Short Code-based One-out-of-Many Proofs and Applications

Xindong Liu[1,2] and Li-Ping Wang[1,2] (✉)

[1] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{liuxindong, wangliping}@iie.ac.cn

Abstract. In this work, we propose two novel succinct one-out-of-many proofs from coding theory, which can be seen as extensions of the Stern's framework and Veron's framework from proving knowledge of a preimage to proving knowledge of a preimage for one element in a set, respectively. The size of each proof is short and scales better with the size of the public set than the code-based accumulator in [34]. Based on our new constructions, we further present a logarithmic-size ring signature scheme and a logarithmic-size group signature scheme. Our schemes feature a short signature size, especially our group signature. To our best knowledge, it is the most compact code-based group signature scheme so far. At 128-bit security level, our group signature size is about 144 KB for a group with 2^{20} members while the group signature size of the previously most compact code-based group signature constructed by the above accumulator exceeds 3200 KB.

Keywords: code-based cryptography · one-out-of-many proofs · set-membership proofs · ring signatures · group signatures.

1 Introduction

Code-based cryptography is the study of cryptosystems based on error-correcting codes that originated from the pioneering work of McEliece [28]. It is able to resist quantum attacks and is widely regarded as an important research branch in post-quantum cryptography. In particular, NIST's recent call for post-quantum standardization has propelled advancements in this area.

Zero-knowledge proofs are a fundamental primitive in cryptography. In 1993, Stern proposed the first code-based zero-knowledge argument of knowledge (ZKA oK) based on the hardness of the syndrome decoding (SD) problem [38]. This proof enables one to demonstrate knowledge of a low-weight preimage for a syndrome. Later, Veron introduced a ZKAoK for the general syndrome decoding (GSD) problem, which is the "dual" problem of the SD problem [39]. Subsequently, several optimization schemes have been proposed within this framework [30,5,7,11,39], as well as applied to other hard problem settings [20,24,26].

Moreover, variants of the Stern protocol or the Veron protocol have been utilized to construct numerous advanced cryptographic schemes, such as proofs of valid opening for code-based commitment schemes [34], verifiable encryption [33], ring signatures [9,10,29], group signatures [1,8,17] and accumulators [3,34].

An important tool in zero-knowledge proofs is one-out-of-many proofs, which enable one to demonstrate knowledge of an opening to a commitment within a list of commitments. This concept is closely related to some primitives such as set-membership proofs and ring signatures. Groth and Kohlweiss initially provided an efficient one-out-of-many proof based on discrete logarithms and applied it in ring signatures [21]. Subsequently, Esgin et al. constructed a lattice-based one-out-of-many proof [16]. In 2022, Lyubashevsky et al. further constructed a more efficient lattice-based one-out-of-many proof and based on this proof, they proposed a logarithmic ring signature and a logarithmic group signature [27].

However, similar constructions in code-based cryptography are not practical. In 2015, Ezerman et al. proposed a construction for proving knowledge of a secret for one syndrome in a public set, and applied it to build a code-based group signature scheme [17]. The group signature size is linear to the number N of group members. While it has a short signature size with a small N, as N grows to 2^{20}, its signature size increases to 19 MB under the 128-bit security level. The similar construction can be found in [2]. In 2019, Nguyen et al. put forward a code-based Merkle-tree accumulator, which is a logarithmic-size set-membership proof [34]. Based on this building block, the authors constructed logarithmic-size ring signature and group signature schemes. In 2020, Beullers et al. developed a general group-action based ring signature framework [6]. Subsequently, Barenghi et al. and Chou et al. instantiated the group action using the code equivalence problem [4] and the matrix code equivalence problem [13], respectively, and proposed efficient ring signature schemes. However, no group signature scheme has been constructed based on these two problems so far.

1.1 Our contributions

Our main cryptographic results are summarized as follows (we describe them in more detail in our technique overview):

- We propose a novel framework of one-out-of-many proofs. Our construction can be seen as an extension of Stern's framework from proving knowledge of a preimage to proving knowledge of a preimage for one element in a public set. The main advantage of our framework is that the growth rate of the proof size relative to the public set size N is very low even compared to the code-based accumulator (Asiacrypt 2019). This is because the term related to N in the expression of the signature size of our schemes is simply determined by the path of a $\log N$-deep Merkle tree, that is $2\lambda \log N$, where λ is the security level, and so is independent of the parameters of the hard problem. The code-based accumulator achieves membership proof by proving knowledge of a hash chain that is linearly related to the parameters of the regular SD problem. This makes the coefficient of $\log N$ much larger than ours.

- To further reduce the proof size, we propose a more efficient GSD-based one-out-of-many proof based on Veron's framework and optimization techniques proposed in [30,23]. Then, we apply this one-out-of-many proof to the construction of a logarithm-size ring signature. Next, we compare our ring signature with other code-based ring signatures in Table 1.
- We construct a new set membership proof by transforming the framework of building one-out-of-many proofs into the framework of building set membership proofs. This transformation is similar to the ideas in [21]. Moreover, this set membership proof serves as a building block in our group signature.
- Based on our GSD-based one-out-of-many proof and the set-membership proof, we present a logarithm-size group signature, which is the most compact code-based group signature scheme to date. We also make the comparison with other code-based group signatures in Table 2. For convenience, we use the parameter sets for SD problem in [18], and the parameter sets for the McEliece encryption scheme in [12]. The above comparison indicates that the signature sizes of both our ring signature and group signature schemes are significantly shorter than previous schemes except the BBNPS in [4], whose security relies on the code equivalence problem.

Table 1. Comparison of ring signature size (KB) under the 128-bit security level.

	N			asymptotic sig.size	hardness assumption
	2^8	2^{12}	2^{20}		
ELLNW[17]	55	124	19273	$\mathcal{O}(N)$	SD
BGM [8]	134	205	19354	$\mathcal{O}(N)$	rank SD
NTWZ [34]	1189	1741	2847	$\mathcal{O}(\log N)$	regular SD
BBNPS[4]	16	20	28	$\mathcal{O}(\log N)$	code equivalence
Our work	55	65	83	$\mathcal{O}(\log N)$	GSD

Table 2. Comparison of group signature size (KB) under the 128-bit security level.

	N			asymptotic sig.size	Anonymity
	2^8	2^{12}	2^{20}		
ELLNW[17]	171	241	19391	$\mathcal{O}(N)$	CPA
BGM[8]	1322	1392	20542	$\mathcal{O}(N)$	CPA
NTWZ [34]	1570	2122	3228	$\mathcal{O}(\log N)$	CCA
Our work	116	126	144	$\mathcal{O}(\log N)$	CPA

1.2 Technical overview

SD-based one-out-of-many proofs. Our first new framework of one-out-of-many proof is an extension to Stern's framework. To construct our protocol, we

begin by making a modification to the original Stern protocol. Let $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ and $\mathbf{s} \in \mathbb{F}_2^{n-k}$ denote a matrix and a syndrome, respectively. To prove the possession of a small-weight vector $\mathbf{e} \in \mathbb{F}_2^n$ such that $\mathbf{He}^\top = \mathbf{s}^\top$, a prover samples a random vector $\mathbf{r} \in \mathbb{F}_2^n$ and a random permutation ϕ, and then sends commitments $c_1 = \mathrm{Com}(\phi, \mathbf{Hr}^\top + \mathbf{s}^\top; \rho_1)$, $c_2 = \mathrm{Com}(\phi(\mathbf{r}); \rho_2)$ and $c_3 = \mathrm{Com}(\phi(\mathbf{r} + \mathbf{e}); \rho_3)$ to a verifier. If the challenge ch is 1, the prover opens c_2 and c_3. If $ch = 2$, it opens c_1 and c_3. If $ch = 3$, it opens c_1 and c_2. The modified Stern protocol is depicted in Fig. 1. The only difference between our protocol in Fig. 1 and the original Stern protocol is that in the former $c_1 = \mathrm{Com}(\phi, \mathbf{Hr}^\top + \mathbf{s}^\top; \rho_1)$ while in the latter $c_1 = \mathrm{Com}(\phi, \mathbf{Hr}^\top; \rho_1)$. Clearly, this modification does not affect the completeness, soundness, and zero-knowledge property of the modified protocol.

Observe two key facts: (1) The modified protocol's c_1 is related to the public key $\mathbf{s}$, while c_2 and c_3 are not related to $\mathbf{s}$. (2) During the verification phase, only when $ch = 3$, the verifier needs to use the public key $\mathbf{s}$ to check the value of c_1. These two observations inspire us to construct a one-out-of-many proof based on our modified protocol.

For a statement composing of a matrix $\mathbf{H}$ and N syndromes $(\mathbf{s}_1, \cdots, \mathbf{s}_N)$, a prover claims that it knows the small-weight preimage $\mathbf{e}$ of one of the syndromes satisfying $\mathbf{He}^\top = \mathbf{s}_I^\top$ for some $I \in [N]$, $[N] := \{1, \cdots, N\}$. To demonstrate this, the prover begins by sampling a random mask vector $\mathbf{r}$, a random permutation ϕ and N random coins $\{b_i\}_{i=1}^N$ to simulate $c_1^i = \mathrm{Com}(\phi, \mathbf{He}^\top + \mathbf{s}_i^\top; b_i)$ for all $i \in [N]$. Subsequently, it samples two random coins ρ_2 and ρ_3 to generate $c_2 = \mathrm{Com}(\phi(\mathbf{r}); \rho_2)$ and $c_3 = \mathrm{Com}(\phi(\mathbf{r} + \mathbf{e}); \rho_3)$. The prover then permutes $(c_1^1, \cdots, c_1^N)$ in random order and sends them along with c_2 and c_3 to a verifier. If the verifier returns 1, the prover will open c_2 and c_3 by revealing $(\phi(\mathbf{r}), \phi(\mathbf{r} + \mathbf{e}), \rho_2, \rho_3)$. This does not leak any information about the witness $\mathbf{e}$ and the index I. If the verifier returns 2, the prover will open c_1^I and c_3 by revealing $(\phi, \mathbf{r} + \mathbf{e}, \rho_1, b_I, \rho_3)$. Since the verifier receives a random permutation of $(c_1^1, \cdots, c_1^N)$ and only verifies whether c_1^I is in it, this also does not leak any information about the index I and witness $\mathbf{e}$. If the verifier returns 3, the prover will open all $\{c_1^i\}_{i=1}^N$ and c_2 by outputting ϕ, $\mathbf{r}$ and all random coins $\{b_i\}_{i=1}^N$. Therefore, no information about the witness $\mathbf{e}$ and the index I is leaked.

In the above process, the prover sends $N + 2$ commitments $(\{c_1^i\}_{i=1}^N, c_2, c_3)$ during the commitment phase. In the response phase, when the received challenge is 3, the prover needs to open these $N + 2$ commitments by outputting $N + 2$ random coins. Therefore, the proof size grows linearly with N. To reduce the proof size, we use a seedtree to generate N random coins required for these N commitments $\{c_1^i\}_{i=1}^N$, and compress these commitments into a root using a Merkle tree. Specifically, the prover first samples a random seed, and then uses a pseudo-random number generator (PRNG) to iteratively generate N random coins required for the N commitments $\{c_1^i\}_{i=1}^N$. Subsequently, the prover arranges these N commitments in lexicographical order and compresses the sorted list into a root using a Merkle tree. This trick is inspired by [6]. Next, the prover sends $(\mathrm{root}, c_2, c_3)$ to the verifier. If the verifier returns 1, the prover's output remains unchanged. If the verifier returns 2, the prover opens c_I and c_3

Round 1: $\mathcal{P}((\mathbf{H}, \mathbf{s}), \mathbf{e})$ chooses the following random objects:

$$\mathbf{r} \xleftarrow{\$} \mathbb{F}_2^n, \phi \xleftarrow{\$} \mathcal{S}_n, \rho_1, \rho_2, \rho_3 \xleftarrow{\$} \{0,1\}^\lambda.$$

and sends the commitment CMT $:= (c_1, c_2, c_3)$ to $\mathcal{V}$, where

$$c_1 = \mathrm{Com}(\phi, \mathbf{H}\mathbf{r}^\top + \mathbf{s}^\top; \rho_1), c_2 = \mathrm{Com}(\phi(\mathbf{r}); \rho_2), c_3 = \mathrm{Com}(\phi(\mathbf{r} + \mathbf{e}); \rho_3).$$

Round 2: $\mathcal{V}$(CMT) sends a $ch \leftarrow \{1, 2, 3\}$ to $\mathcal{P}$.
Round 3: $\mathcal{P}((\mathbf{H}, \mathbf{s}), \mathbf{e}, ch)$:
- $ch = 1$: Set $\mathbf{v}_1 = \phi(\mathbf{r}), \mathbf{v}_2 = \phi(\mathbf{e})$ and send RSP $= (\mathbf{v}_1, \mathbf{v}_2, \rho_2, \rho_3)$ to $\mathcal{V}$.
- $ch = 2$: Set $\mathbf{u}_1 = \mathbf{r} + \mathbf{e}, \pi_1 = \phi$ and send RSP $= (\mathbf{u}_1, \pi_1, \rho_1, \rho_3)$ to $\mathcal{V}$.
- $ch = 3$: Set $\mathbf{u}_2 = \mathbf{r}, \pi_2 = \phi$ and send RSP $= (\mathbf{u}_2, \pi_2, \rho_1, \rho_2)$ to $\mathcal{V}$.

Verification: $\mathcal{V}$(CMT, ch, RSP):
- $ch = 1$: Check if $c_2 = \mathrm{Com}(\mathbf{v}_1; \rho_2), c_3 = \mathrm{Com}(\mathbf{v}_1 + \mathbf{v}_2; \rho_3)$ and $\mathbf{v}_2 \in B(n, t)$.
- $ch = 2$: Check if $c_1 = \mathrm{Com}(\pi_1, \mathbf{H}\mathbf{u}_1^\top; \rho_1)$ and $c_3 = \mathrm{Com}(\pi_1(\mathbf{u}_1); \rho_3)$.
- $ch = 3$: Check if $c_1 = \mathrm{Com}(\pi_2, \mathbf{H}\mathbf{u}_2^\top + \mathbf{s}^\top; \rho_1)$ and $c_2 = \mathrm{Com}(\pi_2(\mathbf{u}_2); \rho_2)$.

Fig. 1. The modified Stern protocol: $B(n, t)$ denotes the set of vectors $\mathbf{v} \in \mathbb{F}_2^n$ such that its Hamming weight $w(\mathbf{v}) = t$. $\mathcal{S}_n$ denotes the symmetric group of all permutations of n elements. Com denotes a commitment scheme with the binding and hiding property.

by outputting $(\phi, \mathbf{r} + \mathbf{e}, \rho_1, b_I, \rho_3)$ and the the path of c_I in the Merkle tree. So the length of the path is $\log N$. If the verifier returns 3, the verifier opens all $\{c_1^i\}_{i=1}^N$ and c_2 by outputting $(\phi, \mathbf{r})$ and seed.

GSD-based one-out-of-many proofs. In 1997, Veron pointed that the GSD-based protocol with a ternary challenge space has lower communication cost than the Stern protocol [39]. The GSD problem is to find two vectors $\mathbf{x} \in \mathbb{F}_2^k$ and $\mathbf{e} \in B(n, t)$ such that $\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e}$ given a matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ and a vector $\mathbf{y} \in \mathbb{F}_2^n$. Although the security proof of this construction [39] has been pointed out to have an issue [22], it has been fixed by Gaborit et al. [5]. The fixed protocol still maintains low communication cost. This improvement is due to the use of the GSD problem instead of the SD problem. In fact, the hardness of two problems is equivalent, while the only difference lies in the use of a generator matrix in the former and a parity-check matrix in the latter. Inspired by this, we construct a GSD-based one-out-of-many proof. Additionally, we use three techniques, namely small-weight vector compression functions [30], seedtrees and Merkle trees [23] to reduce the communication cost of multi-iteration protocols. Seedtrees are used to generate random objects required while Merkle trees are employed to compress commitments in multi-iteration protocols.

Our GSD-based one-out-of-many proof naturally lead to a ring signature scheme by the Fiat-Shamir transform [19]. Specifically, each user i has a public-private key pair $(\mathbf{y}_i, (\mathbf{x}_i, \mathbf{e}_i))$, where $\mathbf{y}_i = \mathbf{x}_i\mathbf{G} + \mathbf{e}_i$. The signature of user i for a message μ is a zero-knowledge proof using our GSD-based one-out-of-many proof for the pair $(\mathbf{x}_i, \mathbf{e}_i)$ satisfying the above equation, where μ is the input in the random oracle in the Fiat-Shamir heuristic.

Set-membership proofs. First we briefly introduce the code-based commitment scheme [32]. For a message $\mathbf{m} \in \{0,1\}^{k_2}$, one initially chooses two random vectors $\mathbf{v} \in \mathbb{F}_2^{k_1}$, $\mathbf{e} \in B(n,t)$, and obtains a McEliece-type commitment

$$\mathrm{Com}_{\mathrm{McE}}(\mathbf{m};(\mathbf{v},\mathbf{e})) = (\mathbf{v}||\mathbf{m})\mathbf{G} + \mathbf{e},$$

where $\mathbf{G} = \begin{pmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \end{pmatrix} \in \mathbb{F}_2^{k\times n}$ is randomly generated, $\mathbf{G}_1 \in \mathbb{F}_2^{k_1\times n}, \mathbf{G}_2 \in \mathbb{F}_2^{k_2\times n}, k = k_1 + k_2$. To open a commitment $\mathbf{c}$, one reveals $\mathbf{m}, \mathbf{v}, \mathbf{e}$ and a receiver verifies if $\mathbf{c} = (\mathbf{v}||\mathbf{m})\mathbf{G} + \mathbf{e}$.

In our set-membership proof, the public information includes a public set $\mathcal{I} = \{\alpha_1, \cdots, \alpha_N\}$ and a commitment $\mathbf{c}$ for some α_i. A prover's goal is to demonstrate that $\mathbf{c}$ is a commitment to an element in the set $\mathcal{I}$. To achieve this, the prover first generates $[\mathbf{c}_1 = \mathbf{c} + \mathrm{Com}_{\mathrm{McE}}(\alpha_1,(\mathbf{0},\mathbf{0})), \cdots, \mathbf{c}_N = \mathbf{c} + \mathrm{Com}_{\mathrm{McE}}(\alpha_N,(\mathbf{0},\mathbf{0}))]$, and then proves that one of $(\{\mathbf{c}_i\}_{i=1}^N)$ is a commitment to $\mathbf{0}$. This is equivalent to proving that a prover knows a certain $\mathbf{c}_i$ having the form of $\mathbf{v}\mathbf{G}_1 + \mathbf{e}$, where $\mathbf{e} \in B(n,t)$ and $i \in [N]$.

Group signatures. We use the enc-then-prove framework to construct our group signature scheme. This framework typically requires three cryptographic layers: a secure signature scheme, a semantically secure encryption scheme and a zero-knowledge protocol connecting the first two layers. Let's now explain the construction of the three components used in our group signature scheme.

Consider a group of size N, where each user is indexed by an integer $i \in [N]$. For each i, the public key of the i-th user is $(\mathbf{G}, \mathbf{y}_i = \mathbf{x}_i\mathbf{G} + \mathbf{e}_i) \in \mathbb{F}_2^{k\times n} \times \mathbb{F}_2^n$, and the private key is $(\mathbf{x}_i, \mathbf{e}_i) \in \mathbb{F}_2^k \times B(n,t)$.

The signature layer. The construction in this layer is similar to our ring signature. User i uses our GSD-based one-out-of-many proof to prove that it has a pair $(\mathbf{x}_i, \mathbf{e}_i) \in \mathbb{F}_2^k \times B(n,t)$ satisfying $\mathbf{x}_i\mathbf{G} + \mathbf{e}_i = \mathbf{y}_i$.

The encryption layer. To achieve the traceability of group signatures, User i encrypts its index i using the randomized McEliece encryption scheme [35] as follows:

$$\mathbf{ct} = \mathrm{Enc}_{\mathrm{McE}}(\mathrm{bin}(i),(\mathbf{z},\mathbf{s})) = (\mathbf{z}||\,\mathrm{bin}(i))\mathbf{G}_{\mathrm{McE}} + \mathbf{s},$$

where $\mathbf{G}_{\mathrm{McE}} \in \mathbb{F}_2^{\ell\times m}$ is the public key of McEliece encryption scheme, $\mathrm{bin}(i)$ is the binary representation of i with length ℓ_2, $\mathbf{z} \in \mathbb{F}_2^{\ell_1}$, $\ell = \ell_1 + \ell_2$ and $\mathbf{s} \in B(m,w)$. Then, User i needs to prove that the ciphertext $\mathbf{ct}$ is an encryption to an index in the set $\{1, \cdots, N\}$. This is similar to our set-membership proof. Specifically, User i first generates the set

$$[\mathbf{ct}_1 = \mathbf{ct} + \mathrm{Enc}_{\mathrm{McE}}(1,(\mathbf{0},\mathbf{0})), \cdots, \mathbf{ct}_N + \mathrm{Enc}_{\mathrm{McE}}(N,(\mathbf{0},\mathbf{0}))],$$

and proves that $\mathbf{ct}_i$ is the encryption of 0, which means that $\mathbf{ct}_i$ has the form of $(\mathbf{z}||0)\mathbf{G}_{\mathrm{McE}} + \mathbf{s}$, where $\mathbf{z} \in \mathbb{F}_2^{\ell_1}$ and $\mathbf{s} \in B(m,w)$.

The third layer. User i must prove that it encrypts its own index i honestly, which requires combining the former one-out-of-many proof with the latter set-memberships proof. The high-level idea is to pair the $N+2$ commitments from the former proof with the $N+2$ commitments from the latter proof respectively,

forming $N+2$ pairs sequentially from 1 to $N+2$. Then shuffle the last N commitment pairs related to $\{\mathbf{y}_i\}_{i=1}^{N}$ and $\{\mathbf{ct}_i\}_{i=1}^{N}$, and send them along with another two pairs to a verifier. The remaining steps are similar to our GSD-based one-out-of-many proof, with the difference being what needs to be revealed and verified is the commitment pair.

Finally, we obtain a logarithmic-size group signature through the Fiat-Shamir transform, and we also prove its correctness, traceability, and CPA-anonymity.

1.3 Roadmap

The rest of the article is organized as follows. Section 2 provides some required preliminaries for our study. In Section 3, we present our novel SD-based and GSD-based one-out-of-many protocols. Our logarithmic-size ring signature scheme and its security proof is provided in Section 4. In Section 5, we give our logarithmic-size group signature scheme. Finally, we choose our parameter sets for our ring signature scheme and group signature scheme in Section 6.

2 Preliminaries

2.1 Hard problems

For integers a, b and $a \leq b$, the notation $[a; b]$ denotes the set $\{a, \cdots, b\}$. If $a = 1$, then it simplifies to $[b]$. Bold lowercase and uppercase letters denote row vectors and matrices respectively. The transpose of a vector $\mathbf{x}$ is represented by $\mathbf{x}^\top$. Let $B(n, t)$ be a set of vectors $\mathbf{v} \in \mathbb{F}_2^n$ such that the Hamming weight $w(\mathbf{v}) = t$. For a set X, $x \xleftarrow{\$} X$ means that x is sampled from X randomly and $\mathbf{x} \xleftarrow{\$,\zeta} X$ denotes $\mathbf{x}$ is sampled from X using the seed ζ. Let $\mathcal{O}(\cdot)$ denote a random oracle and $\mathcal{A}^{\mathcal{O}(\cdot)}$ denote that an algorithm $\mathcal{A}$ has access to $\mathcal{O}(\cdot)$.

Problem 1 (SD Problem). On input a matrix $\mathbf{H} \xleftarrow{\$} \mathbb{F}_2^{(n-k)\times n}$ and a syndrome $\mathbf{s} \in \mathbb{F}_2^{n-k}$, the syndrome decoding problem $\mathrm{SD}(n, k, t)$ asks to find a vector $\mathbf{e} \in \mathbb{F}_2^n$ such that $\mathbf{s}^\top = \mathbf{H}\mathbf{e}^\top$ and $\mathbf{e} \xleftarrow{\$} B(n, t)$.

We only present search version of SD problem. In [31], a reduction from the search version to the decision version is provided. Its dual problem is as follows.

Problem 2 (GSD Problem). On input a matrix $\mathbf{G} \xleftarrow{\$} \mathbb{F}_2^{k\times n}$ and a vector $\mathbf{y} \in \mathbb{F}_2^n$, the general syndrome decoding problem $\mathrm{GSD}(n, k, t)$ asks to find a vector $\mathbf{x} \in \mathbb{F}_2^k$ and a vector $\mathbf{e} \in \mathbb{F}_2^n$ such that $\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e}$, $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^k$ and $\mathbf{e} \xleftarrow{\$} B(n, t)$.

The hardness of the GSD problem is equivalent to that of the SD problem [39].

Problem 3 (DOOM Problem). On input a matrix $\mathbf{H} \xleftarrow{\$} \mathbb{F}_2^{k\times n}$ and a set of N syndromes $\{\mathbf{s}_i\}_{i=1}^{N}$, the decoding one-out-of-many problem $\mathrm{DOOM}(n, k, t, N)$ asks to find a vector $\mathbf{e}_i \in \mathbb{F}_2^n$ for some $i \in [N]$ such that $\mathbf{s}_i^\top = \mathbf{H}\mathbf{e}_i^\top$ and $\mathbf{e}_i \xleftarrow{\$} B(n, t)$.

As [37] stated, a variant of information set decoding algorithms can be adapted to the DOOM(n, k, t, N) problem, resulting in a speedup factor of approximately $\sqrt{N}$. We give the dual version of the DOOM problem.

Problem 4 (GDOOM Problem). On input a matrix $\mathbf{G} \xleftarrow{\$} \mathbb{F}_2^{k\times n}$ and a set of N vectors $\{\mathbf{y}_i\}_{i=1}^N$, the general decoding one-out-of-many problem GDOOM(n, k, t, N) asks to find a vector $\mathbf{x}_i \in \mathbb{F}_2^k$ and a vector $\mathbf{e}_i \in \mathbb{F}_2^n$ for some $i \in [N]$ such that $\mathbf{y}_i = \mathbf{x}_i\mathbf{G} + \mathbf{e}_i, \mathbf{x}_i \xleftarrow{\$} \mathbb{F}_2^k$ and $\mathbf{e}_i \xleftarrow{\$} B(n, t)$.

2.2 Merkle trees

A Merkle tree is a binary tree that compresses a list of data into a value. It is constructed layer by layer from bottom to top using a collision-resistant hash function. Each node in each layer is a hash value of the concatenation of its associated child nodes. In [6] a special type of Merkle trees called index-hiding Merkle trees was introduced. This tree has the characteristic of sorting the leaf nodes in lexicographical order, rather than based on their indices. Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ denote a collision-resistant hash function, and $D = (d_0, \cdots, d_{2^\ell-1})$ denote a data list. A Merkle tree includes the following algorithms.

1. Mtree$(D) \to$ (root, tree): With a data list D as input, set $T_{\ell,j} = \mathcal{H}(d_j)$ for $j \in [0, 2^\ell - 1]$, and iteratively calculate

$$T_{i,j} = \mathcal{H}(T_{i+1,2j-1}, T_{i+1,2j}), i \in [0, \ell - 1], j \in [0, 2^i - 1].$$

   Output $T_{0,0}$ as the root.
2. IH-Mtree$(D) \to$ (root, tree): With a data list D as input, set $T_{\ell,j} = \mathcal{H}(d_j)$ for $j \in [0, 2^\ell - 1]$, and iteratively calculate

$$T_{i,j} = \mathcal{H}((T_{i+1,2j-1}, T_{i+1,2j})_{lex}), i \in [0, \ell - 1], j \in [0, 2^i - 1].$$

   Output $T_{0,0}$ as the root.
3. Gpath(tree, B) $\to$ path: With the structure of a tree and a subset B of D as input, output a list of intermediate nodes that cover all $D \setminus B$. Here, one says that a node set U covers a leaf set L if the union of the leaves in a subtree rooted at each node $u \in U$ is exactly the set L.
4. Rebuild(path, B) $\to$ root$'$: With a subset B and a path as input, output a rebuilt root$'$.

There are two important properties about Merkle trees: the binding property and the index-hiding property of the index-hiding Merkle trees. The binding property means that for any subset S that does not belong to the set D, finding a path such that Rebuild$(S,$ path$)$ = Mtree(D) is the same as discovering a collision in $\mathcal{H}$. The index-hiding property means that for any subset B belonging to the set D, the path of index-hiding Merkle trees will not reveal any information about the set B.

Lemma 1. *[6] For a Merkle tree generated by a data list D, there exists an algorithm $\mathcal{F}$ that uses the tree and a pair $(S, path)$ satisfying $S \notin D$ and $Rebuild(S, path) = root$ to generate a collision in $\mathcal{H}$.*

Lemma 2. *[6] Given an integer $N = 2^\ell$ and two distributions $\mathcal{X}_1$ and $\mathcal{X}_2$ over $\{0,1\}^*$, the distribution $\mathcal{L}_I$, for any $I \in [N]$, is defined as*

$$\mathcal{L}_I = \left[(a_I, \text{path}, \text{root}) \middle| \begin{array}{c} d_I \xleftarrow{\$} \mathcal{X}_1, \\ d_i \xleftarrow{\$} \mathcal{X}_2, \forall 1 \le i \le N, i \ne I, \\ (\text{tree}, \text{root}) \leftarrow \text{IH-Mtree}(D), \\ \text{path} \leftarrow \text{Gpath}(\text{tree}, I). \end{array} \right],$$

where $D = (d_1, \cdots, d_N)$. Then $\mathcal{L}_I = \mathcal{L}_J$ for $\forall I, J \in [N]$.

2.3 Seedtrees

A seedtree is also a completely balanced binary tree, but its construction differs from that of a Merkle tree. In this case, a sender starts by selecting a seed as the root of a tree. Then, a pseudo-random number generator is used to create intermediate nodes from top to bottom. By sending some intermediate nodes, the sender reveals specific leaf nodes without disclosing any information about the remaining leaves. Let $\text{PRNG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ represent a pseudo-random number generator. A seedtree includes the following algorithms.

1. $\text{Stree}(\text{root}, N) \to (l_0, \cdots, l_{2^\ell - 1})$: With a seed root and an integer N as input, set $T_{0,0} = \text{root}$ and iteratively compute
$$(T_{i+1,2j-1}, T_{i+1,2j}) = \text{PRNG}(T_{i,j}), i \in [0, \ell - 1], j \in [0, 2^i - 1].$$
Define $(T_{\ell,0}, \cdots, T_{\ell,2^\ell-1})$ as leaf nodes $(l_0, \cdots, l_{2^\ell-1})$ and then output them.
2. $\text{Oseeds}(\text{root}, \mathbf{ch}) \to \text{seed}_{\text{inter}}$: With a root and a challenge $\mathbf{ch}$ as input, return a set $\text{seed}_{\text{inter}}$ that covers all the leaves with index i such that $ch_i = 1$.
3. $\text{Recover}(\text{seed}_{\text{inter}}, \mathbf{ch}) \to \{l_i\}_{i,s.t.ch_i=1}$: With a set $\text{seed}_{\text{inter}}$ and a challenge $\mathbf{ch}$ as input, return all leaf nodes rooted at the nodes in $\text{seed}_{\text{inter}}$.
4. $\text{Simseeds}(\mathbf{ch}) \to \text{seed}_{\text{inter}}$: With a challenge $\mathbf{ch}$ as input, return a set $\text{seed}_{\text{inter}}$ via random sampling, enabling $\text{seed}_{\text{inter}}$ to cover all leaves with index i satisfying $ch_i = 0$.

Lemma 3. *[6] Given an integer N and a challenge $\mathbf{ch}$, the distributions $\mathcal{X}_1$ and $\mathcal{X}_2$ are defined by*

$$\mathcal{X}_1 = \left[\text{seed}_{\text{inter}}, \{\text{leaf}_i\}_{ch_i=0} \middle| \begin{array}{c} \text{seed} \leftarrow \{0,1\}^\lambda, \\ \{\text{leaf}_i\}_{i=1}^N \leftarrow \text{Stree}(\text{root}, N), \\ \text{seed}_{\text{inter}} \leftarrow \text{Oseeds}(\text{seed}_{\text{root}}, \mathbf{ch}). \end{array} \right] \text{ and}$$

$$\mathcal{X}_2 = \left[\text{seed}_{\text{inter}}, \{\text{leaf}_i\}_{ch_i=0} \middle| \begin{array}{c} \{\text{leaf}_i\}_{ch_i=0} \leftarrow \{0,1\}^\lambda, \\ \text{seed}_{\text{inter}} \leftarrow \text{Simseeds}(\mathbf{ch}). \end{array} \right].$$

For any adversary who queries an oracle $\mathcal{Q}$ times, the advantage of distinguishing the two distributions $\mathcal{X}_1$ and $\mathcal{X}_2$ is at most $\mathcal{Q}/2^\lambda$.

3 Short one-out-of-many proofs from coding theory

We first propose an SD-based one-out-of-many proof along with the security proof. To achieve a lower communication cost, we introduce a GSD-based one-out-of-many proof, in which we decrease the communication cost by employing optimization techniques [30,23].

3.1 The SD-based one-out-of-many proof

In this subsection, we put forward our SD-based one-out-of-many proof in Fig. 2 for proving knowledge of the preimage for one syndrome in a public set $(\mathbf{s}_1, \cdots, \mathbf{s}_N)$. More specifically, the proof is a ZKAoK for the following relation:

$$R = \{(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N), (\mathbf{e}_I, I) | \text{ for some } I \in [N], \mathbf{s}_I^\top = \mathbf{H}\mathbf{e}_I^\top, \mathbf{e}_I \in B(n,t)\}, \quad (1)$$

where $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ and $\mathbf{s}_i \in \mathbb{F}_2^{n-k}$ for all $i \in [N]$.

Our idea stems from a key observation that in the modified Stern protocol in Fig. 1, the commitment $c_1 = \mathrm{Com}(\delta, \mathbf{H}\mathbf{r}^\top + \mathbf{s}^\top; \rho_1)$ needs to be verified in two different ways, in which one is related to the syndrome $\mathbf{s}$ while the other is unrelated to $\mathbf{s}$. Based on this, we replace c_1 with the following set

$$(\mathrm{Com}(\delta, \mathbf{H}\mathbf{r}^\top + \mathbf{s}_1^\top; b_1), \cdots, \mathrm{Com}(\delta, \mathbf{H}\mathbf{r}^\top + \mathbf{s}_N^\top; b_N)),$$

while keeping c_2 and c_3 unchanged, and compress it into a root using an index-hiding Merkle tree. Then, the root, c_2 and c_3 are sent to a verifier.

When the challenge is 1, the verifier checks c_2 and c_3 as the original Stern protocol. When the challenge is 2, the verifier checks the root and c_3 by using the path of the index-hiding Merkle tree, $\mathbf{r} + \mathbf{e}_I$ and δ. When the challenge is 3, the verifier calculates all leaf nodes using $\mathbf{r}$ and checks the root and c_2 by leaf nodes and δ.

In the following we first show that our protocol in Fig. 2 satisfies perfect completeness. That is, if $\mathcal{P}$, which possesses the witness $(\mathbf{e}_I, I)$, faithfully executes the protocol, $\mathcal{V}$ will output "accept" with a probability of 1. If $ch = 1$, $\mathcal{V}$ only needs to repeat the calculation process of c_2 and c_3, and so it always outputs "accept". If $ch = 2$, $\mathcal{V}$ needs to reconstruct the root using the I-th leaf node and repeat the calculation process of c_3. The reconstructed root is the same as the one constructed using all leaves, and hence it always outputs "accept". If $ch = 3$, $\mathcal{V}$ only needs to repeat the calculation process of the root and c_2, and also it always outputs "accept".

Next, the following Theorems 1 and 2 state that our protocol in Fig. 2 is sound and zero-knowledge.

Theorem 1. *Assuming that Com is a computational binding commitment scheme and the hash function $\mathcal{H}$ used in the Merkle tree is collision-resistant, the protocol in Fig. 2 is an argument of knowledge with soundness error 2/3.*

Proof. Assuming there is an adversary $\mathcal{A}$ who is accepted with a probability greater than 2/3, i.e., $\mathcal{A}$ can effectively respond all challenges. Then, we can construct an extractor $\mathcal{E}$ which either breaks the binding property of Com, or outputs a collision in $\mathcal{H}$, or $\mathbf{e} \in B(n,t)$ such that $\mathbf{H}\mathbf{e}^\top = \mathbf{s}_I^\top$ for a certain $I \in [N]$. Formally, given a CMT(c_1, c_2, c_3) and its three valid responses

$$\mathrm{RSP}_1 = (\mathbf{w}_1, \mathbf{w}_2, \rho_2, \rho_3), \mathrm{RSP}_2 = (\mathbf{w}_3, \xi_\delta, b, \mathrm{path}, \rho_3), \mathrm{RSP}_3 = (\xi_s, \xi_r, \xi'_\delta, \rho_2),$$

each of them corresponds to distinct challenges $ch = 1$, $ch = 2$ and $ch = 3$ respectively. The extractor $\mathcal{E}$ first calculates

$$\begin{cases} \pi_1 \xleftarrow{\$,\xi_\delta} \mathcal{S}_n, \mathbf{p}_1 \xleftarrow{\$,\xi_r} \mathbb{F}_2^n, \pi_2 \xleftarrow{\$,\xi'_\delta} \mathcal{S}_n; \\ (b'_1, \cdots, b'_N) = \mathrm{Stree}(\xi_s, N); \\ \mathrm{leaf}' = (\mathrm{Com}(\pi_2, \mathbf{H}\mathbf{p}_1^\top + \mathbf{s}_1^\top; b'_1), \cdots, \mathrm{Com}(\pi_2, \mathbf{H}\mathbf{p}_1^\top + \mathbf{s}_N^\top; b'_N)); \\ \mathrm{R}'_1 = \text{IH-Mtree}(\mathrm{leaf}'); \\ \mathrm{R}'_2 = \mathrm{Rebuild}(\mathrm{path}, \mathrm{Com}(\pi_1, \mathbf{H}\mathbf{w}_3^\top; b)). \end{cases}$$

Due to the validity of these responses, we have

$$\begin{cases} c_1 = \mathrm{R}'_1 = \mathrm{R}'_2, \mathbf{w}_2 \in B(n,t); \\ c_2 = \mathrm{Com}(\mathbf{w}_1; \rho_2) = \mathrm{Com}(\pi_2(\mathbf{p}_1); \rho_2); \\ c_3 = \mathrm{Com}(\mathbf{w}_1 + \mathbf{w}_2; \rho_3) = \mathrm{Com}(\pi_1(\mathbf{w}_3); \rho_3). \end{cases}$$

Then, $\mathcal{E}$ checks if $(\mathbf{w}_1, \rho_2) \neq (\pi_2(\mathbf{p}_1), \rho_2)$. If so, $\mathcal{E}$ breaks the binding property of Com. Similarly, $\mathcal{E}$ checks if $(\mathbf{w}_1 + \mathbf{w}_2, \rho_3) \neq (\pi_1(\mathbf{w}_3), \rho_3)$. If neither of these two inequalities holds true, we have

$$\mathbf{w}_1 = \pi_2(\mathbf{p}_1), \mathbf{w}_1 + \mathbf{w}_2 = \pi_1(\mathbf{w}_3). \tag{2}$$

Next, $\mathcal{E}$ checks if $\mathrm{Com}(\pi_1, \mathbf{H}\mathbf{w}_3^\top; b) \neq \mathrm{leaf}'_i$ for all $i \in [N]$. If so, it finds a collision in $\mathcal{H}$ by employing the Merkle tree extractor in Lemma 1 with the input $(\mathrm{tree}, \mathrm{Com}(\pi_1, \mathbf{H}\mathbf{w}_3^\top; b), \mathrm{path})$. Otherwise, there must exist an index $I \in [N]$ satisfying $\mathrm{Com}(\pi_1, \mathbf{H}\mathbf{w}_3^\top; b) = \mathrm{leaf}'_I$. Furthermore, $\mathcal{E}$ checks if $(\pi_1, \mathbf{H}\mathbf{w}_3^\top, b) \neq (\pi_2, \mathbf{H}\mathbf{p}_1^\top + \mathbf{s}_I^\top, b_I)$. If so, $\mathcal{E}$ breaks the binding property of Com. Otherwise, $\mathcal{E}$ gets $\pi_1 = \pi_2$, $\mathbf{H}\mathbf{w}_3^\top = \mathbf{H}\mathbf{p}_1^\top + \mathbf{s}_I^\top$ and $b = b_I$. From this and Equation (2), we deduce that $\mathbf{H}\pi_1^{-1}(\mathbf{w}_2^\top) = \mathbf{s}_I^\top$, where $\pi_1^{-1}(\mathbf{w}_2) \in B(n,t)$. This means that $\mathcal{E}$ outputs the witness $\mathbf{e} = \pi_1^{-1}(\mathbf{w}_2) \in B(n,t)$ such that $\mathbf{H}\mathbf{e} = \mathbf{s}_I$, $I \in [N]$. □

Theorem 2. *The protocol in Fig. 2 is honest-verifier zero-knowledge, that is, there exists a simulator Sim, such that for any statement-witness pair (s, w) belonging to the relation (1), any $ch \in \{1, 2, 3\}$ and any adversary $\mathcal{A}$ that accesses the oracle $\mathcal{Q}$ times, the following holds*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(\mathcal{P}(s, w, ch)) \to 1] - \Pr[\mathcal{A}^{\mathcal{O}}(Sim(s, ch)) \to 1]\right| \leq \frac{2\mathcal{Q}}{2^\lambda}.$$

Round 1: $\mathcal{P}_1^h((\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N), (\mathbf{e}_I, I))$:

$$\{\zeta_s, \zeta_r, \zeta_\delta, \rho_1, \rho_2\} \xleftarrow{\$} \{0,1\}^\lambda, \mathbf{r} \xleftarrow{\$,\zeta_r} \mathbb{F}_2^n, \delta \xleftarrow{\$,\zeta_\delta} \mathcal{S}_n.$$

- $(b_1, \cdots, b_N) = \text{Stree}(\zeta_s, N)$.
- leaf $= (\text{Com}(\delta, \mathbf{H}\mathbf{r}^\top + \mathbf{s}_1^\top; b_1), \cdots, \text{Com}(\delta, \mathbf{H}\mathbf{r}^\top + \mathbf{s}_N^\top; b_N))$.
- $(\text{root}, \text{tree}) = \text{IH-Mtree(leaf)}$.
- Send the commitment $\text{CMT} := (c_1, c_2, c_3)$ to $\mathcal{V}$, where

$$c_1 = \text{root}, c_2 = \text{Com}(\delta(\mathbf{r}); \rho_2), c_3 = \text{Com}(\delta(\mathbf{r} + \mathbf{e}_I); \rho_3).$$

Round 2: $\mathcal{V}_1^h(\text{CMT})$ sends a $ch \xleftarrow{\$} \{1,2,3\}$ to $\mathcal{P}$.

Round 3: $\mathcal{P}_2^h((\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N), (\mathbf{e}_I, I), ch)$:

- Case $ch = 1$: Set $\mathbf{w}_1 = \delta(\mathbf{r}), \mathbf{w}_2 = \delta(\mathbf{e}_I)$.
  Send $\text{RSP} = (\mathbf{w}_1, \mathbf{w}_2, \rho_2, \rho_3)$ to $\mathcal{V}$.
- Case $ch = 2$: Set $\mathbf{w}_3 = \mathbf{r} + \mathbf{e}_I, \xi_\delta = \zeta_\delta, b = b_I, \text{path} = \text{Gpath}(\text{tree}, I)$.
  Send $\text{RSP} = (\mathbf{w}_3, \xi_\delta, b, \text{path}, \rho_3)$ to $\mathcal{V}$.
- Case $ch = 3$: Set $\xi_s = \zeta_s, \xi_r = \zeta_r, \xi'_\delta = \zeta_\delta$.
  Send $\text{RSP} = (\xi_s, \xi_r, \xi'_\delta, \rho_2)$ to $\mathcal{V}$.

Verification: $\mathcal{V}_2^h((\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N), \text{CMT}, ch, \text{RSP})$:

- Case $ch = 1$: Check if $\mathbf{w}_2 \in B(n,t)$ and

$$c_2 = \text{Com}(\mathbf{w}_1; \rho_2), c_3 = \text{Com}(\mathbf{w}_1 + \mathbf{w}_2; \rho_3).$$

- Case $ch = 2$: Set $\pi_1 \xleftarrow{\$,\xi_\delta} \mathcal{S}_n$, $\text{R}_1 = \text{Rebuild}(\text{path}, \text{Com}(\pi_1, \mathbf{H}\mathbf{w}_3^\top; b))$, and check if

$$c_1 = \text{R}_1, c_3 = \text{Com}(\pi_1(\mathbf{w}_3); \rho_3).$$

- Case $ch = 3$: Set $(b_1, \cdots, b_N) = \text{Stree}(\xi_s, N)$, $\mathbf{p}_1 \xleftarrow{\$,\xi_r} \mathbb{F}_2^n$, $\pi_2 \xleftarrow{\$,\xi'_\delta} \mathcal{S}_n$ and leaf $= (\text{Com}(\pi_2, \mathbf{H}\mathbf{p}_1^\top + \mathbf{s}_1^\top; b_1), \cdots, \text{Com}(\pi_2, \mathbf{H}\mathbf{p}_1^\top + \mathbf{s}_N^\top; b_N))$.
  Compute R_2 as IH-Mtree(leaf) and check if

$$c_1 = \text{R}_2, c_2 = \text{Com}(\pi_2(\mathbf{p}_1); \rho_2).$$

$\mathcal{V}$ outputs 1 when all conditions are met, else it outputs 0.

Fig. 2. The SD-based one-out-of-many proof $\Pi^h = (\mathcal{P}^h = (\mathcal{P}_1^h, \mathcal{P}_2^h), \mathcal{V}^h = (\mathcal{V}_1^h, \mathcal{V}_2^h))$.

Proof. To simplify the proof, we use the random oracle $\mathcal{O}^*(\cdot)$ to instantiate the hash function, the algorithm Stree, and Com, where $*$ denotes the instantiated object. The simulator Sim is constructed as follows:

Case $ch = 1$:

1. Sim selects the following random objects:

$$\mathbf{w}_1 \xleftarrow{\$} \mathbb{F}_2^n, \mathbf{w}_2 \xleftarrow{\$} B(n,t), \{\rho_2, \rho_3\} \xleftarrow{\$} \{0,1\}^\lambda, c_1 \xleftarrow{\$} \{0,1\}^{2\lambda}.$$

2. Sim lets $\text{CMT} = (c'_1, c'_2, c'_3)$, where

$$c'_1 = c_1, c'_2 = \text{Com}(\mathbf{w}_1; \rho_2), c'_3 = \text{Com}(\mathbf{w_1} + \mathbf{w}_2; \rho_3).$$

3. Sim lets $\mathrm{RSP} = (\mathbf{w}_1, \mathbf{w}_2, \rho_2, \rho_3)$ and returns $(\mathrm{CMT}, 1, \mathrm{RSP})$.

Case $ch = 2$:

1. Sim selects the following random objects:

$$\mathbf{w}_3 \xleftarrow{\$} \mathbb{F}_2^n, \{b, \xi_\delta, \rho_3, \{\mathrm{leaf}_i\}_{i=2}^N\} \xleftarrow{\$} \{0,1\}^\lambda, c_2 \xleftarrow{\$} \{0,1\}^{2\lambda}, \pi_1 \xleftarrow{\$,\xi_\delta} \mathcal{S}_n.$$

2. Sim sets $\mathrm{leaf}_1 = \mathrm{Com}(\pi_1, \mathbf{H}\mathbf{w}_3^\top; b)$ and obtains $(\mathrm{tree}, \mathrm{root}) = \text{IH-Mtree}(\mathrm{leaf})$.
3. Sim lets $\mathrm{CMT} = (c_1', c_2', c_3')$, where

$$c_1' = \mathrm{root}, c_2' = c_2, c_3' = \mathrm{Com}(\pi_1(\mathbf{w}_3); \rho_3).$$

4. Sim runs the algorithm $\mathrm{GPath}(\mathrm{tree}, 1)$ to retrieve the path.
5. Sim lets $\mathrm{RSP} = (\mathbf{w}_3, \xi_\delta, b, \mathrm{path}, \rho_3)$ and returns $(\mathrm{CMT}, 2, \mathrm{RSP})$.

Case $ch = 3$:

1. Sim selects the following random objects:

$$\{\xi_s, \xi_r, \xi_\delta, \rho_2\} \xleftarrow{\$} \{0,1\}^\lambda, c_3 \xleftarrow{\$} \{0,1\}^{2\lambda}, \mathbf{v}_2 \xleftarrow{\$,\xi_r} \mathbb{F}_2^n, \pi_2 \xleftarrow{\$,\xi_\delta} \mathcal{S}_n.$$

2. Sim sets $\mathrm{leaf} = (\mathrm{Com}(\mathbf{H}\mathbf{p}_1^\top + \mathbf{s}_1^\top; b_1), \cdots, \mathrm{Com}(\mathbf{H}\mathbf{p}_1^\top + \mathbf{s}_N^\top; b_N))$ and calculates its root, where $(b_1, \cdots, b_N) = \mathrm{Stree}(\xi_s, N)$.
3. Sim lets $\mathrm{CMT} = (c_1', c_2', c_3')$, where

$$c_1' = \mathrm{root}, c_2' = \mathrm{Com}(\pi_2(\mathbf{v}_2); \rho_2), c_3' = c_3.$$

4. Sim lets $\mathrm{RSP} = (\xi_s, \xi_r, \xi_\delta, \rho_2)$ and returns $(\mathrm{CMT}, 3, \mathrm{RSP})$.

If Com is statistically hiding, then two CMTs generated by Sim and a honest prover, respectively, are statistically indistinguishable. Therefore, we only need to check the case of RSP.

$ch = 1$: Since Sim draws $(\mathbf{w}_1, \mathbf{w}_2)$ at random from $\mathbb{F}_2^n \times B(n,t)$, both RSPs generated by a honest prover and Sim respectively, follow the random distribution on $\mathbb{F}_2^n \times B(n,t)$.

$ch = 2$: Set five simulators $\{Sim_i\}_{i=1}^5$ to prove that Sim and a honest prover are indistinguishable. Sim_1 and Sim_5 represent a honest prover and Sim, respectively. Let E_i denote $\mathcal{A}^{\mathcal{O}}(Sim_i(s, ch)) = 1$, for $i \in [5]$.

Sim_2: The only difference between Sim_2 and Sim_1 is that $\{b_i\}_{i=1}^N$ are randomly sampled from $\{0,1\}^\lambda$ instead of being generated by the $\mathcal{O}^{\mathrm{Stree}}(\cdot)$ algorithm Stree with an input ζ_s. If ζ_s has not been queried by $\mathcal{A}$ to the oracle $\mathcal{O}^{\mathrm{Stree}}(\cdot)$, Sim_2 and Sim_1 are indistinguishable. Thus, if $\mathcal{O}^{\mathrm{Stree}}(\cdot)$ is accessed $\mathcal{Q}$ times and the probability of colliding with ζ_s is 2^λ for each query, the probability of colliding with ζ_s after $\mathcal{Q}$ queries is $\mathcal{Q}/2^\lambda$. Thus, we have $|\Pr[E_2] - \Pr[E_1]| \leq \mathcal{Q}/2^\lambda$.

Sim_3: The only difference between Sim_3 and Sim_2 is that $\{\mathrm{leaf}_i\}_{i=1, i\neq I}^N$ are randomly sampled from $\{0,1\}^{2\lambda}$ instead of being generated by $\mathcal{O}^{\mathrm{Com}}(\cdot)$ with the tuple $(\pi_1, \mathbf{H}\mathbf{r}^\top + \mathbf{s}_i^\top; b_i)$ for $i \neq I$. If all tuples have not been queried by $\mathcal{A}$ to the oracle $\mathcal{O}^{\mathrm{Com}}(\cdot)$, Sim_3 and Sim_2 are indistinguishable. We use $\mathcal{Q}_{\mathrm{com}_i}$ to represent the number of times $\mathcal{O}^{\mathrm{Com}_i}(\cdot)$ is accessed. Since the minimum entropy of b_i is $1/2^\lambda$, the minimum entropy of tuple $(\pi_1, \mathbf{H}\mathbf{r}^\top + \mathbf{s}_i^\top; b_i)$ is at most $1/2^\lambda$.

Therefore, the probability of collision with the tuple in each query is at most $1/2^\lambda$. Furthermore, since $\sum_{i=1}^{N} \mathcal{Q}_{\text{leaf}_i} \leq \mathcal{Q}$, we have $|\Pr[E_3] - \Pr[E_2]| \leq Q/2^\lambda$.

Sim_4: There are two differences between Sim_4 and Sim_3. Firstly, c_3 is randomly sampled from $\{0,1\}^{2\lambda}$. Secondly, $\text{leaf}_I = \text{Com}(\pi_1, \mathbf{H}\mathbf{w}_3^\top; b_I)$, where $\mathbf{w}_3$ is randomly sampled from $\mathbb{F}_2^n$. Since $\mathbf{w}_3$ and c_3 follows the random distribution as the real transcript, we have $\Pr[E_4] = \Pr[E_3]$.

Sim_5: The only difference between Sim_5 and Sim_4 is that 1 is used instead of I in witness. Lemma 2 states that regardless of whether the selected index is 1 or I, the root and path follow the same distribution. Therefore, we have $\Pr[E_5] = \Pr[E_4]$.

Thus, we have $\left|\Pr[\mathcal{A}^{\mathcal{O}}(\mathcal{P}(s,w,2)) \to 1] - \Pr[\mathcal{A}^{\mathcal{O}}(Sim(s,2)) \to 1]\right| \leq \frac{2\mathcal{Q}}{2^\lambda}$.

$ch = 3$: Since a honest prover does not use the witness, Sim can perfectly simulate RSP.

In summary, we obtain the required result. □

3.2 The GSD-based one-out-of-many proof

We first propose a GSD-based one-out-of-many proof in one-iteration in Fig. 3. The protocol in Fig. 3 is for the following relation:

$$R = \{(\mathbf{G}, \{\mathbf{y}_i\}_{i=1}^N), (\mathbf{x}, \mathbf{e}, I) | \text{ for some } I \in [N], \mathbf{y}_I = \mathbf{x}\mathbf{G} + \mathbf{e}, \mathbf{x} \in \mathbb{F}_2^k, \mathbf{e} \in B(n,t)\}, \tag{3}$$

where $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ and $\mathbf{y}_i \in \mathbb{F}_2^n$ for all $i \in [N]$. Next, we present its multi-iteration version in Fig. 4 by using additional optimization techniques such as commitments compression [30,23], seedtrees [6,23] and small-weight vector compression functions [30]. In the following we first explain how to use them to obtain the multi-iteration version, which is called the GSD-based one-out-of-many proof.

Commitments compression: The soundness error of the GSD-based one-out-of-many proof in one-iteration in Fig. 3 is 2/3, and so this protocol needs to be repeated κ times to reduce the soundness error. Since three commitments $(c_1^j, c_2^j, c_3^j), j \in [\kappa]$ need to be output in each iteration, the total cost of commitments is $3\kappa|\,\text{Com}\,|$. To reduce the total cost of commitments, we optimize the above protocol by using three Merkle trees Tree_1, Tree_2 and Tree_3 to compress $(c_1^1, \cdots, c_1^\kappa)$, $(c_2^1, \cdots, c_2^\kappa)$ and $(c_3^1, \cdots, c_3^\kappa)$ as root_1, root_2 and root_3 respectively, where (c_1^j, c_2^j, c_3^j) denotes the commitment of the j-th iteration. The prover sends $\text{CMT} = \mathcal{H}(\text{root}_1, \text{root}_2, \text{root}_3)$. Since the verifier can reconstruct two out of the three commitments (c_1^j, c_2^j, c_3^j) in each iteration, the prover transmits those commitments that cannot be computed by the verifier through certain intermediate nodes of the tree. In summary, this will reduce the cost of commitments from $3\kappa|\,\text{Com}\,|$ to $|\mathcal{H}| + \frac{5\kappa|\,\text{Com}\,|}{6}$.

Seedtrees : In one-iteration protocol, a set of seeds needs to be generated for sampling random objects and so certain seeds are revealed based on the challenge. To reduce the communication cost of seeds, we use a set of master seeds along with a set of seedtrees to generate κ sets of seeds in κ iterations. During the response phase, we provide those revealed seeds by outputting certain intermediate nodes in the seedtree. If the probability of a seed being transmitted in

one iteration is $1/p$, then using a seedtree reduces the transmission cost of this seed by about $1/2p$.

Small-weight vector compression: Since the protocol in Fig. 3 may transmit a small weight vector, we employ a small vector compression function in multi-iteration version to reduce the cost of transmitting small weight vectors from n to approximately $n/2$.

By using the above optimization techniques, a multi-iteration version is proposed in Fig. 4. As mentioned in [6], to ensure a tighter security proof and avoid multi-target attacks [14], we use a "salt", a 2λ-bit prefix string, in these seedtrees to distinguish the random oracles used in the seedtrees of different iterations. The "salt" has a negligible impact on practice.

The following theorem provides the security of the protocol in Fig.4.

Theorem 3. *The protocol described in Fig. 4 is an argument of knowledge with the perfect completeness and honest-verifier zero-knowledge.*

Proof. **Completeness:** This protocol has perfect completeness, which is immediately obtained by the correctness of the seedtrees, Merkle trees, and index-hiding Merkle trees.

Soundness: Assume there is an adversary $\mathcal{A}$ who is accepted with a probability $> (2/3)^{\kappa}$, i.e., $\mathcal{A}$ is able to successfully answer all three challenges in some iteration j where $j \in [\kappa]$. We construct an extractor $\mathcal{E}$ which either breaks the binding property of Com, or outputs a collision in $\mathcal{H}$, or outputs $\mathbf{x} \in \mathbb{F}_2^k, \mathbf{e} \in B(n,t)$ such that $\mathbf{xG} + \mathbf{e} = \mathbf{y}_I$ for some $I \in [N]$. First, $\mathcal{E}$ obtains the seeds, random coins and the commitment of the j-th iteration by $\mathrm{RSP}_1, \mathrm{RSP}_2, \mathrm{RSP}_3$ and the algorithms of seedtrees and Merkle trees.

$$\{\xi_s^j, \xi_u^j, \xi_\delta^j, \xi_{\delta'}^j, \rho_1^j, \rho_2^j, c_1^j, c_2^j, c_3^j\} \longleftarrow (\mathrm{RSP}_1, \mathrm{RSP}_2, \mathrm{RSP}_3).$$

Then, $\mathcal{E}$ performs the following steps:

$$\begin{cases} (b_1^j, \cdots, b_N^j) = \mathrm{Stree}(\xi_s^j, N); \\ \mathbf{p}_1^j \xleftarrow{\$,\xi_{\delta'}^j} \mathbb{F}_2^n, \pi_1^j \xleftarrow{\$,\xi_{\delta'}^j} \mathcal{S}_n, \mathbf{p}_2^j \xleftarrow{\$,\xi_u^j} \mathbb{F}_2^{k_1}, \mathbf{p}_2^3 \xleftarrow{\$,\xi_\delta^j} \mathbb{F}_2^n, \pi_2^j \xleftarrow{\$,\xi_\delta^j} \mathcal{S}_n; \\ \mathrm{leaf}^j = (\mathrm{Com}(\pi_2^j(\mathbf{p}_2^j\mathbf{G}_1 + \mathbf{y}_1) + \mathbf{p}_3^j; b_1^j), \cdots, \mathrm{Com}(\pi_2^j(\mathbf{p}_2^j\mathbf{G}_1 + \mathbf{y}_N) + \mathbf{p}_3^j; b_N^j)); \\ \mathrm{R}_1^j = \text{IH-Mtree}(\mathrm{leaf}^j); \\ \mathrm{R}_2^j = \mathrm{Rebuild}(\mathrm{path}^j, \mathrm{Com}(\mathbf{w}_2^j + \mathbf{w}_3^j; b^j)). \end{cases}$$

Due to the validity of these responses, we have

$$\begin{cases} c_1^j = \mathrm{R}_1^j = \mathrm{R}_2^j, \mathbf{w}_3^j \in B(n,t); \\ c_2^j = \mathrm{Com}(\pi_1^j, \mathbf{p}_1^j; \rho_2^j) = \mathrm{Com}(\pi_2^j, \mathbf{p}_3^j; \rho_2^j); \\ c_3^j = \mathrm{Com}(\mathbf{w}_2^j, \rho_3^j) = \mathrm{Com}(\pi_1^j(\mathbf{w}_1^j\mathbf{G}_1) + \mathbf{p}_1^j; \rho_3^j). \end{cases}$$

Round 1: $\mathcal{P}_1^g((\mathbf{G}, \mathbf{y}_1, \cdots, \mathbf{y}_N), (I, \mathbf{x}, \mathbf{e}))$ chooses the following random objects:

$$\{\zeta_s, \zeta_u, \zeta_\delta, \rho_1, \rho_2\} \xleftarrow{\$} \{0,1\}^\lambda; \mathbf{u} \xleftarrow{\$,\zeta_u} \mathbb{F}_2^k, \mathbf{v} \xleftarrow{\$,\zeta_\delta} \mathbb{F}_2^n, \delta \xleftarrow{\$,\zeta_\delta} \mathcal{S}_n.$$

- $(b_1, \cdots, b_N) = \text{Stree}(\zeta_s, N)$.
- leaf $= (\text{Com}(\delta(\mathbf{uG} + \mathbf{y}_1) + \mathbf{v}; b_1), \cdots, \text{Com}(\delta(\mathbf{uG} + \mathbf{y}_N) + \mathbf{v}; b_N))$.
- $(\text{root}, \text{tree}) = \text{IH-Mtree}(\text{leaf})$.
- Send CMT $:= (c_1, c_2, c_3)$ to $\mathcal{V}$, where

$$c_1 = \text{root}, c_2 = \text{Com}(\delta, \mathbf{v}; \rho_2), c_3 = \text{Com}(\delta((\mathbf{u} + \mathbf{x})\mathbf{G}) + \mathbf{v}; \rho_3).$$

Round 2: $\mathcal{V}_1^g(\text{CMT})$ sends a challenge $ch \xleftarrow{\$} \{1, 2, 3\}$ to $\mathcal{P}$.

Round 3: $\mathcal{P}_2^g((\mathbf{G}, \mathbf{y}_1, \cdots, \mathbf{y}_N), (I, \mathbf{x}, \mathbf{e}), ch)$ responds as follows:

- Case $ch = 1$: Set $\xi'_\delta = \zeta_\delta, \mathbf{w}_1 = \mathbf{u} + \mathbf{x}$.
  Send RSP $= (\xi'_\delta, \mathbf{w}_1, \rho_2, \rho_3)$ to $\mathcal{V}$.
- Case $ch = 2$: Set

$$\mathbf{w}_2 = \delta((\mathbf{u} + \mathbf{x})\mathbf{G}) + \mathbf{v}, \mathbf{w}_3 = \delta(\mathbf{e}); \ b = b_I, \text{path} = \text{Gpath}(\text{tree}, I).$$

  Send RSP $= (\mathbf{w}_2, \mathbf{w}_3, b, \text{path}, \rho_3)$ to $\mathcal{V}$.
- Case $ch = 3$: Set $\xi_s = \zeta_s, \xi_u = \zeta_u, \xi_\delta = \zeta_\delta$.
  Send RSP $= (\xi_s, \xi_u, \xi_\delta, \rho_2)$ to $\mathcal{V}$.

Verification: $\mathcal{V}_2^g((\mathbf{G}, \mathbf{y}_1, \cdots, \mathbf{y}_N), \text{CMT}, ch, \text{RSP})$ proceeds as follows:

- Case $ch = 1$: Set $\mathbf{p}_1 \xleftarrow{\$,\xi'_\delta} \mathbb{F}_2^n, \pi_1 \xleftarrow{\$,\xi'_\delta} \mathcal{S}_n$ and check if

$$c_2 = \text{Com}(\pi_1, \mathbf{p}_1; \rho_2), c_3 = \text{Com}(\pi_1(\mathbf{w}_1\mathbf{G}_1) + \mathbf{p}_1; \rho_3).$$

- Case $ch = 2$: Compute R_1 as Rebuild(path,Com$(\mathbf{w}_2 + \mathbf{w}_3; b)$),
  Check if $\mathbf{w}_3 \in B(n, t)$ and

$$c_1 = \text{R}_1, c_3 = \text{Com}(\mathbf{w}_2; \rho_3).$$

- Case $ch = 3$: Set $\mathbf{p}_2 \xleftarrow{\$,\xi_u} \mathbb{F}_2^k, \mathbf{p}_3 \xleftarrow{\$,\xi_\delta} \mathbb{F}_2^n, \pi_2 \xleftarrow{\$,\xi_\delta} \mathcal{S}_n$.
  $(b_1, \cdots, b_N) = \text{Stree}(\xi_s, N)$.
  leaf $= (\text{Com}(\pi_2(\mathbf{p}_2\mathbf{G}_1 + \mathbf{y}_1) + \mathbf{p}_3; b_1), \cdots, \text{Com}(\pi_2(\mathbf{p}_2\mathbf{G}_1 + \mathbf{y}_N) + \mathbf{p}_3; b_N))$.
  Obtain R_2 as IH-Mtree(leaf) and check if

$$c_1 = \text{R}_2, c_2 = \text{Com}(\pi_2, \mathbf{p}_3; \rho_2).$$

$\mathcal{V}$ outputs 1 when all conditions are met, else it outputs 0.

Fig. 3. The one-iteration GSD-based one-out-of-many proof $\Pi^g = (\mathcal{P}^g = (\mathcal{P}_1^g, \mathcal{P}_2^g), \mathcal{V}^g = (\mathcal{V}_1^g, \mathcal{V}_2^g))$.

$\mathcal{E}$ first checks if $(\pi_1^j, \mathbf{p}_1^j, \rho_2^j) \neq (\pi_2^j, \mathbf{p}_3^j, \rho_2^j)$. If so, $\mathcal{E}$ breaks the binding property of Com. Similarly, $\mathcal{E}$ checks if $(\mathbf{w}_2^j, \rho_3^j) \neq (\pi_1^j(\mathbf{w}_1^j\mathbf{G}_1) + \mathbf{p}_1^j, \rho_3^j)$. If neither of these two inequalities holds true, we have

$$\pi_1^j = \pi_2^j, \mathbf{p}_1^j = \mathbf{p}_3^j, \mathbf{w}_2^j = \pi_1^j(\mathbf{w}_1^j\mathbf{G}_1) + \mathbf{p}_1^j. \tag{4}$$

Round 1: $\mathcal{P}_1^{oom}((\mathbf{G}, \{\mathbf{y}_i\}_{i=1}^N), (I, \mathbf{x}, \mathbf{e}))$ chooses the following random objects:

$$\{\zeta_s^{\text{root}}, \zeta_u^{\text{root}}, \zeta_\delta^{\text{root}}, \rho_1^{\text{root}}, \rho_2^{\text{root}}\} \xleftarrow{\$} \{0,1\}^\lambda, \theta \xleftarrow{\$} \{0,1\}^{2\lambda}.$$

- $(\{\zeta_s^i\}_{i=1}^\kappa) = \text{Stree}(\zeta_s^{\text{root}}, \kappa)$, $(\{\zeta_u^i\}_{i=1}^\kappa) = \text{Stree}(\zeta_u^{\text{root}}, \kappa)$, $(\{\zeta_\delta^i\}_{i=1}^\kappa) = \text{Stree}(\zeta_\delta^{\text{root}}, \kappa)$.
- $(\{\rho_1^i\}_{i=1}^\kappa) = \text{Stree}(\rho_1^{\text{root}}, \kappa)$, $(\{\rho_2^i\}_{i=1}^\kappa) = \text{Stree}(\rho_2^{\text{root}}, \kappa)$.
- **For** j from 1 to κ **do**
  $(c_1^j, c_2^j, c_3^j) \leftarrow \mathcal{P}_1^g((\mathbf{G}, \{\mathbf{y}_i\}_{i=1}^N), (I, \mathbf{x}, \mathbf{e}), (\zeta_s^j, \zeta_u^j, \zeta_\delta^j, \rho_1^j, \rho_2^j), \theta)$.
  $(\text{Mtree}_1, C_1) = \text{Mtree}(c_1^1, \cdots, c_1^\kappa), (\text{Mtree}_2, C_2) = \text{Mtree}(c_2^1, \cdots, c_2^\kappa)$,
  $(\text{Mtree}_3, C_3) = \text{Mtree}(c_3^1, \cdots, c_3^\kappa)$.
- Send the commitment $\text{CMT} := (\mathcal{H}(C_1, C_2, C_3), \theta)$ to $\mathcal{V}$.

Round 2: $\mathcal{V}_1^{oom}(\text{CMT})$ sends a challenge $\mathbf{ch} = (ch_1, \cdots, ch_\kappa) \xleftarrow{\$} \{1,2,3\}^\kappa$ to $\mathcal{P}$.

Round 3: $\mathcal{P}_2^{oom}((\mathbf{G}, \{\mathbf{y}_i\}_{i=1}^N), (I, \mathbf{x}, \mathbf{e}), \mathbf{ch})$ responds as follows:

- $\text{path}_1 = \text{Gpath}(\text{Mtree}_1, \{j\}_{ch^j=1})$.
- $\text{path}_2 = \text{Gpath}(\text{Mtree}_2, \{j\}_{ch^j=2}), \text{path}_3 = \text{Gpath}(\text{Mtree}_3, \{j\}_{ch^j=3})$.
- $\zeta_s^{\text{inter}} \leftarrow \text{Oseeds}(\zeta_s^{\text{root}}, \{j\}_{ch^j=2})$.
- $\zeta_\delta^{\text{inter}} \leftarrow \text{Oseeds}(\zeta_\delta^{\text{root}}, \{j\}_{ch^j\neq 1}), \zeta_u^{\text{inter}} \leftarrow \text{Oseeds}(\zeta_u^{\text{root}}, \{j\}_{ch^j=2})$.
- $\zeta_{\rho_1}^{\text{inter}} \leftarrow \text{Oseeds}(\zeta_{\rho_1}^{\text{root}}, \{j\}_{ch^j\neq 1}), \zeta_{\rho_2}^{\text{inter}} \leftarrow \text{Oseeds}(\zeta_{\rho_2}^{\text{root}}, \{j\}_{ch^j\neq 2})$.
- **For** j from 1 to κ **do**
  If $ch^j = 1$ Then
  Set $\mathbf{w}_1^j = \mathbf{u}^j + \mathbf{x}$.
  $\text{rsp}^j = \mathbf{w}_1^j$.
  If $ch^j = 2$ Then
  Set $\mathbf{w}_2^j = \delta^j((\mathbf{u}^j + \mathbf{x})\mathbf{G}_1) + \mathbf{v}^j, \mathbf{w}_3^j = \text{compress}(\delta^j(\mathbf{e}))$.
  $b^j = b_I^j, \text{path}^j = \text{Gpath}(\text{tree}^j, I)$.
  $\text{rsp}^j = (\mathbf{w}_2^j, \mathbf{w}_3^j, b^j, \text{path}^j)$.
- Send $\text{RSP} = (\{\text{rsp}^i\}_{i=1}^\kappa, \text{path}_1, \text{path}_2, \text{path}_3, \zeta_*^{\text{inter}})$.

Verification: $\mathcal{V}_2^{oom}((\mathbf{G}, \{\mathbf{y}_i\}_{i=1}^N), \text{CMT}, \mathbf{ch}, \text{RSP}, \theta)$:

- $\{\zeta_s^j\}_{ch^j=2} \leftarrow \text{Recover}(\zeta_s^{\text{inter}}, \{j\}_{ch^j=2})$.
- $\{\zeta_\delta^j\}_{ch^j\neq 1} \leftarrow \text{Recover}(\zeta_\delta^{\text{inter}}, \{j\}_{ch^j\neq 1}), \{\zeta_u^j\}_{ch^j=2} \leftarrow \text{Recover}(\zeta_u^{\text{inter}}, \{j\}_{ch^j=2})$.
- $\{\zeta_{\rho_1}^j\}_{ch^j\neq 1} \leftarrow \text{Recover}(\zeta_{\rho_1}^{\text{inter}}, \{j\}_{ch^j\neq 1}), \{\zeta_{\rho_2}^j\}_{ch^j\neq 2} \leftarrow \text{Recover}(\zeta_{\rho_2}^{\text{inter}}, \{j\}_{ch^j\neq 2})$.
- **For** j from 1 to κ **do**
  If $ch^j = 1$ Then
  Set $(c_2^j, c_3^j) \leftarrow \mathcal{V}_2^g(ch^j, rsp_j, \zeta_{\delta'}^j, \rho_2^j, \rho_3^j, \theta)$.
  If $ch^j = 2$ Then
  Set $(c_1^j, c_3^j) \leftarrow \mathcal{V}_2^g(ch^j, \text{rsp}^j, \rho_3^j, \theta)$.
  If $ch^j = 3$ Then
  Set $(c_1^j, c_2^j) \leftarrow \mathcal{V}_2^g(ch^j, \text{rsp}^j, \zeta_s^j, \zeta_r^j, \zeta_\delta^j, \rho_2^j, \theta)$.
- $C_1 = \text{Rebuild}(\{c_1^j\}_{ch^j\neq 1}, \text{path}_1), C_2 = \text{Rebuild}(\{c_2^j\}_{ch^j\neq 2}, \text{path}_2)$,
- $C_3 = \text{Rebuild}(\{c_3^j\}_{ch^j\neq 3}, \text{path}_3)$.
- Check if $\text{CMT} = \mathcal{H}(C_1, C_2, C_3)$. If it holds, output 1; otherwise, output 0.

Fig. 4. The GSD-based one-out-of-many proof $\Pi^{\text{oom}} = (\mathcal{P}^{\text{oom}} = (\mathcal{P}_1^{\text{oom}}, \mathcal{P}_2^{\text{oom}}), \mathcal{V}^{\text{oom}} = (\mathcal{V}_1^{\text{oom}}, \mathcal{V}_2^{\text{oom}}))$. Let $\zeta_*^{\text{inter}} = (\zeta_s^{\text{inter}}, \zeta_u^{\text{inter}}, \zeta_\delta^{\text{inter}}, \zeta_{\rho_1}^{\text{inter}}, \zeta_{\rho_2}^{\text{inter}})$.

For convenience, in later proof we let $\pi^j = \pi_1^j, \mathbf{p}^j = \mathbf{p}_1^j$. $\mathcal{E}$ then checks if $\text{Com}(\mathbf{w}_2^j + \mathbf{w}_3^j; b^j)) \neq \text{leaf}_i^j$ for all $i \in [N]$. If so, by employing the Merkle tree ex-

tractor in Lemma 1 with input $(\text{tree}^j, \text{Com}(\mathbf{w}_2^j+\mathbf{w}_3^j; b^j)), \text{path})$, it outputs a collision in $\mathcal{H}$. Otherwise, there exists an index $I \in [N]$ satisfying $\text{Com}(\mathbf{w}_2^j+\mathbf{w}_3^j; b^j) = \text{leaf}_I^j$ and $\mathcal{E}$ further checks if $(\mathbf{w}_2^j + \mathbf{w}_3^j, b^j) \neq (\pi^j(\mathbf{p}_2^j\mathbf{G}_1^j + \mathbf{y}_I^j) + \mathbf{p}^j, b_I^j)$. If so, $\mathcal{E}$ breaks the binding property of Com. Otherwise, $\mathcal{E}$ gets $\mathbf{w}_2^j + \mathbf{w}_3^j = \pi^j(\mathbf{p}_2^j\mathbf{G}_1 + \mathbf{y}_I^j) + \mathbf{p}^j$ and $b^j = b_I^j$ and so it deduces that $(\mathbf{w}_1^j - \mathbf{p}_2^j)\mathbf{G}_1 + (\pi^j)^{-1}(\mathbf{w}_3^j) = \mathbf{y}_I$ by Equation (4), where $(\pi^j)^{-1}(\mathbf{w}_3^j) \in B(n,t)$. This means that for some $I \in [N]$, $\mathcal{E}$ outputs the witness $(\mathbf{w}_1^j - \mathbf{p}_2^j, (\pi^j)^{-1}(\mathbf{w}_3^j))$.

Honest-verifier zero-knowledge: Assume that the adversary $\mathcal{A}$ has accessed the oracle $\mathcal{O}^*(\cdot)$ a total of $\mathcal{Q}$ times, where $\mathcal{O}^*(\cdot)$ instantiates the hash function, algorithm Stree, and Com. Let E_i denote the $\mathcal{A}^{\mathcal{O}}(Sim_i(s, \mathbf{ch})) = 1$, for $i = 1, \cdots, 6$. Sim is built as follows:

Sim first runs

$$\begin{cases} \theta \leftarrow \{0,1\}^{2\lambda}, \zeta_\delta^{\text{inter}} \leftarrow \text{Simseeds}(\{j\}_{ch^j \neq 1}, \theta); \\ \zeta_{\rho_1}^{\text{inter}} \leftarrow \text{Simseeds}(\{j\}_{ch^j \neq 1}, \theta), \zeta_{\rho_2}^{\text{inter}} \leftarrow \text{Simseeds}(\{j\}_{ch^j \neq 2}, \theta); \\ \zeta_s^{\text{inter}} \leftarrow \text{Simseeds}(\{j\}_{ch^j = 2}, \theta), \zeta_u^{\text{inter}} \leftarrow \text{Simseeds}(\{j\}_{ch^j = 2}, \theta). \end{cases}$$

Then, Sim obtains random coins by

$$\begin{cases} \{\zeta_{\rho_1}^j\}_{ch^j \neq 1} \leftarrow \text{Recover}(\zeta_{\rho_1}^{\text{inter}}, \{j\}_{ch^j \neq 1}); \ \{\zeta_{\rho_2}^j\}_{ch^j \neq 2} \leftarrow \text{Recover}(\zeta_{\rho_2}^{\text{inter}}, \{j\}_{ch^j \neq 2}); \\ \{\zeta_s^j\}_{ch^j = 2} \leftarrow \text{Recover}(\zeta_s^{\text{inter}}, \{j\}_{ch^j = 2}); \ \{\zeta_u^j\}_{ch^j = 2} \leftarrow \text{Recover}(\zeta_u^{\text{inter}}, \{j\}_{ch^j = 2}); \\ \{\zeta_\delta^j\}_{ch^j \neq 1} \leftarrow \text{Recover}(\zeta_\delta^{\text{inter}}, \{j\}_{ch^j \neq 1}). \end{cases}$$

For $j = 1$ to κ : Sim performs

Case $ch^j = 1$:

1. Sim samples $\mathbf{w}_1^j \xleftarrow{\$} \mathbb{F}_q^k, c_1^j \xleftarrow{\$} \{0,1\}^{2\lambda}$ and sets $\mathbf{p}_1^j \xleftarrow{\$,\xi_\delta^j} \mathbb{F}_2^n, \pi_1^j \xleftarrow{\$,\xi_\delta^j} \mathcal{S}_n$.
2. Sim sets $c_2^j = \text{Com}(\pi_1^j, \mathbf{p}_1^j; \rho_2^j)$ and $c_3^j = \text{Com}(\pi_1^j(\mathbf{w}_1^j\mathbf{G}_1) + \mathbf{p}_1^j; \rho_3^j)$.
3. Sim sets $\text{rsp}^j = \mathbf{w}_1^j$.

Sim computes $\text{CMT} := \mathcal{H}(C_1, C_2, C_3)$, where

$$C_1 \leftarrow \text{Mtree}(c_1^1, \cdots, c_1^\kappa), C_2 \leftarrow \text{Mtree}(c_2^1, \cdots, c_2^\kappa), C_3 \leftarrow \text{Mtree}(c_3^1, \cdots, c_3^\kappa).$$

and obtains $\text{path}_i = \text{Gpath}(\text{Mtree}_i, \{j\}_{ch^j = i})$ for $i = 1, 2, 3$. It sets

$$\text{RSP} = (\{\text{rsp}^i\}_{i=1}^\kappa, \text{path}_1, \text{path}_2, \text{path}_3, \zeta_*^{\text{inter}})$$

and returns $(\text{CMT}, \mathbf{ch}, \text{RSP}, \theta)$.

Case $ch^j = 2$:

1. Sim chooses the following random objects:

$$\mathbf{w}_2^j \xleftarrow{\$} \mathbb{F}_2^n, \mathbf{w}_3^j \xleftarrow{\$} B(n,t), b^j \xleftarrow{\$} \{0,1\}^\lambda; \ \text{leaf}_i^j \xleftarrow{\$} \{0,1\}^{2\lambda}, \forall i \in [2; N], c_2^j \xleftarrow{\$} \{0,1\}^{2\lambda}.$$

2. Sim sets $\text{leaf}_1^j = \text{Com}(\mathbf{w}_2^j + \mathbf{w}_3^j; b^j)$ and $(\text{tree}^j, \text{root}^j) = \text{IH-Mtree}(\text{leaf}^j)$.
3. Sim sets $c_1^j = \text{root}^j$ and $c_3^j = \text{Com}(\mathbf{w}_2^j; \rho_3^j)$.
4. Sim runs $\text{Gpath}(\text{tree}^j, 1)$ to retrieve the path and sets $\text{rsp}^j = (\mathbf{w}_2^j, \mathbf{w}_3^j, b^j, \text{path}^j)$.

Case $ch^j = 3$:

1. Sim samples the following random objects:

$$\mathbf{p}_2^j \xleftarrow{\$,\xi_u^j} \mathbb{F}_2^{k_1}; \mathbf{p}_3^j \xleftarrow{\$,\xi_\delta^j} \mathbb{F}_2^n; \pi_2^j \xleftarrow{\$,\xi_\delta^j} \mathcal{S}_n, c_3^j \xleftarrow{\$} \{0,1\}^{2\lambda}.$$

2. Sim sets $\text{leaf}^j = (\text{Com}(\pi_2^j(\mathbf{p}_2^j\mathbf{G}_1+\mathbf{y}_1)+\mathbf{p}_3^j; b_1^j), \cdots, \text{Com}(\pi_2^j(\mathbf{p}_2^j\mathbf{G}_1+\mathbf{y}_N)+\mathbf{p}_3^j; b_N^j))$ and calculates its root^j, where $(b_1^j, \cdots, b_N^j) = \text{Stree}(\xi_s^j, N)$.

3. Sim sets $c_1^j = \text{root}^j$ and $c_2^j = \text{Com}(\pi_2^j, \mathbf{p}_3^j; \rho_2^j)$.

Similar to the proof of Theorem 2, we only need to check the case of RSP. We use a sequence of simulators $\{Sim_i\}_{i=1}^6$ to prove that Sim and an honest prover are indistinguishable, in whcih Sim_1 and Sim_6 represent the honest prover and Sim, respectively.

Sim_2: The only difference between Sim_2 and the honest prover is that the tuple internal seeds ζ_*^{inter} are generated by the algorithm Simseeds with the input **ch** intead of being generated by the algorithms Stree and Oseeds with inputs ζ_*^{root} and **ch**. According to Lemma 3, the advantage of distinguishing between these two tuples internal seeds is $\mathcal{Q}/2^\lambda$ when the oracle $\mathcal{O}^{\text{Stree}}(\cdot)$ is accessed $\mathcal{Q}$ times. Therefore, we have $|\Pr[E_2] - \Pr[E_1]| \leq \mathcal{Q}/2^\lambda$.

Observe that, if ζ_*^{inter} has not been queried, when $ch^j = 1$ and $ch^j = 3$, Sim_2 can perfectly simulate. We only prove the case of $ch^j = 2$.

Sim_3: The only difference between Sim_3 and Sim_2 is that $\{b_i^j\}_{i=1}^N$ are randomly sampled from $\{0,1\}^\lambda$ instead of being generated by $\mathcal{O}^{\text{Stree}}(\cdot)$ with the input ζ_s^j, where ζ_s^j is generated by $\mathcal{O}^{\text{Oseeds}}(\zeta_s^{\text{inter}})$. If ζ_s^j has not been queried by $\mathcal{A}$ to the oracle $\mathcal{O}^{\text{Stree}}(\cdot)$, Sim_3 and Sim_2 are indistinguishable. Thus, if $\mathcal{O}^{\text{Stree}}(\cdot)$ is accessed $\mathcal{Q}$ times and the probability of colliding with ζ_s^j each query is 2^λ, the probability of colliding with ζ_s^j after $\mathcal{Q}$ queries is $\mathcal{Q}/2^\lambda$. Thus, we have $|\Pr[E_3] - \Pr[E_2]| \leq \mathcal{Q}/2^\lambda$.

Sim_4: The only difference between Sim_3 and Sim_2 is that $\{\text{leaf}_i^j\}_{i=1,i\neq I}^N$ are randomly sampled from $\{0,1\}^{2\lambda}$ instead of being generated by $\mathcal{O}^{\text{Com}}(\cdot)$ with the tuple $(\delta^j(\mathbf{p}_2^j\mathbf{G} + \mathbf{y}_i) + \mathbf{p}_3^j; b_i^j)$ for $i \neq I$. If all tuples have not been queried by $\mathcal{A}$ to $\mathcal{O}^{\text{Com}}(\cdot)$, Sim_3 and Sim_2 are indistinguishable. Let $\mathcal{Q}_{\text{com}_i}$ represent the number of times $\mathcal{O}^{\text{Com}_i}(\cdot)$ is accessed. Since the minimum entropy of b_i^j is $1/2^\lambda$, the minimum entropy of tuple $(\delta^j(\mathbf{p}_2^j\mathbf{G} + \mathbf{y}_i) + \mathbf{p}_3^j; b_i^j)$ is at most $1/2^\lambda$. Hence, the probability of collision with the tuple in each query is at most $1/2^\lambda$. Furthermore, since $\sum_{i=1}^N \mathcal{Q}_{\text{com}_i} \leq \mathcal{Q}$, we have $|\Pr[E_4] - \Pr[E_3]| \leq Q/2^\lambda$.

Sim_5: There are two differences between Sim_5 and Sim_4. Firstly, c_3^j is randomly sampled from $\{0,1\}^{2\lambda}$. Secondly, $\text{leaf}_I^j = \text{Com}(\mathbf{w}_2^j+\mathbf{w}_3^j; b_I^j)$, where $\mathbf{w}_2^j$ and $\mathbf{w}_3^j$ are randomly sampled from $\mathbb{F}_2^n$ and $B(n,t)$ respectively. Since $\mathbf{w}_2^j$ and $\mathbf{w}_3^j$ follows the random distribution as the real transcript, we have $\Pr[E_5] = \Pr[E_4]$.

Sim_6: The only difference between Sim_6 and Sim_5 is that 1 is used instead of I in witness. Lemma 2 states that regardless of whether the selected index is 1 or I, the root and path follow the same distribution. Therefore, we have $\Pr[E_6] = \Pr[E_5]$.

Thus, we have $\left|\Pr[\mathcal{A}^{\mathcal{O}}(\mathcal{P}(s,w,\mathbf{ch})) \to 1] - \Pr[\mathcal{A}^{\mathcal{O}}(Sim(s,\mathbf{ch})) \to 1]\right| \leq \frac{3\mathcal{Q}}{2^\lambda}$. □

Communication cost: *(1)* The cost of commitments is $2\lambda(5\kappa/6+1)$. *(2)* The cost of seeds is about $(\kappa/3)(20\lambda/3)+2\lambda$. *(3)* The cost of vectors and the path of the Merkle tree is about $(\kappa/3)(\frac{3n}{2}+k+2\lambda\log N)$.

3.3 Our set-membership proof

A set-membership proof is a concept similar to an one-out-of-many proof. It allows one to prove that an element in a public set satisfies a given property, i.e. given a publicly set $\mathcal{I}$ and a property G, one proves the existence of an element α_i such that $\alpha_i \in \mathcal{I}$ and $G(\alpha_i)$ holds. Consider an example where a commitment $\mathbf{c} = \mathrm{Com}_{\mathrm{McE}}$ and a public set $\mathcal{I} = \{\alpha_1, \cdots, \alpha_N\}$ are given. The goal is to demonstrate that $\mathbf{c}$ is a commitment to an element in $\mathcal{I}$ and so our set-membership proof can be achieved by the protocol in Fig. 4 with the input of the set $[\mathbf{c}_1 = \mathbf{c} + \mathrm{Com}_{\mathrm{McE}}(\alpha_1; (\mathbf{0}, \mathbf{0})), \cdots, \mathbf{c}_N = \mathbf{c} + \mathrm{Com}_{\mathrm{McE}}(\alpha_N; (\mathbf{0}, \mathbf{0}))]$ and public commitment key $\mathbf{G}$.

4 Our code-based logarithmic-size ring signature scheme

A ring signature enables a ring member to sign a message on behalf of the ring anonymously. Our GSD-based one-out-of-many proof can be transformed into a ring signature scheme through Fiat-Shamir transform. Specifically, each user has a public-private key pair $(\mathbf{y}_i, (\mathbf{x}_i, \mathbf{e}_i))$, where $\mathbf{y}_i = \mathbf{x}_i\mathbf{G} + \mathbf{e}_i$ and $\mathbf{e}_i$ is small. The signature for a message μ is our GSD-based one-out-of-many proof for $(\mathbf{x}_i, \mathbf{e}_i)$ satisfying Equation (3), where μ is the input into the random oracle of Fiat-Shamir transform. Our ring signature scheme is presented in Fig. 5.

RS.Setup(1^λ):
- $\mathbf{G} \xleftarrow{\$} \mathbb{F}_2^{k\times n}$, where $n = \mathcal{O}(\lambda)$ and $k = \mathcal{O}(\lambda)$.
- $\mathcal{H}_{com} \xleftarrow{\$} \mathcal{H}(1^\lambda)$, $\mathcal{H}_{FS} \xleftarrow{\$} \mathcal{H}(1^\lambda, \kappa)$, $\mathcal{H}_{Merkle} \xleftarrow{\$} \mathcal{H}(1^\lambda, N)$, where $\kappa = \omega(\log(\lambda))$.
- Return $pp = (\mathbf{G}, \mathcal{H}_{FS}, \mathcal{H}_{com})$.

RS.KeGen(pp, N):
- $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^k, \mathbf{e} \xleftarrow{\$} B(n,t)$, $\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e}$.
- Return $(pk, sk) = (\mathbf{y}, (\mathbf{x}, \mathbf{e}))$.

RS.Sign(sk_j, M, R):
- $\mathrm{CMT} \leftarrow \mathcal{P}_1^{oom}(\mathbf{G}, R, sk_j)$.
- $\mathbf{ch} \leftarrow \mathcal{H}_{FS}(M, R, \mathrm{CMT})$.
- $\mathrm{RSP} \leftarrow \mathcal{P}_2^{oom}(\mathbf{G}, R, \mathbf{ch}, sk_j)$.
- Return $\sigma = (\mathrm{CMT}, \mathrm{RSP})$.

RS.Verify(M, σ):
- $(\mathrm{CMT}, \mathrm{RSP}) \leftarrow \sigma$.
- $\mathbf{ch} \leftarrow \mathcal{H}_{FS}(M, R, \mathrm{CMT})$.
- Return $\mathcal{V}_2^{oom}(\mathbf{G}, R, (\mathrm{CMT}, \mathbf{ch}, \mathrm{RSP}))$.

Fig. 5. Our ring signature scheme, where $R := (pk_1, \cdots, pk_N)$.

First, we introduce the definition and security requirements of a ring signature scheme.

Definition 1. *A ring signature scheme contains four polynomial-time algorithms* $\mathcal{RS} = (\mathbf{RS.Setup}, \mathbf{RS.KeyGen}, \mathbf{RS.Sign}, \mathbf{RS.Verify})$,

- $pp \leftarrow \mathbf{RS.Setup}(1^\lambda)$: *Taking a security parameter* 1^λ *as input, output the public parameters* pp.
- $(pk, sk) \leftarrow \mathbf{RS.KeyGen}(pp, N)$: *Taking the public parameter* pp *and the number* N *of ring users as input, publish a pair of public and private keys* (pk_i, sk_i) *for the* i*-th user* $i \in [N]$.
- $\sigma \leftarrow \mathbf{RS.Sign}(R, M, sk)$: *Taking the list of public keys* $R = (pk_1, \cdots, pk_N)$, *a private key* sk *and a message* M *as input, generate a signature* σ.
- $b \leftarrow \mathbf{RS.Verify}(R, M, \sigma)$: *Taking the list of public keys* R, *a message* M, *and a signature* σ *as input, output* b *either* 1 *(accept) or* 0 *(reject).*

A ring signature needs to satisfy three properties: correctness, unforgeability, and anonymity. Correctness ensures that a valid signature can always be verified. Unforgeability guarantees that only users in the ring can generate valid signatures. Anonymity ensures that the output signature does not leak the identity of the signer.

Definition 2 (Correctness). *A ring signature scheme achieves correctness if for any* $\lambda \in \mathbb{N}$, $N = poly(\lambda)$, $j \in [N]$ *and every message* M, *the following holds:*

$$\Pr\left[\mathbf{RS.Verify}(R, M, \sigma) = 1 \middle| \begin{array}{c} pp \leftarrow \mathbf{RS.Setup}(1^\lambda), \\ (pk_i, sk_i) \leftarrow \mathbf{RS.KeyGen}(pp), \forall i \in [N], \\ R = (pk_1, \cdots, pk_N), \\ \sigma \leftarrow \mathbf{RS.Sign}(R, M, sk_j). \end{array}\right] = 1.$$

Definition 3 (Unforgeability w.r.t. insider corruption). *A ring signature scheme* $\mathcal{RS} = (\mathbf{RS.Setup}, \mathbf{RS.KeyGen}, \mathbf{RS.Sign}, \mathbf{RS.Verify})$ *is unforgeable if the advantage of* $\mathcal{A}$ *is negligible in the following game:*

1. $pp \leftarrow \mathbf{RS.Setup}(1^\lambda)$, $(pk_i, sk_i) \leftarrow \mathbf{RS.KeyGen}(pp, N), \forall i \in [N]$;
2. $\mathcal{A}$ *can have access to the corrupted oracle* $\mathbf{Co}(\cdot)$ *with any* $pk_j \in R$ *and* $\mathbf{Co}(\cdot)$ *returns the* sk_j *to* $\mathcal{A}$ *and adds* pk_j *to the set* CU;
3. $\mathcal{A}$ *can have access to the* $\mathbf{RS.Sign}(\cdot)$ *with* (pk_j, M) *and then* $\mathbf{RS.Sign}(\cdot)$ *returns a signature* σ *using the secert key* sk_j;
4. $\mathcal{A}$ *outputs* (R^*, M^*, σ^*) *such that* $\mathbf{RS.Verify}(R^*, M^*, \sigma^*) = 1$, *where* (R^*, M^*) *has never been asked and* $R^* \cap CU = \emptyset$.

The advantage of breaking unforgeability is

$$\mathrm{Adv}_{\mathcal{A}}^{Unf}(\lambda) = \Pr[\mathbf{RS.Verify}(R, M, \sigma) = 1].$$

Definition 4 (Anonymity). *A ring scheme* $\mathcal{RS} = (\mathbf{RS.Setup}, \mathbf{RS.KeyGen}, \mathbf{RS.Sign}, \mathbf{RS.Verify})$ *is anonymous if the advantage of* $\mathcal{A}$ *is negligible in the following game:*

1. $pp \leftarrow \mathbf{RS.Setup}(1^\lambda), (pk_i, sk_i) \leftarrow \mathbf{RS.KeyGen}(pp, N), \forall i \in [N]$;
2. $\mathcal{A}$ *selects a tuple* $(M, pk_{i_0}, pk_{i_1}), i_0, i_1 \in [N]$;
3. $\sigma \leftarrow \mathbf{RS.Sign}(M, sk_{i_b}), b \xleftarrow{\$} \{0, 1\}$;
4. $\mathcal{A}$ *returns a* b'.

The advantage of breaking anonymity is

$$\mathrm{Adv}_{\mathcal{A}}^{Anon}(\lambda) = |\Pr[b = b'] - 1/2|. \tag{5}$$

Theorem 4. *Our ring signature scheme in Fig. 5 achieves correctness, anonymity and unforgeability with regard to. insider corruption in the random oracle model.*

Proof. **Correctness:** The correctness in Fig. 5 can be directly inferred from the completeness of the protocol in Fig. 4.
Anonymity: Since the protocol in Fig. 4 is zero-knowledge, it implies that the protocol is witness-indistinguishable. As a result, the anonymity in Fig. 5 can be obtained immediately.
Unforgeability: We only provide the proof framework without providing specific details. The proof framework is as follows: The challenger first selects an index j^* and set the pk_{j^*} as the challenge instance. The rest pk_j are generated as the same as the real **RS.KeGen**. If $\mathcal{A}$ queries $\mathbf{RS.Sign}(\cdot)$ with pk_{j^*}, the challenger uses the simulator of Fig. 4 to output the signature. Otherwise, the challenger honestly generates the signature. Under the condition that j^* has not been queried with $\mathbf{Co}(\cdot)$ and $\mathcal{A}$ outputs a forged signature about pk_{j^*}, the challenger can extract the solution to the challenge problem via rewind techniques. □

5 Code-based group signatures

Our GSD-based one-out-of-many proof can be applied in the construction of a group signature. However, unlike ring signatures, a group signature scheme cannot be obtained by directly applying the Fiat-Shamir transform to our one-out-of-many proofs. Therefore, we first construct a ZKAoK that allows a signer to prove membership in the group and the honest encryption of its own identity using the public key of the openers. Then, we apply the Fiat-Shamir transform on the ZKAoK to obtain our group signature scheme. The high-level idea of the construction of this ZKAoK is to combine our GSD-based one-out-of-many proof and our set-memberships proof. We first give the definitions and security requirements of a group signature.

Definition 5. *A group signature scheme contains five polynomial-time algorithms* $(\mathbf{GS.Setup}, \mathbf{GS.KeyGen}, \mathbf{GS.Sign}, \mathbf{GS.Verify}, \mathbf{GS.Open})$ *in which:*

- $(pp, mpk, msk) \leftarrow \mathbf{GS.Setup}(1^\lambda)$: *Taking a security parameter* λ *as input, output the the public parameters pp and the group manager's public-secret key pair* (mpk, msk).

- $(pk, sk) \leftarrow \mathbf{GS.KeyGen}(pp, N)$: *Taking the public parameter pp and the number N of group members as input, publish a pair of public and private keys (pk_i, sk_i) for each user i, $i \in [N]$.*
- $\sigma \leftarrow \mathbf{GS.Sign}(R, mpk, M, sk)$: *Taking the list of public keys $R = (pk_1, \cdots, pk_N)$, the manager's public key mpk, a private key sk and a message M as input, generate a signature σ.*
- $b \leftarrow \mathbf{GS.Verify}(R, M, \sigma)$: *Taking the public key R, a message M, and a signature σ as input, output b either 1 (accept) or 0 (reject).*
- $i \leftarrow \mathbf{GS.Open}(R, msk, M, \sigma)$: *Taking the public key R, the group manager's secret key msk, a message M and a group signature σ as input, output an index $i \in [N]$ or $\perp$, indicating failure.*

A group signature scheme needs to achieve three requirements: correctness, anonymity and traceability. We give the relaxed anonymity requirement, namely CPA-anonymity.

Definition 6 (Correctness). *A group signature scheme is correct if for any $\lambda \in \mathbb{N}$, $N = poly(\lambda)$, $j \in [N]$ and every message M, the following holds:*

$$\Pr\left[\begin{array}{c}\mathbf{GS.Verify}(R, M, \sigma) = 1 \\ \mathbf{GS.Open}(msk, R, M, \sigma) = j.\end{array}\middle|\begin{array}{c}(pp, mpk, msk) \leftarrow \mathbf{GS.Setup}(1^\lambda), \\ (pk_i, sk_i) \leftarrow \mathbf{GS.KeyGen}(pp), \forall i \in [N], \\ R = (pk_1, \cdots, pk_N), \\ \sigma \leftarrow \mathbf{GS.Sign}(R, mpk, M, sk_j).\end{array}\right] = 1.$$

Definition 7 (CPA-anonymity). *A group signature scheme $\mathcal{GS} = (\mathbf{GS.Setup}, \mathbf{GS.KeyGen}, \mathbf{GS.Sign}, \mathbf{GS.Verify}, \mathbf{GS.Open})$ is CPA-anonymous if the advantage of any PPT adversary $\mathcal{A}$ is negligible in the following game:*

1. $(pp, mpk, msk) \leftarrow \mathbf{GS.Setup}1^\lambda)$;
2. $(pk_i, sk_i) \leftarrow \mathbf{GS.KeyGen}(pp, N), \forall i \in [N]$;
3. $\mathcal{A}$ *selects a tuple* $(M, pk_{i_0}, pk_{i_1}), i_0, i_1 \in [N]$;
4. $\sigma \leftarrow \mathbf{GS.Sign}(R, mpk, M, sk_{i_b}), b \xleftarrow{\$} \{0, 1\}$;
5. $\mathcal{A}$ *outputs a guess b'.*

The advantage of $\mathcal{A}$ in breaking CPA-anonymity is denoted by

$$\mathrm{Adv}_{\mathcal{A}}^{Anon}(\lambda) = |\Pr[b = b'] - 1/2|.$$

Definition 8 (Traceability). *A group scheme $\mathcal{GS} = (\mathbf{GS.Setup}, \mathbf{GS.KeyGen}, \mathbf{GS.Sign}, \mathbf{GS.Verify}, \mathbf{GS.Open})$ is traceable if the advantage of any PPT $\mathcal{A}$ is negligible in the following game:*

1. $(pp, mpk, msk) \leftarrow \mathbf{GS.Setup}(1^\lambda)$;
2. $(pk_i, sk_i) \leftarrow \mathbf{GS.KeyGen}(pp, N), \forall i \in [N]$;
3. $\mathcal{A}$ *can have access to the corrupted oracle* $\mathbf{Co}(\cdot)$ *with any* $pk_j \in R$, *and* $\mathbf{Co}(\cdot)$ *returns the sk_j to $\mathcal{A}$ and adds pk_j to the set CU*;

4. $\mathcal{A}$ *can have access to the* $\mathbf{GS.Sign}(\cdot)$ *with* (pk_j, M) *and* $\mathbf{GS.Sign}(\cdot)$ *returns a signature* σ *using the secert key* sk_j;
5. $\mathcal{A}$ *outputs* (R^*, M^*, σ^*) *such that* $\mathbf{GS.Verify}(R^*, M^*, \sigma^*) = 1$*, where* (R^*, M^*) *have never been asked and* $R^* \cap CU = \emptyset$.

The advantage of breaking traceability is denoted by

$$\mathrm{Adv}_{\mathcal{A}}^{Trac}(\lambda) = \Pr\left[\begin{array}{l}\mathbf{GS.Verify}(R, mpk, M^*, \sigma^*) = 1,\\ \mathbf{GS.Open}(R, msk, M^*, \sigma^*) \notin CU.\end{array}\right].$$

CPA-McEliece: We review the randomized McEliece encryption scheme [35]. It includes the following three algorithms: $\mathrm{KeyGen}_{\mathrm{McE}}$, $\mathrm{Enc}_{\mathrm{McE}}$, and $\mathrm{Dec}_{\mathrm{McE}}$.

- $(pk = \mathbf{G}_{\mathrm{McE}}, sk = (\mathbf{S}, \mathbf{G}', \mathbf{P})) \leftarrow \mathrm{KeyGen}_{\mathrm{McE}}(1^\lambda)$: With an integer λ as input, select a generator matrix $\mathbf{G}'$ of a random w-error-correcting (m, ℓ) code, and sample a random matrix $\mathbf{S} \in \mathbb{F}_2^{\ell \times \ell}$ and a random m-dimension permutation matrix $\mathbf{P}$, where $m = \mathcal{O}(\lambda), \ell = \mathcal{O}(\lambda), w = \mathcal{O}(\lambda)$. Output the encryption key as $\mathbf{G}_{\mathrm{McE}} = \mathbf{SG'P}$ and the decryption key as $(\mathbf{S}, \mathbf{G}', \mathbf{P})$.
- $\mathbf{ct} \leftarrow \mathrm{Enc}_{\mathrm{McE}}(\mathbf{G}_{\mathrm{McE}}, \mathbf{m})$: With a plaintext $\mathbf{m} \in \{0,1\}^{\ell_2}$ and the $\mathbf{G}_{\mathrm{McE}}$ as input, sample two random vectors $\mathbf{z} \in \mathbb{F}_2^{\ell_1}$ and $\mathbf{s} \in B(m, w)$, where $\ell = \ell_1 + \ell_2$. Output the ciphertext $\mathbf{ct} = (\mathbf{z}||\mathbf{m})\mathbf{G} + \mathbf{s}$.
- $\mathbf{m} \leftarrow \mathrm{Dec}_{\mathrm{McE}}(\mathbf{ct}, sk)$: With the ciphertext $\mathbf{ct}$ and sk as input, compute $\mathbf{m}' = \mathbf{S}^{-1}\mathcal{D}_{\mathbf{G}'}(\mathbf{ctP}^{-1})$, where $\mathcal{D}_{\mathbf{G}'}$ is the error-correcting algorithm. Parse the $\mathbf{m}' = (\mathbf{z}, \mathbf{m}) \in \mathbb{F}_2^{\ell_1} \times \mathbb{F}_2^{\ell_2}$ and outputs $\mathbf{m}$.

The above scheme's CPA-security is based on the following two problems.

Problem 5 (Decisional McEliece problem [35]). Given a matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$, determine whether it is randomly sampled or generated by the $\mathrm{KeyGen}_{\mathrm{McE}}$.

Problem 6 (Decisional Learning Parity with (fixed-weight) Noise problem [15]). Given a matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ and a vector $\mathbf{y} \in \mathbb{F}_2^n$, determine whether $\mathbf{y}$ is randomly generated or generated by $\mathbf{xG} + \mathbf{e}$, where $\mathbf{x} \in \mathbb{F}_2^k$ and $\mathbf{e} \in B(n, t)$.

5.1 The underlying protocol of our group signature

In this subsection, we construct a ZKAoK in Fig. 6 to serve as the building block of our group signature scheme. We first give an overview of our construction. Let $k, n, \ell = \ell_1 + \ell_2$, m denote integers, and $\mathrm{bin}(j)$ denote the binary representation of j with length ℓ_2. The public input includes two matrices $\mathbf{G} \in \mathbb{F}_2^{k \times n}, \mathbf{G}_{\mathrm{McE}} = \begin{pmatrix}\mathbf{G}^1_{\mathrm{McE}}\\ \mathbf{G}^2_{\mathrm{McE}}\end{pmatrix} \in \mathbb{F}_2^{\ell \times m}$, any vector $\mathbf{y}_i \in \mathbb{F}_2^n, i \in [N]$ and a ciphertext $\mathbf{ct} \in \mathbb{F}_2^m$. The protocol allows one to prove the following relation in zero-knowledge

$$-\ \mathbf{xG} + \mathbf{e} = \mathbf{y}_I \wedge \mathbf{x} \in \mathbb{F}_2^k, \mathbf{e} \in B(n, t), \tag{6}$$

$$-\ (\mathbf{z}||\,\mathrm{bin}(I))\mathbf{G}_{\mathrm{McE}} + \mathbf{s} = \mathbf{ct} \wedge \mathbf{z} \in \mathbb{F}_2^{\ell_1}, \mathbf{s} \in B(m, w), \tag{7}$$

$$-\ I \in [N]. \tag{8}$$

Round 1: $\mathcal{P}_1^{\mathrm{GS}}(\triangle, (I, \mathbf{x}, \mathbf{e}, \mathbf{z}, \mathbf{s}))$:

$$\begin{cases} \{\zeta_s, \zeta_{u,r}, \zeta_{\delta,\phi}, \rho_1, \rho_2\} \xleftarrow{\$} \{0,1\}^\lambda; \\ \mathbf{u} \xleftarrow{\$,\zeta_{u,r}} \mathbb{F}_2^k, \mathbf{v} \xleftarrow{\$,\zeta_{\delta,\phi}} \mathbb{F}_2^n, \delta \xleftarrow{\$,\zeta_{\delta,\phi}} \mathcal{S}_n; \\ \mathbf{r} \xleftarrow{\$,\zeta_{u,r}} \mathbb{F}_2^{\ell_1}, \mathbf{f} \xleftarrow{\$,\zeta_{\delta,\phi}} \mathbb{F}_2^m, \phi \xleftarrow{\$,\zeta_{\delta,\phi}} \mathcal{S}_m. \end{cases}$$

- $(b_1, \cdots, b_N) = \mathrm{Stree}(\zeta_s, N)$.
- $\{\mathrm{leaf}_i\}_{i=1}^N = \{\mathrm{Com}(\delta(\mathbf{uG} + \mathbf{y}_i) + \mathbf{v}, \phi((\mathbf{r}||\,\mathrm{bin}(i))\mathbf{G}_{\mathrm{McE}} + \mathbf{ct}) + \mathbf{f}; b_i)\}_{i=1}^N$.
- $(\mathrm{root}, \mathrm{tree}) = \text{IH-Mtree}(\mathrm{tree})$.
- Send the commitment $\mathrm{CMT} := (c_1, c_2, c_3)$ to $\mathcal{V}$, where

$$\begin{cases} c_1 = \mathrm{root}; \\ c_2 = \mathrm{Com}(\delta, \phi, \mathbf{v}, \mathbf{f}; \rho_2); \\ c_3 = \mathrm{Com}(\delta((\mathbf{u}+\mathbf{x})\mathbf{G}) + \mathbf{v}, \phi((\mathbf{r}+\mathbf{z}||\mathbf{0})\mathbf{G}_{\mathrm{McE}}) + \mathbf{f}; \rho_3). \end{cases}$$

Round 2: $\mathcal{V}_1^{\mathrm{GS}}(\mathrm{CMT})$ sends a challenge $ch \xleftarrow{\$} \{1,2,3\}$ to $\mathcal{P}$.

Round 3: $\mathcal{P}_2^{\mathrm{GS}}(\triangle, (I, \mathbf{x}, \mathbf{e}, \mathbf{z}, \mathbf{s}), ch)$:

- Case $ch = 1$: Set $\xi'_{\delta,\phi} = \zeta_{\delta,\phi}, \mathbf{w}_1 = \mathbf{u} + \mathbf{x}, \mathbf{w}_2 = \mathbf{r} + \mathbf{z}$.
  Send $\mathrm{RSP} = (\xi'_{\delta,\phi}, \mathbf{w}_1, \mathbf{w}_2, \rho_2, \rho_3)$ to $\mathcal{V}$.
- Case $ch = 2$: Set

$$\begin{cases} \mathbf{w}_3 = \delta((\mathbf{u}+\mathbf{x})\mathbf{G}) + \mathbf{v}, \mathbf{w}_4 = \delta(\mathbf{e}), \mathbf{w}_5 = \phi((\mathbf{r}+\mathbf{z}||\mathbf{0})\mathbf{G}_{\mathrm{McE}}) + \mathbf{f}; \\ \mathbf{w}_6 = \phi(\mathbf{s}), b = b_I, \mathrm{path} = \mathrm{Gpath}(\mathrm{tree}, I). \end{cases}$$

  Send $\mathrm{RSP} = (\mathbf{w}_3, \mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6, b, \mathrm{path}, \rho_3)$ to $\mathcal{V}$.
- Case $ch = 3$: Set $\xi_s = \zeta_s, \xi_{u,r} = \zeta_{u,r}, \xi_{\delta,\phi} = \zeta_{\delta,\phi}$.
  Send $\mathrm{RSP} = (\xi_s, \xi_{u,r}, \xi_{\delta,\phi}, \rho_2)$ to $\mathcal{V}$.

Verification: $\mathcal{V}_2^{\mathrm{GS}}(\triangle, \mathrm{CMT}, ch, \mathrm{RSP})$:

- Case $ch = 1$: Set $\mathbf{p}_1 \xleftarrow{\$,\xi_{\delta,\phi}} \mathbb{F}_2^n, \pi_1 \xleftarrow{\$,\xi_{\delta,\phi}} \mathcal{S}_n, \mathbf{p}_2 \xleftarrow{\$,\xi_{\delta,\phi}} \mathbb{F}_2^m, \pi_2 \xleftarrow{\$,\xi_{\delta,\phi}} \mathcal{S}_m$
  and check if

$$c_2 = \mathrm{Com}(\pi_1, \pi_2, \mathbf{p}_1, \mathbf{p}_2; \rho_2), c_3 = \mathrm{Com}(\pi_1(\mathbf{w}_1\mathbf{G}) + \mathbf{p}_1, \pi_2(\mathbf{w}_2\mathbf{G}^1_{\mathrm{McE}}) + \mathbf{p}_2; \rho_2).$$

- Case $ch = 2$: Compute R_1 as Rebuild(path,$\mathrm{Com}(\mathbf{w}_3 + \mathbf{w}_4, \mathbf{w}_5 + \mathbf{w}_6; b)$),
  and check if $\mathbf{w}_4 \in B(n,t)$, $\mathbf{w}_6 \in B(m,w)$ and

$$c_1 = \mathrm{R}_1, c_3 = \mathrm{Com}(\mathbf{w}_3, \mathbf{w}_5; \rho_3).$$

- Case $ch = 3$: Set $\mathbf{p}_3 \xleftarrow{\$,\xi_{u,r}} \mathbb{F}_2^k, \mathbf{p}_4 \xleftarrow{\$,\xi_{\delta,\phi}} \mathbb{F}_2^n, \mathbf{p}_5 \xleftarrow{\$,\xi_{u,r}} \mathbb{F}_2^{\ell_1}, \mathbf{p}_6 \xleftarrow{\$,\xi_{\delta,\phi}} \mathbb{F}_2^m, \pi_3 \xleftarrow{\$,\xi_{\delta,\phi}} \mathcal{S}_n,$
  $\pi_4 \xleftarrow{\$,\xi_{\delta,\phi}} \mathcal{S}_m$. $(b_1, \cdots, b_N) = \mathrm{Stree}(\xi_s, N)$.
  $\{\mathrm{leaf}_i\}_{i=1}^N = \{\mathrm{Com}(\pi_3(\mathbf{p}_3\mathbf{G} + \mathbf{y}_i) + \mathbf{p}_4, \pi_4((\mathbf{p}_5||\,\mathrm{bin}(i))\mathbf{G}_{\mathrm{McE}} + \mathbf{ct}) + \mathbf{p}_6; b_i)\}_{i=1}^N$.
  Obtain R_2 as IH-Mtree(leaf) and check if

$$c_1 = \mathrm{R}_2, c_2 = \mathrm{Com}(\pi_3, \pi_4, \mathbf{p}_4, \mathbf{p}_6; \rho_2).$$

$\mathcal{V}$ outputs 1 when all conditions are met, else it outputs 0.

Fig. 6. The underlying ZKAoK $\Pi^{\mathrm{GS}} = (\mathcal{P}^{\mathrm{GS}} = (\mathcal{P}_1^{\mathrm{GS}}, \mathcal{P}_2^{\mathrm{GS}}), \mathcal{V}^{\mathrm{GS}} = (\mathcal{V}_1^{\mathrm{GS}}, \mathcal{V}_2^{\mathrm{GS}}))$ of our group signature scheme. We use $\triangle$ to denote $(\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, \mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{ct})$.

To prove Equations (6) and (8), we can utilize our GSD-based one-out-of-many proof with the public input $(\mathbf{G}, \{\mathbf{y}\}_{i=1}^{N})$. Likewise, to prove Equations (7) and (8), we can employ our set-membership proof with the public input $(\mathbf{G}_{\mathrm{McE}}, \mathbf{ct})$. To prove Equations (6), (7), and (8) simultaneously, we need to merge these two proofs. The high-level idea is to pair the $N+2$ corresponding commitments from the former proof with the $N+2$ commitments from the latter proof in sequential order, and then compress the last N pairs of commitments related to $(\{\mathbf{y}_i\}_{i=1}^{N}, \mathbf{ct})$ into a single root using the IH-Mtree. Next, the prover sends the root along with the remaining two pairs to the verifier. The rest steps are similar to the Fig. 4, with the difference being that what needs to be revealed and verified is the commitment pair. The protocol for this relation (6)(7)(8) is in Fig. 6.

Theorem 5. *The protocol in Fig. 6 is an argument of knowledge with the perfect completeness and zero-knowledge.*

Proof. We only provide the framework of this proof. Completeness: This can be obtained from the correctness of the Merkle tree and seedtree. Soundness: If a adversary can be accepted with a probability greater than $2/3$, then we proceed as in Theorem 3. Namely, we construct an extractor to extract $(I, \mathbf{x}, \mathbf{e}, \mathbf{z}, \mathbf{s})$ from three valid responses. Zero-knowledge: Using the similar steps in Theorem 3, we can construct a simulator for this protocol. □

Communication cost: *(1)* The cost of commitments is about $2\lambda(5\kappa/6+1)$. *(2)* The cost of seeds is about $(\kappa/3)(20\lambda/3)+2\lambda$. *(3)* The cost of vectors and the path of Merkle tree is about $(\kappa/3)(\frac{3n}{2}+\frac{3m}{2}+k+\ell+2\lambda\log N)$.

5.2 Our logrithm-size group signature scheme

Our group signature scheme is described in Fig. 7, and its correctness can be directly inferred from the completeness of the protocol in Fig. 6.

Theorem 6. *Our group signature scheme in Fig. 7 is CPA-anonymous based on the hardness of* $\mathrm{DMcE}(m,\ell,w)$ *and* $\mathrm{DLPN}(m,\ell_1,w)$ *problems, and the zero-knowledge property of protocol in Fig. 6.*

Proof. Let $\mathcal{A}$ denote a PPT adversary who breaks the CPA-anonymity of Fig. 7 with advantage ϵ. We will prove that ϵ is negligible through the sequence of indistinguishable games $\mathbf{Game}_0$, $\mathbf{Game}_1$, $\mathbf{Game}_2$, $\mathbf{Game}_3$ and $\mathbf{Game}_4$. The advantage of $\mathcal{A}$ in $\mathbf{Game}_i$ is represented by $\mathrm{Adv}_{\mathcal{A},\mathbf{G}_i}^{anon}(\lambda)$.

$\mathbf{Game}_0$ is a real CPA-anonymous game, and its process is as follows:

1. $\mathcal{C}$ runs the algorithms **GS.Setup** and **GS.KeGen** to get

   $$(mpk, pk) = (\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, \mathbf{y}_1, \cdots, \mathbf{y}_N), msk = sk_{\mathrm{McE}}, \{sk_j\}_{j=1}^{N} = \{(\mathbf{x}_j, \mathbf{e}_j)\}_{j=1}^{N}.$$

   Then, it provides $\mathcal{A}$ with (mpk, pk) and $sk_j, j \in [N]$.
2. $\mathcal{A}$ returns a message M and two challenge indices $j_0, j_1 \in [N]$ to $\mathcal{C}$.

3. $\mathcal{C}$ samples $b \in \{0,1\}$ and returns the signature $\sigma^* = (\mathbf{ct}^*, \Pi^{\mathrm{GS}^*}) \leftarrow \mathrm{Sign}(pk, sk_{j_b})$, where $\mathbf{ct}^* = (\mathbf{z}||\,\mathrm{bin}(j_b))\mathbf{G}_{\mathrm{McE}} + \mathbf{s}$, with $\mathbf{z} \leftarrow \mathbb{F}_2^{\ell_1}$ and $\mathbf{s} \leftarrow B(m,w)$.
4. Finally, $\mathcal{A}$ outputs a bit b which is either 0 or 1.

Therefore, one has $\mathrm{Adv}^{anon}_{\mathcal{A},\mathbf{G}_0}(\lambda) = \epsilon$.

$\mathbf{Game}_1$: Compared to $\mathbf{Game}_0$, the change in $\mathbf{Game}_1$ is that $\mathcal{C}$ uses the simulator of Fig. 6 to generate a signature for message M. Specifically, $\mathcal{C}$ first computes $\mathbf{ct}^*$ honestly just like $\mathbf{Game}_0$. Then, $\mathcal{C}$ runs the simulator κ times with the input $(\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, \mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{ct}^*)$ to generate its proof Π^{GS^*} and programs the corresponding hash function. Finally, $\mathcal{C}$ returns the signature.

Due to the zero-knowledge property of the protocol in Fig. 6, the output Π^{GS^*} of the simulator is indistinguishable from the real output in Fig. 6. Therefore, the signature returned by the challenger in $\mathbf{Game}_1$ is indistinguishable from the signature returned by the challenger in $\mathbf{Game}_0$. So, $\mathbf{Game}_1$ and $\mathbf{Game}_0$ cannot be differentiated by $\mathcal{A}$. Therefore, one has $\mathrm{Adv}^{anon}_{\mathcal{A},\mathbf{G}_1}(\lambda) = \mathrm{Adv}^{anon}_{\mathcal{A},\mathbf{G}_0}(\lambda)$.

$\mathbf{Game}_2$: Compared to $\mathbf{Game}_1$, the change in $\mathbf{Game}_2$ is that $\mathcal{C}$ use a random matrix $\mathbf{G}'$ to replace the matrix $\mathbf{G}_{\mathrm{McE}}$ in the encryption scheme. We will prove that $\mathbf{Game}_2$ and $\mathbf{Game}_2$ are indistinguishable.

Assuming there is an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Game}_1$ and $\mathbf{Game}_2$ with a probability of $1/2 + \epsilon_1$, then we can construct an algorithm $\mathcal{F}_1$ that can solve the DMcE problem with the same probability.

When given an instance $\mathbf{G}' \in \mathbb{F}_2^{\ell \times m}$ of a DMcE problem, $\mathcal{F}_1$ can distinguish whether $\mathbf{G}$ is randomly sampled or generated by the key generation algorithm in the encryption scheme by running the $\mathcal{A}$. The details are as follows:

1. Algorithm $\mathcal{F}_1$ runs the algorithms **GS.Setup** and **GS.KeGen** to get the public and private keys, and replaces the matrix $\mathbf{G}_{\mathrm{McE}}$ in the encryption algorithm with the matrix $\mathbf{G}'$ in the challenge instance.

2. $\mathcal{A}$ returns a message M and two challenge indices $j_0, j_1 \in [N]$ to $\mathcal{C}$.

3. $\mathcal{F}_1$ selects $b \in \{0,1\}$ and computes the signature σ^* as follows:

- Calculate $\mathbf{ct}^* = (\mathbf{z}||\,\mathrm{bin}(j_b))\mathbf{G}' + \mathbf{s}$, where $\mathbf{z} \leftarrow \mathbb{F}_2^{\ell_1}$ and $\mathbf{s} \leftarrow B(m,w)$.

- Run the simulator of Fig. 6 with the input $(\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, \{\mathbf{y}_i\}_{i=1}^N, \mathbf{ct}^*)$ to generate Π^* and program the corresponding hash function. Return $(\mathbf{ct}^*, \Pi^{\mathrm{GS}^*})$.

4. $\mathcal{A}$ outputs $b = 0$ when guessing the process is $\mathbf{Game}_1$ and $b = 1$ when guessing $\mathbf{Game}_2$.

Clearly, if $\mathbf{G}'$ is generated by the key generation algorithm in the encryption scheme, the above process is equivalent to $\mathbf{Game}_1$. If $\mathbf{G}'$ is randomly generated, the above process is equivalent to $\mathbf{Game}_2$. So, the advantage that $\mathcal{A}$ successfully distinguishes $\mathbf{Game}_1$ from $\mathbf{Game}_2$ is equal to the probability that algorithm $\mathcal{F}_1$ breaks the DMcE problem. Therefore, one has $\mathrm{Adv}^{anon}_{\mathcal{A},\mathbf{G}_2}(\lambda) \approx \mathrm{Adv}^{anon}_{\mathcal{A},\mathbf{G}_1}(\lambda)$.

$\mathbf{Game}_3$: Compared to $\mathbf{Game}_2$, the change in $\mathbf{Game}_3$ is in the way $\mathbf{ct}^*$ is generated. Specifically, $\mathbf{ct}^*$ in $\mathbf{Game}_2$ is

$$\mathbf{ct}^* = (\mathbf{z}||\,\mathrm{bin}(j_b))\mathbf{G}_{\mathrm{McE}} + \mathbf{s} = \mathbf{z}\mathbf{G}^1_{\mathrm{McE}} + \mathbf{s} + \mathrm{bin}(j_b)\mathbf{G}^2_{\mathrm{McE}}.$$

where $\mathbf{G}^1_{\mathrm{McE}} \in \mathbb{F}_2^{\ell_1 \times m}, \mathbf{G}^2_{\mathrm{McE}} \in \mathbb{F}_2^{\ell_2 \times m}$ satisfying $\mathbf{G}_{\mathrm{McE}} = \begin{pmatrix} \mathbf{G}^1_{\mathrm{McE}} \\ \mathbf{G}^2_{\mathrm{McE}} \end{pmatrix}$ and $\mathbf{z} \leftarrow \mathbb{F}_2^{\ell_1}, \mathbf{s} \leftarrow B(m,w)$.

In $\mathbf{Game}_3$, we replace $\mathbf{z}\mathbf{G}_{\mathrm{McE}}^1 + \mathbf{s}$ with a random vector $\mathbf{r} \in \mathbb{F}_2^m$. We will prove that $\mathbf{Game}_3$ and $\mathbf{Game}_2$ are indistinguishable.

Assuming there is an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Game}_2$ and $\mathbf{Game}_3$ with a probability of $1/2 + \epsilon_2$, then we can construct an algorithm $\mathcal{F}_2$ that can solve the DLPN problem with the same probability.

When given an instance of the DLPN problem $(\mathbf{A}, \mathbf{w}) \in \mathbb{F}_2^{\ell_1 \times m} \times \mathbb{F}_2^m$, $\mathcal{F}_2$ can distinguish whether $\mathbf{w}$ is randomly sampled or generated by $\mathbf{w} = \mathbf{z}\mathbf{A} + \mathbf{s}$ for $\mathbf{z} \leftarrow \mathbb{F}_2^{\ell_1}$ and $\mathbf{s} \leftarrow B(m, w)$ by running $\mathcal{A}$. The details are as follows:

1. $\mathcal{F}_2$ samples a random matrix $\mathbf{G}_2 \in \mathbb{F}_2^{\ell_2 \times m}$ and runs the algorithms **GS.Setup** and **GS.KeGen** to get the public and private keys. It then replaces the matrix $\mathbf{G}_{\mathrm{McE}}$ in the encryption algorithm with the matrix $\mathbf{G}^* = \begin{pmatrix} \mathbf{A} \\ \mathbf{G}_2 \end{pmatrix}$, where $\mathbf{A}$ is from the challenge instance.

2. $\mathcal{A}$ returns a message M and two challenge indices $j_0, j_1 \in [N]$ to $\mathcal{F}_2$.

3. $\mathcal{F}_2$ selects $b \in \{0, 1\}$ and computes the signature σ^* as follows:

- Calculate $\mathbf{ct}^* = \mathbf{w} + \mathrm{bin}(j_b)\mathbf{G}_2$, where $\mathbf{w}$ is from the challenge instance.

- Run the simulator of Fig. 6 with the input $(\mathbf{G}, \mathbf{G}^*, \{\mathbf{y}_i\}_{i=1}^N, \mathbf{ct}^*)$ to generate Π^{GS^*} and program the corresponding hash function. Return $\sigma^* = (\mathbf{ct}^*, \Pi^{\mathrm{GS}^*})$.

4. $\mathcal{A}$ outputs $b = 0$ when guessing the process is $\mathbf{Game}_2$ and $b = 1$ when guessing $\mathbf{Game}_3$.

Clearly, if $\mathbf{w}$ has the form $\mathbf{w} = \mathbf{z}\mathbf{A} + \mathbf{s}$, where $\mathbf{z} \in \mathbb{F}_2^{\ell_1}$ and $\mathbf{s} \in B(m, w)$, the above process is equivalent to $\mathbf{Game}_2$. If $\mathbf{w}$ is randomly generated, the above process is equivalent to $\mathbf{Game}_3$. So, the advantage that $\mathcal{A}$ successfully distinguishes $\mathbf{Game}_2$ from $\mathbf{Game}_3$ is equal to the probability that algorithm $\mathcal{F}_1$ breaks the DLPN problem. Therefore, one has $\mathrm{Adv}_{\mathcal{A},\mathbf{G}_3}^{anon}(\lambda) \approx \mathrm{Adv}_{\mathcal{A},\mathbf{G}_2}^{anon}(\lambda)$.

$\mathbf{Game}_4$: Compared to $\mathbf{Game}_3$, the change in $\mathbf{Game}_4$ is in the way $\mathbf{ct}^*$ is generated. In this game, $\mathbf{ct}^*$ is randomly sampled from $\mathbb{F}_2^m$. It is important to note that the distribution of $\mathbf{ct}^*$ in $\mathbf{Game}_4$ and $\mathbf{Game}_3$ are identical. Hence, one has $\mathrm{Adv}_{\mathcal{A},\mathbf{G}_4}^{anon}(\lambda) = \mathrm{Adv}_{\mathcal{A},\mathbf{G}_3}^{anon}(\lambda)$. However, $\mathbf{Game}_4$ is independent of the indices j_0, j_1 which implies that $\mathrm{Adv}_{\mathcal{A},\mathbf{G}_4}^{anon}(\lambda) = 0$.

Due to the indistinguishability of the above 5 games and $\mathrm{Adv}_{\mathcal{A},\mathbf{G}_4}^{anon}(\lambda) = 0$, the advantage of $\mathcal{A}$ breaking the CPA-anonymity is negligible. □

Theorem 7. *Our group scheme satisfies full traceability based on the hardness of GSD problem in the random oracle model.*

Proof. Assuming there exists a PPT adversary $\mathcal{A}$ with a probability of ϵ to break the traceability of our group signature, then we can construct an algorithm $\mathcal{B}$ to break the GSD problem with a probability polynomially related to ϵ.

When $\mathcal{B}$ receives a challenge instance $\mathrm{GSD}(n, k, t)$, that is, $(\widetilde{\mathbf{G}}, \widetilde{\mathbf{y}}) \in \mathbb{F}_2^{k \times n} \times \mathbb{F}_2^n$, it performs the following steps:

1. Select a random $\overline{j} \in [N]$.
2. Set $\mathbf{G} = \widetilde{\mathbf{G}}$ and sample $\mathbf{x}_j \leftarrow \mathbb{F}_2^k$, $\mathbf{e}_j \leftarrow B(n, t)$ for $\forall j \in [N], j \neq \overline{j}$. Then, compute pk_j as $\mathbf{y}_j = \mathbf{x}_j\mathbf{G} + \mathbf{e}_j$ for $\forall j \in [N], j \neq \overline{j}$ and set $pk_{\overline{j}}$ as $\mathbf{y}_{\overline{j}} = \widetilde{\mathbf{y}}$.
3. Obtain the key pair $(\mathbf{G}_{\mathrm{McE}}, sk_{\mathrm{McE}})$ by running the algorithm $\mathrm{KeyGen}_{\mathrm{McE}}$.
4. Return $(\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, \mathbf{y}_1, \cdots, \mathbf{y}_N)$ to $\mathcal{A}$.

Clearly, the output of this process is perfect indistinguishable from the output of the real **GS.KeyGen** algorithm for any $\mathcal{A}$. To respond to the inquiry of $\mathcal{A}$, $\mathcal{B}$ first initializes an empty set CU, and then executes the following steps:

1. If $\mathcal{A}$ makes a query to the hash oracle, $\mathcal{B}$ returns a random value from the set $\{1,2,3\}^{\kappa}$. Assuming $\mathcal{A}$ makes t queries to the hash oracle, we use $(\mathbf{r}_1, \cdots, \mathbf{r}_t)$ to denote the result of the querying.

2. If $\mathcal{A}$ makes a query to $\mathbf{Co}(\cdot)$ with an index j such that $j \neq \overline{j}$, $\mathcal{B}$ will return sk_j to $\mathcal{A}$ and add j to the set CU. If $j = \overline{j}$, $\mathcal{B}$ will abort.

3. If $\mathcal{A}$ makes a query to $\mathbf{GS.Sig}(\cdot)$ with an index-message pair (j, M) such that $j \neq \overline{j}$, $\mathcal{B}$ will honestly generate signature σ with secret key $(\mathbf{x}_j, \mathbf{e}_j)$. If $j = \overline{j}$, $\mathcal{B}$ first encrypts the index j^* with $\mathbf{G}_{\mathrm{McE}}$ to get the ciphertext $\mathbf{ct}$. Then, $\mathcal{B}$ uses the simulator of the protocol in Fig. 6 with the input $(\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, \mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{ct})$ to generate proof Π^{GS} , and programs the corresponding hash function. $\mathcal{B}$ then returns the signature $\sigma = (\mathbf{ct}, \Pi^{\mathrm{GS}})$.

Assume that $\mathcal{A}$ outputs a forged signature about message M^* at some point:

$$\begin{cases} \sigma^* = (\mathbf{ct}^*; \{\mathrm{CMT}^i\}_{i=1}^{\kappa}; \{\mathrm{Ch}^i\}_{i=1}^{\kappa}; \{\mathrm{RSP}^i\}_{i=1}^{\kappa}), \\ \mathbf{GS.Verify}(gpk, M^*, \sigma^*) = 1, \\ \mathbf{GS.Open}(sk_{\mathrm{McE}}, M^*, \sigma^*) = j^* \wedge j^* \notin CU. \end{cases}$$

$\mathcal{B}$ employs sk_{McE} to open σ^*. If the opening algorithm fails to return $\overline{j}$, $\mathcal{B}$ aborts. Since no information about the selection of $\overline{j}$ was leaked and the key generation process is perfect indistinguishable from **GS.Setup**, the probability that $\mathcal{B}$ aborts is no more than $(N-1)/N+(2/3)^{\kappa}$, where $(2/3)^{\kappa}$ is the probability of the soundness of protocol being compromised. Therefore, with at least a $1/N-(2/3)^{\kappa}$ probability, the following holds

$$\mathbf{GS.Verify}(gpk, M^*, \sigma^*) = 1 \wedge \mathbf{GS.Open}(sk_{\mathrm{McE}}, M^*, \sigma^*) = \overline{j}. \tag{9}$$

Assuming that Equation (9) holds, $\mathcal{B}$ then runs $\mathcal{A}$ in the following manner. Let Ω represents the tuple $(M^*; \{\mathrm{CMT}^i\}_{i=1}^{\kappa}; \mathbf{ct}^*; \mathbf{G}, \mathbf{G}_{\mathrm{McE}}, R)$. If Ω is not in the query-result list, then the probability of $(\mathrm{Ch}^1, \cdots, \mathrm{Ch}^{\kappa})$ being identical to $\mathcal{H}(\Omega)$ is at most $3^{-\kappa}$. Hence, with probability at least $\epsilon - 3^{-\kappa}$, there exists certain $\eta^* \leq t$ such that Ω was the input of the η^*-th query. Then, $\mathcal{B}$ selects η^* as the target forking point and repeats $\mathcal{A}$ multiple times using the identical random tape and input as in the initial run. During each rerun, for the first η^*-1 queries, $\mathcal{B}$ uses the same values as in the original run as responses, while starting from the η^*-th query, $\mathcal{B}$ uses new random values $r'_{\eta^*}, \cdots, r'_t \leftarrow \{1,2,3\}^{\kappa}$ as responses.

According to the Forking Lemma [36], $\mathcal{B}$ can get a 3-fork regarding the tuple Ω with probability greater than $1/2$ by rerunning $\mathcal{A}$ at most $32Q_{\mathcal{H}}/(\epsilon - 3^{-\kappa})$ times. The responses of $\mathcal{B}$ with regard to the 3-fork branches is denoted by

$$r_{1,\eta^*} = \left(\mathrm{Ch}_1^1, \cdots, \mathrm{Ch}_1^{\kappa}\right); r_{2,\eta^*} = \left(\mathrm{Ch}_2^1, \cdots, \mathrm{Ch}_2^{\kappa}\right); r_{3,\eta^*} = \left(\mathrm{Ch}_3^1, \cdots, \mathrm{Ch}_3^{\kappa}\right).$$

After a simple probability calculation, one immediately obtains

$$\Pr\left[\exists \theta \in \{1, \cdots, \kappa\} : \left\{\mathrm{Ch}_1^{\theta}, \mathrm{Ch}_2^{\theta}, \mathrm{Ch}_3^{\theta}\right\} = \{1,2,3\}\right] = 1 - (7/9)^{\kappa}.$$

Therefore, $\mathcal{B}$ get three responses RSP_1^θ, RSP_2^θ and RSP_3^θ with a probability of $1-(7/9)^\kappa$, and they are the responses to three different challenges corresponding to the same CMT^θ. Then, using the extractor of the protocol in Fig. 6, $\mathcal{B}$ efficiently extract the tuple $(j', \mathbf{x}', \mathbf{e}', \mathbf{z}', \mathbf{s}') \in [N] \times \mathbb{F}_2^k \times \mathbb{F}_2^n \times \mathbb{F}_2^{\ell_1} \times \mathbb{F}_2^m$ satisfying:

$$\begin{cases} \mathbf{x}'\mathbf{G} + \mathbf{e}' = \mathbf{y}_{j'} \wedge \mathbf{x}' \in \mathbb{F}_2^k, \mathbf{e}' \in B(n,t), \\ (\mathbf{z}' || \operatorname{bin}(j'))\mathbf{G}_{\mathrm{McE}} + \mathbf{s}' = \mathbf{ct}^* \wedge \mathbf{z}' \in \mathbb{F}_2^{\ell_1}, \mathbf{s}' \in B(m,w). \end{cases} \tag{10}$$

On one hand, due to Equation (10) and the correctness of our group signature scheme, the output of the opening algorithm is j'. On the other hand, since Equation (9) holds, we have $\mathbf{GS.Open}(sk_{\mathrm{McE}}, M^*, \sigma^*) = \overline{j}$, which implies $j' = \overline{j}$. Therefore, we have $\mathbf{x}'\mathbf{G} + \mathbf{e}' = \mathbf{x}'\widetilde{\mathbf{G}} + \mathbf{e}' = \mathbf{y}_{j'} = \widetilde{\mathbf{y}}$, and $\mathbf{e}' \in B(n,t)$. In other words, $\mathcal{B}$ successfully finds a valid solution to the challenge instance $(\widetilde{\mathbf{G}}, \widetilde{\mathbf{y}})$.

In summary, if $\mathcal{A}$ can break the traceability of our group signature scheme with a non-negligible probability ϵ within time T, then $\mathcal{B}$ can solve GSD problem with a non-negligible probability $1/2\,(1/N-(2/3)^\kappa)\,(1-(7/9)^\kappa)$ within time $32 \cdot T \cdot Q_{\mathcal{H}} / (\epsilon - 3^{-\kappa})$. It contradicts the hardness of GSD problem. □

GS.Setup(1^λ):
- $\mathbf{G} \leftarrow \mathbb{F}_2^{k\times n}$, where $n = \mathcal{O}(\lambda)$ and $k = \mathcal{O}(\lambda)$.
- $(\mathbf{G}_{\mathrm{McE}}, sk_{\mathrm{McE}}) \leftarrow \mathrm{KeyGen}_{\mathrm{McE}}(1^\lambda)$, where $\mathbf{G}_{\mathrm{McE}} \in \mathbb{F}_2^{\ell\times m}$.
- $\mathcal{H}_{com} \xleftarrow{\$} \mathcal{H}(1^\lambda)$, $\mathcal{H}_{FS} \xleftarrow{\$} \mathcal{H}(1^\lambda, \kappa)$, $\mathcal{H}_{Merkle} \xleftarrow{\$} \mathcal{H}(1^\lambda, N)$ where $\kappa = \omega(\log(\lambda))$.
- Return $pp = (\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, \mathcal{H}_{FS}, \mathcal{H}_{com})$.

GS.KeGen(pp):
- $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^k, \mathbf{e} \xleftarrow{\$} \mathrm{B}(n,t)$, $\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e}$.
- Return $(pk, sk) = (\mathbf{y}, (\mathbf{x}, \mathbf{e}))$.

GS.Sign$(sk_j, M, R = (pk_1, \cdots, pk_N))$:
- $\mathbf{z} \xleftarrow{\$} \mathbb{F}_2^\ell, \mathbf{s} \xleftarrow{\$} B(m,w)$.
- $\mathbf{ct} = \mathrm{Enc}_{\mathrm{McE}}(M, (\mathbf{z}, \mathbf{s})) = (\mathbf{z}||\operatorname{bin}(j))\mathbf{G}_{\mathrm{McE}} + \mathbf{s}$.
- **for** $i = 1$ to κ : $\mathrm{CMT}_i \leftarrow \mathcal{P}_1^{\mathrm{GS}}(\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, \mathbf{ct}, R, sk_j)$.
- $\mathrm{CMT} \leftarrow (\mathrm{CMT}_1, \cdots, \mathrm{CMT}_\kappa)$.
- $\mathbf{ch} \leftarrow \mathcal{H}_{FS}(M, R, \mathrm{CMT})$.
- **for** $i = 1$ to κ : $\mathrm{RSP}_i \leftarrow \mathcal{P}_2^{\mathrm{GS}}(\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, \mathbf{ct}, R, \mathbf{ch}_i, sk_j)$.
- $\mathrm{RSP} \leftarrow (\mathrm{RSP}_1, \cdots, \mathrm{RSP}_\kappa)$.
- Return $\sigma = (\mathbf{ct}, \mathrm{CMT}, \mathrm{RSP})$.

GS.Verify(R, M, σ):
- $(\mathbf{ct}, \mathrm{CMT}, \mathrm{RSP}) \leftarrow \sigma$.
- $\mathbf{ch} = \mathcal{H}_{FS}(M, R, \mathrm{CMT}, \mathbf{ct})$.
- Return $\mathcal{V}_2^{\mathrm{GS}}\big(\mathbf{G}, \mathbf{G}_{\mathrm{McE}}, R, (\mathrm{CMT}, \mathbf{ch}, \mathrm{RSP}), \mathbf{ct}\big)$.

GS.Open$(sk_{\mathrm{McE}}, M, \sigma)$:
- $(\mathbf{ct}, \mathrm{CMT}, \mathrm{RSP}) \leftarrow \sigma$.
- If $\mathrm{Dec}_{\mathrm{McE}}(sk_{\mathrm{McE}}, \mathbf{ct})$ fails, return $\perp$, else return $j \leftarrow \mathrm{Dec}_{\mathrm{McE}}(sk_{\mathrm{McE}}, \mathbf{ct})$.

Fig. 7. Our group signature scheme.

6 Concrete instantiation

We choose parameter sets for our ring signature and group signature schemes under the 128-bit security level. The parameters are set as follows:

1. The parameters (n, k, t) for GSD problem and the parameters (m, ℓ, w) for McEliece encryption scheme are set to achieve the 128-bit security level.
2. To ensure the one-wayness of the public and private keys, the parameters (n, k, t, N) for GDOOM problem 4 are set to achieve the 128-bit security level.
3. The repetition number κ of the protocols in Fig. 4 and Fig. 6 is set to 220 in order to achieve soundness error 2^{-128}.
4. We use cSHAKE to instantiate the hash functions in our scheme and the Merkle tree, as well as the pseudorandom number generator in the seedtree [25].
5. The signature size of our ring signature scheme is equal to the proof size in Fig. 4, i.e. $2\lambda(\frac{5\kappa}{6}+2)+\frac{\kappa}{3}(\frac{20\lambda}{3}+\frac{3n}{2}+k+2\lambda\log N)$.
6. The signature size of our group signature scheme is equal to the proof size in Fig. 6 plus the size of $\mathbf{ct}$, i.e. $2\lambda(\frac{5\kappa}{6}+2)+\frac{\kappa}{3}(\frac{20\lambda}{3}+\frac{3n}{2}+k+\frac{3m}{2}+\ell+2\lambda\log N)+m$.

We set $(n, k, t) = (1280, 640, 132), (1300, 650, 135), (1360, 680, 141)$ for $N = 2^6, 2^{12}, 2^{21}$, respectively, and $(m, \ell, w) = (3488, 2720, 64)$ as in [12]. Then, we present the signature sizes of our ring signature scheme and group signature scheme under different N in Table 3 and Table 4.

Table 3. Ring signature sizes for N.

N	(user) PK size	Signature Size
2^6	0.240KB	51KB
2^{12}	0.247KB	65KB
2^{21}	0.255KB	87KB

Table 4. Group signature sizes for N.

N	(user) PK size	Signature Size
2^6	0.240KB	112KB
2^{12}	0.247KB	126KB
2^{21}	0.255KB	148KB

References

1. Alamélou, Q., Blazy, O., Cauchie, S., Gaborit, P.: A code-based group signature scheme. Designs, Codes and Cryptography **82**, 469–493 (2017)
2. Assidi, H., Ayebie, E.B., Souidi, E.M.: An efficient code-based threshold ring signature scheme. Journal of information security and applications **45**, 52–60 (2019)
3. Ayebie, E.B., Souidi, E.M.: New code-based cryptographic accumulator and fully dynamic group signature. Designs, Codes and Cryptography **90**(12), 2861–2891 (2022)
4. Barenghi, A., Biasse, J.F., Ngo, T., Persichetti, E., Santini, P.: Advanced signature functionalities from the code equivalence problem. International Journal of Computer Mathematics: Computer Systems Theory **7**(2), 112–128 (2022)
5. Bettaieb, S., Bidoux, L., Blazy, O., Gaborit, P.: Zero-knowledge reparation of the véron and AGS code-based identification schemes. In: ISIT 2021. pp. 55–60. IEEE (2021), https://doi.org/10.1109/ISIT45174.2021.9517937

6. Beullens, W., Katsumata, S., Pintore, F.: Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12492, pp. 464–492. Springer (2020), https://doi.org/10.1007/978-3-030-64834-3_16
7. Bidoux, L., Gaborit, P., Kulkarni, M., Sendrier, N.: Quasi-cyclic stern proof of knowledge. In: ISIT 2022. pp. 1459–1464. IEEE (2022), https://doi.org/10.1109/ISIT50566.2022.9834642
8. Blazy, O., Gaborit, P., Mac, D.T.: A rank metric code-based group signature scheme. In: Wachter-Zeh, A., Bartz, H., Liva, G. (eds.) CBCrypto 2021. LNCS, vol. 13150, pp. 1–21. Springer (2021), https://doi.org/10.1007/978-3-030-98365-9_1
9. Branco, P., Mateus, P.: A code-based linkable ring signature scheme. In: Baek, J., Susilo, W., Kim, J. (eds.) ProvSec 2018. LNCS, vol. 11192, pp. 203–219. Springer (2018), https://doi.org/10.1007/978-3-030-01446-9_12
10. Branco, P., Mateus, P.: A traceable ring signature scheme based on coding theory. In: Ding, J., Steinwandt, R. (eds.) PQCrypto 2019. LNCS, vol. 11505, pp. 387–403. Springer (2019), https://doi.org/10.1007/978-3-030-25510-7_21
11. Cayrel, P., Véron, P., Alaoui, S.M.E.Y.: A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 171–186. Springer (2010), https://doi.org/10.1007/978-3-642-19574-7_12
12. Chen, L., Moody, D., Liu, Y.K.: Post-quantum cryptography round 4 submissions. NIST (2022), https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions
13. Chou, T., Niederhagen, R., Persichetti, E., Randrianarisoa, T.H., Reijnders, K., Samardjiska, S., Trimoska, M.: Take your MEDS: digital signatures from matrix code equivalence. In: Mrabet, N.E., Feo, L.D., Duquesne, S. (eds.) AFRICACRYPT 2023. LNCS, vol. 14064, pp. 28–52. Springer (2023), https://doi.org/10.1007/978-3-031-37679-5_2
14. Dinur, I., Nadler, N.: Multi-target attacks on the picnic signature scheme and related protocols. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 699–727. Springer (2019), https://doi.org/10.1007/978-3-030-17659-4_24
15. Döttling, N.: Cryptography based on the Hardness of Decoding. Ph.D. thesis, Karlsruhe, Karlsruher Institut für Technologie (KIT), Diss., 2014 (2014)
16. Esgin, M.F., Steinfeld, R., Sakzad, A., Liu, J.K., Liu, D.: Short lattice-based one-out-of-many proofs and applications to ring signatures. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 67–88. Springer (2019), https://doi.org/10.1007/978-3-030-21568-2_4
17. Ezerman, M.F., Lee, H.T., Ling, S., Nguyen, K., Wang, H.: A provably secure group signature scheme from code-based assumptions. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 260–285. Springer (2015), https://doi.org/10.1007/978-3-662-48797-6_12
18. Feneuil, T., Joux, A., Rivain, M.: Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. LNCS, vol. 13508, pp. 541–572. Springer (2022), https://doi.org/10.1007/978-3-031-15979-4_19
19. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO '86. LNCS, vol. 263, pp. 186–194. Springer (1986), https://doi.org/10.1007/3-540-47721-7_12

20. Gaborit, P., Schrek, J., Zémor, G.: Full cryptanalysis of the chen identification protocol. In: Yang, B. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 35–50. Springer (2011), https://doi.org/10.1007/978-3-642-25405-5_3
21. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 253–280. Springer (2015), https://doi.org/10.1007/978-3-662-46803-6_9
22. Jain, A., Krenn, S., Pietrzak, K., Tentes, A.: Commitments and efficient zero-knowledge proofs from learning parity with noise. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 663–680. Springer (2012), https://doi.org/10.1007/978-3-642-34961-4_40
23. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) CCS 2018. pp. 525–537. ACM (2018), https://doi.org/10.1145/3243734.3243805
24. Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 372–389. Springer (2008), https://doi.org/10.1007/978-3-540-89255-7_23
25. Kelsey, J., Chang, S.j., Perlner, R.: Sha-3 derived functions: cshake, kmac, tuplehash, and parallelhash. NIST special publication **800**, 185 (2016)
26. Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 107–124. Springer (2013), https://doi.org/10.1007/978-3-642-36362-7_8
27. Lyubashevsky, V., Nguyen, N.K.: BLOOM: bimodal lattice one-out-of-many proofs and applications. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022. LNCS, vol. 13794, pp. 95–125. Springer (2022), https://doi.org/10.1007/978-3-031-22972-5_4
28. McEliece, R.J.: A public-key cryptosystem based on algebraic. Coding Thv **4244**, 114–116 (1978)
29. Melchor, C.A., Cayrel, P.L., Gaborit, P., Laguillaumie, F.: A new efficient threshold ring signature scheme based on coding theory. IEEE Transactions on Information Theory **57**(7), 4833–4842 (2011)
30. Melchor, C.A., Gaborit, P., Schrek, J.: A new zero-knowledge code based identification scheme with reduced communication. In: ITW 2011. pp. 648–652. IEEE (2011), https://doi.org/10.1109/ITW.2011.6089577
31. Meurer, A.: A coding-theoretic approach to cryptanalysis. Ph.D. thesis, Verlag nicht ermittelbar (2013)
32. Morozov, K., Roy, P.S., Sakurai, K.: On unconditionally binding code-based commitment schemes. In: IMCOM 2017. p. 101. ACM (2017), https://doi.org/10.1145/3022227.3022327
33. Morozov, K., Takagi, T.: Zero-knowledge protocols for the mceliece encryption. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 180–193. Springer (2012), https://doi.org/10.1007/978-3-642-31448-3_14
34. Nguyen, K., Tang, H., Wang, H., Zeng, N.: New code-based privacy-preserving cryptographic constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11922, pp. 25–55. Springer (2019), https://doi.org/10.1007/978-3-030-34621-8_2
35. Nojima, R., Imai, H., Kobara, K., Morozov, K.: Semantic security for the mceliece cryptosystem without random oracles. Designs, Codes and Cryptography **49**, 289–305 (2008)

36. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of cryptology **13**, 361–396 (2000)
37. Sendrier, N.: Decoding one out of many. In: Yang, B. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 51–67. Springer (2011), https://doi.org/10.1007/978-3-642-25405-5_4
38. Stern, J.: A new paradigm for public key identification. IEEE Transactions on Information Theory **42**(6), 1757–1768 (1996)
39. Véron, P.: Improved identification schemes based on error-correcting codes. Applicable Algebra in Engineering, Communication and Computing **8**, 57–69 (1997)