New Code-Based Privacy-Preserving Cryptographic Constructions

Abstract. This works introduces 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, which enables interesting applications such as range proofs. Our second contribution is a new and simple zero-knowledge argument for Boolean circuits. Given circuit C and output $(u_1, \ldots, u_t)$, our protocol allows to prove knowledge of $(x_1, \ldots, x_\ell)$ such that $C(x_1, \ldots, x_\ell) = (u_1, \ldots, u_t)$. The protocol is relatively fast: When implementing with the AES circuit, the protocol can run in less than 1 second on a personal computer. Moreover, the underlying zero-knowledge techniques can be easily extended to simultaneously handle code-based relations. Our third contribution is a code-based Merkle-tree accumulator supported by ZK argument of membership, which have 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 [53] 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 [34,19] and obtaining worst-case hardness for Learning Parity with Noise (LPN) [20] have been introduced. On the other hand, plausible algorithms for practical applications are being recognized by the community: with 7 PKE/KEM from codes coming into the second round of the NIST Post-Quantum Cryptography Standardization process, the field stands together with lattice-based cryptography [3] 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 design of advanced constructions aiming to protect both privacy and security of users, namely, privacy-preserving cryptographic protocols. These protocols typically require sophisticated combinations of cryptographic algorithms (e.g., signature, encryption,

hashing, commitment), linking together by means of zero-knowledge proofs [38], i.e., proofs that reveal nothing but the truth of the statement. An important example of such protocols is group signature [28], which allows members of a group to anonymously and accountably sign messages on behalf of the group. To design such a scheme, one usually needs a signature scheme, an encryption scheme and a zero-knowledge proof for proving that one has a valid signing certificate and has honestly encrypted some identifying information. To enable such appealing applications, it is thus essential to equip cryptographic algorithms with companions zero-knowledge proofs. As we will discuss briefly later, this important line of research, however, is still rather underdeveloped in the code-based setting.

Our Results. In this work, we introduce 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) [8,9], similar variants of which was recently studied in [6,67,20]. 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 [37]. 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 [42], 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 with larger dimension before being hashed. This makes proving knowledge of a valid opening quite challenging, since one has to prove that the encoding process is done correctly. We overcome this problem by employing a specific permuting technique that works in the framework of Stern's protocol [65] 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. For instance, when combining our building block with the recent techniques from [49], we obtain two immediate applications, both of which have not been previously known under code-based assumptions: (i) ZK arguments for relations among committed integers (i.e., additions and multiplications over $\mathbb{Z}$); (ii) range arguments for committed integers.

Our second contribution is a new and simple ZK argument for Boolean circuit evaluation that interacts smoothly with our first building block. More

specifically, for a given circuit C and a given t-bit output $(u_1, \ldots, u_t)$, our protocol allows to prove knowledge of an ℓ-bit input $(x_1, \ldots, x_\ell)$ such that $C(x_1, \ldots, x_\ell) = (u_1, \ldots, u_t)$, where $(x_1, \ldots, x_\ell)$ could possibly be additionally committed and satisfy other code-based relations. Zero-knowledge protocols addressing both non-algebraic and algebraic statements can find applications in privacy-preserving credentials, two-party computations, cryptocurrencies, etc., and have been recently receiving noticeable attentions [27,2]. However, these protocols employ relatively complex cryptographic mechanisms such as garbled circuits and SNARKs, and only address algebraic statements in the number-theoretic setting. In the code-based setting, a ZK protocol for Boolean circuits was proposed by Jain et al. [42], but it requires the prover to commit to intermediate values in the circuit evaluation process, and thus, has communication and computation costs $\mathcal{O}(\kappa \cdot \lambda \cdot N)$, where N is the circuit size, λ is the security parameter and κ is the number of protocol repetitions (in order to make the soundness error negligibly small). Our protocol significantly improves this result by a factor $\mathcal{O}(\lambda)$, with communication and computation costs $\mathcal{O}(\kappa \cdot N)$. More specifically, for each round of the protocol, on average the prover only needs to send less than 7 bits for each gate in the circuit, and the computation costs of both prover and verifier are relatively small. Our protocol is certainly not as fast as recent proposals [36,26,5,43] based on the MPC-in-the-head approach [41]. However, these protocols are specially designed for handling circuits and are not known to be compatible with algebraic statements. Our protocol, on the other hand, follows a new design approach, in which we preserve a fine-grained control over the secret bits, which allows us to easily combine it with other protocols for code-based relations. While not being practical for large circuits, the protocol is easy to implement, can address all 16 types of binary gates in a universal manner, and can handle NOT gates *for free*. The protocol therefore can be considered as a suitable option when one works with average-size circuits, and when one wants to combine the circuit relation with code-based statements. In Section 7, we provide our implementation results. For the AES Boolean circuit and on a personal computer, the variant of the protocol achieving soundness error 2^{-80} can run in less than 1 second.

Our third contribution is the first code-based accumulator [12] supported by ZK arguments of valid accumulated values. Accumulators are essential building blocks in numerous authentication mechanisms, including ring and group signatures [66,31,48,30], anonymous credentials [24,23,1,50], digital cash [7,25,57], broadcast encryption [35], and authenticated data structures [62,61]. Accumulators with companion ZK proofs have been proposed from number-theoretic assumptions [12,59,45], lattice assumptions [48] and from symmetric-key primitives [30,17], 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 follows Libert et al.'s approach for lattices [48], which relies on Merkle hash trees [56] 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 employ the blueprint of [48] to design 2 prominent anonymity-oriented constructions: ring signature [63] and group signature [28].

- Our ring signature is the first one from code-based assumptions that achieves signature size logarithmic in the cardinality of the ring. Previous constructions [54,29,55,21] 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 compatible with ZK protocols.
- Our group signature is also the first one that produces logarithmic-size signatures in the scope of code-based cryptography. Compared with previous works [33,4], our scheme not only has shorter signatures (for large groups), but also achieve the stronger notion of CCA-anonymity. The scheme follows the BMW model [11] for static groups, but it can be easily extended in the same manner as [51] to the first code-based fully-dynamic scheme [18], where members can join and leave the group after the setup phase.

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 $j \in [k/c]$. 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 $\mathsf{B}_{2^c}^1$ whose sole 1 entry is at the t_j-th position for $j \in [0, 2^c - 1]$. 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 [65], 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 basic technique above 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 [37] 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 [48] and our techniques for proving correctness of re-encoding at each step in a tree path, we get a Merkle-tree-accumulator supported by 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 [48], 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 [60] and make it CCA2-secure in the random oracle model via the Naor-Yung transformation [58]. Both our ring and group signatures feature logarithmic-size signatures, thanks to the tree structure.

Let us now discuss our technique for handling boolean circuits. Let C be a boolean circuit that has N gates, each of which is labelled by a binary operation. Note that there are 16 types of binary operations in total, including AND, OR, XOR, NAND and NOR. The goal is to prove in ZK the possession of an $(x_1, \ldots, x_\ell)$ such that $C(x_1, \ldots, x_\ell) = (u_1, \ldots, u_t)$. The protocol is based on 2 crucial ideas.

The first idea is to reduce the problem of proving the circuit relation to the problem of proving equations over $\mathbb{Z}_2$. Suppose that a gate labelled by operation $\circ$ takes as inputs bits x, y and outputs bit z. That is, we have $x \circ y = z$. We then view the relation as the equation

$$z \ \oplus \ x \circ y = 0 \bmod 2, \tag{3}$$

where $\oplus$ denotes the addition modulo 2, and capture the circuit evaluation process by N such equations. For the last t equations, the right hand side are $(u_1, \ldots, u_t)$, indicating the circuit output.

Now, if we divide the reduced problem into N sub-problems of proving a single equation, then things become tricky. Firstly, we have to demonstrate the possession of bits x, y, z satisfying the equation, with respect to operation $\circ$. Secondly, x, y, z may appear in other equations, e.g., they may be the inputs or outputs of other gates. We therefore need a mechanism that allows to prove equation of the form (3) for arbitrary binary operation $\circ$, that preserves the ability to prove that bits x, y, z satisfy other relations.

In the literature of Stern-like protocols [65], a technique for handling a single bit z was proposed in [48], and a technique for treating bit product $x \cdot y$ was suggested in [47]. These techniques consists of extending z and $x \cdot y$ into vectors:

$$\mathsf{enc}(z) \stackrel{\mathsf{def}}{=} (\overline{z}, z) \in \{0,1\}^2; \quad \mathsf{ext}(x, y) \stackrel{\mathsf{def}}{=} (\ \overline{x} \cdot \overline{y},\ \overline{x} \cdot y,\ x \cdot \overline{y},\ x \cdot y\ ) \in \{0,1\}^4,$$

where $\bar{b}$ denotes the bit $1 - b$, then applying specific types of permutations on the extended vector. To this end:

- For bit c, define F_c that transforms $\mathbf{u} = (u_0, u_1) \in \mathbb{Z}^2$ into $F_c(\mathbf{u}) = (u_c, u_{\overline{c}})$.
- For bits c_1, c_2, define T_{c_1,c_2} that transforms $\mathbf{v} = (v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}) \in \mathbb{Z}^4$ into $T_{c_1,c_2}(\mathbf{v}) = (\ v_{c_1,c_2},\ v_{c_1,\overline{c}_2},\ v_{\overline{c}_1,c_2},\ v_{\overline{c}_1,\overline{c}_2}\ )$.

The goal is to enable the following equivalences:

$$\mathbf{u} = \mathsf{enc}(z) \iff F_c(\mathbf{u}) = \mathsf{enc}(z \oplus c) \tag{4}$$

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

In the framework of Stern's protocol [65], these techniques work as follows. The prover prepares $\mathbf{u} = \mathsf{enc}(z), \mathbf{v} = \mathsf{ext}(x, y)$, permutes them using uniformly random bits c, c_1, c_2, and demonstrates to the verifier that the right-hand sides of (4) and (5) hold. The verifier is thus convinced that the left-hand sides also hold, which implies that the prover indeed knows a bit z and a bit of the form $x \cdot y$. Meanwhile, he cannot learn any additional information about z, x, y, thanks to the "one-time pads" c, c_1, c_2. Furthermore, to prove that z, x, y also appear in other equations, it suffices to use the *same* "one-time pads" at their other appearances.

The above techniques allow us to prove knowledge of z and $x \cdot y = x\ \mathsf{AND}\ y$. However, we would like to handle not just the AND operation, but all 16 operations. Here comes our second idea. Indeed, somewhat interestingly, we can develop the technique from [47] to make it applicable to arbitrary binary boolean operation $\circ$. Specifically, for any operation $\circ : \{0,1\} \times \{0,1\} \to \{0,1\}$ and any two bits x, y, we define the vector

$$\mathsf{ext}_\circ(x, y) \stackrel{\mathsf{def}}{=} \big(\overline{x} \circ \overline{y},\ \overline{x} \circ y,\ x \circ \overline{y},\ x \circ y\big) \in \{0,1\}^4.$$

Then we prove that the generalized version of (5) also holds, i.e.,

$$\mathbf{v} = \mathsf{ext}_\circ(x, y) \iff T_{c_1,c_2}(\mathbf{v}) = \mathsf{ext}_\circ(x \oplus c_1, y \oplus c_2).$$

Having equipped with this insight, we are now ready to describe how our sub-protocol for an equation of the form (3) goes. 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 the equation can be equivalently written as:

$$\mathbf{M}_2 \cdot \mathsf{enc}(z) + \mathbf{M}_4 \cdot \mathsf{ext}_\circ(x,y) = 0 \bmod 2. \tag{6}$$

Then, to prove that $\mathsf{enc}(z)$ and $\mathsf{ext}_\circ(x,y)$ are well-formed while x, y, z may simultaneously satisfy other equations, we use the permuting techniques discussed above. To prove that (6) holds in ZK, we sample uniformly random $\mathbf{r}_2 \in \mathbb{Z}_2^2$ and $\mathbf{r}_4 \in \mathbb{Z}_2^4$, and convince the verifier that $\mathbf{M}_2 \cdot (\mathsf{enc}(z) + \mathbf{r}_2) + \mathbf{M}_4 \cdot (\mathsf{ext}_\circ(x,y) + \mathbf{r}_4) = \mathbf{M}_2 \cdot \mathbf{r}_2 + \mathbf{M}_4 \cdot \mathbf{r}_4 \bmod 2$.

By combining N such sub-protocols, we obtain a mechanism for proving the system of equations that capture the circuit evaluation process. Furthermore, whenever we need to prove that the bits $x_1, \ldots, x_\ell$ satisfy other algebraic relations, e.g., they are committed via our commitment scheme, we can use the same "one-time pads" to mask them in multiple layers.

ORGANIZATION. In Section 2, we recall the background on code-based hash functions and zero-knowledge arguments. In Section 3, we present the core techniques underlying our protocols. Our commitment scheme and its companion ZK arguments are presented in Section 4. Our protocol for circuits is given in Section 5. Our accumulator and its supporting protocol is presented in Section 6. Implementation results of our protocol for circuits is given in Section 7. Due to space restriction, we defer the descriptions and analyses of our ring and group signature schemes to the Appendix.

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) [8,9], 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 [6,67,20], 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 [16,15]. 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 hamming wight 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 $j \in [0, 2^c - 1]$. $\mathsf{RE}(\mathbf{x})$ 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 [8,9], 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 [52] and binary Shortest Vector Problem [6], 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 [8]. In practice, for appropriate choices of n, k, c, the best known algorithms require exponential times in the security parameter. See [15] 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* [39], where the latter assumes that there exists a PPT extractor which takes as input a set of valid transcripts with respect to 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 [65] protocols. In particular, they are Σ-protocols in the generalized sense defined in [42,14] (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 proposed in [46], 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 [46]. 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:

$$\begin{cases} \mathbf{w} \in \mathsf{VALID} \iff \Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}, \\ \text{If } \mathbf{w} \in \mathsf{VALID} \text{ and } \phi \text{ is uniform in } \mathcal{S}, \text{ then } \Gamma_\phi(\mathbf{w}) \text{ is uniform in } \mathsf{VALID}. \end{cases} \tag{7}$$

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 (7) 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 let 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 deferred to the Appendix A.

Theorem 1 ([46]). *Assume that* COM *is a statistically hiding and computationally binding string commitment scheme. Then, the protocol in Figure 3 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* $\mathbf{M} \cdot \mathbf{w}' = \mathbf{v}$*.*

3 Permuting Techniques

3.1 Techniques for Handling Well-Formed Regular Words

In our constructions of commitments and accumulators, we encode inputs $\mathbf{v} \in \{0,1\}^m$, for some $m \in \mathbb{Z}^+$, to regular words $\mathbf{y} = \mathsf{RE}(\mathbf{v}) \in \mathsf{Regular}(m, c)$, where c divides m. Thus, in order to design companion zero-knowledge arguments for these constructions, we need a mechanism allowing to prove the correctness of the encoding process. 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:

$$\mathbf{x} = \Delta_{2^c}(\mathbf{v}) \iff E_{\mathbf{s}}(\mathbf{x}) = \Delta_{2^c}(\mathbf{v} \oplus \mathbf{s}). \tag{8}$$

- 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:

$$\mathbf{y} = \mathsf{RE}(\mathbf{v}) \iff E'_{\mathbf{t}}(\mathbf{y}) = \mathsf{RE}(\mathbf{v} \oplus \mathbf{t}). \tag{9}$$

The equivalence in (9) 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 (9) 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" $\mathbf{t}$.

3.2 Techniques for Handling Arbitrary Binary Operations

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:

$$\mathbf{v} = \mathsf{enc}(b) \iff F_c(\mathbf{v}) = \mathsf{enc}(b \oplus c). \tag{10}$$

- 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:

$$\mathbf{w} = \mathsf{Encode}(\mathbf{b}) \iff F'_{\mathbf{t}}(\mathbf{w}) = \mathsf{Encode}(\mathbf{b} \oplus \mathbf{t}). \tag{11}$$

The authors of [48] showed that the equivalence (10) 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 [47], 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 $b_1 \cdot b_2$. Next, define a specific type of permutation, associated with two bits c_1, c_2, which is formalized as follows.

Definition 2 (Permutation $T_{\cdot,\cdot}(\cdot)$). *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*

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

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

$$\mathbf{v} = \mathsf{ext}(b_1, b_2) \iff T_{c_1,c_2}(\mathbf{v}) = \mathsf{ext}(b_1 \oplus c_1, b_2 \oplus c_2). \tag{12}$$

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.

3.3 From 1 Gate to 16 Gates

We aim to design a mechanism that can be used to prove that a bit has the form $x_1 \circ x_2$, where x_1, x_2 are secret bits, and $\circ$ could be an arbitrary binary boolean operation. Note that, in total, there are 16 such operations, including AND, OR, XOR, NAND and NOR. Although it may be sufficient to handle just a single functionally complete gate, e.g., NAND, we aim to address all 16 gates using a universal method. Such a treatment does provide flexibility, since we do not have to convert a given circuit into an all-NAND one before running the protocol.

Our starting point is the technique from [47] for proving bit product, which we recalled above. As the bit-wise multiplication operation "$\cdot$" can be interpreted as the AND operation, this would give a method for proving the well-formedness of x_1 AND x_2, but would not be sufficient for our purpose.

We then investigate the core ideas underlying the extending-then-permuting technique from [47], and observe that, the bit-wise operation in question does not necessarily have to be the AND operation. Indeed, somewhat interestingly, we find that the technique can be generalized to be applicable to arbitrary binary boolean operation $\circ$. Let us begin with the following definition.

Definition 3 (Extended vector $\mathsf{ext}_\circ(\cdot,\cdot)$). *For any binary boolean operation $\circ : \{0,1\} \times \{0,1\} \to \{0,1\}$, and for any $b_1, b_2 \in \{0,1\}$, define the vector $\mathsf{ext}_\circ(b_1, b_2) \in \{0,1\}^4$ as follows:*

$$\mathsf{ext}_\circ(b_1, b_2) = \left(\bar{b}_1 \circ \bar{b}_2,\ \bar{b}_1 \circ b_2,\ b_1 \circ \bar{b}_2,\ b_1 \circ b_2\right).$$

Our definition of $\mathsf{ext}_\circ(\cdot,\cdot)$ subsumes the definition of $\mathsf{ext}(\cdot,\cdot)$ from [47] as a special case when $\circ$ is AND. Next, we prove in Theorem 1 that the equivalence (12) still holds with respect to $\mathsf{ext}_\circ(\cdot,\cdot)$ and permutation $T_{\cdot,\cdot}(\cdot)$.

Lemma 1. *For any binary boolean operation $\circ$, any $b_1, b_2, c_1, c_2 \in \{0,1\}$ and any $\mathbf{v} = (v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}) \in \mathbb{Z}^4$, it holds that:*

$$\mathbf{v} = \mathsf{ext}_\circ(b_1, b_2) \iff T_{c_1,c_2}(\mathbf{v}) = \mathsf{ext}_\circ(b_1 \oplus c_1, b_2 \oplus c_2). \tag{13}$$

Proof. Let $\mathbf{w} = (w_{0,0}, w_{0,1}, w_{1,0}, w_{1,1}) \in \mathbb{Z}^4$ be the vector $T_{c_1,c_2}(\mathbf{v})$. By definition 2, we have:

$$\begin{aligned} & w_{0,0} = v_{c_1,c_2},\ w_{0,1} = v_{c_1,\bar{c}_2},\ w_{1,0} = v_{\bar{c}_1,c_2},\ w_{1,1} = v_{\bar{c}_1,\bar{c}_2} \\ \iff & w_{i,j} = v_{c_1 \oplus i,\, c_2 \oplus j},\ \forall (i,j) \in \{0,1\} \times \{0,1\} \end{aligned}$$

Meanwhile, let $\mathbf{t} = (t_{0,0}, t_{0,1}, t_{1,0}, t_{1,1}) \in \{0,1\}^4$ be the vector $\mathsf{ext}_\circ(b_1 \oplus c_1, b_2 \oplus c_2)$. By Definition 3, we have:

$$\begin{aligned} & \begin{cases} t_{0,0} = (\overline{b_1 \oplus c_1}) \circ (\overline{b_2 \oplus c_2}) = (\bar{b}_1 \oplus c_1 \oplus 0) \circ (\bar{b}_2 \oplus c_2 \oplus 0) \\ t_{0,1} = (\overline{b_1 \oplus c_1}) \circ (b_2 \oplus c_2) = (\bar{b}_1 \oplus c_1 \oplus 0) \circ (\bar{b}_2 \oplus c_2 \oplus 1) \\ t_{1,0} = (b_1 \oplus c_1) \circ (\overline{b_2 \oplus c_2}) = (\bar{b}_1 \oplus c_1 \oplus 1) \circ (\bar{b}_2 \oplus c_2 \oplus 0) \\ t_{1,1} = (b_1 \oplus c_1) \circ (b_2 \oplus c_2) = (\bar{b}_1 \oplus c_1 \oplus 1) \circ (\bar{b}_2 \oplus c_2 \oplus 1) \end{cases} \\ \iff & t_{i,j} = (\bar{b}_1 \oplus c_1 \oplus i) \circ (\bar{b}_2 \oplus c_2 \oplus j),\ \forall (i,j) \in \{0,1\} \times \{0,1\}. \end{aligned}$$

Hence, the following equivalences hold for any $b_1, b_2, c_1, c_2 \in \{0,1\}$ and any $\mathbf{v} = (v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}) \in \mathbb{Z}^4$:

$$\begin{aligned} & T_{c_1,c_2}(\mathbf{v}) = \mathsf{ext}_\circ(b_1 \oplus c_1, b_2 \oplus c_2) \iff \mathbf{w} = \mathbf{t} \\ \iff & v_{c_1 \oplus i,\, c_2 \oplus j} = (\bar{b}_1 \oplus c_1 \oplus i) \circ (\bar{b}_2 \oplus c_2 \oplus j),\ \forall (i,j) \in \{0,1\} \times \{0,1\} \\ \iff & v_{i,j} = (\bar{b}_1 \oplus i) \circ (\bar{b}_2 \oplus j),\ \forall (i,j) \in \{0,1\} \times \{0,1\} \\ \iff & v_{0,0} = \bar{b}_1 \circ \bar{b}_2;\ v_{0,1} = \bar{b}_1 \circ b_2;\ v_{1,0} = b_1 \circ \bar{b}_2;\ v_{1,1} = b_1 \circ b_2 \\ \iff & \mathbf{v} = \mathsf{ext}_\circ(b_1, b_2). \end{aligned}$$

This concludes the lemma. □

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

In this section, we first recall the definitions of statistically hiding and computationally binding commitments. We then present our code-based instantiation in Section 4.2. To widen its potential applications in code-based privacy-preserving protocols, we put forward a ZKAoK of a valid opening, described in Section 4.3.

4.1 Definitions

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

Definition 4. *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.

4.2 Our Construction

Given security parameter λ, choose $n = \mathcal{O}(\lambda)$, $k \geq n + 2\lambda + \mathcal{O}(1)$. Let the message space be $\mathcal{M} = \{0,1\}^\ell$, and let c be a constant dividing ℓ and k. Let $m_0 = 2^c \cdot \ell / c$, $m_1 = 2^c \cdot k / c$ 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 $\mathbf{x} \in \{0,1\}^\ell$ 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 $(\mathbf{c}, \mathbf{s})$. 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 $\mathbf{x} \in \{0,1\}^\ell$ 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 2. *The scheme is computationally binding, assuming the hardness of the 2-*$\mathsf{RNSD}_{n,\ell+k,c}$ *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 $\mathbf{t} \neq \mathbf{0}$, $\mathbf{t} \in \mathsf{2\text{-}Regular}(\ell + k, c)$ and $\mathbf{B} \cdot \mathbf{t} = \mathbf{0}$. In other words, $\mathbf{t}$ is a solution to the $\mathsf{2\text{-}RNSD}_{n,\ell+k,c}$ problem associated with uniformly random matrix $\mathbf{B}$. □

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

Lemma 3 (Leftover hash lemma, adapted from [37]). *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 $\mathbf{x} \in \{0,1\}^\ell$, 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 [44], 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.

4.3 ZKAoK of a Valid Opening

We now describe a ZKAoK of a valid opening for the commitment scheme from Section 4.2. 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 3.1. Let $\mathbf{z} = (\mathbf{x} \parallel \mathbf{s}) \in \{0,1\}^{\ell+k}$ and $\mathbf{w}_{\mathsf{com}} = \mathsf{RE}(\mathbf{z}) \in \mathsf{Regular}(\ell + k, c) \subset \{0,1\}^{2^c \cdot (\ell+k)/c}$. 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 $\mathsf{VALID}_{\mathsf{com}} = \mathsf{Regular}(\ell + k, c)$ and $\mathcal{S} = \{0,1\}^{\ell+k}$. For $\mathbf{t} = (\mathbf{t}_1 \| \ldots \| \mathbf{t}_{(\ell+k)/c}) \in \mathcal{S}$ consisting of $(\ell + k)/c$ blocks of length c, it follows from (9) that we have

$$\mathbf{w}_{\mathsf{com}} = \mathsf{RE}(\mathbf{z}) \iff E'_{\mathbf{t}}(\mathbf{w}_{\mathsf{com}}) = \mathsf{RE}(\mathbf{z} \oplus \mathbf{t}). \tag{14}$$

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 order words, the conditions of (7) 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 3 with public input $(\mathbf{B}, \mathbf{c})$ and prover's witness $\mathbf{w}_{\mathsf{com}}$, and obtain a ZKAoK for R_{com}.

5 Zero-Knowledge Arguments for Boolean Circuits

5.1 Description of the Protocol

Given the techniques in Section 3.2 and the development in Section 3.3, one can think of a natural method to solve the problem of proving boolean circuit satisfiability. This method consists of committing to the input bits and output bits at each gate, and addressing the problem *indirectly* by proving relations $x \circ y = z$ among committed bits x, y, z. While this method is common in the literature (in particular, in the code-based protocol from [42]), it incurs communication cost at least $\mathcal{O}(\lambda) \cdot N$, where λ is the security parameter and N is the circuit size. Here, we employ a fundamentally different idea that will help to significantly reduce the cost: We capture the circuit evaluation process by a simple system of equations over $\mathbb{Z}_2$, which can be proved *directly* in ZK. Our protocol can handle arbitrary circuits of fan-ins at most 2, but for simplicity of description, we consider circuits represented entirely by binary gates, of polynomial size N and input size ℓ.

Let $C : \{0,1\}^\ell \to 0,1^t$ be a boolean circuit that has N gates labelled by binary operations $\circ_1, \ldots, \circ_N$. The topology of C is specified by two publicly known functions g and h mapping $\{1, \ldots, N\}$ to $\{1, \ldots, \ell + N - t\}$. For an ℓ-bit input $(x_1, \ldots, x_\ell)$ and t-bit output $(u_1, \ldots, u_t)$, the assignments to non-input wires in C are denoted as $x_{\ell+1}, \ldots,\ x_{\ell+N-t},\ x_{\ell+N-t+1}, \ldots,\ x_{\ell+N}$ - where the last t elements are assignments to output wires, and are computed as:

$$\forall\, j = 1, \ldots, N : \ x_{\ell+j} = x_{g(j)} \ \circ_j \ x_{h(j)}.$$

Note that, $C(x_1, \ldots, x_\ell) = (u_1, \ldots, u_t)$ if and only if:

$$\forall\ i \in [t] : \quad x_{\ell+N-t+i} \ = \ x_{g(N-t+i)} \quad \circ_{N-t+i} \quad x_{h(N-t+i)} \ = \ u_i.$$

We will build a zero-knowledge argument of knowledge for the relation R_{circuit}, defined as

$$R_{\text{circuit}} = \big\{ \big((C, g, h, u_1, \ldots, u_t), x_1, \ldots, x_\ell\big) : C(x_1, \ldots, x_\ell) = (u_1, \ldots, u_t) \big\}.$$

Our approach is to capture the evaluation process by the following N equations modulo 2:

$$\begin{aligned}
x_{\ell+1} \ \oplus \ x_{g(1)} \ \circ_1 \ x_{h(1)} &= 0, \\
&\vdots \\
x_{\ell+N-t} \ \oplus \ x_{g(N-t)} \ \circ_{N-t} \ x_{h(N-t)} &= 0, \\
x_{g(N-t+1)} \ \circ_{N-t+1} \ x_{h(N-t+1)} &= u_1, \\
&\vdots \\
x_{g(N)} \circ_N x_{h(N)} &= u_t.
\end{aligned}$$

Given our interpretation, to prove knowledge of secret input $(x_1, \ldots, x_\ell)$ such that $C(x_1, \ldots, x_\ell) = (u_1, \ldots, u_t)$ for a given output $(u_1, \ldots, u_t)$, it suffices to prove knowledge of bits $x_1, \ldots, x_\ell, x_{\ell+1}, \ldots, x_{\ell+N-t}$ such that the above system of N equations hold[1]. To this end, we will translate the system into an equivalent form that is compatible with the techniques in Sections 3.2 and 3.3. At first, we perform the following extensions:

- For each $j = \ell+1, \ldots, \ell+N-t$, we extend bit $x_j \in \{0,1\}$ to vector $\mathsf{enc}(x_j) = (\overline{x}_j, x_j) \in \{0,1\}^2$. Note that $x_j = \mathbf{M}_2 \cdot \mathsf{enc}(x_j)$, where $\mathbf{M}_2$ is the matrix $[0,1] \in \mathbb{Z}_2^{1\times 2}$.
- For each $j = 1, \ldots, N$, we extend bit $x_{g(j)} \circ_j x_{h(j)} \in \{0,1\}$ to vector $\mathsf{ext}_{\circ_j}(x_{g(j)}, x_{h(j)}) \in \{0,1\}^4$. Note that $x_{g(j)} \circ_j x_{h(j)} = \mathbf{M}_4 \cdot \mathsf{ext}_{\circ_j}(x_{g(j)}, x_{h(j)})$, where $\mathbf{M}_4$ is the matrix $[0,0,0,1] \in \mathbb{Z}_2^{1\times 4}$.

Then, the given equations can be equivalently written as:

$$\begin{aligned}
\mathbf{M}_2 \cdot \mathsf{enc}(x_{\ell+1}) \oplus \mathbf{M}_4 \cdot \mathsf{ext}_{\circ_1}\big(x_{g(1)}, x_{h(1)}\big) &= 0,\\
&\vdots\\
\mathbf{M}_2 \cdot \mathsf{enc}(x_{\ell+N-t}) \oplus \mathbf{M}_4 \cdot \mathsf{ext}_{\circ_{N-t}}\big(x_{g(N-t)}, x_{h(N-t)}\big) &= 0,\\
\mathbf{M}_4 \cdot \mathsf{ext}_{\circ_{N-t+1}}\big(x_{g(N-t+1)}, x_{h(N-t+1)}\big) &= u_1,\\
&\vdots\\
\mathbf{M}_4 \cdot \mathsf{ext}_{\circ_N}\big(x_{g(N)}, x_{h(N)}\big) &= u_t.
\end{aligned}$$

Next, using linear algebra, we unify the newly obtained equations into an equation of the form $\mathbf{M}_C \cdot \mathbf{w}_C = \mathbf{v}_C \bmod 2$, where matrix $\mathbf{M}_C$ and vector $\mathbf{v}_C$ are of the form,

$$\mathbf{M}_C = \begin{bmatrix}
\mathbf{M}_2 & & \mathbf{M}_4 & & & \\
& \ddots & & \ddots & & \\
& & \mathbf{M}_2 & & \mathbf{M}_4 & \\
& & & & & \mathbf{M}_4 \\
& & & & & & \ddots \\
& & & & & & & \mathbf{M}_4
\end{bmatrix} \in \mathbb{Z}_2^{N\times(6N-2t)}; \quad \mathbf{v}_C = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ u_1 \\ \vdots \\ u_t \end{pmatrix} \in \mathbb{Z}_2^N,$$

and vector $\mathbf{w}_C \in \{0,1\}^{6N-2t}$ is formed as follows:

$$\begin{aligned}
\mathbf{w}_C = \big(\mathsf{enc}(x_{\ell+1}) \,\|\, \ldots \,\|\, \mathsf{enc}(x_{\ell+N-t}) \|\\
\|\, \mathsf{ext}_{\circ_1}(x_{g(1)}, x_{h(1)}) \,\|\, \ldots \,\|\, \mathsf{ext}_{\circ_N}(x_{g(N)}, x_{h(N)})\big). \quad (15)
\end{aligned}$$

We therefore have translated the equation $C(x_1, \ldots, x_\ell) = (u_1, \ldots, u_t)$ into equation $\mathbf{M}_C \cdot \mathbf{w}_C = \mathbf{v}_C \bmod 2$, where vector $\mathbf{w}_C$ is as in (15).

[1] Note that, $x_{g(j)}, x_{h(j)} \in \{x_1, \ldots, x_\ell, x_{\ell+1}, \ldots, x_{\ell+N-t}\}$, for all $j = 1, \ldots, N$.

To prove in ZK that $\mathbf{w}_C$ is well-formed, we introduce the following technique. Let $L_0 = \ell + N - t$ and $L = 6N - 2t$. For any $\mathbf{b} = (b_1, \ldots, b_{L_0}) \in \{0,1\}^{L_0}$, define $\mathsf{good}(\mathbf{b}) \in \{0,1\}^L$ as the vector of the form

$$\big(\mathsf{enc}(b_{\ell+1}) \,\|\, \ldots \,\|\, \mathsf{enc}(b_{\ell+N-t}) \,\|\, \mathsf{ext}_{\circ_1}(b_{g(1)}, b_{h(1)}) \,\|\, \ldots \,\|\, \mathsf{ext}_{\circ_N}(b_{g(N)}, b_{h(N)})\big).$$

Let $\mathbf{x}^\star = (x_1, \ldots, x_\ell, x_{\ell+1}, \ldots, x_{\ell+N-t}) \in \{0,1\}^{L_0}$, then, by construction, we have $\mathbf{w}_C = \mathsf{good}(\mathbf{x}^\star)$. Now, to prove in ZK that $\mathbf{w}_C = \mathsf{good}(\mathbf{x}^\star)$, we will employ a combination of permuting techniques $F_{\cdot}(\cdot)$ and $T_{\cdot,\cdot}(\cdot)$ from Sections 3.2 and 3.3. To this end, let $\mathcal{S} = \{0,1\}^{L_0}$, and for any $\mathbf{e} = (e_1, \ldots, e_{L_0}) \in \mathcal{S}$, we define the permutation $\Theta_{\mathbf{e}}$ of L elements as follows. When applied to vector

$$\mathbf{a} = \big(\, \mathbf{a}_{\ell+1} \,\|\, \ldots \,\|\, \mathbf{a}_{\ell+N-t} \,\|\, \widehat{\mathbf{a}}_1 \,\|\, \ldots \,\|\, \widehat{\mathbf{a}}_N \big) \in \mathbb{Z}^L,$$

where $\mathbf{a}_{\ell+1}, \ldots, \mathbf{a}_{\ell+N-t} \in \mathbb{Z}^2$ and $\widehat{\mathbf{a}}_1, \ldots, \widehat{\mathbf{a}}_N \in \mathbb{Z}^4$, it transforms vector $\mathbf{a}$ into vector $\Theta_{\mathbf{e}}(\mathbf{a})$ of the form

$$\big(\;\; F_{e_{\ell+1}}(\mathbf{a}_{\ell+1}) \,\|\, \ldots \,\|\, F_{e_{\ell+N-t}}(\mathbf{a}_{\ell+N-t}) \,\|\, T_{e_{g(1)},e_{h(1)}}(\widehat{\mathbf{a}}_1) \,\|\, \ldots \,\|\, T_{e_{g(N)},e_{h(N)}}(\widehat{\mathbf{a}}_N) \;\;\big).$$

Then, based on facts (10) and (13), we obtain the following equivalence.

$$\mathbf{w}_C = \mathsf{good}(\mathbf{x}^\star) \quad \Longleftrightarrow \quad \Theta_{\mathbf{e}}(\mathbf{w}_C) = \mathsf{good}(\mathbf{x}^\star \oplus \mathbf{e}). \tag{16}$$

Now, define the set $\mathsf{VALID}_C = \{\mathsf{good}(\mathbf{z}) : \;\; \mathbf{z} \in \{0,1\}^{L_0}\} \subset \{0,1\}^L$. Then we note that $\mathbf{w}_C \in \mathsf{VALID}_C$, and for uniformly random $\mathbf{e} \in \mathcal{S}$, the vector $\Theta_{\mathbf{e}} = \mathsf{good}(\mathbf{x}^\star \oplus \mathbf{e})$ is uniformly random in VALID_C. In other words, the conditions of (7) hold with respect to the sets VALID_C, $\mathcal{S}$ and permutations $\{\Theta_{\mathbf{e}} : \;\; \mathbf{e} \in \mathcal{S}\}$ defined above.

We therefore have reduced the circuit relation to an instance of the abstract protocol from Section 2.2. Prior to the interaction, both prover and verifier form matrix $\mathbf{M}_C$ and vector $\mathbf{v}_C$ based on the description of C and $(u_1, \ldots, u_t)$, while the prover computes witness $\mathbf{w}_C \in \mathsf{VALID}_C$ as described. They then interact as in Figure 3. The protocol employs a statistically hiding and computationally binding commitment scheme COM. Based on Theorem 1, we obtain the following result.

Theorem 2. *Assume that* COM *is a statistically hiding and computationally binding string commitment scheme. Then, the protocol described in this section is a statistical* ZKAoK *for the relation* R_{circuit} *with perfect completeness, soundness error* $2/3$*, and communication cost* $\mathcal{O}(\ell + N - t)$*.*

Proof. The perfect completeness and the zero-knowledge property follow from those of the protocol in Figure 3. As for extraction, by running the knowledge extractor of Theorem 1, we obtain $\mathbf{w}'_C \in \mathsf{VALID}_C$ such that $\mathbf{M}_C \cdot \mathbf{w}'_C = \mathbf{v}_C$. Parse $\mathbf{w}'_C$ as

$$\begin{aligned}\mathbf{w}'_C = \big(\mathsf{enc}(x'_{\ell+1}) \,\|\, \ldots \;\; &\| \, \mathsf{enc}(x'_{\ell+N-t}) \| \\ &\| \, \mathsf{ext}_{\circ_1}(x'_{g(1)}, x'_{h(1)}) \,\|\, \ldots \,\|\, \mathsf{ext}_{\circ_N}(x'_{g(N)}, x'_{h(N)})\big).\end{aligned}$$

Then, the bits $x'_{\ell+1}, \ldots, x'_{\ell+N-t}, x'_{g(1)}, x'_{h(1)}, \ldots, x'_{g(N)}, x'_{h(N)}$ satisfy

$$\begin{aligned} x'_{\ell+1} \ \oplus \ x'_{g(1)} \ \circ_1 \ x'_{h(1)} &= 0, \\ &\vdots \\ x'_{\ell+N-t} \ \oplus \ x'_{g(N-t)} \ \circ_{N-t} \ x'_{h(N-t)} &= 0, \\ x'_{g(N-t+1)} \ \circ_{N-t+1} \ x'_{h(N-t+1)} &= u_1, \\ &\vdots \\ x'_{g(N)} \circ_N x'_{h(N)} &= u_t. \end{aligned}$$

In other words, we can extract a set $\{x'_1, \ldots, x'_\ell\} \subseteq \{x'_{g(1)}, x'_{h(1)}, \ldots, x'_{g(N)}, x'_{h(N)}\}$ such that $C(x'_1, \ldots, x'_\ell) = (u_1, \ldots, u_t)$. □

5.2 Remarks

We give a few remarks on the performance of our protocol for Boolean circuits.

1. **Computation cost:** the prover's cost consists of extending and permuting a few vectors of length at most $6N - 2t$, carrying out $8N - 3t$ bit-wise additions, and computing 3 commitments (where COM can be replaced by a hash function like SHA256 in practice). The verifier's cost is similar to that of the prover.
2. **Communication:** As we pointed out in Remark 2, in the case $Ch = 1$, it suffices for the prover to send $\mathbf{x}^\star + \mathbf{e} \in \{0,1\}^{\ell+N-t}$ instead of $\Theta_{\mathbf{e}}(\mathbf{w}_C) = \mathsf{good}(\mathbf{x}^\star + \mathbf{e}) \in \{0,1\}^{6N-2t}$, since the former fully determines the latter. The communication cost for every value of Ch is thus $\zeta = 3|\mathsf{COM}| + 2|\rho_i| + \ell + 7N - 3t$ bits. (In practice, the cost can be reduced around one half by sending the seeds of the PRG used to generate randomness ϕ and $\mathbf{r}_w$, instead of sending these objects.)
3. **NOT-free:** In the description of Section 5.1, for simplicity, we do not consider gates NOT. The protocol can be easily adapted to work with circuits containing many NOT, and interestingly, this can be done for free. For instance, if x_1 appears in the circuit in both forms x_1 and $\mathsf{NOT}(x_1) = \overline{x}_1$, then the prover can compute vector $\mathsf{good}(\cdot)$ with respect to x_1 and $\overline{x}_1$, but only sends $x_1 + e$ (as part of $\mathbf{x}^\star + \mathbf{e}$) to the verifier. The latter, computing $\mathsf{good}(\cdot)$ with respect to $x_1 + e$ and $\overline{x_1 + e}$, should be able to obtain the required value. Here, we make use of the identity $\overline{x}_1 + e = \overline{x_1 + e}$.

We give a toy example on a circuit containing 3 binary gates $\mathsf{AND}, \mathsf{OR}, \mathsf{NAND}$ and 2 gates NOT in Appendix C.

5.3 Proving $C(x_1, \ldots, x_\ell) = (u_1, \ldots, u_t)$ for Committed Inputs

We now demonstrate that our ZK protocol for Boolean circuits from Section 5.1 can be easily extended to additionally prove the algebraic statement that input

$(x_1, \ldots, x_\ell)$ was committed via the commitment scheme of Section 4. Specifically, we consider the relation:

$$R_{\text{combine}} = \big\{ \big((C, g, h, u_1, \ldots, u_t, \mathbf{B} = [\mathbf{B}_0 \mid \mathbf{B}_1], \mathbf{c}), \mathbf{x} = (x_1, \ldots, x_\ell), \mathbf{r}\big) : \\ C(x_1, \ldots, x_\ell) = (u_1, \ldots, u_t) \quad \wedge \quad \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}) + \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{r}) = \mathbf{c} \big\}.$$

A ZKAoK for R_{combine} can be obtained in a rather straightforward manner by combining the protocols from Section 5.1 and Section 4.3, which handle witnesses $\mathbf{w}_C = \mathsf{good}(\mathbf{x}^\star)$ defined in (15) and $\mathbf{w}_{\mathsf{com}} = (\mathsf{RE}(\mathbf{x}) \parallel \mathsf{RE}(\mathbf{r}))$, respectively. Let $\mathbf{w} = (\mathbf{w}_C \parallel \mathbf{w}_{\mathsf{com}}) \in \{0,1\}^{6N-2t+m}$, then we have equation $\mathbf{M} \cdot \mathbf{w} = \mathbf{v}$, where $\mathbf{M} = [\mathbf{M}_C \mid \mathbf{B}]$ and $\mathbf{v} = (\mathbf{v}_C \parallel \mathbf{c})$.

Define $\mathsf{VALID} \subset \{0,1\}^{6N-2t+m}$ as the set of all vectors of the form $\mathbf{w} = (\mathbf{w}_C \parallel \mathbf{w}_{\mathsf{com}})$, such that there exist $\mathbf{x} \in \{0,1\}^\ell$, $x_{\ell+1}, \ldots, x_{\ell+N-t} \in \{0,1\}$, $\mathbf{r} \in \{0,1\}^k$ satisfying

$$\mathbf{w}_C = \mathsf{good}(\mathbf{x}, x_{\ell+1}, \ldots, x_{\ell+N-t}); \quad \mathbf{w}_{\mathsf{com}} = (\ \mathsf{RE}(\mathbf{x}) \parallel \mathsf{RE}(\mathbf{r})\ ).$$

By construction, we have $\mathbf{w} \in \mathsf{VALID}$. Next, let $\mathcal{S} := \{0,1\}^{\ell+N-t} \times \{0,1\}^k$, and for every $\phi = (\mathbf{e}, \mathbf{t}_r) \in \mathcal{S}$, where $\mathbf{e} = (\mathbf{e}_x, e_{\ell+1}, \ldots, e_{\ell+N-t})$ define the permutation Γ_ϕ that transforms vector $\mathbf{w} = (\mathbf{w}_C \parallel \mathbf{w}_{\mathsf{com}}) \in \mathbb{Z}^{6N-2t+m}$ into vector

$$\Gamma_\phi(\mathbf{w}) = \big(\Theta_{\mathbf{e}}(\mathbf{w}_C) \parallel E'_{(\mathbf{e}_x \| \mathbf{t}_r)}(\mathbf{w}_{\mathsf{com}})\big).$$

Note that, Γ_ϕ is defined in a way such that it permutes the bits of $\mathbf{x} = (x_1, \ldots, x_\ell)$ using the same random bits of vector $\mathbf{e}_x$. In the framework of Stern's protocol, this allows us to prove that the same $(x_1, \ldots, x_\ell)$ are involved in both circuit layer and commitment layer.

It can be checked that the sets VALID, $\mathcal{S}$ and permutations $\{\Gamma_\phi : \ \phi \in \mathcal{S}\}$ satisfy the conditions of (7). In other words, we have reduce the relation R_{combine} to an instance of R_{abstract}. Therefore, by running the protocol of Figure 3, we obtain a ZKAoK for R_{combine}.

6 Code-Based Accumulators With Companion Zero-Knowledge Protocols

In this section, we aim to provide our code-based Merkle-tree accumulator from collision-resistant hashing. In Section 6.1, we first recall the definition of a cryptographic accumulator scheme, as well as the security requirement. We then modify the AFS hash functions to support hashing of two inputs in Section 6.2, upon which we build our Merkle-tree accumulator in Section 6.3. In Section 6.4, we describe our zero-knowledge argument of knowledge of an accumulated value. Two interesting applications based on the accumulator: logarithmic size ring signature scheme and group signature scheme are given in the Appendix G and Appendix H.

6.1 Cryptographic Accumulators

We recall the definition of accumulators in the following.

Definition 5. *An accumulator scheme consists of the following a tuple of polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Accu}, \mathsf{WitGen}, \mathsf{Verify})$*:*

- $\mathit{Setup}(1^\lambda)$ *Given a security parameter* 1^λ*, outputs the public parameter* pp*.*
- $\mathit{Accu}_{pp}(R)$ *Take as input a set* R *with* n *data values as* $R = \{\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}\}$*, outputs an accumulator value* $\mathbf{u}$*.*
- $\mathit{WitGen}_{pp}(R, \mathbf{d})$ *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* $\mathbf{d} \notin R$*.*
- $\mathit{Verify}_{pp}(\mathbf{u}, (\mathbf{d}, w))$ *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. The correctness requires that the correctly computed accumulator value has valid verifying witnesses. In other words, 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 [10,24], if it is infeasible to output a valid witness for a value $\mathbf{d}^*$ that is not chosen from the data value set.

Definition 6. A*n accumulator scheme is secure if for all PPT adversaries* $\mathcal{A}$*:*

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

6.2 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 7. *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 7 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 $\mathbf{B}$. □

6.3 Code-Based Merkle-tree Accumulator

In this section, we describe our Merkle-tree accumulator based on the code-based hash functions $\mathcal{H}$ in Definition 7. The construction is adapted from the blueprint by Libert et al. [48], which operates in the setting of lattices.

$\mathsf{Setup}(\lambda)$. 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,

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

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

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

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.

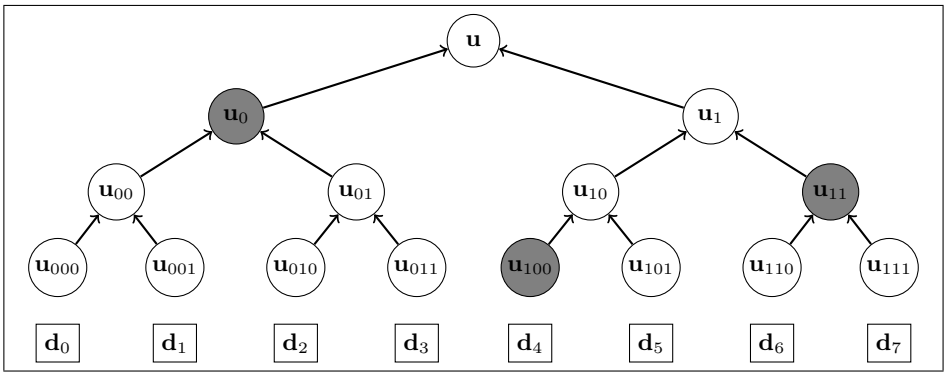

Fig. 1: A Merkle tree with $2^3 = 8$ leaves $\mathbf{d}_0, \ldots, \mathbf{d}_7$. It recursively accumulates the leaf values into value $\mathbf{u}$. The witness of $\mathbf{d}_5$ contains the binary representation of 5, which is (101), and all the gray nodes.

The correctness of the Merkle-tree accumulator scheme follows immediately from the construction. In terms of the security, it is guaranteed by the collision-resistance hash function family $\mathcal{H}$, which means that if the adversary can break the security of the accumulator, then it can find a solution to the $\mathsf{2\text{-}RNSD}_{n,2n,c}$ problem.

Theorem 3. *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 probabilistic polynomial time adversary $\mathcal{B}$ who breaks the security of accumulator scheme with non-negligible probability. By the security definition, $\mathcal{B}$ first receives a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_2^{n\times m}$ that output 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 path $\mathbf{d}^*, \mathbf{k}_{\ell-1}^*, \ldots, \mathbf{k}_1^*, \mathbf{u}^*$, 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 $\mathbf{u}_{j_1^*,\ldots,j_i^*}$. □

6.4 Zero-Knowledge Argument for Code-Based Accumulator

In this section, we describe a statistical zero-knowledge argument that will allow prover $\mathcal{P}$ to convince verifier $\mathcal{V}$ that $\mathcal{P}$ knows a secret value that is correctly accumulated into the root of the above code-based Merkle tree. Specifically, 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:

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

Before we look at the above relation, we first present an additional permuting technique, which is developed from previous work [48] and the permuting technique in Section 3.1.

- 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 works on $\mathbf{z} = \begin{pmatrix} \mathbf{z}_0 \\ \mathbf{z}_1 \end{pmatrix} \in \{0,1\}^m$ as follows, where $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^{m/2}$. It transforms $\mathbf{z}$ to $\Psi_{e,\mathbf{t}}(\mathbf{z}) = \begin{pmatrix} E'_{\mathbf{t}}(\mathbf{z}_e) \\ E'_{\mathbf{t}}(\mathbf{z}_{\bar{e}}) \end{pmatrix}$. Precisely, 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 (9) that the following equivalences hold:

$$\begin{cases} \mathbf{z} = \mathsf{Ext}(b, \mathsf{RE}(\mathbf{v})) & \iff \Psi_{e,\mathbf{t}}(\mathbf{z}) = \mathsf{Ext}(b \oplus e, \mathsf{RE}(\mathbf{v} \oplus \mathbf{t})) \\ \mathbf{y} = \mathsf{Ext}(\bar{b}, \mathsf{RE}(\mathbf{w})) & \iff \Psi_{\bar{e},\mathbf{t}}(\mathbf{y}) = \mathsf{Ext}(\bar{b} \oplus \bar{e}, \mathsf{RE}(\mathbf{w} \oplus \mathbf{t})). \end{cases} \tag{17}$$

Now let us look the the equations associated with the relation R_{acc}. This 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:

$$\mathbf{v}_i = \begin{cases} h_{\mathbf{B}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), \text{ if } j_{i+1} = 0; \\ h_{\mathbf{B}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), \text{ if } j_{i+1} = 1. \end{cases} \tag{18}$$

Rewrite equation (18) 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, it is necessary and sufficient to construct an argument system in which $\mathcal{P}$ convinces $\mathcal{V}$ 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

$$\begin{cases} \mathbf{B} \cdot \mathsf{Ext}(j_1, \mathsf{RE}(\mathbf{v}_1)) \oplus \mathbf{B} \cdot \mathsf{Ext}(\bar{j}_1, \mathsf{RE}(\mathbf{w}_1)) = \mathbf{u}; \\ \mathbf{B} \cdot \mathsf{Ext}(j_2, \mathsf{RE}(\mathbf{v}_2)) \oplus \mathbf{B} \cdot \mathsf{Ext}(\bar{j}_2, \mathsf{RE}(\mathbf{w}_2)) \oplus \mathbf{v}_1 = \mathbf{0}. \\ \qquad\qquad\qquad\qquad \cdots\cdots\cdots \\ \mathbf{B} \cdot \mathsf{Ext}(j_\ell, \mathsf{RE}(\mathbf{v}_\ell)) \oplus \mathbf{B} \cdot \mathsf{Ext}(\bar{j}_\ell, \mathsf{RE}(\mathbf{w}_\ell)) \oplus \mathbf{v}_{\ell-1} = \mathbf{0}. \end{cases} \tag{19}$$

Next, we apply the function Encode defined in Section 3.2 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

$$\begin{cases} \mathbf{y}_i = \mathsf{Ext}(j_i, \mathsf{RE}(\mathbf{v}_i)) \in \{0,1\}^m \\ \mathbf{z}_i = \mathsf{Ext}(\bar{j}_i, \mathsf{RE}(\mathbf{w}_i)) \in \{0,1\}^m \end{cases} \tag{20}$$

Therefore, the equations in (19) is equivalent to the following.

$$\begin{cases} \mathbf{B} \cdot \mathbf{y}_1 \oplus \mathbf{B} \cdot \mathbf{z}_1 = \mathbf{u}; \\ \mathbf{B} \cdot \mathbf{y}_2 \oplus \mathbf{B} \cdot \mathbf{z}_2 \oplus \mathbf{I}_n^* \cdot \mathbf{x}_1 = \mathbf{0}. \\ \qquad \cdots\cdots\cdots \\ \mathbf{B} \cdot \mathbf{y}_\ell \oplus \mathbf{B} \cdot \mathbf{z}_\ell \oplus \mathbf{I}_n^* \cdot \mathbf{x}_{\ell-1} = \mathbf{0}. \end{cases} \tag{21}$$

Now through some basic algebra, we can transform the equations in (26) into an 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

$$\mathbf{w}_A = \left(\mathbf{y}_1 \parallel \ldots \parallel \mathbf{y}_\ell \parallel \mathbf{z}_1 \parallel \ldots \parallel \mathbf{z}_\ell \parallel \mathbf{x}_1 \parallel \ldots \parallel \mathbf{x}_{\ell-1} \right) \tag{22}$$

Now we specify the set VALID_A that contains 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 (7) hold. Let VALID_A contains all vectors $\mathbf{w}'_A = \left(\mathbf{y}'_1 \| \ldots \| \mathbf{y}'_\ell \| \mathbf{z}'_1 \| \ldots \| \mathbf{z}'_\ell \| \mathbf{x}'_1 \| \ldots \| \mathbf{x}'_{\ell-1}\right) \in \{0,1\}^L$ 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 $\mathcal{S}_A$ be of the following form:

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

Then, for each $\phi = \left(\mathbf{b}_1 \parallel \ldots \parallel \mathbf{b}_\ell \parallel \mathbf{e}_1 \parallel \ldots \parallel \mathbf{e}_\ell \parallel g_1 \parallel \ldots \parallel g_\ell \right)$, 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 $i \in [\ell-1]$ to

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

Based on the equivalences observed in (11) and (17), it can be checked that the conditions in (7) are satisfied. We thus have reduced the considered relation into an instance of $\mathrm{R}_{\mathrm{abstract}}$.

The interactive protocol. 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 3. The protocol utilizes the statistically hiding and computationally binding string commitment scheme from Section 4 to obtain the desired statistical ZKAoK. The protocol has communication cost $\mathcal{O}(L) = \ell \cdot \mathcal{O}(m+n)$ bits.

7 Implementation for AES Boolean Circuit

Our implementation runs on a 64-bit Windows 10 Enterprise (Version 1703 - OS Build 15063.1563) with Intel(R) Core(TM) i7-7700 CPU at 3.60 GHz and 16 GB of RAM. The implementation was conducted by using Visual C++ on Visual Studio 2017 Community. For parallelized computing, we employed OpenMP with a maximal number of 8 threads.

For pseudorandom generators, we used Salsa20 stream cipher from an online source code [2] for generating randomness during execution of the protocol. For commitments, we used Win64 OpenSSL v1.1.1a Light for calling SHA-256 cryptographic hash function. The randomness of each commitment has 256 bits generated from Salsa20 stream cipher.

We consider AES boolean circuit with non-expanded key size, taken from website [3], for the experiment. The number of protocol repetitions is set as $\kappa = 69$ (corresponding to soundness error 2^{-40}) and $\kappa = 137$ (corresponding to soundness error 2^{-80}), with different numbers of threads (for parallelized computing) spanning from 1 to 8. For each case and each number of threads, we run the protocol 100 times to measure the average running time of the protocol. For 1-thread cases, we also measured the average committing time before communicating and the average verifying time for the verifier. Fig. 2 describes the average running time of our protocol with different values of κ and numbers of threads. Table 1 describes the average committing time and average verifying time in case of 1-thread computing of our protocol.

	Average Commitment Time	Average Verifying Time
kappa = 69	0.769 (s)	0.7382 (s)
kappa = 137	1.526 (s)	1.4738 (s)

Table 1: Average committing time and average verifying time for $\kappa = 69$ and $\kappa = 137$.

[2] https://github.com/everard/Salsa20
[3] https://homes.esat.kuleuven.be/ nsmart/MPC/

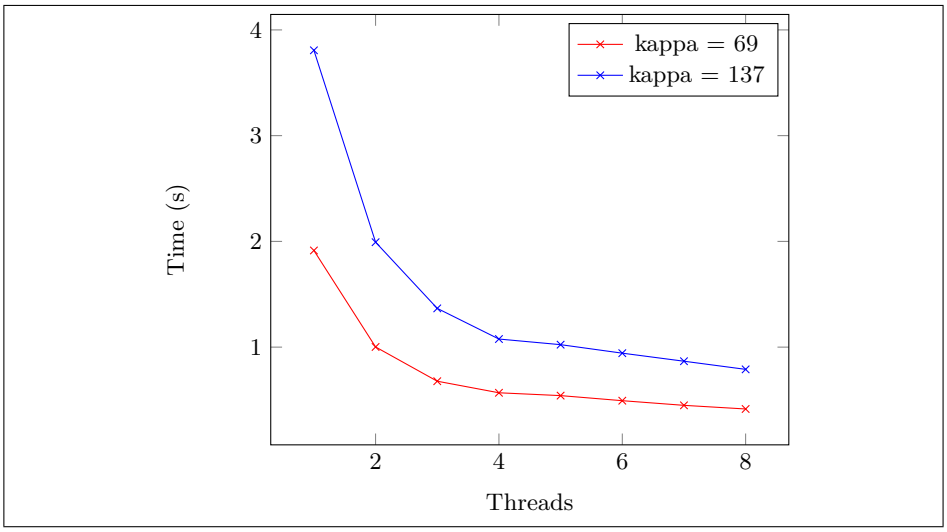

Fig. 2: Running time for AES boolean circuit.

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. S. Agrawal, C. Ganesh, and P. Mohassel. Non-interactive zero-knowledge proofs for composite statements. In *CRYPTO 2018*, volume 10993 of *LNCS*, pages 643–673. Springer, 2018.
3. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC 1996*, pages 99–108. ACM, 1996.
4. Q. Alamélou, O. Blazy, S. Cauchie, and P. Gaborit. A code-based group signature scheme. *Des. Codes Cryptography*, 82(1-2):469–493, 2017.
5. S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *CCS 2017*, pages 2087–2104. ACM, 2017.
6. 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.
7. 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.
8. D. Augot, M. Finiasz, and N. Sendrier. A fast provably secure cryptographic hash function. *IACR Cryptology ePrint Archive*, 2003:230, 2003.
9. 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.
10. 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.
11. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general as-

sumptions. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.
12. 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.
13. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *J. Cryptology*, 22(1):114–138, 2009.
14. 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.
15. 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.
16. 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.
17. D. Boneh, S. Eskandarian, and B. Fisch. Post-quantum group signatures from symmetric primitives. *IACR Cryptology ePrint Archive*, 2018:261, 2018.
18. J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth. Foundations of fully dynamic group signatures. In *ACNS 2016*, volume 9696 of *LNCS*, pages 117–136. Springer, 2016.
19. 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.
20. 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.
21. P. Branco and P. Mateus. A code-based linkable ring signature scheme. In *ProvSec 2018*, volume 11192 of *LNCS*, pages 203–219. Springer, 2018.
22. 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.
23. 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.
24. 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.
25. S. Canard and A. Gouget. Multiple denominations in e-cash with compact transaction data. In *FC 2010*, volume 6052 of *LNCS*, pages 82–97. Springer, 2010.
26. M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *CCS 2017*, pages 1825–1842. ACM, 2017.
27. M. Chase, C. Ganesh, and P. Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In *CRYPTO 2016*, volume 9816 of *LNCS*, pages 499–530. Springer, 2016.
28. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.
29. L. Dallot and D. Vergnaud. Provably secure code-based threshold ring signatures. In *IMACC 2009*, volume 5921 of *LNCS*, pages 222–235. Springer, 2009.

30. 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.
31. 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.
32. 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`.
33. 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.
34. 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.
35. C. Gentry and Z. Ramzan. Rsa accumulator based broadcast encryption. In *ISC 2004*, volume 3225 of *LNCS*, pages 73–86. Springer, 2004.
36. I. Giacomelli, J. Madsen, and C. Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *USENIX 2016*, pages 1069–1083. USENIX Association, 2016.
37. 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.
38. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
39. J. Groth. Evaluating security of voting schemes in the universal composability framework. In *ACNS 2004*, volume 3089 of *LNCS*, pages 46–60. Springer, 2004.
40. 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.
41. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30. ACM, 2007.
42. 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.
43. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. *IACR Cryptology ePrint Archive*, 2018:475, 2018.
44. 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.
45. J. Li, N. Li, and R. Xue. Universal accumulators with efficient nonmembership proofs. In *ACNS 2007*, volume 4521 of *LNCS*, pages 253–269. Springer, 2007.
46. 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.
47. 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.
48. 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.

49. 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.
50. Z. Lin and N. Hopper. Jack: Scalable accumulator-based nymble system. In *WPES 2010*, pages 53–62. ACM, 2010.
51. S. Ling, K. Nguyen, H. Wang, and Y. Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In *ACNS 2017*, volume 10355 of *LNCS*, pages 293–312. Springer, 2017.
52. V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. *J. Cryptology*, 31(3):774–797, 2018.
53. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, 1978.
54. 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.
55. 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.
56. R. C. Merkle. A certified digital signature. In *CRYPTO 1989*, volume 435 of *LNCS*, pages 218–238. Springer, 1989.
57. I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *IEEE S&P 2013*, pages 397–411. IEEE, 2013.
58. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC 1990*, pages 427–437. ACM, 1990.
59. L. Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292. Springer, 2005.
60. 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.
61. 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.
62. C. Papamanthou, R. Tamassia, and N. Triandopoulos. Authenticated hash tables. In *ACM-CCS 2008*, pages 437–448. ACM, 2008.
63. 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.
64. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS 1999*, pages 543–553, 1999.
65. J. Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
66. G. Tsudik and S. Xu. Accumulating composites and improved group signing. In *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 269–286. Springer, 2003.
67. 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 The Interactive Protocol

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

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

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

B Proof of Theorem 1

Proof. Completeness: If an honest prover follows the protocol, then he always gets accepted by the verifier. 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 RSP $= \big(\phi, \mathbf{w}' \oplus \mathbf{r}_w, \rho_1, \rho_3\big)$.
- If $Ch = 3$: Send 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 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 $C'_2 = \mathsf{COM}(\Gamma_\phi(\mathbf{r}_w); \rho_2)$, $C'_3 = \mathsf{COM}(\Gamma_\phi(\mathbf{w}' \oplus \mathbf{r}_w); \rho_3)$ 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 $\text{RSP}_1 = (\mathbf{t}_w, \mathbf{t}_r, \rho_2, \rho_3)$, $\text{RSP}_2 = (\phi_2, \mathbf{w}_2, \rho_1, \rho_3)$, $\text{RSP}_3 = (\phi_3, \mathbf{w}_3, \rho_1, \rho_2)$ 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

$$\begin{cases} \mathbf{t}_w \in \mathsf{VALID};\ \phi_2 = \phi_3;\ \mathbf{t}_r = \mathit{\Gamma}_{\phi_3}(\mathbf{w}_3);\ \mathbf{t}_w \oplus \mathbf{t}_r = \mathit{\Gamma}_{\phi_2}(\mathbf{w}_2); \\ \mathbf{M} \cdot \mathbf{w}_2 \oplus \mathbf{v} = \mathbf{M} \cdot \mathbf{w}_3. \end{cases} \tag{23}$$

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 (e.g., our protocol for circuits in Section 5), 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 $\mathbf{t}_w$.

C Example of A Circuit Evaluation

Consider the circuit evaluation $C(x_1, x_2, x_3) = u$, containing 3 binary gates $\mathsf{AND}, \mathsf{OR}, \mathsf{NAND}$ and 2 gates NOT, as follows:

$$x_4 = x_1\ \mathsf{AND}\ x_2, \quad x_5 = \overline{x}_1\ \mathsf{OR}\ x_3, \quad x_4\ \mathsf{NAND}\ \overline{x}_5 = u.$$

Let $x_1 = 0, x_2 = 1, x_3 = 1$. Then $x_4 = 0$, $x_5 = 1$, and $u = 1$. The prover forms vector $\mathbf{x}^\star = (x_1, x_2, x_3, x_4, x_5) = (01101)$ and computes

$$\begin{aligned} \mathbf{w}_C &= \big(\ \mathsf{enc}(x_4) \parallel \mathsf{enc}(x_5) \parallel \mathsf{ext}_{\mathsf{AND}}(x_1, x_2) \parallel \mathsf{ext}_{\mathsf{OR}}(\overline{x}_1, x_3) \parallel \mathsf{ext}_{\mathsf{NAND}}(x_4, \overline{x}_5)\ \big) \\ &= \big(\ 10 \parallel 01 \parallel 0100 \parallel 0111 \parallel 0111\ \big) \in \{0,1\}^{16}. \end{aligned}$$

In the interaction, suppose that the randomness sampled by the prover are $\mathbf{r}_w = (r_1, \ldots, r_{16}) = (\ 10\,01\,0111\,1100\,0010\ )$ and $\phi = \mathbf{e} = (10011)$. We have $\mathit{\Theta}_\phi(\mathbf{w}_C) = \big(F_1(10)\ \ F_1(01)\ \ T_{1,0}(0100)\ \ T_{1,0}(0111)\ \ T_{1,1}(0111)\big)$

He then computes $\mathbf{M}_C \cdot \mathbf{r}_w = (r_2 + r_8, r_4 + r_{12}, r_{16}) = (110)$, $\mathbf{w}_2 = \mathbf{w}_C + \mathbf{r}_w = \big(\ 00\ 00\ 0011\ 1011\ 0101\ \big)$,

$$\begin{aligned}
\Theta_\phi(\mathbf{w}_C) &= \big(F_1(10)\ \ F_1(01)\ \ T_{1,0}(0100)\ \ T_{1,0}(0111)\ \ T_{1,1}(0111)\big) = (01\ 10\ 0001\ 1101\ 1110)\\
\Theta_\phi(\mathbf{w}_2) &= \big(F_1(00)\ \ F_1(00)\ \ T_{1,0}(0011)\ \ T_{1,0}(1011)\ \ T_{1,1}(0101)\big) = (00\ 00\ 1100\ 1110\ 1010),\\
\Theta_\phi(\mathbf{r}) &= \big(F_1(10)\ \ F_1(01)\ \ T_{1,0}(0111)\ \ T_{1,0}(1100)\ \ T_{1,1}(0010)\big) = (01\ 10\ 1101\ 0011\ 0100),
\end{aligned}$$

and generates commitment C_1, C_2, C_3 as per the protocol description.

$$\begin{cases}
C_1 = \mathsf{COM}(\phi = (10011), \mathbf{M}_C \cdot \mathbf{r}_w = (110); \rho_1)\\
C_2 = \mathsf{COM}(\Theta_\phi(r) = (01\ 10\ 1101\ 0011\ 0100); \rho_2)\\
C_3 = \mathsf{COM}(\Theta_\phi(\mathbf{w}_C + \mathbf{r}_w) = \Theta_\phi(\mathbf{w}_2) = (00\ 00\ 1100\ 1110\ 1010); \rho_3).
\end{cases}$$

- If $Ch = 1$, prover sends ρ_2, ρ_3 and $\Theta_\phi(\mathbf{r})$ together with $\mathbf{z} = \mathbf{x}^\star + \mathbf{e} = (11110) = (z_1, z_2, z_3, z_4, z_5)$. Note that the latter allows the verifier to construct vector

$$\begin{aligned}
\mathsf{good}(\mathbf{z}) &= \big(\ \mathsf{enc}(z_4) \parallel \mathsf{enc}(z_5) \parallel \mathsf{ext}_{\mathsf{AND}}(z_1, z_2) \parallel \mathsf{ext}_{\mathsf{OR}}(\overline{z}_1, z_3) \parallel \mathsf{ext}_{\mathsf{NAND}}(z_4, \overline{z}_5)\ \big)\\
&= \big(\ 01 \parallel 10 \parallel 0001 \parallel 1101 \parallel 1110\ \big) \in \{0,1\}^{16} = \Theta_\phi(\mathbf{w}_C),
\end{aligned}$$

  and to check that C_2 and C_3 are well-formed.
- If $Ch = 2$, prover sends ρ_1, ρ_3, ϕ and $\mathbf{w}_2 = (00\ 00\ 0011\ 1011\ 0101)$. The verifier checks that C_3 is well-formed and ϕ together with $\mathbf{M}_C \cdot \mathbf{w}_2 - u = (111) - (001) = (110)$ were committed in C_1.
- If $Ch = 3$, prover sends ρ_1, ρ_2, ϕ, $\mathbf{r}_w$, allowing verifier to check the well-formedness of C_1 and C_2.

D Definition and Security Requirement of Ring Signature

Now we recall the standard definition and security requirements for ring signatures, as put forward in [13,40].

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 the distribution of a signature on a message M under a ring R signed using the key sk_{j_0} looks statistically close to the one of a signature on the same message under the same ring R signed using the key sk_{j_1}. 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

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

A ring signature scheme is said to be statistically anonymous if the distribution of a signature on a message M under a ring R signed using the key sk_{j_0} looks statistically close to the one of a signature on the same message under the same ring R signed using the key sk_{j_1}. The formal definition is given below.

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

E Definition and Security Requirement of Group Signature

In this section, we give the syntax and the security model of the static group signatures [11]. 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 4. 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 4. 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.

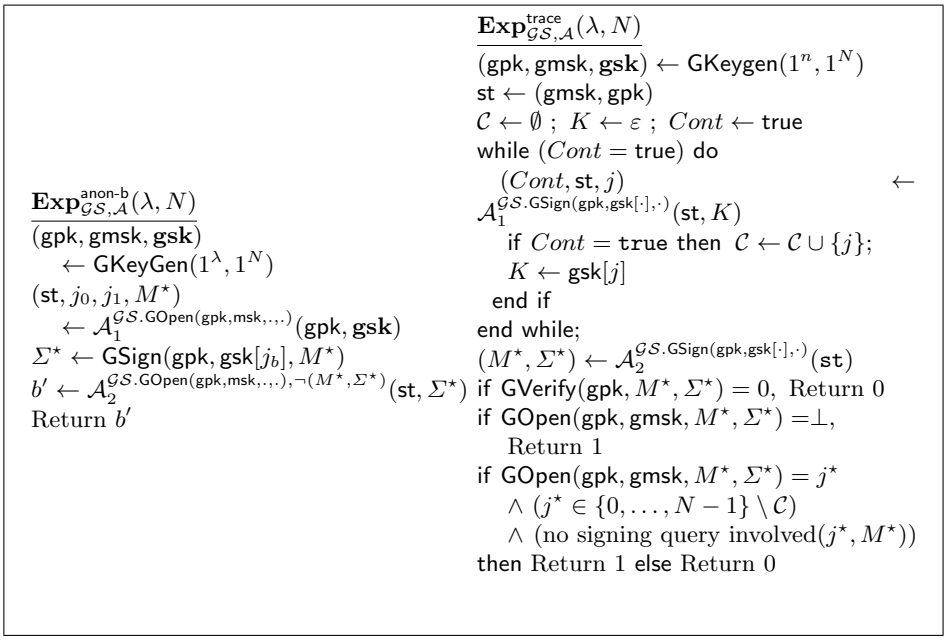

Fig. 4: Experiments for the definitions of anonymity and full traceability

F The Randomized McEliece Encryption Scheme

Now we recall a randomized variant of the McEliece [53] encryption scheme as suggested in [60], 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.

- $\mathsf{ME.KeyGen}(n_\mathsf{e}, k_\mathsf{e}, t_\mathsf{e})$: 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 $\mathbf{G} = \mathbf{P}\mathbf{G}'\mathbf{S} \in \mathbb{Z}_2^{n_\mathsf{e} \times k_\mathsf{e}}$. 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 [60,32], 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 $1/2 + \mathsf{negl}(\lambda)$.

G Code-Based Logarithmic Size Ring Signatures

In this section, we construct a code-based ring signature scheme [63] with logarithmic size in the number of ring members involved. We use the accumulator and its associated ZKAoK given in Section 6.3 and Section 6.4 as the building blocks. The definition of ring signature is put in Appendix D and the construction is demonstrated in Section G.1.

G.1 Our Logarithmic-Size Ring Signature Scheme

$\mathsf{RSetup}(1^\lambda)$: 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 Appendix G.3. The protocol is able to achieve negligible soundness error with $\kappa = \omega(\log \lambda)$ times repetition and 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 Appendix G.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.

G.2 Analysis of the Scheme

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

Theorem 4. *The ring signature scheme described in Section G.1 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.*

Unforgeability with respect to insider corruption. Assumes that the cardinality of each ring $R^\star$ is a power of 2 in the proof of unforgeability for simplicity. However, this restriction can be easily eliminated, as we will see later on.

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 5. *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.*

The above proof is given in the appendix.

Theorem 6. *The scheme provides statistical anonymity in the random oracle model.*

The proof of the above theorem relies on the statistical witness indistinguishability of the argument system of Theorem 1. The proof is straightforward and omitted.

G.3 Zero-Knowledge Argument Underlying the Ring Signature Scheme

In this section, we describe our statistical ZKAoK for the ring signature scheme that is invoked by the signer when producing signatures. This protocol is an extension of the one for the accumulator from Section 6.4. 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

$$\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{x}^{(1)}) = \mathbf{d}. \tag{24}$$

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 6.4, we only need to consider the new equation (24). 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 (24) is equivalent to the following:

$$\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{x}^{(1)}) \oplus \mathbf{I}_n^* \cdot \mathbf{x}_\ell = \mathbf{0}. \tag{25}$$

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

$$\begin{cases} \mathbf{B} \cdot \mathbf{y}_1 \oplus \mathbf{B} \cdot \mathbf{z}_1 = \mathbf{u} \\ \mathbf{B} \cdot \mathbf{y}_2 \oplus \mathbf{B} \cdot \mathbf{z}_2 \oplus \mathbf{I}_n^* \cdot \mathbf{x}_1 = \mathbf{0} \\ \qquad \cdots\cdots\cdots \\ \mathbf{B} \cdot \mathbf{y}_\ell \oplus \mathbf{B} \cdot \mathbf{z}_\ell \oplus \mathbf{I}_n^* \cdot \mathbf{x}_{\ell-1} = \mathbf{0} \\ \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{x}^{(1)}) \oplus \mathbf{I}_n^* \cdot \mathbf{x}_\ell = \mathbf{0}, \end{cases} \tag{26}$$

where

$$\begin{cases} \mathbf{y}_i = \mathsf{Ext}(j_i, \mathsf{RE}(\mathbf{v}_i)) \in \{0,1\}^m, i \in [\ell] \\ \mathbf{z}_i = \mathsf{Ext}(\bar{j}_i, \mathsf{RE}(\mathbf{w}_i)) \in \{0,1\}^m, i \in [\ell] \\ \mathbf{x}_i = \mathsf{Encode}(\mathbf{v}_i) \in \{0,1\}^{2n}, i \in [\ell]. \end{cases} \tag{27}$$

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

$$\mathbf{w}_R = (\mathbf{y}_1 \| \ldots \| \mathbf{y}_\ell \| \mathbf{z}_1 \| \ldots \| \mathbf{z}_\ell \| \mathbf{x}_1 \| \ldots \| \mathbf{x}_\ell \| \mathsf{RE}(\mathbf{x}^{(0)}) \| \mathsf{RE}(\mathbf{x}^{(1)})) \tag{28}$$

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 (7) hold. Let VALID_R contain all vectors of the form

$$\mathbf{w}'_R = \big(\mathbf{y}'_1 \| \ldots \| \mathbf{y}'_\ell \| \mathbf{z}'_1 \| \ldots \| \mathbf{z}'_\ell \| \mathbf{x}'_1 \| \ldots \| \mathbf{x}'_\ell \| \mathbf{u}'_0 \| \mathbf{u}'_1\big) \in \{0,1\}^{L_R} \tag{29}$$

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:

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

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

$$\mathbf{w}^*_R = \big(\mathbf{y}^*_1 \| \ldots \| \mathbf{y}^*_\ell \| \mathbf{z}^*_1 \| \ldots \| \mathbf{z}^*_\ell \| \mathbf{x}^*_1 \| \ldots \| \mathbf{x}^*_\ell \| \mathbf{u}^*_0 \| \mathbf{u}^*_1\big) \in \{0,1\}^{L_R} \quad (30)$$

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

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

Based on the equivalences observed in (17), (11) and (9), it can be checked that the conditions in (7) 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 3. The protocol has communication cost $\mathcal{O}(L_R) = \ell \cdot \mathcal{O}(m+n)$ bits.

H Code-Based Logarithmic Size Group Signatures

This section shows how to use the powerful supporting technique of our accumulator and ZK argument system to build our code-based group signature which has shorter signatures, more efficient compared to previous proposals and achieves CCA-anonymity. To this end, we apply the double encryption technique [58] to a variant of randomized McEliece encryption scheme [53] as suggested in [60], obtaining a CCA2-secure encryption scheme. The construction is demonstrated in Section H.1.

H.1 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 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.

   In Appendix H.3, 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 able to achieve negligible soundness error with $\kappa = \omega(\log \lambda)$ times repetition 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 Appendix H.3 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 $(j'_1, \ldots, j'_\ell)$.

H.2 Analysis of the Scheme

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

Theorem 7. *The group signature scheme described in Section H.1 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 8. *The scheme provides full traceability in the random oracle model if the* $\mathsf{2\text{-}RNSD}_{n,2n,c}$ *problem is hard.*

The proof of Theorem 9 is similar to [64] and is given in the appendix.

Theorem 9. *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.*

The proof described above is also given in the appendix.

H.3 The Associated Zero-Knowledge Argument

In this section, we present the zero-knowledge protocol that is exploited by signers when generating signatures in Section H.1. This protocol is extended from the one in Appendix G.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

$$\mathbf{c}_1 = \mathbf{G}_1 \cdot \begin{pmatrix} \mathbf{r}_1 \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_1, \quad \text{and} \quad \mathbf{c}_2 = \mathbf{G}_2 \cdot \begin{pmatrix} \mathbf{r}_2 \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_2, \tag{31}$$

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 $\mathrm{R}_{\mathrm{group}}$ is given below.

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

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 [65]. 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

$$\mathbf{e} \in \mathsf{B}(n_e, t_e) \Longleftrightarrow \sigma(\mathbf{e}) \in \mathsf{B}(n_e, t_e), \tag{32}$$

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 $\mathrm{R}_{\mathrm{group}}$. As the transformation for the ring signature layer has been demonstrated in Appendix G.3, we consider the newly appeared relations in (31).

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 (31) is equivalent to

$$\mathbf{c}_1 = \mathbf{G}_1^* \cdot \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{f} \end{pmatrix} \oplus \mathbf{e}_1, \quad \text{and} \quad \mathbf{c}_2 = \mathbf{G}_2^* \cdot \begin{pmatrix} \mathbf{h}_2 \\ \mathbf{f} \end{pmatrix} \oplus \mathbf{e}_2. \tag{33}$$

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 (33). 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 (28). 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 (7) 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 (29).
- $\mathbf{f}' = \mathsf{Encode}(\mathsf{bin}(j'))$ 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 Appendix G.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 Appendix G.3.

Based on the equivalences observed in (17), (11), (9) and (32), it can be checked that the conditions in (7) 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 3. 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.

I Proof of Theorem 5

Proof. We prove unforgeability w.r.t. insider corruption by contradiction. 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 $\Sigma^\star = \Pi^\star_{\mathsf{ring}} = (\{\mathsf{CMT}_i^\star\}_{i=1}^\kappa, \mathrm{CH}^\star, \{\mathsf{RSP}^\star\}_{i=1}^\kappa)$, where $\mathsf{CH}^\star = \mathcal{H}_{\mathsf{FS}}\big(M^\star, (\{\mathsf{CMT}_i^\star\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}, R^\star\big)$ and $\mathsf{RSP}^\star$ is a valid response w.r.t. $\mathsf{CMT}_i^\star$ 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. [22], 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 [22] 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

$$h_{\mathbf{A}}(\mathbf{x}^{\star(0)}, \mathbf{x}^{\star(1)}) = \mathbf{d}^\star, \tag{34}$$

$$\mathsf{TVerify}_{\mathbf{A}}\big(\mathbf{u}^\star, \mathbf{d}^\star, w^\star\big) = 1. \tag{35}$$

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 (35).

Case (ii) implies that, the extracted witnesses $(\mathbf{d}_{j^\star}, \mathbf{x}^\star)$ satisfy equation (34) 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

$$\mathbf{d}_{j^\star} = \mathbf{A}_0 \cdot \mathsf{RE}(\mathbf{x}_{j^\star}^{(0)}) \oplus \mathbf{A}_1 \cdot \mathsf{RE}(\mathbf{x}_{j^\star}^{(1)}). \tag{36}$$

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 (34) and equation (36), 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 $\text{2-}\mathsf{RNSD}_{n,2n,c}$ 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 $\text{2-}\mathsf{RNSD}_{n,2n,c}$ problem. □

J Proof of Theorem 8

Proof. We prove full traceability by contradiction. 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. [22], 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 [22] 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

$$h_{\mathbf{A}}(\mathbf{x}_0^\star, \mathbf{x}_1^\star) = \mathbf{d}^\star, \tag{37}$$

$$\mathsf{TVerify}_{\mathbf{A}}\left(\mathbf{u}^{\star}, \mathbf{d}^{\star}, w^{\star}\right) = 1. \tag{38}$$

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

$$\mathbf{d}_{j^{\star}} = \mathbf{A}_0 \cdot \mathsf{RE}(\mathbf{x}_{j^{\star}}^{(0)}) \oplus \mathbf{A}_1 \cdot \mathsf{RE}(\mathbf{x}_{j^{\star}}^{(1)}). \tag{39}$$

Since $j^{\star}$ is uncorrupted, $\mathbf{x}_{j^{\star}} \neq \mathbf{x}^{\star}$ with probability at least $1/2$. Combining equation (37) and equation (39), 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. □

K Proof of Theorem 9

Proof. We prove the theorem using a sequence of indistinguishable games. The first game is experiment $\mathbf{Exp}_{\mathcal{GS},\mathcal{A}}^{\mathsf{anon\text{-}0}}(\lambda, N)$ whereas the last game is experiment $\mathbf{Exp}_{\mathcal{GS},\mathcal{A}}^{\mathsf{anon\text{-}1}}(\lambda, N)$. 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 $\mathbf{Exp}_{\mathcal{GS},\mathcal{A}}^{\mathsf{anon\text{-}0}}(\lambda, N)$, 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 $\Sigma^{\star} = (\Pi_{\mathsf{group}}^{\star}, \mathbf{c}_1^{\star}, \mathbf{c}_2^{\star})$. Then we have $\Pr[W_0] = \Pr[\mathbf{Exp}_{\mathcal{GS},\mathcal{A}}^{\mathsf{anon\text{-}0}}(\lambda, N) = 1]$.

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 $\mathsf{sk}_{\mathsf{ME}}^{(2)}$. 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 $\mathsf{sk}_{\mathsf{ME}}^{(2)}$ instead of the first one $\mathsf{sk}_{\mathsf{ME}}^{(1)}$. 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 $|\Pr[W_2] - \Pr[W_1]| \leq \Pr[F_1] \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{sound}}(\lambda) \in \mathsf{negl}(\lambda)$.

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. □