \mathbf{a}_2, \sigma_1, \sigma_2) \in \mathcal{S}_G$, the associated permutation Γ_{ϕ_G} performs as follows. It transforms vector of the form $\mathbf{w}^*_G = (\mathbf{w}^*_R \| \mathbf{f}^* \| \mathbf{h}^*_1 \| \mathbf{h}^*_2 \| \mathbf{e}^*_1 \| \mathbf{e}^*_2)$ into

$$\Gamma_{\phi_G}(\mathbf{w}^*_G) = (\, \Gamma_{\phi_R}(\mathbf{w}^*_R) \,\|\, F'_{\mathbf{g}}(\mathbf{f}^*) \,\|\, F'_{\mathbf{a}_1}(\mathbf{h}^*_1) \,\|\, F'_{\mathbf{a}_2}(\mathbf{h}^*_2) \,\|\, \sigma_1(\mathbf{e}^*_1) \,\|\, \sigma_2(\mathbf{e}^*_2)\,),$$

where $\mathbf{g} = (g_1, \ldots, g_\ell)^\top$, g_i is the one appearing in ϕ_R for all $i \in [\ell]$, and Γ_{ϕ_R} is defined as in Section C.3.

Based on the equivalences observed in (14), (13), (7) and (31), it can be checked that the conditions in (3) are satisfied. We thus have reduced the considered relation into an instance of $\mathrm{R}_{\mathrm{abstract}}$ and the desired statistical ZKAoK protocol can be obtained by running the protocol in Figure 1. The protocol has communication cost $\mathcal{O}(L_G) = \mathcal{O}((2\ell+1)m + 2\ell n + 2n_e + 4k_e - 2\ell)$ bits.

D.5 Analysis of the Scheme

We summarize the properties of the given group signature scheme in the following theorem.

Theorem 9. *The group signature scheme described in Section D.3 is correct, and produces signatures of bit-size $\mathcal{O}((2\ell+1)m + 2\ell n + 2n_e + 4k_e - 2\ell)$ with $\ell = \lceil \log N \rceil$.*

Theorem 10. *The scheme provides full traceability in the random oracle model if the $\mathsf{2\text{-}RNSD}_{n,2n,c}$ problem is hard.*

Proof. The proof is done by contraposition. Assuming that an adversary $\mathcal{A}$ succeeds with non-negligible advantage ϵ, then we construct a PPT algorithm $\mathcal{B}$ that solves a $\mathsf{2\text{-}RNSD}_{n,2n,c}$ problem with non-negligible probability.

Given a matrix $\mathbf{A} \in \mathbb{Z}_2^{n \times m}$, $\mathcal{B}$ faithfully runs the $\mathsf{GKeygen}$ algorithm and then provides the adversary with gpk and gmsk. At the same time, $\mathcal{B}$ keeps all the users' private keys $\mathsf{gsk}[j] = (\mathbf{x}_j, \mathbf{d}_j, w^{(j)})$, where $\mathbf{u} = \mathsf{TAcc}_{\mathbf{A}}(\mathbf{d}_0, \ldots, \mathbf{d}_1)$ and $\mathbf{d}_j = h_{\mathbf{A}}(\mathbf{x}_j^{(0)}, \mathbf{x}_j^{(1)}) \in \{0,1\}^n$ for each $j \in [0, N-1]$. This allows $\mathcal{B}$ to answer all the corruption and signing queries. Note that the oracle $\mathcal{H}_{\mathsf{FS}}(\cdot)$ outputs uniformly random elements in $\{1,2,3\}^\kappa$ and the same answer is output when a hash query is made more than once.

When $\mathcal{A}$ halts and outputs $(M^\star, \Sigma^\star)$. Consider the case that $\mathcal{A}$ succeeds in breaking the traceability. It then follows that $\Sigma^\star$ is a valid signature on message $M^\star$. Let $(M^\star, \Sigma^\star)$ open to some uncorrupted user $j^\star$, for which $(j^\star, M^\star)$ is not queried to the signing oracle. Parse $\Sigma^\star$ as $(\Pi^\star_{\mathsf{group}}, \mathbf{c}_1^\star, \mathbf{c}_2^\star)$ and $\Pi^\star_{\mathsf{group}}$ as $(\{\mathsf{CMT}_i^\star\}_{i=1}^\kappa, \mathsf{CH}^\star, \{\mathsf{RSP}^\star\}_{i=1}^\kappa)$.

We claim that $\mathcal{A}$ had queried $\big(M^\star, \{\mathsf{CMT}_i^\star\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1^\star, \mathbf{c}_2^\star\big)$ with overwhelming probability. Otherwise, the probability of guessing the value of $\mathsf{CH}^\star = \mathcal{H}_{\mathsf{FS}}\big(M^\star, \{\mathsf{CMT}_i^\star\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1^\star, \mathbf{c}_2^\star\big)$ would be at most $3^{-\kappa}$, which is negligible. Therefore, with probability at least $\epsilon' := \epsilon - 3^{-\kappa}$, $\mathcal{A}$ had queried the hash oracle $\mathcal{H}_{\mathsf{FS}}$ and denote by $t^\star \in \{1, \ldots, Q_H\}$ the index of this specific query, where Q_H is the total number of hash queries made by $\mathcal{A}$.

Algorithm $\mathcal{B}$ then runs at most $32 \cdot Q_H/(\epsilon - 3^{-\kappa})$ extra executions of the adversary $\mathcal{A}$. In each new run, all queries receive exactly the same answers as in the original run until the point of $t^\star$-th query to the hash oracle. From the $t^\star$-th query on, $\mathcal{B}$ replies $\mathcal{A}$ with uniformly random and independent values for each new run. As a result, the input of $t^\star$-th query is the same as in the initial run while the output is uniformly random and independent from the original run. By the Forking Lemma of Brickell et al. [21], with probability at least $1/2$, $\mathcal{B}$ can obtain a 3-fork involving the same tuple $\big(M^\star, \{\mathsf{CMT}_i^\star\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1^\star, \mathbf{c}_2^\star\big)$ with pairwise distinct hash values $\mathsf{CH}_{t^\star}^{(1)}, \mathsf{CH}_{t^\star}^{(2)}, \mathsf{CH}_{t^\star}^{(3)} \in \{1,2,3\}^\kappa$ and corresponding valid responses $\mathsf{RSP}_{t^\star}^{(1)}, \mathsf{RSP}_{t^\star}^{(2)}, \mathsf{RSP}_{t^\star}^{(3)}$. With probability $1 - (7/9)^\kappa$, the results of [21] imply that there exists some $j \in \{1, \ldots, \kappa\}$ such that $\{\mathsf{CH}_{t^\star,j}^{(1)}, \mathsf{CH}_{t^\star,j}^{(2)}, \mathsf{CH}_{t^\star,j}^{(3)}\} = \{1,2,3\}$.

From 3 valid responses $(\mathsf{RSP}_{t^\star,j}^{(1)}, \mathsf{RSP}_{t^\star,j}^{(2)}, \mathsf{RSP}_{t^\star,j}^{(3)})$ w.r.t. the same commitment $\mathsf{CMT}_j^\star$ and all challenges $1, 2, 3$, Theorem 1 ensures that $\mathcal{B}$ is able to extract witnesses

$$(\mathbf{x}^\star, \mathbf{d}^\star, w^\star, \mathbf{r}_1^\star, \mathbf{r}_2^\star, \mathbf{e}_1^\star, \mathbf{e}_2^\star),$$

where $\mathbf{x}^\star = \begin{pmatrix} \mathbf{x}^{\star(0)} \\ \mathbf{x}^{\star(1)} \end{pmatrix} \in \{0,1\}^{2n}$, $w^\star = \big((j_1^\star, \ldots, j_\ell^\star), (\mathbf{w}_\ell^\star, \ldots, \mathbf{w}_1^\star)\big)$ such that $(j_1^\star, \ldots, j_\ell^\star) \in \{0,1\}^\ell$ is the binary expansion of some integer $j^\star \in [0, N-1]$ and

$$h_{\mathbf{A}}(\mathbf{x}_0^\star, \mathbf{x}_1^\star) = \mathbf{d}^\star, \tag{33}$$

$$\mathsf{TVerify}_{\mathbf{A}}\big(\mathbf{u}^\star, \mathbf{d}^\star, w^\star\big) = 1. \tag{34}$$

Since $\mathcal{A}$ wins the game, either we have (i) $\mathbf{d}^\star \notin R = (\mathbf{d}_0, \ldots, \mathbf{d}_{N-1})$ or (ii) $\mathbf{d}^\star \in R = (\mathbf{d}_0, \ldots, \mathbf{d}_{N-1})$ and $\mathbf{d}^\star = \mathbf{d}_{j^\star}$. Note that $\{\mathbf{d}_j\}_{j=0}^{N-1}$ are pairwise distinct, as ensured by the GKeyen algorithm.

Case (i) implies a violation of the security of the underlying accumulator.

Case (ii) implies that, $\mathbf{c}_1^\star$ decrypts to $(j_1^\star, \ldots, j_\ell^\star) \in \{0,1\}^\ell$ from the soundness of the argument system. Therefore, $\mathcal{A}$ did not obtain the private key $\mathsf{gsk}[j^\star]$, which contains a vector $\mathbf{x}_{j^\star} = \begin{pmatrix} \mathbf{x}_{j^\star}^{(0)} \\ \mathbf{x}_{j^\star}^{(1)} \end{pmatrix} \in \{0,1\}^{2n}$ that satisfies

$$\mathbf{d}_{j^\star} = \mathbf{A}_0 \cdot \mathsf{RE}(\mathbf{x}_{j^\star}^{(0)}) \oplus \mathbf{A}_1 \cdot \mathsf{RE}(\mathbf{x}_{j^\star}^{(1)}). \tag{35}$$

Since $j^\star$ is uncorrupted, $\mathbf{x}_{j^\star} \neq \mathbf{x}^\star$ with probability at least $1/2$. Combining equation (33) and equation (35), it follows that

$$\mathbf{A}_0 \cdot (\mathsf{RE}(\mathbf{x}_{j^\star}^{(0)}) \oplus \mathsf{RE}(\mathbf{x}^{\star(0)})) \oplus \mathbf{A}_1 \cdot (\mathsf{RE}(\mathbf{x}_{j^\star}^{(1)}) \oplus \mathsf{RE}(\mathbf{x}^{\star(1)})) = \mathbf{0},$$

which yields a valid $\mathsf{2\text{-}RNSD}_{n,2n,c}$ solution $\mathbf{w} = \mathsf{RE}(\mathbf{x}_{j^\star}) \oplus \mathsf{RE}(\mathbf{x}^\star)$. To argue $\mathbf{x}_{j^\star} \neq \mathbf{x}^\star$ with probability at least $1/2$, we observe that $\mathcal{A}$ may learn $\mathbf{d}_{j^\star} = \mathbf{A}_0 \cdot \mathsf{RE}(\mathbf{x}_{j^\star}^{(0)}) \oplus \mathbf{A}_1 \cdot \mathsf{RE}(\mathbf{x}_{j^\star}^{(1)})$ by corrupting $\mathsf{gsk}[j^\star + 1]$ or $\mathsf{gsk}[j^\star - 1]$. However, there exists at least one another vector $\mathbf{x}^\star \neq \mathbf{x}_{j^\star}$. What is more, the statistical WI property implies that a negligible amount of information regarding which witness among $\mathbf{x}^\star$ and $\mathbf{x}_{j^\star}$ is leaked when replying the signing queries $(j^\star, \cdot)$.

Therefore, with probability at least $1/2 \cdot (\epsilon - 3^{-\kappa}) \cdot (1 - (7/9)^\kappa) \cdot 1/2$, which is non-negligible, $\mathcal{B}$ solves an instance of the $\mathsf{2\text{-}RNSD}_{n,2n,c}$ problem. □

Theorem 11. *Assume that the* $\mathsf{DMcE}(n_\mathsf{e}, k_\mathsf{e}, t_\mathsf{e})$ *and the* $\mathsf{DLPN}(k_e, n_\mathsf{e}, \mathsf{B}(n_\mathsf{e}, t_\mathsf{e}))$ *problems are hard and the argument system is simulation-sound, then the group signature scheme is fully anonymous.*

Proof. We prove the theorem using a sequence of indistinguishable games. The first game is experiment $\mathbf{Exp}^{\mathsf{anon\text{-}0}}_{\mathcal{GS},\mathcal{A}}(\lambda, N)$ whereas the last game is experiment $\mathbf{Exp}^{\mathsf{anon\text{-}1}}_{\mathcal{GS},\mathcal{A}}(\lambda, N)$. It then follows that our group signature scheme is fully anonymous. For each i, denote by W_i the event that the adversary outputs 1 in Game i.

Game 0**:** In this game, the challenger $\mathcal{B}$ runs experiment $\mathbf{Exp}^{\mathsf{anon\text{-}0}}_{\mathcal{GS},\mathcal{A}}(\lambda, N)$, in which the adversary $\mathcal{A}$ receives signature $\Sigma^\star \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{gsk}[j_0], M^\star)$ in the challenge phase. Let $\Sigma^\star = (\Pi^\star_{\mathsf{group}}, \mathbf{c}_1^\star, \mathbf{c}_2^\star)$. Then we have $\Pr[W_0] = \Pr[\mathbf{Exp}^{\mathsf{anon\text{-}0}}_{\mathcal{GS},\mathcal{A}}(\lambda, N) = 1]$.

Game 1**:** This game is the same as Game 0 except that $\mathcal{B}$ keeps both decryption keys of McEliece encryption scheme instead of wiping out the second one $\mathsf{sk}^{(2)}_{\mathsf{ME}}$. We can see that this change does not affect $\mathcal{A}$'s view. Therefore $\Pr[W_1] = \Pr[W_0]$.

Game 2**:** This game modifies Game 1 as follows: The opening oracle employs the second McEliece decryption key $\mathsf{sk}^{(2)}_{\mathsf{ME}}$ instead of the first one $\mathsf{sk}^{(1)}_{\mathsf{ME}}$. The view of $\mathcal{A}$ is the same as in Game 1 until the event F_1, that $\mathcal{A}$ queries the opening oracle a signature $\Sigma = (\Pi_{\mathsf{group}}, \mathbf{c}_1, \mathbf{c}_2)$ with $\mathbf{c}_1$ and $\mathbf{c}_2$ encrypting distinct ℓ-bit strings, happens. Since event F_1 breaks the soundness of the underlying argument system, we have $|\Pr[W_2] - \Pr[W_1]| \leq \Pr[F_1] \leq \mathbf{Adv}^{\mathsf{sound}}_{\mathcal{B}}(\lambda) \in \mathsf{negl}(\lambda)$.

Game 3**:** This game changes Game 2 by generating simulated proof instead of real proof in the challenged phase. In other words, $\Pi_{\mathsf{group}^\star}$ is now a simulated proof while $\mathbf{c}_1^\star$ and $\mathbf{c}_2^\star$ encrypt the same ℓ-bit string. Due to the statistical zero-knowledge property of the underlying argument system, the view of the adversary in Game 2 and Game 3 are statistically indistinguishable, implying that $|\Pr[W_3] - \Pr[W_2]| \in \mathsf{negl}(\lambda)$.

Game 4**:** This game modifies game 3 by changing the distribution of the challenged signature $\Sigma^\star$. Here, we compute $\mathbf{c}_1^\star$ by encrypting the ℓ-bit binary representation of j_1 instead of j_0 and keep $\mathbf{c}_2^\star$ unchanged. By the semantic security of McEliece's encryption scheme for the public key $\mathbf{G}_1$, we have $|\Pr[W_4] - \Pr[W_4]| \in \mathsf{negl}(\lambda)$. We remark that the semantic security can be relied on since we only utilize $\mathsf{sk}_{\mathsf{ME}}^{(2)}$ in the opening oracle queries.

Game 5**:** This game is identical to Game 4 except that we switch back to the key $\mathsf{sk}_{\mathsf{ME}}^{(1)}$ in the opening oracle queries. Note that $\mathcal{A}$'s view remains unchanged unless the event F_2, that $\mathcal{A}$ queries the opening oracle a signature $\Sigma = (\Pi_{\mathsf{group}}, \mathbf{c}_1, \mathbf{c}_2)$ with $\mathbf{c}_1$ and $\mathbf{c}_2$ encrypting different ℓ-bit strings, occurs. However, event F_2 would violate the simulation-soundness of the underlying argument system, we thus have $|\Pr[W_5] - \Pr[W_4]| \leq \Pr[F_2] \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{ss\text{-}sound}}(\lambda)$.

Game 6**:** In this game, we modify again the distribution of the challenged signature $\Sigma^\star$. It changes $\mathbf{c}_2^\star$ to be encryption of the binary representation of j_1 instead of j_0. By the semantic security of McEliece's encryption scheme with respect to $\mathbf{G}_2$, we have $|\Pr[W_6] - \Pr[W_5]| \in \mathsf{negl}(\lambda)$. Note that now both $\mathbf{c}_1^\star$ and $\mathbf{c}_2^\star$ encrypt the same message, therefore $\Pi_{\mathsf{group}}^\star$ is a simulated proof for a true statement.

Game 7**:** This game modifies Game 6 in one aspect: it generates a real proof $\Pi_{\mathsf{group}}^\star$ in the challenged phase. Since the underlying argument system is statistically zero-knowledge, we have $|\Pr[W_7] - \Pr[W_6]| \in \mathsf{negl}(\lambda)$. This is in fact experiment $\mathbf{Exp}_{\mathcal{GS},\mathcal{A}}^{\mathsf{anon\text{-}1}}(\lambda, N)$, hence $\Pr[W_7] = \Pr[\mathbf{Exp}_{\mathcal{GS},\mathcal{A}}^{\mathsf{anon\text{-}1}}(\lambda, N) = 1]$.

As a result, we find that

$$|\Pr[\mathbf{Exp}_{\mathcal{GS},\mathcal{A}}^{\mathsf{anon\text{-}1}}(n, N) = 1] - \Pr[\mathbf{Exp}_{\mathcal{GS},\mathcal{A}}^{\mathsf{anon\text{-}0}}(n, N) = 1]| \in \mathsf{negl}(n),$$

this concludes the proof. □