Code-based Fully Dynamic Accountable Ring Signatures and Group Signatures using the Helper Methodology

Rishiraj Bhattacharyya[1], Sreehari Kollath[1], Christophe Petit[1,2]

[1] University of Birmingham
rishiraj.bhattacharyya@gmail.com,sreeharikollath@gmail.com
[2] Université Libre de Bruxelles christophe.petit@ulb.be

Abstract. In this paper, we propose an efficient code-based accountable ring signature scheme based on the *proof with helper* paradigm. Our main technical contribution is a zero-knowledge OR proof based on the helper protocol of Beullens (Eurocrypt 2020). Combined with the Niederreiter encryption scheme, we leverage the framework of Beullens *et al* (Eurocrypt 2022) to construct our accountable ring signature. We apply our construction to develop a code-based dynamic group signature scheme that simultaneously achieves a logarithmic signature size and the strongest security notion for full anonymity and unforgeability. Our construction results in a relatively short group signatures—just 51kB for a group of 2^{20} members at the 128-bit security level—while maintaining the strongest security guarantees. In comparison, two recent code-based group signature schemes proposed at PKC 2024 and Asiacrypt 2024 achieve signature sizes of 146kB and 124kB, respectively, for the same number of users.

1 Introduction

An Accountable Ring Signature (ARS) [14] is a special type of digital signature that allows a user to sign a message on behalf of a group while remaining anonymous—but with an accountability mechanism. Unlike standard ring signatures, where the signer's identity remains permanently hidden, an ARS includes a built-in accountability feature. This means that a trusted authority (called an opener) can reveal the real signer's identity if necessary. The first formalized ARS scheme was proposed by Xu *et al* [36], introducing a designated authority capable of identifying the actual signer in cases of misuse. In [7], Beullens *et al* proposed a generic construction of accountable ring signatures based on group actions. They instantiated their construction in both lattice-based and isogeny-based settings.

A group signature scheme [16] allows a user to anonymously sign messages on behalf of a group of users. The membership of the group is managed by a group manager, who has the ability to trace a signature to its signing member. The topic of group signatures has been a tremendously popular research topic

with a plethora of works (cf. [12,26,9,2]) appearing in the last thirty years. We have seen many post-quantum group signature schemes as well. Starting from [20], notable lattice-based constructions include [17,23,25,24]. Group signatures can be built from accountable ring signature in a generic fashion [14]. Following this construction, Beullens *et al* [7] also obtain group signatures based on group actions from isogenies and lattices.

In comparison, the number of *code-based* accountable ring signature and group signature proposals has been scarce. The earliest code-based group signature proposal was [17]. Almost at the same time, [1] proposed a novel group signature scheme. Recently, Liu and Wang proposed a code-based group signature scheme with logarithmic signature size [27]. These constructions use Stern's protocol [34] as a building block, which suffers from a high soundness error and assumes that the group manager is completely honest. More recently, some code-based group signature schemes [31,35] were proposed which satisfies the strongest (and arguably standard by now) security criteria proposed by Bootle *et al* [13] where even the group manager is held accountable.

One can imagine combining Stern's protocol with the technique of [7] to achieve stronger security properties. In that case, however, we would still loose the efficiency as the resulting sigma protocol will have a soundness error of 2/3, as inherent in Stern's protocol. We ask the following question.

Can we construct efficient code-based accountable ring signatures while also achieving a reduced soundness error?

In this paper, we answer the question positively. Our contributions are three-fold,

1. We construct a *traceable* OR proof based on the helper paradigm [6] leading to a sigma protocol with low soundness error.
2. We employ the techniques of [36] (coupling a randomness-recoverable IND-CPA secure encryption scheme with an OR proof) to build an **accountable ring signature** using our new code-based OR proof and Niederreiter encryption scheme [29].
3. We propose an efficient code-based group signature.

To the best of our knowledge, ours is the first accountable ring signature proposal using the helper paradigm. Following the roadmap of [14] of restricting accountable ring signatures to get group signatures, we get a code-based group signature achieving the strongest properties as in [13].

Overview of our construction. The signature generated by our construction (following the framework of encrypt-then-prove approach [15] has two components, a) a ciphertext encrypting an efficiently invertible function of the user index I with the opener's public key, and b) a proof of knowledge of the secret key of the corresponding public key of *one-out-of-N* group members. In order to trace the signer, the opener can decrypt the ciphertext to find the index I. To establish the correctness of the tracing to a judge, the opener will recover the

randomness used in the encryption[3], and provide a proof of knowledge relating the ciphertext to the user index.

The framework of [7] requires a perfectly correct IND-CPA secure encryption and a Non-Interactive Zero-Knowledge (NIZK) proof of an "OR protocol" that they introduced with online extractability (of a witness). The difficulty of employing this framework in the code-based setting comes from the complexity of constructing an *efficient* NIZK with online-extractability[4]. A detailed analysis of the [7] approach reveals that the security of their online extractor is closely linked to the (special) soundness error. In the case of Stern-based schemes, the soundness error of one round is 2/3, which is prohibitively high.

Our main technical contribution is to solve this problem via constructing an OR proof under the helper framework, which significantly reduces the soundness error ($1/K$ for a large constant K bounded above by the group size N, and the underlying field size q). The helper framework allows one, with the knowledge of the seed, to extract a valid witness for every possible value of the challenge. Thus, we side-step any complexity arising from a possible challenge value where the response could be simulated without knowing the secret. The only thing we now need to prove for online extractability is that a random seed can (almost) always be extracted from random oracle queries while constructing a valid proof; an event whose probability can be bounded by the unpredictability of the Random Oracle outputs.

Related works. As previously mentioned, we follow the framework from [7] for constructing an Accountable Ring Signature (ARS). Their ARS and the resulting group signature rely on a simple, generic construction based on cryptographically hard group actions. Recently, [35] proposed the first fully dynamic group signature that satisfies the standard security notions from [13] while supporting dynamic membership, allowing users to join and leave the group at any time. Another notable work in this area was presented in [31], where the authors introduced new proofs of knowledge and several privacy-preserving systems based on codes. Their approach leverages efficient code-based zero-knowledge protocols using the VOLE-in-the-Head technique [3], a variant of the MPC-in-the-Head approach [22]. In comparison, we construct an intermediate Sigma protocol with a helper, as introduced by [6], to develop an accountable ring signature, which we then extend to a group signature as its application. Our resulting group signature achieves a significantly smaller signature size, being less than half of the sizes proposed in [3,27] for a group of 2^{20} users.

2 Preliminaries

For a finite set S, we write $|S|$ for the size of S. We write $x \xleftarrow{\$} S$ for the process of choosing x uniformly at random from S. We write $[1, n]$ for the set $\{1, \ldots, n\}$.

[3] This requires that the decryption algorithm is randomness-recovering, e.g. OAEP, Padding-based encryption schemes.

[4] The witness-extraction via rewinding (a proof technique for soundness) does not imply full anonymity [5].

For a deterministic (resp. probabilistic) algorithm $\mathcal{A}$, $y \leftarrow \mathcal{A}(x)$ (resp $y \xleftarrow{\$} \mathcal{A}(x)$) denotes y is the (resp. uniformly sampled) output of $\mathcal{A}$ on input x. For the set S, $x \xleftarrow{\$,\theta} S$ denotes that x is sampled uniformly at random from S using the seed θ. We define $\approx$ as the symbol representing approximation in our notation. We define the weight of a vector $y \in \mathbb{F}_q^n$ as the number of non-zero coordinates in the vector and denote it as $|y|$. We write $\mathcal{A}^{\mathcal{O}}$ to denote that $\mathcal{A}$ has access to $\mathcal{O}$ as an oracle. The advantage of an algorithm $\mathcal{A}$ in game G is defined by $\mathbf{Adv}_{\mathcal{A},G} \stackrel{def}{=} \Pr[G^{\mathcal{A}} = 1]$. Define $S(n,\omega)$ and $\tilde{S}(n,\omega)$ as,

$$S(n,\omega) \stackrel{def}{=} \{x \in \mathbb{F}_2^n \mid |x| = \omega\} \qquad \tilde{S}(n,\omega) \stackrel{def}{=} \{x \in \mathbb{F}_2^n \mid |x| \leq \omega\}$$

A PPT (Probabilistic Polynomial Time) algorithm is an algorithm that runs in polynomial time and has access to a source of randomness, meaning its execution may produce different outputs for the same input due to probabilistic choices.

2.1 Binary Linear Code and Syndrome Decoding Problem

For two positive integers k, n with $k < n$ and a prime q, a linear code C is a k-dimensional subspace of $\mathbb{F}_q^n$. The dual code $C^\perp$ is the orthogonal complement of C. The code is generated using a parity check matrix $\mathsf{H} \in \mathbb{F}_q^{(n-k)\times n}$ whose rows form a basis of the dual code $C^\perp$; $C = \{y \in \mathbb{F}_q^n \mid \mathsf{H}y^T = 0\}$. The parameters k and n are called the dimension and the length of the code respectively.

Syndrome Decoding Problem ($\mathsf{SDP}(\mathsf{H}, t, \omega)$): Given positive integers n, k, ω, q, a random parity check matrix $\mathsf{H} \in \mathbb{F}_q^{(n-k)\times n}$, and a vector (also called syndrome) $t \in \mathbb{F}_q^{(n-k)}$, the syndrome decoding problem ($\mathsf{SDP}(\mathsf{H}, t, \omega)$) asks to find a vector $y \in \mathbb{F}_q^n$ such that $\mathsf{H}y^T = t$ satisfying $|y| \leq \omega$ if it exists. We denote $\mathsf{SDP}(\mathsf{H}, \omega)$ as the problem of finding a solution to a random syndrome, assuming that a solution exists.

Although the general decoding problem for binary linear codes is NP-complete [4] and conjectured to be hard on random instances, some well-known families of linear codes admit efficient decoders. Code-based cryptosystems use one such code for decryption, typically quasi-cyclic codes or Goppa codes, in a disguised form. In our constructions, we work with binary codes, taking $q = 2$ as the finite field size.

Niederreiter Encryption [29]: We recall the classical version of the Niederreiter public-key encryption scheme.

- Keygen: Let $n, k, \omega \in \mathbb{Z}^+$. Let H be an $(n-k) \times n$ parity-check matrix for a $[n, k]$ linear code C with a fast decoding algorithm $\mathtt{D}$ that can correct up to ω errors. Generate an $(n-k) \times (n-k)$ binary non-singular matrix S, and an $(n \times n)$ permutation matrix P. Then compute $\mathsf{H}' = \mathsf{SHP}$ and set
  - Public Key: H'
  - Private Key: $(\mathsf{S}, \mathsf{H}, \mathsf{P})$
- $\mathsf{Encrypt}(\mathsf{H}', m)_{m \in \mathbb{F}_2^n, |m| = \omega}$: Output the ciphertext, $c^T = \mathsf{H}'m^T$
- $\mathsf{Decrypt}((\mathsf{S}, \mathsf{H}, \mathsf{P}), c)$: Compute $y = \mathsf{S}^{-1}c^T = \mathsf{HP}m^T$, apply a syndrome decoding algorithm D, to recover $z^T = \mathsf{P}m^T$ and retrieve the message by computing $\mathsf{P}^{-1}z = m^T$.

Sigma Protocol [19] A Sigma protocol for a binary NP relation R is an interactive protocol consisting of three PPT algorithms $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$, in which the prover's input is $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$ and the verifier's input is x. Here x, w are called the statement and the witness, respectively. The protocol consists of three messages:

1. The prover sends the first message $\mathsf{u} \leftarrow \mathsf{P}_1(\mathsf{x}, \mathsf{w})$, called a commitment.
2. The verifier chooses a uniformly random challenge $\mathsf{ch} \xleftarrow{\$} \mathsf{C}$ from some finite challenge space C, and sends it as the second message of the protocol.
3. The prover generates a response $\mathsf{z} \leftarrow \mathsf{P}_2(\mathsf{ch})$ and sends it as the third and final message in the protocol.
4. The verifier runs the final verification algorithm $\mathsf{V}(\mathsf{x}, \mathsf{u}, \mathsf{ch}, \mathsf{z})$ and outputs a bit $b \in \{0, 1\}$, where 0 corresponds to reject and 1 corresponds to accept.

The protocol must satisfy the following properties:

1. **Completeness.** A sigma protocol $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ is said to be correct if for any $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$ and for any honest prover $(\mathsf{P}_1, \mathsf{P}_2)$, with $\mathsf{u} \leftarrow \mathsf{P}_1(\mathsf{x}, \mathsf{w})$, $\mathsf{ch} \leftarrow \mathsf{C}$ and $\mathsf{z} \leftarrow \mathsf{P}_2(\mathsf{ch})$, it holds that $\mathsf{V}(\mathsf{x}, \mathsf{u}, \mathsf{ch}, \mathsf{z}) = 1$.
2. **2-special soundness.** A sigma protocol is said to satisfy 2-special soundness if there exists a polynomial-time extraction algorithm Ext, that outputs a witness w such that $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$ given two valid transcripts $(\mathsf{x}, \mathsf{u}, \mathsf{ch}, \mathsf{z})$, $(\mathsf{x}, \mathsf{u}, \mathsf{ch}', \mathsf{z}')$ with $\mathsf{ch} \neq \mathsf{ch}'$.
3. **Special honest-verifier zero-knowledge.** A sigma protocol $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ is computationally special honest-verifier zero-knowledge (sHVZK) [21] if there exists a probabilistic polynomial time simulator S such that for all interactive probabilistic polynomial time adversaries $\mathcal{A}$,

$$\begin{aligned} &\Pr[\mathcal{A}(u, z) = 1 \mid (\mathsf{x}, \mathsf{w}, \mathsf{ch}) \leftarrow \mathcal{A}(1^\lambda), \mathsf{u} \leftarrow \mathsf{P}_1(\mathsf{x}, \mathsf{w}), \mathsf{z} \leftarrow \mathsf{P}_2(\mathsf{ch}, \mathsf{x}, \mathsf{w}, \mathsf{u})] \\ \approx\ &\Pr[\mathcal{A}(u, z) = 1 \mid (\mathsf{x}, \mathsf{w}, \mathsf{ch}) \leftarrow \mathcal{A}(1^\lambda), (\mathsf{u}, \mathsf{z}) \leftarrow \mathsf{S}(\mathsf{x}, \mathsf{ch})] \end{aligned}$$

Sigma Protocol with Helper [6] A sigma protocol with helper for relation R with challenge space C has a trusted third party called the helper who runs a setup algorithm $\mathtt{Setup}$, based on a random seed, seed at the beginning of each execution of the protocol. The helper sends auxiliary information aux to the verifier and seed used in the $\mathtt{Setup}$ algorithm to the prover.

In real-world scenarios, it is not ideal to assume the presence of an honest helper; therefore, the helper should be removed. The idea is to let the prover pick S different seeds $\mathsf{seed}_1, \ldots, \mathsf{seed}_\mathsf{S}$ and generate S sets of auxiliary information $\mathsf{aux}_i = \mathtt{Setup}(\mathsf{seed}_i)$ for $1 \leq i \leq \mathsf{S}$. After this, the prover sends all the auxiliary information aux_i to the verifier, along with the first messages of the protocol u_i for each setup. The verifier picks a random index j and a single challenge $\alpha \in \mathsf{C}$ and sends this to the prover. The prover now reveals the seed values, seed_i for which $i \neq j$. The prover also sends the response, z to the challenge α at index j. Using the seeds, the verifier then checks whether all the auxiliary information $\mathsf{aux}_{i \neq j}$ was generated honestly and checks if z is a correct response to the challenge at index j.

In a standard, interactive, zero-knowledge proof, the prover initially computes a series of commitments, followed by challenges from the verifier, which the prover responds to. Contrastingly, in a *non-interactive* zero-knowledge proof (NIZK), the verifier's role is assumed by a hash function (or a comparable mechanism). Fiat–Shamir heuristic [18] is a technique that allows for taking interactive proof of knowledge and creating a digital signature based on it.

2.2 Accountable Ring Signature

We recall the definitions of accountable ring signatures from [14]. To remain consistent with the literature, we reproduce the following definitions almost verbatim from [7].

Accountable Ring Signature An accountable ring signature Π_{ARS} consists of PPT algorithms (`Setup`, `OKGen`, `UKGen`, `Sign`, `Verify`, `Open`, `Judge`) defined as follows:

1. $\texttt{Setup}(1^\lambda) \to \mathsf{pp}$: On input a security parameter 1^λ, it returns a public parameter pp used by the scheme. We assume pp defines the opener's public key space $\mathcal{K}_{\mathsf{opk}}$ and the user's verification key space $\mathcal{K}_{\mathsf{v}}$, with efficient algorithms to decide membership.
2. $\texttt{OKGen}(\mathsf{pp}) \to (\mathsf{opk}, \mathsf{osk})$: On input a public parameter pp, it outputs a pair of public and secret keys $(\mathsf{opk}, \mathsf{osk})$ for an opener.
3. $\texttt{UKGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$: On input a public parameter pp, it outputs a pair of verification and signing keys $(\mathsf{vk}, \mathsf{sk})$ for a user.
4. $\texttt{Sign}(\mathsf{opk}, \mathsf{sk}, \mathcal{R}, \mathsf{M}) \to \sigma$: On input an opener's public key opk, a signing key sk, a list of verification keys, i.e., a ring, $\mathcal{R} = \{\mathsf{v}_1, \ldots, \mathsf{v}_N\}$, and a message M, it outputs a signature σ.
5. $\texttt{Verify}(\mathsf{opk}, \mathcal{R}, \mathsf{M}, \sigma) \to 1/0$: On input an opener's public key opk, a ring $\mathcal{R} = \{\mathsf{v}_1, \ldots, \mathsf{v}_N\}$, a message M, and a signature σ, it (deterministically) outputs either 1 (accept) or 0 (reject).
6. $\texttt{Open}(\mathsf{osk}, \mathcal{R}, \mathsf{M}, \sigma) \to (\mathsf{v}, \pi)/0$: On input an opener's secret key osk, a ring $\mathcal{R} = \{\mathsf{v}_1, \ldots, \mathsf{v}_N\}$, a message M, a signature σ, it (deterministically) outputs either a pair of verification key v and a proof π that the owner of v produced the signature, or 0.
7. $\texttt{Judge}(\mathsf{opk}, \mathtt{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi) \to 1/0$: On input an opener's public key opk, a ring $\mathcal{R} = \{\mathsf{v}_1, \ldots, \mathsf{v}_N\}$, a verification key v, a message M, a signature σ, and a proof π, it (deterministically) outputs either 1 (accept) or 0 (reject). We assume without loss of generality that $\texttt{Judge}(\mathsf{opk}, \mathcal{R}, \mathsf{v}, \mathsf{M}, \sigma, \pi)$ outputs 0 if $\texttt{Verify}(\mathsf{opk}, \mathcal{R}, \mathsf{M}, \sigma)$ outputs 0.

Security Properties An accountable ring signature is required to satisfy the following properties: correctness, anonymity, traceability, unforgeability, and tracing soundness.

1. The correctness of a group signature scheme ensures that any valid group member can create a signature that will be accepted as valid by the verifier.

2. Anonymity ensures that a signature does not reveal the identity of the group member who generated it.
3. Unforgeability considers two types of forgeries. The first captures the natural notion of unforgeability where an adversary cannot forge a signature for a ring of honest users, i.e., a ring of users for which it does not know any of the corresponding secret keys. The second captures the fact that an adversary cannot accuse an honest user of producing a signature even if the ring contains malicious users and the opener is malicious.
4. Traceability requires that any opener key pair $(\mathsf{opk}, \mathsf{osk})$ in the range of the opener key generation algorithm can open a valid signature σ to some user verification key v along with a valid proof π, establishing the identity of the user.
5. Tracing soundness requires that a signature cannot be traced to two different users in the ring.
6. Online extractability ensures an extraction algorithm that can always extract a witness with negligible probability of failure when provided with a valid proof π and the list of random oracle queries made by an adversary.

For a more formal definition we refer to [14][7].

2.3 Other Tools

We use the notion of Seed Trees and Index Hiding Merkle Trees. Due to space limitation, we refer the reader to [8] for details.

2.4 From Accountable Ring Signature to Group Signature

An accountable ring signature can be converted into a dynamic group signature. The opener in an accountable ring signature can also act as the group manager in a group signature scheme. The group manager has the power to reveal the identity of the signer if needed, making it responsible for both accountability and the issuance of signing credentials. There is a natural reduction between the properties of accountable ring signatures to those of the group signatures. For a more detailed description, see [13].

3 PoK for *one-out-of-N* SD Problem

3.1 Gaborit and Bidoux PoK for SD Problem[10]

Let n, k, ω be positive integers, and let $\mathsf{H} \in \mathbb{F}_2^{(n-k)\times n}$ be a parity-check matrix. Consider the equation $y = \mathsf{H}e^T$, where $e \in S(n, \omega)$. Stern's protocol [34] enables a prover to demonstrate knowledge of e such that $y = \mathsf{H}e^T$ with $|e| = \omega$.

In Stern's protocol, the prover selects a random vector $u \in \mathbb{F}_2^n$ and a permutation $\pi \in S_n$ to conceal the secret e. In [10], the authors introduced an alternative proof of knowledge for the syndrome decoding problem using a helper. Their approach involves generating multiple permutations π_i (for $i \in [1, K]$) and revealing

all but one to demonstrate knowledge of a solution to a permuted instance of the syndrome decoding problem, represented by s_i (for $i \in [1, K]$). Additionally, the vector $u \in \mathbb{F}_2^n$ is used to mask the secret vector, while $t_i \in \mathbb{F}_2^n$ is introduced to mask $\pi_i(s_{i-1})$. The detailed steps of the protocol are presented in Figure 1.

Lemma 1. *[10] If the hash function* Hash *is a random oracle, then the protocol depicted in Figure 1 is an honest-verifier zero-knowledge proof of knowledge with Helper having soundness error* $1/K$ *assuming the hardness of the syndrome decoding problem,* $\mathsf{SDP}(\mathsf{H}, \mathsf{y}, \omega)$.

3.2 PoK for *one-out-of-N* SD Problem

Let $n_1, n_2, k_1, k_2, \omega_1, \omega_2, N \in \mathbb{Z}^+$. Let $\mathsf{H}_1 \in \mathbb{F}_2^{(n_1-k_1)\times n_1}$ and $\mathsf{H}_2 \in \mathbb{F}_2^{(n_2-k_2)\times n_2}$ be two parity-check matrices. Define

$$x_i = \mathsf{H}_1 d_i^T, y_i = \mathsf{H}_2 e_i^T \quad \text{where } d_i \in S(n_1, \omega_1), e_i \in S(n_2, \omega_2)$$

for $i \in [1, N]$. Assume that the prover knows the solution to *one-out-of-N* syndromes with respect to both H_1 and H_2 with the same index. That is, we assume the prover knows the values of d_I and e_I for a certain index $I \in [1, N]$. We now outline a proof of knowledge for the joint *one-out-of-N* syndrome decoding problem in the set $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ without revealing the particular index.

We construct the proof of knowledge using the proof of knowledge for the syndrome decoding (SD) problem from [10] by using an index-hiding Merkle tree. The helper first uses the pseudorandom generator (PRG) to generate two seed values, $(\mathsf{seed}_1, \mathsf{seed}_2)$, from the master seed seed. The helper then uses seed_1 and seed_2 to uniformly sample

$$a^{(1)} \in S(n_1, \omega_1) \quad \text{and} \quad a^{(2)} \in S(n_2, \omega_2),$$

respectively. Next, the helper computes the masked values:

$$x'_i = x_i + \mathsf{H}_1 (a^{(1)})^T, \quad y'_i = y_i + \mathsf{H}_2 (a^{(2)})^T, \quad \text{for } i \in [1, N].$$

The helper then commits to these values and constructs an index-hiding Merkle tree with root root, where the leaves are defined as

$$\mathsf{C}_i = \mathsf{Hash}(\mathsf{bits}_i \parallel x'_i \parallel y'_i),$$

for $i \in [1, N]$, with bits_i being uniformly sampled from $\{0, 1\}^\lambda$.

Now, assume that the prover knows the secrets d_I and e_I corresponding to the I^{th} instances of the syndrome decoding problems. By design, the weights of $d_I + a^{(1)}$ and $e_I + a^{(2)}$ are at most $2\omega_1$ and $2\omega_2$, respectively.

Using the proof in Figure 1, the prover now reveals (x'_I, y'_I), bits_I, and the Merkle tree path path corresponding to root with respect to C_I. The prover then proceeds to demonstrate knowledge of the secrets $z^{(1)}$ and $z^{(2)}$ such that

$$\mathsf{H}_1 (z^{(1)})^T = x'_I, \quad \text{with } |z^{(1)}| \leq 2\omega_1,$$

Private Data: $sk = (e \mid \mathsf{H}e^{T} = y, \text{ and } \mid e \mid = \omega)$

Public Data: $pk = (\mathsf{H} \in \mathbb{F}_2^{(n-k)\times n}, \omega, \text{ a hash function } \mathsf{Hash} : \{0,1\}^{\star} \to \{0,1\}^{\lambda}, K \in \mathbb{Z}^{+}, y)$

Round 0, Helper $-\ \mathsf{SDHelper}(pk, \mathsf{seed})$

$\mathsf{seed} \xleftarrow{\$} \{0,1\}^{\lambda}$

$(\theta, \epsilon) \in (\{0,1\}^{\lambda})^2 \leftarrow \mathsf{Hash}(\mathsf{seed})$

for $i \in [1, K]$

$\theta_i \xleftarrow{\$,\theta} \{0,1\}^{\lambda}, \quad \phi_i \xleftarrow{\$,\theta_i} \{0,1\}^{\lambda}$

$\pi_i \xleftarrow{\$,\phi_i} S_n, \quad t_i \xleftarrow{\$,\phi_i} \mathbb{F}_2^n, \quad r_{1,i} \xleftarrow{\$,\theta_i} \{0,1\}^{\lambda}$

$\mathsf{com}_{1,i} = \mathsf{Hash}(r_{1,i}, \phi_i)$

endfor

$\pi = \pi_K \circ \cdots \circ \pi_1, \quad t = t_K + \sum_{i=1}^{K-1} \pi_K \circ \ldots \pi_{i+1}(t_i)$

$r \xleftarrow{\$,\epsilon} \mathbb{F}_2, \quad u = \pi^{-1}(r - t) \in \mathbb{F}_2^n$

$\mathsf{com}_1 = \mathsf{Hash}(\mathsf{H}u^T \mid\mid \pi(u) + t \mid\mid (\mathsf{com}_{(1,i)})_{i\in[1,K]})$

$\mathsf{aux} \leftarrow \{\mathsf{com}_1\}$

Send seed to the prover and aux to the verifier

Round 1, Prover $-\ \mathsf{SDP}_1(pk, sk, \mathsf{seed}\ )$

Compute $(\theta_i, \pi_i, t_i)_{i\in[1,K]}, \epsilon$ and u from seed

$s_0 = u + e$

for $i \in [1, K]$

$s_i = \pi_i[s_{i-1}] + t_i$

endfor

$\mathsf{com}_2 = \mathsf{Hash}((s_i)_{i\in[1,K]})$

Send com_2 to the verifier

Round 2, Verifier $-\ \mathsf{SDV}_1()$

$\alpha \xleftarrow{\$} [1, K]$

Send α to the prover

Round 3, Prover$-\mathsf{SDP}_2(pk, sk, \mathsf{seed}, \alpha)$

$z_1 = s_0$

$z_2 = (\theta_i)_{i\in[1,K]\setminus\alpha}$

$z_3 = \pi_\alpha \circ \cdots \circ \pi_1(s_0 - u)$

$\mathsf{rsp} = (z_1, z_2, z_3, \epsilon, \mathsf{com}_{1,\alpha})$

Send rsp to the verifier

Round 4, Verifier $-\ \mathsf{SDV}_2(pk, \mathsf{com}_2, \mathsf{rsp}, \alpha)$

Compute $(\tilde{\phi}_i, \tilde{\pi}_i, \tilde{t}_i, \tilde{r}_{1,i})_{i\in[1,K]\setminus\alpha}$ from z_2 and

$\tilde{q}_K = \tilde{r}$ from ϵ

for $i \in \{K, K-1, \ldots, \alpha+1\}$

$\tilde{q}_{i-1} = (\tilde{\pi}_i)^{-1}(\tilde{q}_i - \tilde{t}_i)$

endfor

$\tilde{s}_0 = z_1, \tilde{s}_\alpha = \tilde{q}_\alpha + z_3$

$\tilde{\mathsf{com}}_{1,\alpha} = \mathsf{com}_{1,\alpha}$

for $i \in [1, K]\setminus\alpha$

$\tilde{s}_i = \tilde{\pi}_i(\tilde{s}_{i-1}) + \tilde{t}_i, \quad \tilde{\mathsf{com}}_{1,i} = \mathsf{Hash}(\tilde{r}_{1,i}, \tilde{\phi}_i)$

endfor

$c_1 \leftarrow (\mathsf{com}_2 = \mathsf{Hash}(z_1 \mid\mid (\tilde{s}_i)_{i\in[1,K]}))$

$c_2 \leftarrow (z_3 \in \tilde{S}(n, \omega))$

$c_3 \leftarrow (\mathsf{com}_1 = \mathsf{Hash}(\mathsf{H}(z_1)^T - y \mid\mid \tilde{r} \mid\mid (\tilde{\mathsf{com}}_{1,i})_{i\in[1,K]}))$

return $c_1 \wedge c_2 \wedge c_3$

Fig. 1. The SD protocol[10]

$$\mathsf{H}_2(z^{(2)})^T = y'_I, \quad \text{with } |z^{(2)}| \leq 2\omega_2.$$

The detailed algorithm is presented in Figure 2.

We now show that our non-interactive zero-knowledge (NIZK) proof ensures correctness, soundness, and zero-knowledge properties.

Lemma 2. *If the hash function* Hash *is a random oracle, then the protocol depicted in Figure 2 is an honest-verifier zero-knowledge PoK with Helper having soundness error* $1/K$ *assuming the hardness of syndrome decoding problems,* $\mathsf{SDP}(\mathsf{H}_1, x_i, \omega_1)$, $\mathsf{SDP}(\mathsf{H}_1, 2\omega_1)$ $\mathsf{SDP}(\mathsf{H}_2, y_i, \omega_2)$ *and* $\mathsf{SDP}(\mathsf{H}_2, 2\omega_2)$ *with* $1 \leq i \leq N$.

Proof. We proceed to prove correctness, soundness and zero-knowledge.

Correctness: Assume the helper is honest. The prover commits to the values $x'_i = x_i + \mathsf{H}_1(a^{(1)})^T$ and $y'_i = y_i + \mathsf{H}_2(a^{(2)})^T$ for $i \in [1, N]$, and then proceeds to prove the knowledge of the secrets corresponding to a pair of the syndromes (x_I, y_I). Assume the user knows the solution to the I^{th} instances of the SD problem x_I and y_I. We have $x'_I = \mathsf{H}_1(a^{(1)} + d_I)^T$ and $y'_I = \mathsf{H}_2(a^{(2)} + e_I)^T$, where $(a^{(1)} + d_I)$, $(a^{(2)} + e_I)$ has weights less than or equal to $2\omega_1$ and $2\omega_2$ respectively. We modified the protocol from [10] in such a way that the verifier checks whether the weights of the secret $d_I + a^{(1)}$ (resp. $e_I + a^{(2)}$) are less than or equal to $2\omega_1$ (resp. $2\omega_2$). The rest of the checks are satisfied due to the correctness of the underlying protocol from [10].

Special Soundness: Assume we are provided with two instances

$$\left(pk_1, pk_2, \mathsf{aux}_1, \mathsf{aux}_2, \mathsf{root}, \mathsf{rsp}_1, \alpha, \mathsf{rsp}_2^{(1)}, \mathsf{rsp}_2^{(2)}\right)$$
$$\left(pk_1, pk_2, \mathsf{aux}_1, \mathsf{aux}_2, \mathsf{root}, \mathsf{rsp}_1, \overline{\alpha}, \overline{\mathsf{rsp}_2^{(1)}}, \overline{\mathsf{rsp}_2^{(2)}}\right)$$

with $\alpha \neq \overline{\alpha}$ generated using a random seed seed. To prove the special soundness, one need to build an efficient knowledge extractor Ext which returns a solution of the *one-out-of-N* instance with respect to pk_1 and pk_2. Given the known values of $z_2^{(1)}$ and $z_2^{(2)}$ from $\mathsf{rsp}_2^{(1)}$ and $\mathsf{rsp}_2^{(2)}$, the extractor Ext can reconstruct all $\theta_i^{(1)}$ and $\theta_i^{(2)}$ for $i \in [1, K] \setminus \alpha$. Furthermore, the values $\theta_\alpha^{(1)}$ and $\theta_\alpha^{(2)}$ can be derived from $\overline{z}_2^{(1)}$ and $\overline{z}_2^{(2)}$, which are obtained from $\overline{\mathsf{rsp}_2^{(1)}}$ and $\overline{\mathsf{rsp}_2^{(2)}}$. Since the masking values $a^{(1)}$, $a^{(2)}$ and the permutations $\pi_i^{(1)}$, $\pi_i^{(2)}$ for $1 \leq i \leq K$ are deterministically generated from the seed values $\theta_i^{(1)}$ and $\theta_i^{(2)}$, the extractor can reconstruct both the masking values and the permutations. Now, for $b \in \{1, 2\}$

$$s_0^{(b)} = d_I + u^{(b)} + a^{(b)} \quad \text{and} \quad z_3^{(b)} = \pi_\alpha^{(b)} \circ \cdots \circ \pi_1^{(b)}(s_0^{(b)} - u^{(b)})$$

are public. From these values, Ext can recover $u^{(1)}, u^{(2)}$ and and then d_I and e_I. Now, $\mathsf{H}_1 d_I^T = x_I$, $\mathsf{H}_2 e_I^T = y_I$ and the extractor has solved the I^{th} instance of the *one-out-of-N* SD problem.

Private Data: $sk_1 = d_I, sk_2 = e_I$

Public Data: $pk_1 = (\mathsf{H}_1 \in \mathbb{F}_2^{(n_1-k_1)\times n_1}, \omega_1, \mathsf{Hash} : \{0,1\}^\star \to \{0,1\}^\lambda, K \in \mathbb{Z}^+), \{x_1, \ldots, x_N\}$

Public Data: $pk_2 = (\mathsf{H}_2 \in \mathbb{F}_2^{(n_2-k_2)\times n_2}, \omega_2, \mathsf{Hash} : \{0,1\}^\star \to \{0,1\}^\lambda, K \in \mathbb{Z}^+), \{y_1, \ldots, y_N\}$

Round 0, Helper $-\ \mathsf{OneOutOfNHelper}(pk_1, pk_2, \mathsf{seed}, \{x_1, \ldots, x_N\}, \{y_1, \ldots, y_N\})$

$\mathsf{PRG}(\mathsf{seed}) = (\mathsf{seed}_1, \mathsf{seed}_2)$

$\mathsf{aux}_1 \leftarrow \mathsf{SDHelper}(pk_1, \mathsf{seed}_1) \quad \mathsf{aux}_2 \leftarrow \mathsf{SDHelper}(pk_2, \mathsf{seed}_2)$

Recompute $\theta_1^{(1)}, \ldots, \theta_K^{(1)}$ from seed_1 and $\theta_1^{(2)}, \ldots, \theta_K^{(2)}$ from the seed_2

$a^{(1)} \xleftarrow{\$, \theta_1^{(1)}, \ldots, \theta_K^{(1)}} S(n_1, \omega_1), \quad a^{(2)} \xleftarrow{\$, \theta_1^{(2)}, \ldots, \theta_K^{(2)}} S(n_2, \omega_2)$

for $i \in [1, N]$

  $\mathsf{bits}_i \xleftarrow{\$, \mathsf{seed}, i} \{0,1\}^\lambda \quad x_i' = x_i + \mathsf{H}_1(a^{(1)})^T \quad y_i' = y_i + \mathsf{H}_2(a^{(2)})^T$

  $\mathsf{C}_i = \mathsf{Hash}(\mathsf{bits}_i \mid\mid x_i' \mid\mid y_i')$

endfor

$(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \ldots, \mathsf{C}_N)$

Send seed to the prover and $\{\mathsf{aux}_1, \mathsf{aux}_2, \mathsf{root}\}$ to the verifier

---

Round 1, Prover $-\ \mathsf{OneOutOfNP}_1(pk_1, pk_2, sk_1, sk_2, \mathsf{seed})$

Update $sk_1 = (d_I + a^{(1)}), \quad pk_1 = (\mathsf{H}_1, |sk_1|, \mathsf{Hash}, K \in \mathbb{Z}^+, x_I')$

Update $sk_2 = (e_I + a^{(2)}), \quad pk_2 = (\mathsf{H}_2, |sk_2|, \mathsf{Hash}, K \in \mathbb{Z}^+, y_I')$

$\mathsf{com}_2^{(1)} \leftarrow \mathsf{SDP}_1(pk_1, sk_1, \mathsf{seed}_1) \quad \mathsf{com}_2^{(2)} \leftarrow \mathsf{SDP}_1(pk_2, sk_2, \mathsf{seed}_2)$

Recompute the index-hiding Merkle tree, tree

$\mathsf{path} \leftarrow \mathsf{MerklePath}(\mathsf{tree}, I)$

Send $\mathsf{rsp}_1 \leftarrow \{\mathsf{com}_2^{(1)}, \mathsf{com}_2^{(2)}, x_I', y_I', \mathsf{bits}_I, \mathsf{path}\}$ to the verifier

---

Round 2, Verifier $-\ \mathsf{OneOutOfNV}_1()$

Send $\alpha \xleftarrow{\$} [1, K]$ to the prover

Round 3, Prover$-\mathsf{OneOutOfNP}_2(pk_1, pk_2, sk_1, sk_2, \mathsf{seed}, \alpha)$

$\mathsf{rsp}_2^{(1)} \leftarrow \mathsf{SDP}_2(pk_1, sk_1, \mathsf{seed}_1, \alpha) \quad \mathsf{rsp}_2^{(2)} \leftarrow \mathsf{SDP}_2(pk_2, sk_2, \mathsf{seed}_2, \alpha)$

Send $\mathsf{rsp}_2^{(1)}, \mathsf{rsp}_2^{(2)}$ to the verifier

---

Round 4, Verifier $-\ \mathsf{OneOutOfNV}_2(pk_1, pk_2, \mathsf{rsp}_1, \mathsf{rsp}_2^{(1)}, \mathsf{rsp}_2^{(2)}, \alpha)$

Parse $z_3^{(1)}$ and $z_3^{(2)}$ from $\mathsf{rsp}_2^{(1)}$ and $\mathsf{rsp}_2^{(2)}$

if $|z_3^{(1)}| > 2\omega_1$ or $|z_3^{(2)}| > 2\omega_2$

  return 0

Update $pk_1 = (\mathsf{H}_1, |z_3^{(1)}|, \mathsf{Hash}, K \in \mathbb{Z}^+, x_I') \quad pk_2 = (\mathsf{H}_2, |z_3^{(2)}|, \mathsf{Hash}, K \in \mathbb{Z}^+, y_I')$

$b_1 \leftarrow \mathsf{SDV}_2(pk_1, \mathsf{com}_2^{(1)}, \mathsf{rsp}_2^{(1)}, \alpha)$

$b_2 \leftarrow \mathsf{SDV}_2(pk_2, \mathsf{com}_2^{(2)}, \mathsf{rsp}_2^{(2)}, \alpha)$

$d \leftarrow (\mathsf{root} = \mathsf{ReconstructRoot}(x_I', y_I', \mathsf{bits}_I, \mathsf{path}))$

return $b_1 \wedge b_2 \wedge d$

Fig. 2. The *one-out-of-N* proof

Zero-Knowledge: The protocol is a *one-out-of-N* proof for joint execution of two independent proof of knowledge instances. We prove the zero-knowledge property for one instance. The other holds exactly the same way.

The prover in the protocol provides a path to the root and a proof of knowledge of the solution to a syndrome y'_I, $I \in [1, N]$ having weight $|sk_2|$. The process is divided in two parts. The first part is to prove the knowledge of the solution I^{th} syndrome decoding problem (knowledge of e_I for the syndrome y_I) using the technique of [10], and the second part is to hide the knowledge of I via the index-hiding Merkle Tree. The details follow.

Let $y_1 = \mathsf{H}e_1^T$. The helper samples $a \in S(n, \omega)$ from seed. The helper commits to $y'_1 = y_1 + \mathsf{H}a^T$. Let S be the following simulator by modifying the simulator S' in [10] that outputs a transcript for the syndrome y_1 that is indistinguishable from an honest execution of the protocol. In the first step we draw a sample following weight distribution of $e_I + a$. Once the weight $\tilde{\omega}$ is fixed then we invoke the S' from the simulator in [10].

1. Let $g_1 \xleftarrow{\$} S(n, \omega_2), g_2 \xleftarrow{\$} S(n, \omega)$. Denote $\tilde{\omega} = |g_1 + g_2|$. Choose $\tilde{e}_2 \xleftarrow{\$} S(n, \tilde{\omega})$
2. Simulate S' on $(H_2, \tilde{\omega}, Hash, , K, y'_1)$ in the following way
   (a) Compute (π_i, t_i) for $i \in [1, K]$ and u from seed
   (b) Recompute y'_1 from seed and compute $\tilde{e}_1 \in \mathbb{F}_2^n$ such that $\mathsf{H}(\tilde{e}_1)^T = y'_1$
   (c) Compute $\tilde{s}_0 = u + \tilde{e}_1$, $\tilde{s}_i = \pi_i[\tilde{s}_{i-1}] + t_i$
   (d) Compute $\tilde{s}_\alpha = \pi_\alpha \circ \cdots \circ \pi_1[u + \tilde{e}_2] + t_\alpha + \sum_{i=1}^{\alpha-1} \pi_\alpha \circ \cdots \circ \pi_{i+1}[t_i]$
   (e) Compute $\tilde{s}_i = \pi_i[\tilde{s}_{i-1}] + t_i$ for all $i \in [\alpha + 1, K]$
   (f) Compute $\tilde{\mathsf{com}}_2 = \mathsf{Hash}(u + \tilde{e}_1 \mid\mid (\tilde{s}_i)_{i \in [1,K]})$
   (g) Compute $\tilde{z}_1 = u + \tilde{e}_1$, $\tilde{z}_2 = (\theta_i)_{i \in [1,K] \setminus \alpha}$, $z_3 = \pi_\alpha \circ \cdots \circ \pi_1[\tilde{e}_2]$
3. Recompute path to root w.r.t $\mathsf{C}_1 = \mathsf{Hash}(\mathsf{bits}_1, y'_1)$
4. Compute $\tilde{\mathsf{rsp}} = (\tilde{z}_1, \tilde{z}_2, \tilde{z}_3, \epsilon, \tilde{\mathsf{com}}_{1,\alpha})$ and output $(\mathsf{H}, \mathsf{root}, y'_1, \mathsf{path}, \tilde{\mathsf{com}}_1, \tilde{\mathsf{com}}_2, \alpha, \tilde{\mathsf{rsp}})$.

To show the indistinguishability of the entire protocol, we consider the following hybrids.

- $\mathsf{Hybrid0}$: This hybrid has the actual prover.
- $\mathsf{Hybrid1}$: In this game, we set $I = 1$. By the property of index-hiding merkle tree, the distribution of $(\mathsf{root}, \mathsf{path})$ is indistinguishable from the ones in $\mathsf{Hybrid0}$.
- The final hybrid is the simulator. The responses by the SDP_1 and SDP_2 are replaced by the output of S'. Since the prover and the simulator have the same weight distribution for the secret, indistinguishability follows as in [10], the output of the simulator S is indistinguishable from the output of the honest prover for the protocol.

□

4 Accountable Ring Signature

4.1 Overview of the construction

In this section, we present an overview of our construction of the accountable ring signature, following techniques of [7]. Let H_2 be a parity check matrix, let

$N \in \mathbb{Z}^+$ denote the number of users in the group, and let F be an efficiently invertible function encoding user indices into low weight codewords.

For $i \in [1, N]$, each user $\mathtt{U}_i$ selects a secret low weight e_i, computes their verification key $\mathsf{v}_i = \mathsf{H}_2 e_i$, and sends it to an opener together with a proof of well-formedness. The opener maintains and publishes the set $\mathcal{R} = \{\mathsf{v}_1, \ldots, \mathsf{v}_N\}$ consisting of all the verification keys The opener also has a public opening key H_1, which is an encryption key for Niederreiter PKE scheme.

To produce a signature, the signer encrypts $F(I)$ using the Niederreiter PKE scheme with encryption key H_1, and it appends to it a non interactive zero-knowledge proof for the following three statements:

1. The ciphertext ct is an encryption of $F(I)$ using the Niederreiter PKE scheme, namely
$$\mathsf{ct} = \mathsf{Encrypt}(F(I), r) = \mathsf{H}_1(F(I) \,||\, r)^T = \mathsf{H}_1^1 F(I)^T + \mathsf{H}_1^2 r^T.$$
2. v_I is included in the ring $\mathcal{R}$;
3. The signer knows a secret key e_I such that $\mathsf{v}_I = \mathsf{H}_2 e_I^T$.

The first statement guarantees that any signature can be traced by the opener and the second and third statements ensure that a user who does not possess a suitable secret signing key will not be able to sign a message on behalf of the set.

Consider the set $\{(\mathsf{ct}_1, \mathsf{v}_1), \ldots, (\mathsf{ct}_N, \mathsf{v}_N)\}$ where $\mathsf{ct}_i = \mathsf{ct} + \mathsf{H}_1^1(F(i))^T$ for $i \in [1, N]$. We then have $(\mathsf{ct}_I, \mathsf{v}_I) = (\mathsf{H}_1^2 r^T, \mathsf{H}_2 e_I^T)$, where the signer knows (r, e_I), and all they need to do is to prove that knowledge. This is achieved with a dedicated "one-out-of-N double syndrome decoding" zero-knowledge protocol which we describe in Section 3.2. Our basic protocol is an interactive protocol with helper with high soundness error; to obtain a signature the helper is removed as in Section 2.1, the protocol is repeated and interaction is removed using the Fiat-Shamir transform. Further tricks can be used to reduce the proof lengths and will be described in Section 5.

4.2 Our base zero-knowledge protocol

As mentioned, we construct the proof of knowledge for the *one-out-of-N* syndrome decoding problem over $\mathbb{F}_2$ following the approach outlined in Section 4.1. This proof is defined with respect to the set $\{(\mathsf{ct}_1, \mathsf{v}_1), \ldots, (\mathsf{ct}_N, \mathsf{v}_N)\}$, where ct_i is given by $\mathsf{ct}_i = \mathsf{ct} + \mathsf{H}_1^1(F(i))^T$, and v_i represents the verification key of user $\mathtt{U}_i$ for $i \in [1, N]$.

We now show that our NIZK ensures completeness, soundness and zero-knowledge properties.

Theorem 1. *If the hash function* Hash *is a random oracle, then the protocol depicted above is an honest-verifier zero-knowledge PoK with Helper having soundness error* $1/K$ *assuming the hardness of four syndrome decoding problems,* $\mathsf{SDP}(\mathsf{H}_1, \mathsf{ct}, \omega_1)$, $\mathsf{SDP}(\mathsf{H}_1^2, 2\omega_1)$ $\mathsf{SDP}(\mathsf{H}_2, \mathsf{v}_i', \omega_2)$ *or* $\mathsf{SDP}(\mathsf{H}_2, 2\omega_2)$*, where* $\mathsf{ct} = \mathsf{Encrypt}(F(I), r)$ *and* v_i *is the verification key of the user* $\mathtt{U}_i$ *for* $i \in [1, N]$.

Proof. Recall that we are proving the knowledge of the

1. The preimage of ct_I with the public key being
   $pk_1 = (\mathsf{H}_1^2 \in \mathbb{F}_2^{(n_r-k_1)\times n_r}, \omega_1, \mathsf{Hash} : \{0,1\}^\star \to \{0,1\}^\lambda, K \in \mathbb{Z}^+, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_N\})$
2. the preimage of v_I with the public key being
   $pk_2 = (\mathsf{H}_2 \in \mathbb{F}_2^{(n_2-k_2)\times n_2}, \omega_2, \mathsf{Hash} : \{0,1\}^\star \to \{0,1\}^\lambda, K \in \mathbb{Z}^+, \{\mathsf{v}_1, \ldots, \mathsf{v}_N\})$

for a certain index $I \in [1, N]$. Since we are making use of the protocol from Section 3.2, correctness and soundness holds. For the zero-knowledge part, note that ct is the output of IND-CPA secure encryption, and thus indistinguishable from a random vector in $\mathbb{F}_2^{(n_1-k_1)}$. Thus, the simulator in this section simply reduces to the simulator from Section 3.2 for (H_1^2 with intermediate commitments ct_i') and ($\mathsf{H}_2, \mathsf{v}_i$) (with same seed to generate the randomness, and the commitment C_i's are generated with the intermediate commitments for both the instances, v_i' and ct_i'). The indistinguishability follows. □

4.3 ARS Algorithms

Here, we present all the algorithms for our accountable ring signature.

1. $\mathtt{Setup}(1^\lambda) \to \mathsf{pp}$
   On input the security a security parameter 1^λ, return the public parameter $\mathsf{pp} = ((n_1, n_2, n_r, n_m, k_1, k_2, \omega_1, \omega_2) \in (\mathbb{Z}^+)^8, \mathsf{seed}_{\mathsf{pub}}\{0,1\}^\lambda)$. Here $n_r + n_m = n_1$. The specific values of n_m and n_r are chosen based on the number of users (See Section 5). We define $\omega_r = \lceil \frac{n_r \omega_1}{n_1} \rceil$ and $\omega_m = \omega_1 - \omega_r$. For security, we need $n_r \geq n_1/2$ [30].
   The public parameters implicitly define $\mathsf{H}_2 \in \mathbb{F}_q^{(n_2-k_2)\times n_2}$ generated from $\mathsf{seed}_{\mathsf{pub}}\{0,1\}^\lambda$, and a relation $\mathsf{R} = \{(\mathsf{v}, e) \mid \mathsf{H}_2 e^T = \mathsf{v} \text{ with } |e| = \omega_2\}$.
2. $\mathtt{OKGen}(\mathsf{pp}) \to (\mathsf{opk} = \mathsf{H}_1, \mathsf{osk})$
   Let $\mathsf{KeyGen}(\mathsf{pp})$ denote the key generation algorithm of an encryption scheme. The $\mathtt{OKGen}$ algorithm outputs a full rank parity-check matrix
   $$\mathsf{H}_1 = [\mathsf{H}_1^1 \mid\mid \mathsf{H}_1^2] \in \mathbb{F}_q^{(n_1-k_1)\times n_1}$$
   with the trapdoor osk using the key-generation algorithm $\mathsf{KeyGen}(\mathsf{pp})$.
3. $\mathtt{UKGen}(\mathsf{pp}) \to (\mathsf{v_i}, \mathsf{e_i})$
   The user $\mathtt{U}_i$ generates an element (e_i, v_i) in the hard relation R and publishes v_i as the verification key. Additionally, they provide a proof of knowledge demonstrating that they know the secret e_i, which has a weight exactly equal to ω_2, using the proof of knowledge method outlined in Section 3.1 (here, the verifier checks whether the weight of the resulting vector is exactly ω_2 instead of less than or equal to ω_2). The user keeps the witness e_i as the secret signing key.
4. $\mathtt{Sign}(\mathsf{opk}, e_I, \mathcal{R}, \mathsf{M}) \to \sigma$ and $\mathtt{Verify}(\mathsf{opk}, \mathcal{R}, \mathsf{M}, \sigma) \to 1/0$
   Recall that the details of the base protocol were described in Section 4.2.
   From Base protocol to Main protocol: For $\lambda \in \mathbb{Z}^+$, let K correspond to the parameter from the *one-out-of-N* protocol. Define, $\ell_\lambda = \lambda / \log_2(K)$

to be the number of rounds we run to reach a soundness error of $1/2^{\lambda}$. Here we convert the base protocol into a signature scheme using the Fiat-Shamir transformation. The details of the protocol are given in Figure 3.

5. $\mathtt{Open}(\mathsf{osk}, \mathcal{R}, \mathsf{M}, \sigma) \rightarrow (\mathsf{vk}, \pi)/0$
   Let π be a signature which passes the verification algorithm. The opener can obtain the ciphertext, ct from the signature. The idea then is to simply decrypt the ciphertext to obtain the identity of the user.

$$\mathsf{Decrpt}(\mathsf{ct}) = \mathsf{Decrypt}(\mathsf{Encrypt}(F(I), r)) = (F(I), r)$$

   From the recovered plaintext, the opener inverts F to identify the user to whom $F(I)$ corresponds and outputs either the corresponding verification key or 0 in case of failure.
6. $\mathtt{Judge}(\mathsf{opk}, \mathcal{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi) \rightarrow 1/0$
   Let π be a signature which passes the verification algorithm. The opener can obtain the ciphertext, ct from the signature. Let, $\mathsf{Decrypt}(\mathsf{ct}) = (F(I), r)$. The opener now outputs the index I and provides a zero-knowledge proof of knowledge of the SD problem [10], $\pi_{\mathtt{judge}}$ for z of weight ω_r satisfying $\mathsf{H}_1^2 z^T = \mathsf{ct} - \mathsf{H}_1^1 (F(I))^T$. If the check is right, the judge outputs 1. Note that since the Neiderreiter decryption algorithm is perfectly correct, there does not exist an $(F(J), r') \neq (F(I), r)$ with $|r'| \leq \omega_r$ such that $\mathsf{Decrypt}(\mathsf{ct}) = (F(J), r')$

Security properties The following theorem captures our main result.

Theorem 2. *If the hash function* Hash *is a random oracle, then the accountable ring signature obtained by applying Fiat-Shamir transformation on our* NIZK *protocol achieves CCA-anonymity against full key exposure, sound traceability, non-frameability, and unforgeability.*

Proof of Theorem 2 Recall that our construction follows the blueprint of [7]. We use the following main theorem of [7].

Theorem 3. *([7]) If the underlying encryption scheme is multi-challenge* $\mathsf{IND\text{-}CPA}$ *secure, and the* NIZK *is online extractable, honest-verifier proof of knowledge, and the* $\pi_{\mathtt{judge}}$ *protocol is honest verifier zero-knowledge proof then the accountable ring signature achieves CCA-anonymity against full key exposure, sound traceability, non-frameability, and unforgeability.*

Recall that we use the randomised Neiderreiter encryption scheme, which is proven to be $\mathsf{IND\text{-}CPA}$ secure [30]. As observed in [7], a single-challenge $\mathsf{IND\text{-}CPA}$ encryption scheme is in-fact multi-challenge $\mathsf{IND\text{-}CPA}$ secure (via a simple hybrid argument).

Lemma 3. *The protocol described in Figure 3 is multi-proof online extractable for the relation*

$$\mathsf{R} = \left\{ \left(X = \{(\mathsf{v}_i)\}_{i \in [N]}, pk, \mathsf{ct} \right), W = (I, e_I, r_I) \mid (\mathsf{H}_2 e_I^T = \mathsf{v}_I) \wedge |e_I| = \omega \wedge \mathsf{ct} = \mathsf{Encrypt}(F(I); r) \right\}$$

Private Data: $sk_1 = r, sk_2 = e_I$

Public Data$(pk) : pk_1 = (\mathsf{H}_1 = [\mathsf{H}_1^1 \mid \mathsf{H}_1^2] \in \mathbb{F}_2^{(n_1-k_1)\times n_1}, \omega_r, \mathsf{Hash} : \{0,1\}^\star \to \{0,1\}^\lambda, K \in \mathbb{Z}^+)$

$pk_2 = (\mathsf{H}_2 \in \mathbb{F}_2^{(n_2-k_2)\times n_2}, \omega_2, \mathsf{Hash} : \{0,1\}^\star \to \{0,1\}^\lambda, K \in \mathbb{Z}^+)$,

$\mathcal{R}, \mathsf{ct} = \mathsf{Encrypt}(F(I), r), N, \ell_\lambda$, HashFS: $\{0,1\}^\star \to [1,\mathsf{S}]^{\ell_\lambda} \times [1,\mathsf{K}]^{\ell_\lambda}$

Round 1, Prover – $\mathsf{GSP}(pk_1, pk_2, sk_1, sk_2, \mathsf{seed}, \mathsf{ct}, \mathcal{R})$

K is the size of the challenge space, N is the number of users, S is the number of randomly generated seeds and ℓ_λ, the number of repetitions to achieve λ-bits of security

for $i \in [1, N]$

  $\mathsf{ct}_i = \mathsf{ct} + \mathsf{H}_1^1 F(i)^T$

endfor

$\mathcal{R} \leftarrow \{v_1, \ldots, v_N\}$ $\mathsf{CT} \leftarrow \{\mathsf{ct}_1, \ldots \mathsf{ct}_N\}$

for $i_1 \in [1, \ell_\lambda]$ and **for** $i_2 \in [1, \mathsf{S}]$

  $\mathsf{seed}_{i_1,i_2} \xleftarrow{i_1,i_2,\mathsf{seed}} \{0,1\}^\lambda$, $(\mathsf{aux}_{i_1,i_2}, \mathsf{root}_{i_1,i_2}) \leftarrow \mathsf{OneOutOfNHelper}(pk_1, pk_2, \mathsf{seed}_{i_1,i_2}, \mathcal{R}, \mathsf{CT})$,

  $\mathsf{rsp}_{1,i_1,i_2} \leftarrow \mathsf{OneOutOfNP}_1(pk_1, pk_2, sk_1, sk_2, \mathsf{seed}_{i_1,i_2})$

endfor

$\mathrm{temp} = M \parallel \mathsf{FS} \parallel (\mathsf{aux}_{i_1,i_2}, \mathsf{root}_{i_1,i_2}, \mathsf{rsp}_{1,i_1,i_2})_{i_1 \in [1,\ell_\lambda], i_2 \in [1,\mathsf{S}]} \parallel pk$

$(j_1, \ldots, j_{\ell_\lambda}, \alpha_1, \ldots, \alpha_{\ell_\lambda}) = \mathsf{HashFS}(\mathrm{temp})$

for $i \in [1, \ell_\lambda]$

  $\mathsf{rsp}_{2,i} \leftarrow \mathsf{OneOutOfNP}_2(pk_1, pk_2, sk_1, sk_2, \mathsf{seed}_{j_i}, \alpha_i)$

endfor

$\mathsf{reveal} \leftarrow \{(i_1, i_2) \in [1, \ell_\lambda] \times [1, \mathsf{S}] \mid i_2 \neq j_{i_1}\}$

return $\pi \leftarrow \{(\mathsf{seed}_{i_1,i_2})_{(i_1,i_2)\in\mathsf{reveal}}, (\mathsf{aux}_{i_1,i_2}, \mathsf{root}_{i_1,i_2}, \mathsf{com}_{2,i_1,i_2})_{i_1\in[1,\mathsf{S}], i_2\in[1,\ell_\lambda]}, (\mathsf{rsp}_{2,i})_{i\in[1,\ell_\lambda]}\}$

Round 2, Verifier – $\mathsf{GSV}(pk_1, pk_2, \mathsf{ct}, \mathcal{R}, \pi')$

$\pi' \leftarrow \{(\mathsf{seed}_{i_1,i_2})_{(i_1,i_2)\in\mathsf{reveal}'}, (\mathsf{aux}'_{i_1,i_2}, \mathsf{root}'_{i_1,i_2}, \mathsf{rsp}'_{1,i_1,i_2})_{i_1\in[1,\mathsf{S}], i_2\in[1,\ell_\lambda]}, (\mathsf{rsp}'_{2,i})_{i\in[1,\ell_\lambda]}\}$

$\mathrm{temp} = \mathsf{msg} \parallel \mathsf{FS} \parallel (\mathsf{aux}'_{i_1,i_2}, \mathsf{root}'_{i_1,i_2}, \mathsf{rsp}'_{1,i_1,i_2})_{i_1\in[1,\ell_\lambda], i_2\in[1,\mathsf{S}]} \parallel pk$

$(j'_1, \ldots, j'_{\ell_\lambda}, \alpha'_1, \ldots, \alpha'_{\ell_\lambda}) = \mathsf{HashFS}(\mathrm{temp})$

$\mathsf{reveal}' \leftarrow \{(i_1, i_2) \in \{1, \ell_\lambda\} \times \{1, \mathsf{S}\} \mid i_2 \neq j'_{i_1}\}$

$\mathcal{R} \leftarrow \{v_1, \ldots, v_N\}$ $\mathsf{CT} \leftarrow \{\mathsf{ct}_1, \ldots \mathsf{ct}_N\}$

for $(i_1, i_2) \in \mathsf{reveal}'$

  if $\mathsf{seed}_{i_1,i_2} \notin \pi'$ **return** 0

endfor

for $(i_1, i_2) \in \mathsf{reveal}'$

  if $(\mathsf{aux}'_{i_1,i_2}, \mathsf{root}'_{i_1,i_2}) \neq \mathsf{OneOutOfNHelper}(pk_1, pk_2, \mathsf{seed}_{i_1,i_2}, \mathcal{R}, \mathsf{CT})$

    return 0

endfor

$b = 1$

for $i \in \{1, \ell_\lambda\}$

  $b = b \wedge \mathsf{OneOutOfNV}_2(pk_1, pk_2, \mathsf{rsp}'_{1,i,j_i}, \mathsf{rsp}'_{2,i}, \alpha'_i)$

endfor

return b

Fig. 3. The final NIZK

5 Parameters and Instantiation for the Group Signature

In this section, we provide concrete parameter choices for our group signature scheme.

Signature size for a single execution of the protocol from Figure 3: We will use superscripts notations (1) or (2) to differentiate outputs when we use pk_1 or pk_2. Recall that a single execution with helper consists of the auxillary information aux_1 and aux_2, root, rsp_1, $\mathsf{rsp}_2^{(1)}$ and $\mathsf{rsp}_2^{(2)}$. We now calculate the cost of the individual components.

1. The helper round requires the prover to send $\mathsf{aux}_1, \mathsf{aux}_2$ and root. This requires 3λ bits.
2. For the first round, the prover has to send $\mathsf{rsp}_1 \leftarrow \{\mathsf{com}_2^{(1)}, \mathsf{com}_2^{(2)}, \mathsf{ct}'_I, \mathsf{v}'_I, \mathsf{bits}_I, \mathsf{path}\}$ to the verifier. The targets ct'_I and v'_I are of length $(n_1 - k_1)$ and $(n_2 - k_2)$ respectively, and the cost of computing the path for a single setup is $\lambda \log_2(N)$ bits. Thus this step requires $3\lambda + (n_1 - k_1) + (n_2 - k_2) + \lambda \log_2(N)$ bits
3. The prover then sends two responses $\mathsf{rsp}_2^{(1)} \leftarrow \mathsf{SDP}_2(pk_1, sk_1, \mathsf{seed}_1, \alpha)$ and $\mathsf{rsp}_2^{(2)} \leftarrow \mathsf{SDP}_2(pk_2, sk_2, \mathsf{seed}_2, \alpha)$. The individual components of the responses include $z_1^{(b)}, z_2^{(b)}, z_3^{(b)}$ and $\epsilon^{(b)}$ for $b \in [1, 2]$. We have $z_1^{(1)}, z_3^{(1)} \in \mathbb{F}_2^{n_r}$ and $z_1^{(2)}, z_3^{(2)} \in \mathbb{F}_2^{n_2}$. The message $z_2^{(b)}$ corresponds to $K - 1$ distinct seeds and $\epsilon^{(b)}$ refers to another seed value for $b \in [1, 2]$. So the cost of this step in the protocol will be $2K\lambda + 2n_r + 2n_2$ bits.

Implementation strategies In order to calculate the parameters of our scheme, we considered the following implementation strategies (from [6] and [11]) to implement the helper protocol and index-hiding Merkle trees.

1. **Communicating seeds via a binary tree:** In our protocol, the prover selects S seed values and sends all except one. The approach involves employing a binary tree format, selecting a root node randomly, and generating values for all S leaf nodes. Rather than sending $\mathsf{S} - 1$ seed values, the prover can now send $\lceil \log_2(\mathsf{S}) \rceil$ node values within the tree, enabling them to recalculate all seeds except the challenged one.
2. **Using Merkle trees for commitments:** Instead of sending Q commitments, com_i for $i \in [1, \mathsf{Q}]$, the prover constructs a Merkle tree on these commitments and only sends the root of the tree. In response, the prover sends $\lceil \log_2(\mathsf{Q}) \rceil$ nodes of the Merkle tree to reconstruct the root.
3. **Parallel repetitions**: Instead of letting the verifier choose 1 out of S setups to execute with a resulting soundness error $1/K$, we now let them choose τ out of M setups to execute. Now suppose a cheating prover does at most τ out of the M setups incorrectly, the soundness error of the scheme is bounded by
$$max_{1 \le t \le \tau} \frac{\binom{M-t}{\tau-t}}{\binom{M}{\tau} K^{(\tau-t)}}$$
4. **Random seed substitution:** The strategy, introduced in [11], allows to substitute a vector $v \in \mathbb{F}_2^n$ by a random seed. This technique implies that our proof of knowledge size scales with $1.5\tau(n_r + n_2)$ instead of $2\tau(n_r + n_2)$.

Total, optimized group signature size We now estimate the size of our group signatures, following [10].

1. Choose (M, τ, K) such that the soundness error of the protocol is less that $2^{-\lambda}$, where λ is the security parameter. One can use a seed tree to generate the seeds and a Merkle tree to store all the auxiliary information. Since only τ setups are executed, the prover has to reveal all except τ seed values and auxiliary information to the verifier by using the Merkle tree setup. This information requires $(\lambda + |\mathsf{com}|)\tau \log_2(M/\tau) = 2\lambda\tau \log_2(M/\tau)$ bits.
2. With the random seed substitution, the cost of sending $z_1^{(b)}, z_3^{(b)}$ for $b \in \{1, 2\}$ will be $1.5(n_1 + n_2)$ bits.
3. With the Merkle tree optimization for $z_2^{(b)}$, the seed values can be revealed with a cost of $\lambda \log_2(K)$ bits.
4. Therefore, the total bit size of the signature is

$$8\lambda + \tau\left(2.5(n_1 + n_2) - (k_1 + k_2) + \lambda(\log_2(N) + 4\log_2(K) + 2\log_2(M/\tau))\right).$$

Concrete parameters The security of the Niederreiter cryptosystem is equivalent to the security of the McEliece cryptosystem [33,28] with the same parameters. The best algorithms to break these systems are the variants of the information set decoding [32]. We take the **level-1** parameter $(n_r, k_1, \omega_r) = (3488, 2720, 64)$ to achieve 128-bit security for the McEliece system.

We set (n_m, ω_m) as $(55, 1)$, $(109, 2)$, and $(164, 3)$ corresponding to around $2^6, 2^{12}$ and 2^{20} users. More generally, our group signature can accommodate $\binom{n_m}{w_m}$ users. Recall that the values of (n_1, ω_1) can be calculated as $(n_1, \omega_1) = (n_r + n_m, \omega_r + \omega_m)$.

Following [27], we set $(n_2, k_2, \omega_2) = (1280, 640, 132)$, $(1300, 650, 135)$, $(1360, 680, 141)$ for $N = 2^6, 2^{12}$ and 2^{20} users respectively. We also set (K, τ, M) to be $(32, 28, 389)$ as in [6] to reach 128−bits of security.

The resulting signature sizes are given in the following table.

N	Sig. size(kB) (ours)	Sig. size(kB)[27]	Sig. size(kB) [31]
2^6	45	112	49
2^{12}	48	126	74
2^{20}	53	146	124

The signature size for the lattice and isogeny-based constructions for achieving 128−bit security in [7] were 96KB and 15.5KB respectively for 2^{20} users.

References

1. Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A code-based group signature scheme. *Designs, Codes and Cryptography*, 82:469–493, 2017.
2. Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. Membership privacy for fully dynamic group signatures. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2181–2198, 2019.

3. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In *Annual International Cryptology Conference*, pages 581–615. Springer, 2023.
4. Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Trans. Inf. Theory*, 24(3), 1978.
5. David Bernhard, Marc Fischlin, and Bogdan Warinschi. Adaptive proofs of knowledge in the random oracle model. In *Public-Key Cryptography–PKC 2015*, pages 629–649. Springer, 2015.
6. Ward Beullens. Sigma protocols for MQ, PKP and SIS, and fishy signature schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 183–211. Springer, 2020.
7. Ward Beullens, Samuel Dobson, Shuichi Katsumata, Yi-Fu Lai, and Federico Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 95–126. Springer, 2022.
8. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 464–492. Springer, 2020.
9. Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In *Security and Cryptography for Networks: 7th International Conference, Proceedings 7*, pages 381–398. Springer, 2010.
10. Loïc Bidoux and Philippe Gaborit. Shorter signatures from proofs of knowledge for the SD, MQ, PKP and RSD problems. *arXiv preprint arXiv:2204.02915*, 2022.
11. Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, and Nicolas Sendrier. Quasi-cyclic Stern proof of knowledge. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 1459–1464. IEEE, 2022.
12. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Annual international cryptology conference*, pages 41–55. Springer, 2004.
13. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In *International Conference on Applied Cryptography and Network Security*, pages 117–136. Springer, 2016.
14. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *European Symposium on Research in Computer Security*, pages 243–265. Springer, 2015.
15. Jan Camenisch. Efficient and generalized group signatures. In *International conference on the theory and applications of cryptographic techniques*, pages 465–479. Springer, 1997.
16. David Chaum and Eugène Van Heyst. Group signatures. In *Advances in Cryptology—EUROCRYPT'91*, pages 257–265. Springer, 1991.
17. Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and Huaxiong Wang. Provably secure group signature schemes from code-based assumptions. *IEEE Transactions on Information Theory*, 66(9):5754–5773, 2020.
18. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

19. Shafi Goldwasser, Silvio Micali, and Chales Rackoff. The knowledge complexity of interactive proof-systems. In *Providing sound foundations for cryptography: On the work of Shafi Goldwasser and Silvio Micali*, pages 203–225. 2019.
20. S Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *Advances in Cryptology-ASIACRYPT 2010. Proceedings 16*, pages 395–412. Springer, 2010.
21. Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 253–280. Springer, 2015.
22. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM Journal on Computing*, 39(3):1121–1152, 2009.
23. Shuichi Katsumata and Shota Yamada. Group signatures without nizk: from lattices in the standard model. In *Advances in Cryptology–EUROCRYPT 2019, Proceedings, Part III 38*, pages 312–344. Springer, 2019.
24. Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In *Advances in Cryptology-ASIACRYPT 2013, Proceedings, Part II 19*, pages 41–61. Springer, 2013.
25. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: logarithmic-size ring signatures and group signatures without trapdoors. In *Advances in Cryptology–EUROCRYPT 2016, Proceedings, Part II 35*, pages 1–31. Springer, 2016.
26. Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: standard model security from simple assumptions. In *Advances in Cryptology–CRYPTO 2015, Proceedings, Part II 35*, pages 296–316. Springer, 2015.
27. Xindong Liu and Li-Ping Wang. Short code-based one-out-of-many proofs and applications. In *IACR International Conference on Public-Key Cryptography*, pages 370–399. Springer, 2024.
28. Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.
29. Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Prob. Contr. Inform. Theory*, 15(2):157–166, 1986.
30. Ryo Nojima, Hideki Imai, Kazukuni Kobara, and Kirill Morozov. Semantic security for the mceliece cryptosystem without random oracles. *Designs, Codes and Cryptography*, 49:289–305, 2008.
31. Ying Ouyang, Deng Tang, and Yanghong Xu. Code-based zero-knowledge from vole-in-the-head and their applications: Simpler, faster. Technical report, and Smaller. Cryptology ePrint Archive, Report 2024/1414, 2024.
32. Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.
33. Nicolas Sendrier. Mceliece public key cryptosystem., 2005.
34. Jacques Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
35. Luping Wang, Jie Chen, Huan Dai, and Chongben Tao. Efficient code-based fully dynamic group signature scheme. *Theoretical Computer Science*, 2024.
36. Shouhuai Xu and Moti Yung. Accountable ring signatures: A smart card approach. In *Smart Card Research and Advanced Applications VI: IFIP 18th World Computer Congress TC8/WG8*, pages 271–286. Springer, 2004.