On Canonical Liftings, Group Actions, and Linkable Ring Signatures

Alessandro Budroni[1], Giuseppe D'Alconzo[2], Leonardo Errati[2], and Edoardo Persichetti[3]

[1] Technology Innovation Institute, UAE, alessandro.budroni@tii.ae
[2] Politecnico di Torino, Italy, {giuseppe.dalconzo,leonardo.errati}@polito.it
[3] Florida Atlantic University, USA, epersichetti@fau.edu

Abstract. We introduce the notion of *canonical lifting* for group actions. This generalises existing techniques for reducing the size of group-action-based signatures, such as the canonical forms used in the LESS signature scheme. Building on this framework, we construct Sigma protocols and a linkable ring signature scheme. Notably, our approach avoids the key-reuse mechanism for linkability used in the framework of Beullens *et al.* (ASIACRYPT'20), which was shown to be insecure for group actions arising from linear code equivalence (ASIACRYPT'24) and lattice isomorphism (CiC'25). The resulting signatures have size logarithmic in the ring size. We provide security proofs at the level of the abstract canonical-lifting framework, which are then inherited by concrete instantiations satisfying the required assumptions. Finally, we instantiate our approach using the linear code equivalence group action and obtain a linkable ring signature scheme CELeReS, achieving signature sizes that compare favourably with the state of the art, and our AVX2-optimised C implementation demonstrates its practical applicability.

Keywords: Canonical Lifting, Group Actions, Sigma Protocols, Linkable Ring Signatures, Linear Code Equivalence, Post Quantum Cryptography

1 Introduction

Group actions provide a general framework for constructing cryptographic primitives in post-quantum cryptography. Starting as a natural generalisation of the structure underlying the discrete logarithm problem, this framework captures computational assumptions arising from isogenies [28,23,12,1], linear codes [25,51,13], lattices [29,9], and other algebraic settings. In particular, the LESS protocol, originally proposed in [13], is a code-based signature scheme whose security relies on the Linear Equivalence Problem (LEP). Since LEP can be modelled as a computational problem arising from a group action, namely, the Group Action Inverse Problem, LESS follows the standard paradigm of applying the Fiat–Shamir transform to a group-action-based Sigma protocol. LESS was submitted to the NIST call for additional post-quantum digital signatures [45] and

advanced to the second round, although it was not selected for the third round. The threshold signature LEAST [5], which builds on LESS, is currently a preview submission to the ongoing NIST call for multi-party threshold schemes [46].

Chou, Persichetti, and Santini [27] introduced a major optimisation based on canonical forms for linear codes, which significantly reduced signature sizes and was later adopted in the LESS specification [4]. The resulting variant relies on the Canonical-Form Linear Equivalence Problem (CF-LEP), shown to be equivalent to LEP. However, for the canonical forms used in LESS, CF-LEP does not admit a formulation as a group-action inverse problem. As a consequence, the latest version of LESS falls outside the standard framework of group-action-based signatures. For this reason, recent works on advanced LESS-based signatures treat the canonical-form and standard group action as separate settings [40,35].

Ring signatures, introduced by Rivest, Shamir, and Tauman-Kalai [50] and later standardised in ISO/IEC 20008 [37], allow a signer to authenticate a message anonymously within a signer-chosen set of public keys, called the *ring*, without requiring prior enrolment or a group manager. They are used, for example, in privacy-preserving cryptocurrencies such as Monero and Bytecoin, as well as in anonymous communication, whistleblowing systems, electronic voting, confidential authentication, and other privacy-sensitive settings [49]. In many of these applications, *linkability* is also needed to detect repeated use by the same anonymous signer. Linkable ring signatures provide this feature by allowing signatures from the same ring member to be publicly linked, preventing exploits such as double-spending or double-voting while preserving anonymity [52].

Related work. Two lines of research focused on post-quantum linkable ring signatures with signature size logarithmic in the size of the ring. The first is based on assumptions from symmetric cryptography. In 2019, Backes, Döttling, Hanzlik, Kluczniak, and Schneider [3] proposed a framework for (linkable) ring signatures from symmetric primitives and NIWI (non-interactive witness-indistinguishable) proofs. More recently, Zhang et al. [53] proposed DualRing$_L$-PRF, based on Legendre and power-residue PRFs.
The second focuses on assumptions from public-key cryptography. In 2019, Lu, Au and Zhang proposed Raptor [43], based on lattices. In 2020, Beullens, Katsumata, and Pintore presented a framework to design (linkable) ring signatures from group actions [11] by compressing an OR proof with Merkle trees, instantiating it with isogenies (Calamari) and lattices (Falafl). Their framework has been instantiated with other group actions: code [8] and matrix code [25] equivalence, lattice isomorphism [38], trilinear form equivalence [14,32]. However, if applied to linear code equivalence or lattice isomorphism, the resulting schemes do not achieve linkability [17,18,39]. Moreover, it was recently discovered that the use of non-commutative group actions leads to correctness and linkability issues [36]. These proposals also provide different security guarantees. Bultel and Olivier-Anclin [22] identify anonymity with adversarially chosen keys and repeated challenge-signature queries as the strongest anonymity notion, and observe that

only the schemes of Backes et al. [3] and Beullens et al. [11] among those surveyed were proved in this model. In this paper, we follow the model of Beullens et al.

Our Contributions. In this work, we introduce the notion of *canonical liftings* for group actions, a new framework for constructing cryptographic primitives. This isolates the structural principles underlying the canonical-form techniques used in LESS, while placing them in a broader setting that is not tied to linear code equivalence. We study the conditions under which this abstraction preserves the hardness of the underlying group-action problem. We then demonstrate its versatility in the construction of advanced cryptographic primitives by building a linkable ring signature. Importantly, in this case, the canonical-lifting framework allows us to diverge from the approach of [11]: instead of reusing the same secret key for both signing and enabling linkability, we use two distinct secret keys that are, however, equivalent under some equivalence relation. Most importantly, our construction does not inherit the linking mechanism of [11] and avoids the correctness issues described in [36]. Finally, we instantiate our constructions from the linear code equivalence group action, obtaining the linkable ring signature schemes CELeReS with signature sizes starting from 26.7 KB for ring size 2 and that compares favourably with the state of the art.

1.1 Technical Overview

Let $(G, \circ)$ be a group and X a set. A (left)[4] *group action* of G on X is a map

$$\star : G \times X \to X$$

compatible with the group operation. This means that, for any $g, g' \in G$ and any $x \in X$, we have that $g \star (g' \star x) = (g \circ g') \star x$, and such that $e \star x = x$, where e is the identity element of G. Given two elements $x, x' \in X$, the *Group Action Inverse Problem (GAIP)* asks to find $g \in G$ such that $x' = g \star x$.

Generally, in group-action-based Sigma protocols, the statement is the pair $x, x' \in X$, while the witness is a group element $g \in G$ such that $x' = g \star x$. The prover samples a random $\rho \in G$ and commits to $\mathsf{com} = \rho \star x'$. Then, depending on the verifier's challenge, it reveals either ρ or $\rho \circ g$. The verifier checks the response against the commitment, i.e. verifies that $\mathsf{com} = \rho \star x'$ or that $\mathsf{com} = (\rho \circ g) \star x$, again depending on the challenge. In the signature schemes derived from this kind of Sigma protocol, a significant part of the signature consists of the group elements $\rho \circ g$ that must be encoded and sent by the signer to the verifier. Finding ways to compress these group elements has therefore been an important research direction for this family of signatures [33,27,26,15].

Canonical lifting. In this work, we introduce the notion of canonical lifting for group actions. A canonical lifting for a group action $(G, X, \star)$ is composed of two

[4] The definition of *right* group actions is analogous.

sets R and C, together with three efficient maps

$$\mathsf{Rep} : G \to R, \qquad \mathsf{Can} : X \to C, \qquad \mathsf{Lift} : R \times X \to C$$

such that $\mathsf{Can}(g \star x) = \mathsf{Lift}(\mathsf{Rep}(g), x)$, for every $g \in G$ and $x \in X$. If elements of R admit a more compact representation than elements of G, canonical lifting can be used to reduce the communication cost of group-action-based Sigma protocols. Indeed, the prover can commit to $\mathsf{Can}(\rho \star x')$, and then, instead of sending $\rho \circ g \in G$, it can send only the representative $\mathsf{Rep}(\rho \circ g) \in R$. The verifier can then check the response by calculating

$$\mathsf{Lift}(\mathsf{Rep}(\rho \circ g), x) = \mathsf{Can}((\rho \circ g) \star x) = \mathsf{Can}(\rho \star (g \star x)) = \mathsf{Can}(\rho \star x').$$

Thus, the verifier recovers the same canonical value in C as in the commitment while receiving a potentially shorter response.

Sigma protocols. We use the canonical-lifting framework to define a Sigma protocol and a OR Sigma protocol with logarithmic ring-size overhead, based on an index-hiding Merkle tree. Since canonical liftings fall outside the standard group-action model, the security analysis cannot be obtained as a direct adaptation of existing group-action-based proofs. We therefore revisit the proof techniques, especially those underlying the unforgeability of the ring signature scheme. This leads to the introduction of the Canonical Lifting Collision Problem (CLCP) as the central hardness assumption: given $x, x' \in X$, find $r, r' \in R$ such that

$$\mathsf{Lift}(r, x) = \mathsf{Lift}(r', x').$$

We then identify sufficient conditions under which CLCP is equivalent to GAIP:

i) for every $r \in R$, one can efficiently compute $g \in G$ such that $\mathsf{Rep}(g) = r$,
ii) whenever $\mathsf{Can}(x) = \mathsf{Can}(x')$, one can efficiently recover a group element $g \in G$ such that $x' = g \star x$.

Under these conditions, the canonical lifting preserves the hardness of the underlying group-action problem while making explicit the effective space of the secret keys, i.e., the set of representatives R rather than the group G.

Linkable ring signature. We obtain an OR Sigma protocol with tag that we use to instantiate a linkable ring signature from it. Rather than reusing the secret key twice for the public/private key pair and the tag as in [11], we use different secrets $g, g' \in G$ that are however equivalent via Rep, that is,

$$\mathsf{Rep}(g) = \mathsf{Rep}(g').$$

This allows to achieve linkability while avoiding the attack scenario studied in the literature of this family of linkable ring signatures [17,18,39]. We use the framework of canonical lifting to identify the minimal computational assumptions to guarantee linkability, linkable anonymity, and non-frameability.

Instantiations. We instantiate the linkable ring signature using a suitable canonical lifting for the group action of code equivalence, resulting in the scheme CELeReS, and study the underlying hardness assumptions over linear codes and discuss the possible cryptanalysis approaches. Compared with the proposed post-quantum linkable ring signatures that are considered both secure and correct, it achieves the smallest sizes, starting from 26.7 KB for ring size 2. We present our comparisons in Table 1.

Artifacts. We provide our implementation[5] of CELeReS in reference C and AVX2-optimised code. Our benchmarks show that it is suitable for practical applications.

Scheme	Security	2^1	2^2	2^6	2^{10}	2^{12}	Assumption
CELeReS	128 bit	26.7	27.8	32.3	36.7	39.0	Sec. 5.2
	192 bit	69.2	72.3	85.1	97.8	104.2	
	256 bit	119.1	123.8	142.6	161.3	170.7	
Raptor [43]	100 bit	–	–	82.7	1301.9	5203.3	NTRU
Calamari [11]	NIST-I[6]	3.6	4.6	14.1	20.1	23.1	CSIDH-512
Falafl [11]	NIST-I	33	34	36	39.3	41	MSIS, MLWE
DualRing-PRF [53]	128 bit	–	–	45	64	–	Residue PRFs

Table 1: Signature-size comparison, in kilobytes, for different ring sizes. Sizes for CELeReS and [11] were retrieved from the implementations and include the tag.

Paper Organisation. Section 2 recalls the necessary background. In Section 3, we introduce canonical liftings and the corresponding Sigma protocols. In Section 4, we construct our linkable ring signature, and instantiate it with the LEP group action in Section 5. Finally, in Section 6, we present our implementation and provide comparisons against other schemes.

2 Preliminaries

Notation. We only deal with finite sets and represent them with uppercase letters. The notation $s \leftarrow\!\!\$\ S$ means that s was sampled uniformly at random from the set S. We denote by $[n]$ the set $\{1, \ldots, n\}$. For a prime q, we denote by $\mathbb{F}_q$ the finite field of q elements. Consider the n-dimensional $\mathbb{F}_q$-vector space $\mathbb{F}_q^n$. Bold lowercase letters denote row vectors $\boldsymbol{v} = (v_1, \ldots, v_n)$, bold uppercase letters denote matrices $\boldsymbol{M} \in \mathbb{F}_q^{k \times n}$. For a set of indices $J \subseteq [n]$, we denote by $\boldsymbol{M}_J$ the matrix obtained by selecting the columns of $\boldsymbol{M}$ indexed by J. We make use of the following sets of full-rank matrices: $\mathsf{GL}_n(\mathbb{F}_q)$, the group of invertible $n \times n$ matrices, $\mathsf{Perm}_n(\mathbb{F}_q)$, the set of permutation matrices of order n, $\mathsf{Perm}_{k,n}(\mathbb{F}_q)$, where $k < n$, the set of permutation matrices $\boldsymbol{P}$ such that, if $\boldsymbol{P}(i,j) = \boldsymbol{P}(i',j') = 1$ for some $i < i'$, then $j < j'$ whenever $i, i' \leq k$ or $i, i' > k$ [27, Sec. 2]. With $\mathsf{Diag}_n(\mathbb{F}_q)$ we denote the set of full-rank $n \times n$ diagonal matrices, and $\mathsf{Mono}_n(\mathbb{F}_q)$ is the set of monomial matrices $\boldsymbol{M} = \boldsymbol{P}\boldsymbol{D}$, for $\boldsymbol{P} \in \mathsf{Perm}_n(\mathbb{F}_q)$ and $\boldsymbol{D} \in \mathsf{Diag}_n(\mathbb{F}_q)$.

[5] Available at `https://anonymous.4open.science/r/celeres-DBB9`.
[6] The claimed security level of Calamari is questionable, see [16,48,24].

Group actions. We consider *effective* (left) group actions [1, Def. 3.4], meaning that the elements of G and X can be efficiently represented, and that the basic operations associated with the action, including sampling, membership testing, group operations, inversion, and evaluation of the action, are efficiently computable. An *automorphism* of $x \in X$ is a group element $g \in G$ such that $g \star x = x$.

Problem 1. *Given a group action $(G, X, \star)$ and two set elements $x, x' \in X$, the* Group Action Inverse Problem (GAIP) *requires finding, if it exists, a group element $g \in G$ such that $x' = g \star x$.*

2.1 Linear Code Equivalence and Canonical Forms

An $[n,k]_q$*-linear code* $\mathscr{C}$ is a k-dimensional subspace of $\mathbb{F}_q^n$, of *length* $n > 0$ and *dimension* $0 < k \leq n$. A matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ whose rows form a basis of $\mathscr{C}$ as vector subspace is called a *generator matrix* of $\mathscr{C}$, and $\mathscr{C} = \left\{\boldsymbol{uG} \mid \boldsymbol{u} \in \mathbb{F}_q^k\right\} = \langle \boldsymbol{G} \rangle$. We may apply a change of basis $\boldsymbol{S} \in \mathsf{GL}_k(\mathbb{F}_q)$ and obtain another, equally valid, generator matrix $\boldsymbol{G'} = \boldsymbol{SG}$. If it is in the form $\boldsymbol{G'} = [\boldsymbol{I}_k \mid \boldsymbol{A}]$, where $\boldsymbol{I}_k$ is the size-k identity matrix and $\boldsymbol{A}$ a $k \times (n-k)$ matrix, we say it is in *systematic form.*

Linear Equivalence Problem. Two $[n,k]_q$-linear codes $\mathscr{C}, \mathscr{C}'$ are said to be *linearly equivalent* if there exists a monomial matrix $\boldsymbol{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ such that $\mathscr{C}' = \mathscr{C}\boldsymbol{Q}$. The computational problem of determining the equivalence between two linear codes can be rephrased using the notion of generator matrices, and is the security assumption underlying the LESS signature scheme [4].

Definition 1. *Given two generator matrices $\boldsymbol{G}, \boldsymbol{G'} \in \mathbb{F}_q^{k \times n}$ of two linearly equivalent codes $\mathscr{C}$ and $\mathscr{C}'$, the* Linear Equivalence Problem (LEP) *requires finding an invertible $\boldsymbol{S} \in \mathsf{GL}_k(\mathbb{F}_q)$ and a monomial $\boldsymbol{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ such that $\boldsymbol{G'} = \boldsymbol{SGQ}$.*

When $\boldsymbol{Q}$ is simply a permutation matrix, the above problem is referred to as *Permutation Equivalence Problem (PEP).* Notice that PEP is considered hard only in limited cases, for example when the code is self-orthogonal.

We define the monomial automorphism group of a code $\mathscr{C}$ as

$$\mathsf{Aut}(\mathscr{C}) = \{\boldsymbol{Q} \in \mathsf{Mono}_n(\mathbb{F}_q) \ : \ \mathscr{C}\boldsymbol{Q} = \mathscr{C}\}.$$

Analogously, the permutation automorphism group is defined for $\boldsymbol{Q} \in \mathsf{Perm}_n(\mathbb{F}_q)$. If $\mathscr{C}$ is generated by $\boldsymbol{G}$, we may write $\mathsf{Aut}(\boldsymbol{G})$. We say that $\mathscr{C}$ has trivial automorphism group if $\mathsf{Aut}(\mathscr{C}) = \{\boldsymbol{I}_n\}$. In the literature, it is folklore to assume that randomly sampled linear codes have trivial automorphism group. Furthermore, we report the following recent result on the matter.

Proposition 1 ([41]). *If $\mathscr{C}$ is a uniformly random $[n,k]_q$-linear code with $\min\{k, n-k\} > (2+\varepsilon)\log_q(n)$, for any $\varepsilon > 0$, then it has a trivial monomial automorphism group with high probability as n goes to infinity. Moreover, $\mathscr{C}$ has no non-trivial permutation automorphisms.*

Linear equivalence as a group action. LEP can be naturally interpreted as an instance of GAIP. Indeed, let $G = \mathsf{GL}_k(\mathbb{F}_q) \times \mathsf{Mono}_n(\mathbb{F}_q)$ act on the set of full-rank matrices in $\mathbb{F}_q^{k \times n}$ via $(\boldsymbol{S}, \boldsymbol{Q}) \star \boldsymbol{G} = \boldsymbol{S}\boldsymbol{G}\boldsymbol{Q}$. With respect to this action, finding an equivalence between two generator matrices $\boldsymbol{G}$ and $\boldsymbol{G}'$ (solving LEP) is exactly the problem of inverting the group action (solving GAIP). This is often not presented as a left group action, since $\boldsymbol{S}$ and $\boldsymbol{Q}$ acts on both the left and the right of $\boldsymbol{G}$. We can still consider it as such by implicitly assuming that the product in the group G is given by $(\boldsymbol{S}_1, \boldsymbol{Q}_1) \circ (\boldsymbol{S}_2, \boldsymbol{Q}_2) = (\boldsymbol{S}_1\boldsymbol{S}_2, \boldsymbol{Q}_2\boldsymbol{Q}_1)$.

2.2 Relations and Sigma Protocols

Definition 2. *Consider a set X of statements and a set W of witnesses. A* sigma protocol Σ *for a binary relation $\mathcal{R} \subseteq X \times W$ is a public-coin interactive protocol between a prover $\mathcal{P}(x, w)$ and a verifier $\mathcal{V}(x)$ with the following structure:*

- $\mathsf{Setup} \to (x, w)$. *The instance-witness pair (x, w) is generated.*
- $\mathcal{P}.\mathsf{Commit} \to \mathsf{com}$. *The prover $\mathcal{P}$ sends an initial message* com *to $\mathcal{V}$.*
- $\mathcal{V}.\mathsf{Challenge} \to \mathsf{ch}$. *On input the commitment* com*, $\mathcal{V}$ sends a challenge* ch *sampled with distribution $\mathcal{D}_{\mathscr{C}}$ from a challenge space $\mathscr{C}$.*
- $\mathcal{P}.\mathsf{Response} \to \mathsf{rsp}$. *On input the challenge* ch*, $\mathcal{P}$ sends a response* rsp.
- $\mathcal{V}.\mathsf{Verify} \to \mathsf{accept}/\mathsf{reject}$. *On input* rsp*, $\mathcal{V}$ either accepts or rejects.*

At the end of their interaction, the verifier outputs accept *or* reject*. The execution trace* $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$ *is called* transcript*, and is* accepting *if $\mathcal{V}$ outputs* accept*. Further, we require the following properties.*

- Completeness: *If both $\mathcal{P}$ and $\mathcal{V}$ follow the protocol honestly and $(x, w) \in \mathcal{R}$, then $\mathcal{V}$ always returns* accept.
- 2-Special Soundness: *Given the statement x, there exists a polynomial-time algorithm $\mathcal{E}$, called* extractor*, that on input two accepting transcripts* $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$ *and* $(\mathsf{com}, \mathsf{ch}', \mathsf{rsp}')$ *for the same commitment* com *and* $\mathsf{ch} \neq \mathsf{ch}'$ *can compute w such that $(x, w) \in \mathcal{R}$.*
- (Perfect) Honest-verifier zero-knowledge: *There exists a PPT algorithm $\mathcal{S}(x, \mathsf{ch})$, called* simulator*, that outputs an accepting transcript* $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$ *with a distribution identical to that of transcripts produced by an honest interaction between $\mathcal{P}$ and $\mathcal{V}$.*

The Fiat-Shamir transform [34] removes the interactive step and allows asynchronous verification. This is a known procedure to build signature schemes.

Relaxed relations. The property of special soundness in Definition 2 can be relaxed by requiring the extractor $\mathcal{E}$ to return a witness satisfying a weaker relation $\tilde{\mathcal{R}} \supseteq \mathcal{R}$ instead of $\mathcal{R}$. For this to be meaningful, once given a statement of this weaker relation, finding a related witness still needs to be sufficiently hard.

OR relations. Given a relation $\mathcal{R}$, we can construct one where the $\mathcal{P}$ proves knowledge of the statement $(x_1, w) \in \mathcal{R} \vee \cdots \vee (x_r, w) \in \mathcal{R}$. This is called its *OR relation* (denoted by $\mathcal{R}^{\mathsf{OR}}$), and is defined as

$$\mathcal{R}^{\mathsf{OR}} = \{\langle (x_1, \ldots, x_r), w \rangle \in X^r \times W \mid \text{ for some } i \in [r], (x_i, w) \in \mathcal{R}\}\,.$$

2.3 Linkable Ring Signature Schemes

A ring signature allows a user to sign a message while concealing their identity among the members of a *ring* R. A *linkable* ring signature additionally allows one to detect whether two distinct signatures were produced by the same signer, without revealing their identity. Most constructions achieve this through a tag uniquely associated with the user, which can be bundled to the signature material σ, and the procedure $\mathsf{LRS.Link}$.

Definition 3. *A* linkable ring signature scheme *is a quintuple of PPT algorithms* $\mathsf{LRS} = (\mathsf{LRS.Setup}, \mathsf{LRS.KeyGen}, \mathsf{LRS.Sign}, \mathsf{LRS.Verify}, \mathsf{LRS.Link})$ *such that:*

- $\mathsf{LRS.Setup}(1^\lambda) \to \mathsf{pp}$. *On input the security parameter* λ*, it returns the public parameters* pp *of the scheme, implicit input to all other algorithms of* LRS.
- $\mathsf{LRS.KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$. *On input the public parameters* pp *of the scheme, it returns a public key* pk *and a secret key* sk.
- $\mathsf{LRS.Sign}(\mathsf{sk}, \mathsf{msg}, \mathsf{R}) \to \sigma$. *On input the secret key* sk*, a message* msg*, and a ring* $\mathsf{R} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ *of public keys, it returns a signature* σ.
- $\mathsf{LRS.Verify}(\mathsf{R}, \mathsf{msg}, \sigma) \to \mathsf{accept/reject}$. *On input a ring* $\mathsf{R} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ *of public keys, a message* msg *and a signature* σ*, it returns* accept *or* reject.
- $\mathsf{LRS.Link}(\sigma_0, \sigma_1) \to \mathsf{accept/reject}$. *On input two signatures* σ_0, σ_1*, it returns* accept *or* reject.

The scheme must be correct. *For every security parameter* $\lambda \in \mathbb{N}$ *and* $N = \mathsf{poly}(\lambda)$ *cosigners, signer* $j \in [N]$*, messages* msg_1 *and* msg_2*, sets* $D_0, D_1 \subseteq [N]$ *such that* $j \in D_0 \cap D_1$*, an honest execution of the protocol is verified with probability 1:*

$$\Pr\left[\begin{array}{c} \mathsf{LRS.Verify}(\mathsf{R}_0, \mathsf{msg}_0, \sigma_0) = \mathsf{accept} \\ \mathsf{LRS.Verify}(\mathsf{R}_1, \mathsf{msg}_1, \sigma_1) = \mathsf{accept} \\ \mathsf{LRS.Link}(\sigma_0, \sigma_1) = \mathsf{accept} \end{array} \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{LRS.Setup}(1^\lambda) \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{LRS.KeyGen}(\mathsf{pp})\ \forall i \in [N] \\ \mathsf{R}_b = \{\mathsf{pk}_i\}_{i \in D_b}\ \textit{for } b = 0, 1\ \textit{and} \\ \sigma_b \leftarrow \mathsf{LRS.Sign}(\mathsf{sk}_j, \mathsf{msg}_b, \mathsf{R}_b)\ \textit{for } j \in [N] \end{array}\right] = 1.$$

To be considered secure, LRS must be *linkable*, *linkable anonymous*, and *non-frameable*. We follow the definitions of Beullens, Katsumata, and Pintore [11]. In particular, the survey [22] refers to their linkable-anonymity notion as *ano-ACK* and distinguishes it from weaker notions.

Definition 4. *A linkable ring signature scheme* LRS *is said* secure *if it satisfies the following security properties.*

- Linkability. *For every security parameter* $\lambda \in \mathbb{N}$ *and* $N = \mathsf{poly}(\lambda)$ *number of cosigners, the advantage of any PPT adversary* $\mathcal{A}$ *playing the game* $\mathsf{Game}^{\mathsf{Link}}_{\mathsf{LRS}}(\lambda)$ *of Figure 1 is negligible in* λ*, where*

$$\mathsf{Adv}^{\mathsf{Link}}_{\mathsf{LRS}}(\mathcal{A}) = \Pr[\mathcal{A}\ \textit{wins}\ \mathsf{Game}^{\mathsf{Link}}_{\mathsf{LRS}}].$$

- Linkable Anonymity. *For every security parameter* $\lambda \in \mathbb{N}$ *and* $N = \mathsf{poly}(\lambda)$ *number of cosigners, the advantage of any PPT adversary* $\mathcal{A}$ *playing the game* $\mathsf{Game}^{\mathsf{L-Anon}}_{\mathsf{LRS}}(\lambda)$ *of Figure 1 is negligible in* λ*, where*

$$\mathsf{Adv}^{\mathsf{L-Anon}}_{\mathsf{LRS}}(\mathcal{A}) = \left|\Pr[\mathcal{A}\ \textit{wins}\ \mathsf{Game}^{\mathsf{L-Anon}}_{\mathsf{LRS}}] - \tfrac{1}{2}\right|.$$

– Non-frameability. *For every security parameter* $\lambda \in \mathbb{N}$ *and* $N = \mathsf{poly}(\lambda)$ *number of cosigners, the advantage of any PPT adversary* $\mathcal{A}$ *playing the game* $\mathsf{Game}^{\mathsf{Non-Frame}}_{\mathsf{LRS}}(\lambda)$ *of Figure 1 is negligible in* λ*, where*

$$\mathsf{Adv}^{\mathsf{Non-Frame}}_{\mathsf{LRS}}(\mathcal{A}) = \Pr[\mathcal{A} \textit{ wins } \mathsf{Game}^{\mathsf{Non-Frame}}_{\mathsf{LRS}}].$$

$\mathsf{Game}^{\mathsf{Link}}_{\mathsf{LRS}}(\lambda)$:
1. The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{LRS.Setup}(1^\lambda)$ and sends pp to $\mathcal{A}$.
2. $\mathcal{A}$ outputs $PK = \{\mathsf{pk}_i\}_{i\in[N]}$ and $\{(\sigma_i, \mathsf{msg}_i, \mathsf{R}_i)\}_{i\in[N+1]}$. It wins if:
   (a) $\forall i \in [N+1] : \mathsf{R}_i \subseteq PK$ and $\mathsf{LRS.Verify}(\mathsf{R}_i, \mathsf{msg}_i, \sigma_i) = \mathsf{accept}$;
   (b) $\forall i, j \in [N+1], i \neq j : \mathsf{LRS.Link}(\sigma_i, \sigma_j) = \mathsf{reject}$.

$\mathsf{Game}^{\mathsf{L\text{-}Anon}}_{\mathsf{LRS}}(\lambda)$:
1. The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{LRS.Setup}(1^\lambda)$, generates $(\mathsf{pk}_i, \mathsf{sk}_i)$ for $i \in [N]$, and samples $b \leftarrow \{0, 1\}$.
2. $\mathcal{C}$ sends pp and $PK = \{\mathsf{pk}_i\}_{i\in[N]}$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs $\mathsf{pk}^*_0, \mathsf{pk}^*_1 \in PK$.
4. $\mathcal{C}$ sends all keypairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ such that $\mathsf{pk}_i \in PK \setminus \{\mathsf{pk}^*_0, \mathsf{pk}^*_1\}$.
5. $\mathcal{A}$ may perform *signing queries* $(\mathsf{sign}, \mathsf{pk}, \mathsf{msg}, \mathsf{R})$, where $\mathsf{pk} \in \{\mathsf{pk}^*_0, \mathsf{pk}^*_1\} \subseteq \mathsf{R}$:
   (a) if $\mathsf{pk} = \mathsf{pk}^*_0$, $\mathcal{C}$ returns $\sigma \leftarrow \mathsf{LRS.Sign}(\mathsf{sk}^*_b, \mathsf{msg}, \mathsf{R})$.
   (b) if $\mathsf{pk} = \mathsf{pk}^*_1$, $\mathcal{C}$ returns $\sigma \leftarrow \mathsf{LRS.Sign}(\mathsf{sk}^*_{1-b}, \mathsf{msg}, \mathsf{R})$.
6. $\mathcal{A}$ outputs b^*. It wins if $b^* = b$.

$\mathsf{Game}^{\mathsf{Non\text{-}Frame}}_{\mathsf{LRS}}(\lambda)$:
1. The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{LRS.Setup}(1^\lambda)$, generates $(\mathsf{pk}_i, \mathsf{sk}_i)$ for $i \in [N]$, and initialises two lists $SL, CL \leftarrow \emptyset$, logging signing and corruption queries.
2. The challenger sends pp and PK to $\mathcal{A}$.
3. $\mathcal{A}$ may adaptively perform two kinds of query:
   (a) signing query $(\mathsf{sign}, i, \mathsf{msg}, \mathsf{R})$, if $\mathsf{pk}_i \in \mathsf{R}$, the challenger $\mathcal{C}$ returns $\sigma \leftarrow \mathsf{LRS.Sign}(\mathsf{sk}_i, \mathsf{msg}, \mathsf{R})$ and adds $(i, \mathsf{msg}, \mathsf{R})$ to SL;
   (b) corruption query $(\mathsf{corrupt}, i)$, $\mathcal{C}$ adds pk_i to CL and returns $(\mathsf{pk}_i, \mathsf{sk}_i)$.
4. $\mathcal{A}$ outputs $(\mathsf{R}^*, \mathsf{msg}^*, \sigma^*)$. It wins if:
   (a) the solution was not part of a signing query, i.e., $(\cdot, \mathsf{msg}^*, \mathsf{R}^*) \notin SL$;
   (b) the signature verifies, i.e., $\mathsf{LRS.Verify}(\mathsf{R}^*, \mathsf{msg}^*, \sigma^*) = \mathsf{accept}$;
   (c) there was some sign query $(i, \mathsf{msg}, \mathsf{R}) \in SL$ which returns σ and $\mathsf{LRS.Link}(\sigma^*, \sigma) = \mathsf{accept}$ for an uncorrupted $\mathsf{pk}_i \in PK \setminus CL$.

Fig. 1: Security games for LRS: linkability, linkable anonymity, non-frameability.

Linkable ring signatures may also be required to satisfy *unforgeability* (with respect to insider corruption), i.e., no PPT adversary $\mathcal{A}$ with access to signing and corruption oracles may feasibly forge a signature of an uncorrupted user [11,22] with respect to a fixed ring. However, with the notation of Definition 4, this is

known to be implied by linkability and non-frameability. Suppose that a PPT unforgeability adversary $\mathcal{A}$ manages to produce a valid forgery with respect to $\mathsf{R} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. Combining its forgery with N honestly generated signatures, one from each ring member, yields $N + 1$ valid signatures. If two of them link to the same member, this contradicts non-frameability. If none link, this contradicts linkability, having produced $N + 1$ signatures for N users.

3 Canonical Liftings for Group Actions

In this section, we first introduce the notion of canonical liftings for group actions. Then, we show how to construct Sigma protocols within this framework.

Definition 5. *Let $(G, X, \star)$ be a group action. A* canonical lifting *for $(G, X, \star)$ is a tuple $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$, where R and C are sets, whose elements we call* representatives *and* canonical values*, respectively, and*

$$\mathsf{Rep} : G \to R, \qquad \mathsf{Can} : X \to C, \qquad \mathsf{Lift} : R \times X \to C$$

are efficiently computable maps such that, for all $g \in G$ and $x \in X$,

$$\mathsf{Can}(g \star x) = \mathsf{Lift}(\mathsf{Rep}(g), x). \tag{1}$$

We call Equation (1) *the* defining property of a canonical lifting.

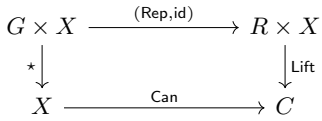

Fig. 2: The maps of Definition 5. On the left, the group action paradigm. On the right, the canonical lifting one.

Intuitively, Equation (1) captures the fact that a canonical value $\mathsf{Can}(x')$ associated to a transformed statement $x' = g \star x$ can be recovered from the original statement x and a suitable representative $\mathsf{Rep}(g)$ of the group element relating the two. This is represented in Figure 2. Notice that the map Rep induces an equivalence relation on G, that is $g \sim g'$ if and only if $\mathsf{Rep}(g) = \mathsf{Rep}(g')$, for $g, g' \in G$. We denote by $[g]_{\mathsf{Rep}}$ the class of g. The notion of canonical lifting generalises the concept of canonical forms with respect to an equivalence relation that was introduced in the context of LEP [33,27].

3.1 Sigma Protocol via Canonical Lifting

The defining property of canonical liftings gives naturally rise to a statement-witness relation. The public statement is $x, x' \in X$, and a witness is a pair

$h, h' \in R$ that leads to the same canonical representation when applied to x and x', respectively. We say that they *lead to a collision*. The relation is

$$\mathcal{R}_{\mathsf{CL}} = \left\{ \langle (x, x'), (h, h') \rangle \in X^2 \times R^2 \mid \mathsf{Lift}(h, x) = \mathsf{Lift}(h', x') \right\}. \tag{2}$$

We introduce in Figure 3 a Sigma protocol Σ_{CL} for this relation. In contrast to typical Sigma protocols arising from group actions, here the prover commits to the canonical value associated with the randomised statement $\rho \star x'$, for a randomiser $\rho \leftarrow\!\$\, G$. Upon receiving $\mathsf{ch} = 0$, the prover answers with ρ. If $\mathsf{ch} = 1$ instead, rather than revealing the full witness $\rho \circ g$, it sends its representative $\mathsf{Rep}(\rho \circ g)$. Thanks to Equation (1), the verifier recovers the same canonical value. If $\mathsf{ch} = 0$, one could similarly send $\mathsf{rsp} = \mathsf{Rep}(\rho)$ and verify by computing $\mathsf{Lift}(\mathsf{rsp}, x')$. However, in the Fiat–Shamir signature, this branch is typically represented by the seed used to generate the randomizer ρ, rather than by ρ itself. Hence, this additional optimisation would not affect the signature size.

Setup
1. Let $x \in X$ be public
2. Sample $g \leftarrow\!\$\, G$
3. Set $x' = g \star x$

PROVER	**VERIFIER**
$\mathsf{Commit}(x')$ 1. Sample $\rho \leftarrow\!\$\, G$ 2. Set $\mathsf{com} = \mathsf{Can}(\rho \star x')$ 3. Send com to the verifier	$\mathsf{Challenge}$ 1. Sample $\mathsf{ch} \leftarrow\!\$\, \{0, 1\}$ 2. Send ch to the prover
$\mathsf{Response}(\mathsf{ch}, g)$ 1. If $\mathsf{ch} = 0$: i. Set $\mathsf{rsp} = \rho$ 2. Else: i. Set $\mathsf{rsp} = \mathsf{Rep}(\rho \circ g)$ 3. Send rsp to the verifier	$\mathsf{Verify}(x, x', \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$ 1. If $\mathsf{ch} = 0$: i. parse rsp as ρ ii. $\mathsf{com}' = \mathsf{Can}(\rho \star x')$ 2. Else: i. parse rsp as h ii. $\mathsf{com}' = \mathsf{Lift}(h, x)$ 3. Return accept iff $\mathsf{com}' = \mathsf{com}$

Fig. 3: Sigma protocol Σ_{CL} from a group action with canonical lifting.

If one takes $R = G$, $C = X$, Lift as the group action $\star$, and Rep and Can as the identity functions, the canonical lifting Sigma protocol generalises the standard group-action one. However, if the elements of R admit a compact encoding, replacing the group element in the response by its representative yields a more efficient Sigma protocol and, consequently, shorter Fiat–Shamir signatures.

Known Instantiations. The protocol in Figure 3 provides an abstraction of the CF-LEP Sigma protocol [27, Figure 8] underlying LESS 2.0, which is generally viewed as an optimised variant of the original group-action-based construction; however, its use of canonical forms places it outside the standard group-action framework. Canonical liftings provide a formal framework for this canonical-form variant and can therefore also describe advanced LESS-derived signature schemes that originate from group actions but rely on canonical forms, such as [40,35]. Concretely, in LESS 2.0, given $\mathsf{SF}(\boldsymbol{G}) = [\boldsymbol{I}_k \mid \boldsymbol{A}]$ the systematic form of $\boldsymbol{G}$, then $\mathsf{Can}(\boldsymbol{G}) = \mathsf{CF}(\boldsymbol{A})$, where CF is the canonical-form algorithm described in [4, Sec. 4.1]. Considering for simplicity directly the group $G = \mathsf{Mono}_n(\mathbb{F}_q)$, the representative map Rep is instantiated as

$$\mathsf{CosetRep} : \mathsf{Mono}_n(\mathbb{F}_q) \longrightarrow \mathsf{Perm}_{k,n}(\mathbb{F}_q),$$

with respect to the subgroup $F_3 \coloneqq \mathsf{Mono}_k(\mathbb{F}_q) \times \mathsf{Mono}_{n-k}(\mathbb{F}_q)$. Namely, the map $\mathsf{CosetRep}(\boldsymbol{Q})$ returns the unique $\boldsymbol{P} \in \mathsf{Perm}_{k,n}(\mathbb{F}_q)$ such that $\boldsymbol{Q} = \boldsymbol{N}\boldsymbol{P}$ for some $\boldsymbol{N} \in F_3$. Such a representative can be encoded by a set $\mu \subseteq [n]$ of k indices, and Lift reconstructs the corresponding canonical value as

$$\mathsf{Lift}(\boldsymbol{G}, \mu) = \mathsf{CF}\big((\boldsymbol{G}_\mu)^{-1}\boldsymbol{G}_{[n]\setminus\mu}\big)\,.$$

The signature-size improvement comes from transmitting these coset representatives, encodable with $\lceil \log_2 \binom{n}{k} \rceil$ bits, instead of full elements of $\mathsf{Mono}_n(\mathbb{F}_q)$. Nevertheless, our framework allows to better understand the scheme at a more abstract level, rather than seeing canonical forms only as an optimization. As an illustration of this, we present in Appendix A an optimization for LESS that we uncover thanks to the study of the scheme under the canonical lifting framework.

Other instantiations naturally fitting the canonical-lifting framework were proposed in [33] to optimize the LESS and MEDS signature schemes via group factorisation. The signature-compression technique for MEDS introduced in [26] and further improved and formalized with a group action notation in [15] follows a similar principle, although in a more instance-dependent form. Indeed, rather than sending the isometry itself, the response consists of codewords from which the verifier can reconstruct the relevant isometry. In the terminology of canonical liftings, this can be viewed as allowing a representative map having the form $\mathsf{Rep} : X \times G \to R$ so that the compressed response depends both on the group element and on the public instance. In this work, we consider the case where Rep depends only on the group element, which suffices for the ring and linkable ring signature instantiations in the following sections.

Proposition 2. *The Sigma protocol Σ_{CL} of Figure 3 satisfies completeness, 2-special soundness, and honest-verifier zero-knowledge.*

Proof. (Completeness) Let $x' = g \star x$ be an honestly generated public statement, and let $\rho \in G$ be the randomiser chosen by the prover during the commitment phase. The prover computes $\mathsf{com} = \mathsf{Can}(\rho \star x')$. If $\mathsf{ch} = 0$, the prover returns

ρ, and the verifier recomputes $\mathsf{com}' = \mathsf{Can}(\rho \star x')$. Since $\mathsf{com}' = \mathsf{com}$, it always accepts. If $\mathsf{ch} = 1$, the prover returns $h = \mathsf{Rep}(\rho \circ g)$ and the verifier computes

$$\begin{aligned}\mathsf{com}' &= \mathsf{Lift}(h, x) = \mathsf{Lift}(\mathsf{Rep}(\rho \circ g), x) \\ &= \mathsf{Can}((\rho \circ g) \star x) = \mathsf{Can}(\rho \star (g \star x)) = \mathsf{Can}(\rho \star x') = \mathsf{com}\end{aligned}$$

where we used the defining property of Equation (1). Thus the verifier accepts.

2-Special Soundness. Given two accepting transcripts for the same commitment com and different challenge ch, namely $(\mathsf{com}, 0, \mathsf{rsp}_0)$ and $(\mathsf{com}, 1, \mathsf{rsp}_1)$. We have $\mathsf{Can}(\mathsf{rsp}_0 \star x') = \mathsf{com} = \mathsf{Lift}(\mathsf{rsp}_1, x)$. Hence, the witness for the statement (x, x') is given by $(\mathsf{rsp}_1, \mathsf{Rep}(\mathsf{rsp}_0))$, as by definition of canonical lifting it holds that $\mathsf{Lift}(\mathsf{rsp}_1, x) = \mathsf{Can}(\mathsf{rsp}_0 \star x') = \mathsf{Lift}(\mathsf{Rep}(\mathsf{rsp}_0), x')$.

Honest-Verifier Zero-Knowledge. We exhibit a polynomial-time simulator $\mathcal{S}$ that, on input a challenge ch and a valid statement (x, x'), returns an accepting transcript with the same distribution of a transcript obtained by a real execution of the protocol. If $\mathsf{ch} = 0$, it follows the instruction of a real prover. If $\mathsf{ch} = 1$, it samples ρ' from G and returns the simulated transcript $(\mathsf{com} = \mathsf{Lift}(\mathsf{Rep}(\rho'), x), \mathsf{ch} = 1, \mathsf{rsp} = \mathsf{Rep}(\rho'))$. By the defining property of canonical lifting we get $\mathsf{com} = \mathsf{Lift}(\mathsf{Rep}(\rho'), x) = \mathsf{Can}(\rho' \star x)$. Since the prover samples ρ uniformly and g is fixed, $\rho \circ g$ is also uniform. Therefore, for a uniformly sampled ρ', the pair $(\mathsf{Lift}(\mathsf{Rep}(\rho \circ g), x), \mathsf{Rep}(\rho \circ g))$ produced by an honest execution of the protocol and $(\mathsf{Lift}(\mathsf{Rep}(\rho'), x), \mathsf{Rep}(\rho'))$ are identically distributed. □

3.2 OR Sigma Protocol via Canonical Lifting

We generalise the Sigma protocol Σ_{CL} to an OR Sigma protocol $\Sigma_{\mathsf{CL}}^{\mathsf{OR}}$, described in Figure 4. The OR relation induced by $\mathcal{R}_{\mathsf{CL}}$ is

$$\mathcal{R}_{\mathsf{CL}}^{\mathsf{OR}} = \left\{ \langle (x, x_1, \ldots, x_N), (h, h', i) \rangle \in X^{N+1} \times \left(R^2 \times [N]\right) \mid \mathsf{Lift}(h, x) = \mathsf{Lift}(h', x_i) \right\}.$$

The prover randomises its witness and commits using an index-hiding Merkle tree, of which we report the definition in Appendix B.2. This compresses the commitments into a single root, while still allowing to only open the branch corresponding to the known witness without revealing its index. We prove relaxed 2-special soundness for the relaxed OR relation:

$$\tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}} = \left\{ \langle (x, x_1, \ldots, x_N), w \rangle \;\middle|\; \begin{array}{c} (x, x_1, \ldots, x_N) \in X^{N+1} \text{ and either} \\ w = (h, h', i) : \mathsf{Lift}(h, x) = \mathsf{Lift}(h', x_i); \text{ or} \\ w = (s, s') : s \neq s', \mathsf{H}_{\mathsf{Coll}}(s) = \mathsf{H}_{\mathsf{Coll}}(s'), \text{ or } \mathcal{O}(s) = \mathcal{O}(s') \end{array} \right\}.$$

This extends $\mathcal{R}_{\mathsf{CL}}^{\mathsf{OR}}$, and is needed due to the use of index-hiding Merkle trees; the function $\mathsf{MT.GetTree}$ introduces a collision-resistant $\mathsf{H}_{\mathsf{Coll}}$, while an oracle $\mathcal{O}$ models commitment schemes by hash functions. Witnesses could arise from the new conditions, as they could reconstruct the root just as $w = (h, h', i)$ would.

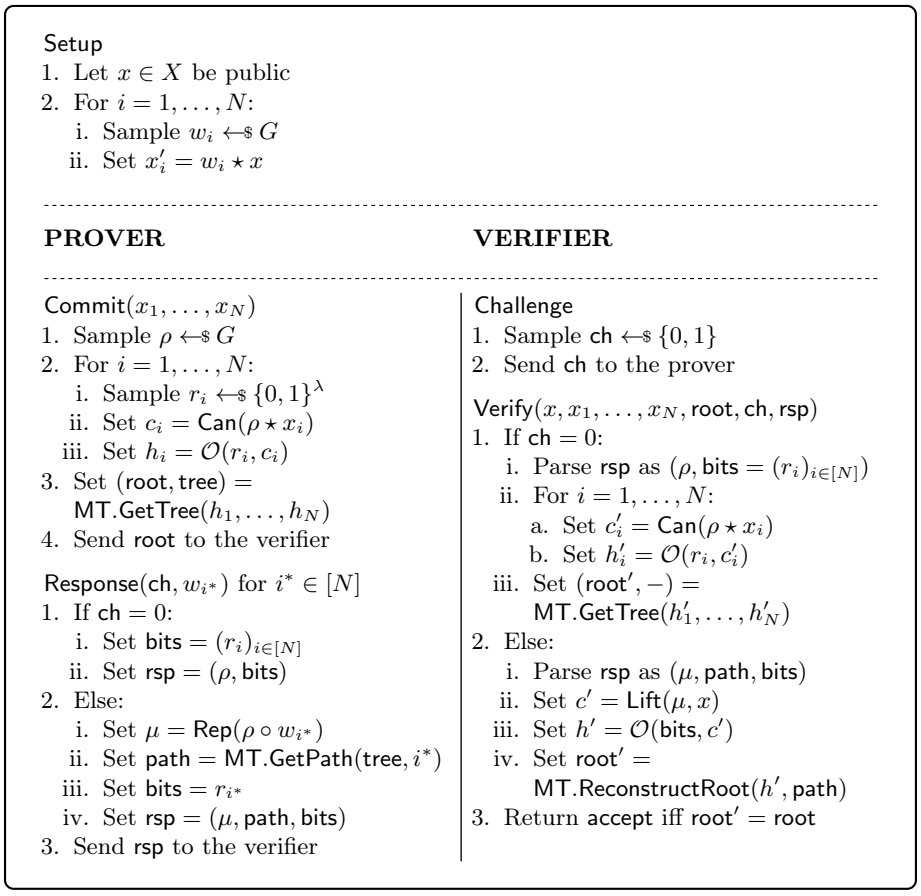

Fig. 4: OR Sigma protocol $\Sigma^{\mathsf{OR}}_{\mathsf{CL}}$.

Proposition 3. *The Sigma protocol $\Sigma^{\mathsf{OR}}_{\mathsf{CL}}$ of Figure 4 satisfies completeness, relaxed 2-special soundness, and honest-verifier zero-knowledge.*

Proof. Completeness. Let $i^* \in [N]$ be the secret index, and let $\rho \in G$ be the randomising element used in the commitment phase. If $\mathsf{ch} = 0$, the prover reveals ρ together with all opening strings $(r_i)_{i \in [N]}$. An honest verifier recomputes every commitment h_i, hence reconstructs the same root. If $\mathsf{ch} = 1$, the prover reveals $h = \mathsf{Rep}(\rho \circ w_{i^*})$ together with the opening string r_{i^*} and the authentication path for the corresponding leaf. By the same arguments of Proposition 2, an honest verifier recomputes the same canonical value for the index i^*, hence the same leaf commitment h_{i^*}. Therefore, the verifier reconstructs the same root.

Relaxed 2-Special Soundness. Given two accepting transcripts $(\mathsf{root}, 0, (\rho, \mathsf{bits}))$ and $(\mathsf{root}, 1, (\mu, \mathsf{path}, r))$ for the same commitment and distinct challenges, set

$h_i = \mathcal{O}(\mathsf{bits}_i, \mathsf{Can}(\rho \star x_i))$ and $(\mathsf{root}_0, -) = \mathsf{MT.GetTree}(h_1, \ldots, h_N)$, then $\mathsf{root} = \mathsf{root}_0$. Also, if we set $h' = \mathcal{O}(r, \mathsf{Lift}(\mu, x))$, $\mathsf{root} = \mathsf{MT.ReconstructRoot}(h', \mathsf{path})$. The authentication path does not reveal the corresponding index, so the extractor $\mathcal{E}$ sets $h_i = \mathcal{O}(\mathsf{bits}_i, \mathsf{Lift}(\mathsf{Rep}(\rho), x_i))$ and checks whether $h_i \neq \mathcal{O}(r, \mathsf{Lift}(\mu, x))$ for every $i \in [N]$. If so, since $\mathsf{root}_0 = \mathsf{root} = \mathsf{MT.ReconstructRoot}(h', \mathsf{path})$, $\mathcal{E}$ finds a hash collision $\mathsf{H}_{\mathsf{Coll}}(s) = \mathsf{H}_{\mathsf{Coll}}(s')$ from the binding property of MT. If there is an index I such that $h_I = \mathcal{O}(r, \mathsf{Lift}(\mu, x))$ and $(r, \mathsf{Lift}(\mu, x)) \neq (\mathsf{bits}_I, \mathsf{Lift}(\mathsf{Rep}(\rho), x_I))$, then $\mathcal{E}$ finds a random oracle collision. Otherwise, the two inputs to $\mathcal{O}$ are equal, so $\mathsf{Lift}(\mu, x) = \mathsf{Lift}(\mathsf{Rep}(\rho), x_I)$, and it finds a witness $(\mu, \mathsf{Rep}(\rho), I)$ for $\tilde{\mathcal{R}}^{\mathsf{OR}}_{\mathsf{CL}}$.

Honest-Verifier Zero-Knowledge. We exhibit a simulator $\mathcal{S}$ producing an accepting transcript which is computationally indistinguishable from a transcript generated by an honest protocol execution. It takes as input a statement $(x, x_1, \ldots, x_N)$ for $\tilde{\mathcal{R}}^{\mathsf{OR}}_{\mathsf{CL}}$ and a challenge $\mathsf{ch} \in \{0, 1\}$. Formally, assume that $\mathcal{A}$ is a (computationally unbounded) adversary making Q calls to the random oracle $\mathcal{O}$. If $\mathsf{ch} = 0$ the simulator simply behaves like an honest prover. Let us focus on $\mathsf{ch} = 1$. The simulator samples $r_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$, $\rho \leftarrow\!\!\$\ G$, and computes $h_1 = \mathcal{O}(r_1, \mathsf{Lift}(\mathsf{Rep}(\rho), x))$. It again samples $h_i \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ for $i = 2, \ldots, N$. Then it computes $(\mathsf{root}, \mathsf{tree}) = \mathsf{MT.GetTree}(h_1, \ldots, h_N)$ and $\mathsf{path} = \mathsf{MT.GetPath}(\mathsf{tree}, 1)$. It returns the transcript $(\mathsf{root}, 1, (\mathsf{Rep}(\rho), \mathsf{path}, r_1))$. This is an accepting transcript by construction. Moreover, following the chain of simulators from [11, Theorem 3.3], we get

$$\left| \Pr\left[\mathcal{A}^{\mathcal{O}}(P^{\mathcal{O}}(x, w, 1)) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}}(\mathcal{S}^{\mathcal{O}}(x, 1)) = 1\right] \right| \leq \frac{2Q}{2^\lambda}$$

and, since the transcripts for $\mathsf{ch} = 0$ are indistinguishable, we obtain the thesis. □

3.3 OR Sigma Protocol with Tag via Canonical Lifting

In this section, we further extend the OR Sigma protocol $\Sigma^{\mathsf{OR}}_{\mathsf{CL}}$ to an OR Sigma protocol with tag $\Sigma^{\mathsf{OR\text{-}t}}_{\mathsf{CL}}$. Looking ahead to the construction of a linkable ring signature, the main idea of our approach is that the witness underlying the public statement and the witness underlying the tag should be distinct, so as to avoid reusing the same secret twice, while still remaining somehow linked. Compared with $\Sigma^{\mathsf{OR}}_{\mathsf{CL}}$, linking a tag forces a compatibility requirement between Rep and left multiplication by randomising elements. We call this property *left-compatibility*.

Definition 6. *Let $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ be a canonical lifting for a group action $(G, X, \star)$. We say that it is* left-compatible *if, for all $g, h, \rho \in G$,*

$$\mathsf{Rep}(g) = \mathsf{Rep}(h) \Rightarrow \mathsf{Rep}(\rho \circ g) = \mathsf{Rep}(\rho \circ h),$$

and one can efficiently sample, uniformly at random, from $[g]_{\mathsf{Rep}}$, for any $g \in G$.

In other words, if $\mathsf{Rep}(\rho \circ g) = \mathsf{Rep}(\rho \circ h)$, we have $[\rho \circ g]_{\mathsf{Rep}} = [\rho \circ h]_{\mathsf{Rep}}$, and one can efficiently sample random elements in G with a prescribed representation

via Rep. This allows to choose two distinct witnesses as public statement and tag, linked by their equivalence class. The relation for this Sigma protocol is

$$\mathcal{R}_{\mathsf{CL}}^{\mathsf{OR}-t} = \left\{ \langle (x, x_1, \ldots, x_N, t, t'), (h, h', i) \rangle \in X^{N+3} \times \left(R^2 \times [N]\right) \;\middle|\; \begin{array}{c} \mathsf{Lift}(h, x) = \mathsf{Lift}(h', x_i), \\ \mathsf{Lift}(h, t) = \mathsf{Lift}(h', t') \end{array} \right\}.$$

Like in the OR Sigma protocol, we prove relaxed 2-special-soundness for the following relaxed relation:

$$\tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}-t} = \left\{ \langle (x, x_1, \ldots, x_N, t, t'), w \rangle \;\middle|\; \begin{array}{c} (x, x_1, \ldots, x_N, t, t') \in X^{N+3} \text{ and either} \\ w = (h, h', i) : h, h' \in R, i \in [N] \text{ satisfying} \\ \mathsf{Lift}(h, x) = \mathsf{Lift}(h', x_i) \text{ and } \mathsf{Lift}(h, t) = \mathsf{Lift}(h', t'); \text{ or} \\ w = (s, s') : s \neq s', \mathsf{H}_{\mathsf{Coll}}(s) = \mathsf{H}_{\mathsf{Coll}}(s'), \text{ or } \mathcal{O}(s) = \mathcal{O}(s') \end{array} \right\}.$$

Witnesses for $\tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}-t}$ are either witnesses for the base relation $\mathcal{R}_{\mathsf{CL}}^{\mathsf{OR}-t}$, collisions for the hash function $\mathsf{H}_{\mathsf{Coll}}$, or collisions for the random oracle $\mathcal{O}$.

Proposition 4. *The Sigma protocol with tag $\Sigma_{\mathsf{CL}}^{\mathsf{OR\text{-}t}}$ of Figure 5 satisfies completeness, relaxed 2-special soundness, and honest-verifier zero-knowledge.*

Proof. Completeness. The completeness of the Merkle-tree part follows exactly as in the proof of Proposition 3. Therefore, it only remains to check the tag component. Let $i^* \in [N]$ be the prover's secret index, let $t_{i^*} = \tilde{w}_{i^*} \star t$ with $\tilde{w}_{i^*} \in [w_{i^*}]_{\mathsf{Rep}}$, and let $\rho \in G$ be the randomizer used in the commitment phase. If $\mathsf{ch} = 0$, the prover returns $\pi = \mathsf{Rep}(\rho)$ together with t_{i^*}. The verifier computes

$$\tau' = \mathsf{Lift}(\pi, t_{i^*}) = \mathsf{Lift}(\mathsf{Rep}(\rho), t_{i^*}) = \mathsf{Can}(\rho \star t_{i^*}) = \tau,$$

so the hash value is recomputed correctly. If $\mathsf{ch} = 1$, the prover returns $h = \mathsf{Rep}(\rho \circ w_{i^*})$. Since $\tilde{w}_{i^*} \in [w_{i^*}]_{\mathsf{Rep}}$, we have $\mathsf{Rep}(\tilde{w}_{i^*}) = \mathsf{Rep}(w_{i^*})$, and by left-compatibility, $\mathsf{Rep}(\rho \circ \tilde{w}_{i^*}) = \mathsf{Rep}(\rho \circ w_{i^*}) = \mu$. Therefore, the verifier computes

$$\tau' = \mathsf{Lift}(\mu, t) = \mathsf{Lift}(\mathsf{Rep}(\rho \circ \tilde{w}_{i^*}), t) = \mathsf{Can}((\rho \circ \tilde{w}_{i^*}) \star t) = \mathsf{Can}(\rho \star t_{i^*}) = \tau,$$

and again the hash value is recomputed correctly.

Relaxed 2-special soundness. The proof follows from the one of Proposition 3, in fact, we assume knowledge of two accepting transcripts $(\mathsf{H}_{\mathsf{Coll}}(\mathsf{root}, \tau), 0, (\pi, \mathsf{bits}))$ and $(\mathsf{H}_{\mathsf{Coll}}(\mathsf{root}, \tau), 1, (\mu, \mathsf{path}, r))$ for a statement $(x, x_1, \ldots, x_N, t, t')$ such that by setting $h_i = \mathcal{O}(\mathsf{bits}_i, \mathsf{Lift}(\pi, x_i))$, $(\mathsf{root}_0, -) = \mathsf{MT.GetTree}(h_1, \ldots, h_N)$, then $\mathsf{root} = \mathsf{root}_0$, and by setting $h' = \mathcal{O}(r, \mathsf{Lift}(\mu, x))$, it holds that $\mathsf{root} = \mathsf{root}_1$, where $\mathsf{root}_1 = \mathsf{MT.ReconstructRoot}(h', \mathsf{path})$. Setting $\tau_0 = \mathsf{Lift}(\pi, t')$ and $\tau_1 = \mathsf{Lift}(\mu, t)$, acceptance gives $\mathsf{H}_{\mathsf{Coll}}(\mathsf{root}_0, \tau_0) = \mathsf{H}_{\mathsf{Coll}}(\mathsf{root}_1, \tau_1)$. If $(\mathsf{root}_0, \tau_0) \neq (\mathsf{root}_1, \tau_1)$, the extractor obtains a collision for $\mathsf{H}_{\mathsf{Coll}}$. Otherwise, $\mathsf{root}_0 = \mathsf{root}_1$ and $\tau_0 = \tau_1$. If $h_i \neq h'$ for every $i \in [N]$, Merkle-tree binding, applied to tree_0 and (h', path), yields a collision for $\mathsf{H}_{\mathsf{Coll}}$. Otherwise, choose I such that $h_I = h'$. If the corresponding inputs to $\mathcal{O}$ differ, this equality is a random oracle collision. In the remaining case, $\mathsf{Lift}(\mu, x) = \mathsf{Lift}(\pi, x_I)$ and $\mathsf{Lift}(\mu, t) = \tau_1 = \tau_0 = \mathsf{Lift}(\pi, t')$, so (μ, π, I) is a witness for $\tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}-t}$.

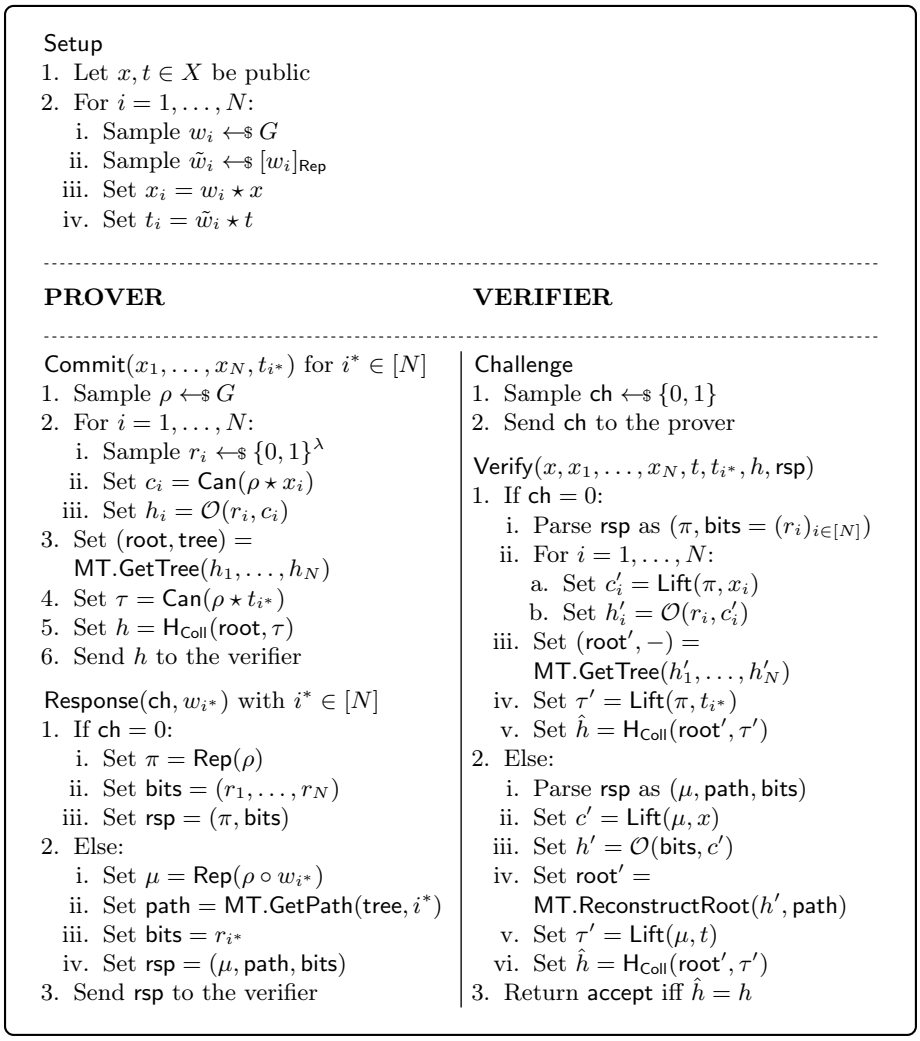

Fig. 5: OR Sigma protocol with tag $\Sigma_{\mathsf{CL}}^{\mathsf{OR\text{-}t}}$.

Honest-verifier zero-knowledge. We build our simulator $\mathcal{S}$ on the simulator $\mathcal{S}'$ given in the proof of Proposition 3. Let $(x, x_1, \ldots, x_N, t, t')$ be a statement of the relation $\tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}-t}$. If $\mathsf{ch} = 0$, the simulator $\mathcal{S}$ produces an indistinguishable transcript following the instructions of the real prover since the latter does not use the witness. Whenever $\mathsf{ch} = 1$, $\mathcal{S}$ runs $\mathcal{S}'$ on $((x, x_1, \ldots, x_N), \mathsf{ch})$, obtaining $(\mathsf{root}, 1, (\mu, \mathsf{path}, r))$, with $\mu \in R$ and $r \in \{0,1\}^{2\lambda}$ such that $\mathsf{root} = \mathsf{root}'$, where

$$\mathsf{root}' = \mathsf{MT.ReconstructRoot}(\mathcal{O}(r, \mathsf{Lift}(\mu, x)), \mathsf{path}).$$

Then the simulator $\mathcal{S}$ returns the transcript $(\mathsf{H}_{\mathsf{Coll}}(\mathsf{root}, \mathsf{Lift}(\mu, t)), 1, (\mu, \mathsf{path}, r))$, which is accepting by construction and, assuming that $\mathcal{A}$ is a (computationally unbounded) adversary making Q calls to the random oracle $\mathcal{O}$, we obtain that

$$\Big|\Pr\big[\mathcal{A}^{\mathcal{O}}(P^{\mathcal{O}}(x, w, \mathsf{ch})) = 1\big] - \Pr\big[\mathcal{A}^{\mathcal{O}}(\mathcal{S}^{\mathcal{O}}(x, \mathsf{ch})) = 1\big]\Big| \leq \frac{2Q}{2^{\lambda}}$$

for any prover $\mathcal{P}$, we have $\langle x, w\rangle \in \tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}-t}$ and $\mathsf{ch} \in \{0, 1\}$. □

3.4 On Constructing Canonical Lifting-based Signatures

Applying standard parallel repetition for soundness amplification, followed by the Fiat–Shamir transform, to Σ_{CL}, $\Sigma_{\mathsf{CL}}^{\mathsf{OR}}$, and $\Sigma_{\mathsf{CL}}^{\mathsf{OR\text{-}t}}$ yields, respectively, an ordinary signature, a ring signature, and a linkable variant. We focus on the latter, which we study in detail in Section 4, and do not spell out the other two constructions to avoid redundancy. Nevertheless, we provide some observations and results concerning the scheme derived from Σ_{CL}, which can itself be viewed as an abstraction of LESS. In particular, its underlying computational problem can be read directly from the witness relation in Equation (2) and is defined as follows.

Problem 2. *Let* $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ *be a canonical lifting for a group action* $(G, X, \star)$. *Given* $x, x' \in X$ *the* Canonical Lifting Collision Problem (CLCP) *requires finding, if they exist, two representatives* $r, r' \in R$ *such that*

$$\mathsf{Lift}(r, x) = \mathsf{Lift}(r', x').$$

Next, we provide a sufficient condition for reducing CLCP to GAIP.

Definition 7. *We say that a canonical lifting* $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ *for a group action* $(G, X, \star)$ *is* effective *if:*
i) given $r \in R$, *it is efficient to find an element* $g \in G$ *such that* $\mathsf{Rep}(g) = r$;
ii) given $x, x' \in X$ *such that* $\mathsf{Can}(x) = \mathsf{Can}(x')$, *it is efficient to find, if it exists,* $h \in G$ *such that* $x' = h \star x$.

For an effective canonical lifting, representatives in R can be efficiently lifted back to compatible group elements in G, and collisions of canonical values can be converted into explicit group-action witnesses. In this sense, CLCP treats the representatives in R as the effective secret space, rather than the full group G.

Lemma 1. *Let* $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ *be an effective canonical lifting for a group action* $(G, X, \star)$. *Then CLCP and GAIP are polynomial-time equivalent.*

Proof. (CLCP $\leq_{\mathsf{poly}}$ GAIP) Let (x, x') be an instance of CLCP. Using an oracle for GAIP, find $g \in G$ such that $x' = g \star x$. By Equation (1), we have

$$\mathsf{Lift}(\mathsf{Rep}(g), x) = \mathsf{Can}(g \star x) = \mathsf{Can}(x') = \mathsf{Lift}(\mathsf{Rep}(e), x'),$$

where e is the identity of G. Hence $(\mathsf{Rep}(g), \mathsf{Rep}(e))$ solves the instance.
(GAIP $\leq_{\mathsf{poly}}$ CLCP) Let (x, x') be an instance of GAIP. Using an oracle for

CLCP, find $r, r' \in R$ such that $\mathsf{Lift}(r, x) = \mathsf{Lift}(r', x')$. Since the canonical lifting is effective, we can efficiently find $g, g' \in G$ with $\mathsf{Rep}(g) = r$ and $\mathsf{Rep}(g') = r'$, so

$$\mathsf{Can}(g \star x) = \mathsf{Lift}(\mathsf{Rep}(g), x) = \mathsf{Lift}(\mathsf{Rep}(g'), x') = \mathsf{Can}(g' \star x').$$

By effectiveness, we can efficiently find $h \in G$ such that $g' \star x' = h \star (g \star x) = (h \circ g) \star x$. Hence $x' = \big((g')^{-1} \circ h \circ g\big) \star x$, and $(g')^{-1} \circ h \circ g$ solves the instance. □

Lemma 1 shows that if a canonical lifting is effective, it can be used to construct signature schemes from group actions without weakening the underlying problem.

4 Linkable Ring Signatures from Canonical Lifting

We construct a linkable ring signature $\mathsf{LRS}_{\mathsf{CL}}$ by applying the Fiat–Shamir transform to the OR Sigma protocol with tag $\Sigma_{\mathsf{CL}}^{\mathsf{OR\text{-}t}}$ from Section 3.3, and we study the underlying canonical lifting hardness assumptions to prove linkability, linkable anonymity, and non-frameability.

First, note that $\Sigma_{\mathsf{CL}}^{\mathsf{OR\text{-}t}}$ has soundness error at most $1/2$. This means that a cheating prover, one who does not know a valid witness, can convince the verifier with probability at most $1/2$. Repeating λ executions of the protocol reduces this bound to $1/2^{\lambda}$, where λ denotes the target bit-security level. Then, by applying the Fiat–Shamir transform to $\Sigma_{\mathsf{CL}}^{\mathsf{OR\text{-}t}}$, together with some classical optimisations techniques presented in Appendix B.1, one obtains the linkable ring signature $\mathsf{LRS}_{\mathsf{CL}}$ reported in Figure 6. Observe that the key generation algorithm requires sampling efficiently from the equivalence class $[w]_{\mathsf{Rep}}$ of an element $w \in G$.

4.1 Computational Problems

Before proving the properties of $\mathsf{LRS}_{\mathsf{CL}}$, we need to introduce the underlying computational problems induced by canonical liftings. Throughout this subsection, the public base elements x, t are output by a polynomial-time parameter-generation algorithm on input 1^{λ}, which may be deterministic.

Definition 8. *Let $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ be a left-compatible canonical lifting for a group action $(G, X, \star)$. We say that it is* non-splitting (NS) *if, for fixed $x, t \in X$, for any PPT algorithm $\mathcal{A}$, the following is negligible in λ*

$$\Pr\left[\begin{array}{c|l} \begin{array}{c}\mathsf{Lift}(g_0, x) = \mathsf{Lift}(h_0, x') \wedge \mathsf{Lift}(g_1, x) = \mathsf{Lift}(h_1, x'),\\ \mathsf{Lift}(g_0, t) = \mathsf{Lift}(h_0, t_0) \wedge \mathsf{Lift}(g_1, t) = \mathsf{Lift}(h_1, t_1),\\ \mathsf{Can}(t_0) \neq \mathsf{Can}(t_1)\end{array} & (g_0, g_1, h_0, h_1, x', t_0, t_1) \leftarrow \mathcal{A}(x, t) \end{array}\right].$$

This property prevents an adversary from finding two pairs (g_0, h_0) and (g_1, h_1) that relate x to the same element x' through Lift, while relating t to two elements t_0, t_1 with distinct canonical values. The following proposition gives the conditions such that breaking the non-splitting assumption is equivalent to finding an element of R that stabilizes x but not t.

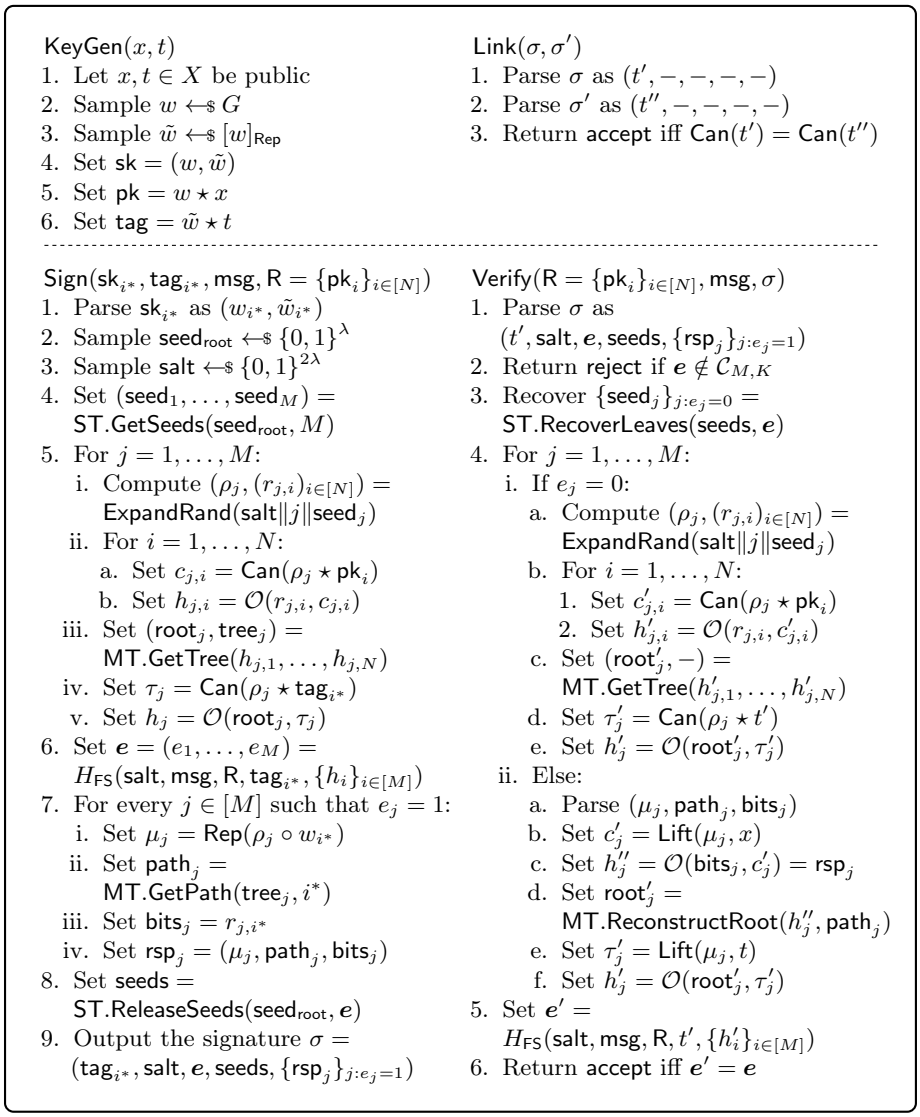

Fig. 6: Linkable ring signature $\mathsf{LRS}_{\mathsf{CL}}$.

Proposition 5. *Let* $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ *be a canonical lifting for a group action* $(G, X, \star)$ *and let* R *be a subgroup of* $G = R \ltimes H$*, with* H *normal subgroup of* G*. Suppose that* Can *is a canonical form for the relation induced by the action of* H*, then breaking the non-splitting property on elements* x, t *is equivalent to finding in* R *an automorphism of* x *which is not an automorphism of* t*.*

Proof. Using the notation of [33], we can consider the quotient group action $(R, X/H, \star_R)$. In fact, we have that $C \subset X$, Can is a representative of the orbit defined by the action of H, and $\mathsf{Lift}(g, y) = \mathsf{Can}(g \star_R y) \in X$.

Let x, t in X. We show that finding an automorphism in R of x which is not an automorphism of t can be used to break the non-splitting property. Let $h \in R$ be such that $h \star_R x = x$, while $h \star_R t \neq t$. Then, sampling g, g_0, g_1 from R, a solution for the task in Definition 8 is given by $(g_0 g, g_1 g h, g_0, g_1, x' = \mathsf{Can}(g \star_R x), t_0 = g \star_R t, t_1 = (gh) \star_R t)$. In fact, we have $\mathsf{Lift}(g_0 g, x) = \mathsf{Lift}(g_0, x')$ and $\mathsf{Lift}(g_1 g h, x) = \mathsf{Lift}(g_1, x')$ while $\mathsf{Lift}(g_0 g, t) = \mathsf{Lift}(g_0, t_0)$ and $\mathsf{Lift}(g_1 g h, t) = \mathsf{Lift}(g_1, t_1)$. Additionally, assume that $\mathsf{Can}(t_0) = \mathsf{Can}(t_1)$. Then there is an element $h' \in H$ such that $t_1 = h' \star t_0$, and hence, $(gh) \star t = (h'g) \star t = (g g^{-1} h' g) \star t$. Therefore, $h \star t = (g^{-1} h' g) \star t$, and since $g^{-1} h' g \in H$ by definition of semi-direct product, we get the contradiction $h \star_R t = t$. Hence, $\mathsf{Can}(t_0) \neq \mathsf{Can}(t_1)$.

The converse is as follows. Let $x, t \in X$, we want to find an automorphism of x which is not an automorphism of t. If $(g_0, g_1, h_0, h_1, x', t_0, t_1)$ is a solution for the task in the NS property, we have $x' = (h_0^{-1} g_0) \star_R x = (h_1^{-1} g_1) \star_R x$, $t_0 = (h_0^{-1} g_0) \star_R t$, and $t_1 = (h_1^{-1} g_1) \star_R t$, with $\mathsf{Can}(t_0) \neq \mathsf{Can}(t_1)$. Therefore $g = (h_1^{-1} g_1)^{-1} (h_0^{-1} g_0)$ is an automorphism of x. We need to show that it is non-trivial and $g \star_R t \neq t$. If $g = e$, then $h_1^{-1} g_1 = h_0^{-1} g_0$, hence $t_0 = t_1$, contradicting $\mathsf{Can}(t_0) \neq \mathsf{Can}(t_1)$, therefore g is non-trivial. Moreover, suppose that $g \star_R t = t$, then $t_0 = (h_0^{-1} g_0) \star_R t = (h_1^{-1} g_1) \star_R t = t_1$, which is another contradiction. □

If x has trivial automorphism group, even an unbounded adversary is unable to break the non-splitting property. However, finding an automorphism of a single element is considered as hard as GAIP [14] and, to obtain (standard) group action-based signature schemes which are secure in the QROM, finding a non-trivial automorphism must be infeasible [14].

Definition 9. *Let* $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ *be a left-compatible canonical lifting for a group action* $(G, X, \star)$*. We say that it is* 2-pseudorandom *if, for fixed* $x, t \in X$*, the distributions of the following two pairs*

$$(g \star x, \widetilde{g} \star t)\,, \qquad (g_0 \star x, g_1 \star t)\,,$$

are computationally indistinguishable, where $g, g_0, g_1 \leftarrow\!\$\; G$ *and* $\widetilde{g} \leftarrow\!\$\; [g]_{\mathsf{Rep}}$*.*

Definition 10. *Let* $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ *be a left-compatible canonical lifting for a group action* $(G, X, \star)$*. We say that it is* frame one-way *if, for fixed* $x, t \in X$*, for every PPT adversary* $\mathcal{A}$ *outputting* $y, z \in X$ *and* $h, h' \in R$*, the following probability is negligible in* λ*:*

$$\Pr\left[\begin{array}{l}\mathsf{Lift}(h,x)=\mathsf{Lift}(h',y)\\ \mathsf{Lift}(h,t)=\mathsf{Lift}(h',z),\\ \mathsf{Can}(z)=\mathsf{Can}(\widetilde{g}\star t)\end{array}\;\middle|\;\begin{array}{l} g\leftarrow\!\!\$\, G,\quad \widetilde{g}\leftarrow\!\!\$\, [g]_{\mathsf{Rep}},\\ (y,z,h,h')\leftarrow\mathcal{A}(x,t,g\star x,\widetilde{g}\star t)\end{array}\right].$$

4.2 Security Properties of the Linkable Ring Signature $\mathsf{LRS}_{\mathsf{CL}}$

Theorem 1. *If $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ is non-splitting and left-compatible for a group action $(G, X, \star)$, and $\mathsf{H}_{\mathsf{Coll}}$ is collision resistant, then the linkable ring signature $\mathsf{LRS}_{\mathsf{CL}}$ achieves linkability in the random oracle model.*

Proof. To prove the linkability property of $\mathsf{LRS}_{\mathsf{CL}}$, we assume that there is a PPT adversary $\mathcal{A}$ against the linkability game having advantage ϵ, running time t, and making at most Q queries to the random oracle. We design a PPT adversary $\mathcal{B}$ that breaks the non-splitting property of the canonical lifting or finds a collision for the hash function $\mathsf{H}_{\mathsf{Coll}}$.

The strategy is the following: $\mathcal{B}$ runs $\mathcal{A}$ until it wins the linkability game, producing a list of public keys $\{\mathsf{pk}_i\}_{i\in[N]}$ and signatures $\{\sigma_i\}_{[N+1]}$. Then, $\mathcal{B}$ rewinds $\mathcal{A}$ until it is able to extract a witness for the relaxed relation $\tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}-t}$ from each signature σ_i. Then, depending on the structure of the obtained witnesses w_i, $\mathcal{B}$ finds a collision for $\mathsf{H}_{\mathsf{Coll}}$ of breaks the non-splitting property. The adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine, providing the randomness needed. For $\ell \in [Q]$, we define as $\mathsf{rand}_{<\ell}$ the randomness used by $\mathcal{B}$ (and hence $\mathcal{A}$) before the ℓ-th call by $\mathcal{A}$ to the random oracle, and as $\mathsf{rand}_{\geq\ell}$ the randomness used after the ℓ-th call.

1. $\mathcal{B}$ runs $\mathcal{A}$ until it wins the game. Hence, $\mathcal{A}$ returns a list of N public keys $\{\mathsf{pk}_i\}_{i\in[N]}$ and a list of $N+1$ tuples $\{(\sigma_i, \mathsf{msg}_i, \mathsf{R}_i)\}_{i\in[N+1]}$. Here, we can forget the salt and assume that $\sigma_i = (t_i, \mathsf{ch}_i, \mathsf{rsp}_i)$. Moreover, let $h_i \in [Q]$ be the index such that the signature σ_i is based on the h_i-th random oracle query, i.e. $\mathsf{ch}_i = \mathcal{O}(\mathsf{msg}, \mathsf{R}_i, t_i, \mathsf{com}_i)$ for a suitable commitment com_i. Then, $\mathcal{B}$ set the list $L = \{(i, \mathsf{msg}_i, \mathsf{R}_i, t_i, \mathsf{com}_i, \mathsf{ch}_i, \mathsf{rsp}_i)\}_{i\in[N+1]}$.
2. For every $j \in [N+1]$, $\mathcal{B}$ rewinds $\mathcal{A}$ before the h_j-th call to the random oracle and uses a fresh randomness $\mathsf{rand}'_{\geq h_j}$ until $\mathcal{A}$ wins the linkability game again, returning the list of N public keys $\{\mathsf{pk}'_i\}_{i\in[N]}$ and the list of $N+1$ tuples $\{(\sigma'_i, \mathsf{msg}'_i, \mathsf{R}'_i)\}_{i\in[N+1]}$. If the signature σ'_j is not based on the h_j-th oracle call, $\mathcal{B}$ rewinds $\mathcal{A}$ and uses a fresh randomness $\mathsf{rand}'_{\geq h_j}$. Otherwise, the list L is updated with the tuple $(j, \mathsf{msg}'_j, \mathsf{R}'_j, t'_j, \mathsf{com}'_j, \mathsf{ch}'_j, \mathsf{rsp}'_j)$. Since the randomness used by $\mathcal{A}$ before the h_j-th call is fixed, i.e., $\mathsf{rand}_{<h_j}$, we have that $(\mathsf{msg}_j, \mathsf{R}_j, t_j, \mathsf{com}_j) = (\mathsf{msg}'_j, \mathsf{R}'_j, t'_j, \mathsf{com}'_j)$.
3. For each $j \in [N+1]$, $\mathcal{B}$ collects two tuples with the same index j, namely $(j, \mathsf{msg}_j, \mathsf{R}_j, t_j, \mathsf{com}_j, \mathsf{ch}_j, \mathsf{rsp}_j)$ and $(j, \mathsf{msg}_j, \mathsf{R}_j, t_j, \mathsf{com}_j, \mathsf{ch}'_j, \mathsf{rsp}'_j)$. If $\mathsf{ch}_j = \mathsf{ch}'_j$, then $\mathcal{B}$ aborts. Otherwise, it runs the extractor for the relaxed $\tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}-t}$ on the statement $(x, \mathsf{R}_j, t, t_j)$, where $\mathsf{R}_j = (x_{j,1}, \ldots, x_{j,|\mathsf{R}_j|})$, obtaining the witness w_j.
4. The pigeonhole principle guarantees the existence of two indices $i, j \in [N+1]$ such that w_i and w_j are witnesses for the (non-relaxed) relation $\mathcal{R}_{\mathsf{CL}}^{\mathsf{OR}-t}$ with the same index in $[N]$, namely $w_i = (h_i, h'_i, I)$ and $w_j = (h_j, h'_j, I)$, $\mathcal{B}$ outputs

$(h_i, h_j, h'_i, h'_j, x_I, t_i, t_j)$. If for some $j \in [N+1]$ the witness w_j is a collision for $\mathsf{H}_{\mathsf{Coll}}$, $\mathcal{B}$ returns w_j. Otherwise, the extractor returned a collision for the random oracle and $\mathcal{B}$ aborts.

On the success of $\mathcal{B}$. We show that, when the output of $\mathcal{B}$ is not a collision for $\mathsf{H}_{\mathsf{Coll}}$, it is a solution for the problem in Definition 8. In fact, given the output $(h_i, h_j, h'_i, h'_j, x_I, t_i, t_j)$, since (h_i, h'_i, I) is a witness for $(x, x_{i,1}, \ldots, x_{i,|\mathsf{R}_i|}, t, t_i)$,

$$\mathsf{Lift}(h_i, x) = \mathsf{Lift}(h'_i, x_I) \wedge \mathsf{Lift}(h_i, t) = \mathsf{Lift}(h'_i, t_i).$$

Analogously, since (h_j, h'_j, I) is a witness for $(x, x_{j,1}, \ldots, x_{j,|\mathsf{R}_j|}, t, t_j)$, we have

$$\mathsf{Lift}(h_j, x) = \mathsf{Lift}(h'_j, x_I) \wedge \mathsf{Lift}(h_j, t) = \mathsf{Lift}(h'_j, t_j).$$

Moreover, since $\mathsf{Link}(t_i, t_j) = \mathsf{reject}$, we have that $\mathsf{Can}(t_i) \neq \mathsf{Can}(t_j)$. This implies that $\mathcal{B}$ breaks the non-splitting property of the canonical lifting.

Running time of $\mathcal{B}$. We now show that $\mathcal{B}$ runs in expected polynomial time $\mathcal{O}(tN^2Q/\epsilon)$, breaking the non-splitting or finding a collision with probability close to 1. Consider the four execution steps of $\mathcal{B}$.

1. $\mathcal{A}$ terminates with expected time $\mathcal{O}(t/\epsilon)$, and so does $\mathcal{B}$ in point 1.
2. We use a variant of the *heavy row lemma* [44]. Recall the original formulation. For $j \in [N+1]$ let $\mathsf{rand}_{<h_j} || \mathsf{rand}_{\geq h_j}$ be the randomness used. Then, if $\mathcal{A}$ wins, conditioned on the randomness $\mathsf{rand}_{<h_j}$, with probability ϵ/Q, then, rewinding with a new randomness $\mathsf{rand}'_{\geq h_j}$ gives another win with probability $\epsilon/2Q$. In this case, we say that $\mathsf{rand}_{<h_j}$ is *heavy*. We need that this event occurs simultaneously for each index in $[N+1]$.
   To this end, we apply the lemma in a finer way: instead of guaranteeing that $\mathsf{rand}_{<h_j}$ is *heavy* except with probability $1/2$, we ask for this to fail with probability at most $1/2(N+1)$, at the cost of going from the starting probability of $\epsilon/2Q$ to $\epsilon/2(N+1)Q$. In practice, for a fixed $j \in [N+1]$, with probability at least $1 - 1/2(N+1)$, the randomness $\mathsf{rand}_{<h_j}$ is such that resampling a new randomness $\mathsf{rand}'_{\geq h_j}$ gives another win on the same h_j-th query with probability at least $\epsilon/2(N+1)Q$. Taking a union bound over all the indexes $j \in [N+1]$, the probability that the above fails for some j is

$$(N+1)\frac{1}{2(N+1)} = \frac{1}{2}.$$

   Conditioned on the event that all $N+1$ prefixes are heavy, which occurs with probability at least $1/2$, the success probability of each rewind is at least $\epsilon/2(N+1)Q$. Consequently, the expected number of rewindings over all indices is at most $2(N+1)^2Q/\epsilon$, and their expected running time is $\mathcal{O}(tN^2Q/\epsilon)$. By truncating this phase after a constant multiple of its conditional expected running time and restarting the entire experiment upon failure, Markov's inequality shows that each experiment succeeds with constant probability. Hence, only $\mathcal{O}(1)$ experiments are expected, and the total expected running time remains $\mathcal{O}(tN^2Q/\epsilon)$.

3. $\mathcal{B}$ aborts if $\mathsf{ch}_j = \mathsf{ch}'_j$ for some j, which happens with negligible probability $\mathcal{O}(Q/2^\lambda)$ since the output of the random oracle is randomly sampled.
4. The extractor $\mathcal{E}$ for $\tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}-t}$ runs in polynomial time, therefore so does $\mathcal{B}$. Exactly as above, it aborts with negligible probability $\mathcal{O}(Q/2^\lambda)$. □

Theorem 2. *If $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ is 2-pseudorandom and left-compatible for a group action $(G, X, \star)$, then the linkable ring signature $\mathsf{LRS}_{\mathsf{CL}}$ achieves linkable anonymity in the random oracle model.*

Proof. Let $\mathcal{A}$ be a PPT adversary for the linkable anonymity game with advantage ϵ. We use an hybrid argument of five games; the first is the linkable anonymity one, the last is independent from the choice of the bit b and the advantage of any adversary is zero. Denote by E_i the event that $\mathcal{A}$ wins game Game_i.

- Game_1 is the linkable anonymity game and $\epsilon = |\Pr[\mathsf{E}_1] - 1/2|$ by definition.
- Game_2 is a modification where, when the adversary $\mathcal{A}$ performs a signature query $(\mathsf{sign}, \mathsf{pk}, \mathsf{msg}, \mathsf{R})$, with $\mathsf{pk} \in \{\mathsf{pk}_0^*, \mathsf{pk}_1^*\} \subseteq \mathsf{R}$, the challenger samples a challenge ch and programs the random oracle so that $\mathsf{H}_{\mathrm{FS}}(\mathsf{msg}, \mathsf{R}, t', \mathsf{com}) = \mathsf{ch}$. It aborts if this was already queried. Otherwise, it continues as in Game_1. This induces at most a negligible difference, so $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\lambda)$.
- Game_3 differs in how $\mathcal{C}$ creates a signature for the query $(\mathsf{sign}, \mathsf{pk}, \mathsf{msg}, \mathsf{R})$, with $\mathsf{pk} \in \{\mathsf{pk}_0^*, \mathsf{pk}_1^*\} \subseteq \mathsf{R} = \{x_1, \ldots, x_{|\mathsf{R}|}\}$. It retrieves the secret key $\mathsf{sk} = (w, \tilde{w})$ of pk and sets $t' = \tilde{w} \star t$. Then, after sampling a random challenge ch, it runs the simulator $\mathcal{S}$ from the proof of Proposition 4 to obtain $\mathcal{S}((x, x_1, \ldots, x_{|\mathsf{R}|}, t, t'), \mathsf{ch}) = (\mathsf{com}, \mathsf{chrsp})$ and programs the random oracle so that $\mathsf{H}_{\mathrm{FS}}(\mathsf{msg}, \mathsf{R}, t', \mathsf{com}) = \mathsf{ch}$. It returns the tuple $(t', \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$. By honest-verifier zero-knowledge of the underlying Sigma protocol, the signature is indistinguishable from a real one and $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\lambda)$.
- In Game_4, the challenger samples N key-pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ and then guesses the indexes (i_0, i_1) chosen by $\mathcal{A}$ in the linkable anonymity game. It generates the two tags t_0 and t_1 corresponding to the i_0-th and i_1-th key pairs, respectively. If the guess on i_0 and i_1 is incorrect, then it aborts. Otherwise, it acts as in Game_3. In this game, the challenger does not need the secret keys during the signing queries. Since it guesses the indexes i_0 and i_1 with probability $1/N^2$, we have $N^2|\Pr[\mathsf{E}_4] - 1/2| = |\Pr[\mathsf{E}_3] - 1/2|$.
- In the last game Game_5, the challenger guesses i_0, i_1 and then samples all the remaining keypairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ for $i \in [N] \setminus \{i_0, i_1\}$. For $b \in \{0, 1\}$, it samples independent elements $g_b, g'_b \leftarrow\!\!\$\ G$ and sets $x_{i_b} = g_b \star x$, $t_{i_b} = g'_b \star t$. It then continues as in the previous game. By 2-pseudorandomness, the honestly generated pair $(x_{i_b}, t_{i_b}) = (g \star x, \widetilde{g} \star t)$, with $\widetilde{g} \leftarrow\!\!\$\ [g]_{\mathsf{Rep}}$, is indistinguishable from the independently generated pair $(g_b \star x, g'_b \star t)$. Observe that the 2-pseudorandom assumption is used twice, for $b = 0$ and $b = 1$. Hence, $|\Pr[E_4] - \Pr[E_5]| = \mathsf{negl}(\lambda)$. Additionally, we have that the challenge public keys and tags are independently sampled and the signing key is no longer used to sign messages. Hence, the bit b remains information-theoretically hidden from $\mathcal{A}$, thus $\Pr[E_5] = 1/2$.

Summing up, we have that $\epsilon = \mathsf{negl}(\lambda)$. □

Theorem 3. *If* $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ *is frame one-way and left-compatible for a group action* $(G, X, \star)$*, and* $\mathsf{H}_{\mathsf{Coll}}$ *is collision resistant, then the linkable ring signature* $\mathsf{LRS}_{\mathsf{CL}}$ *achieves non-frameability in the random oracle model.*

Proof. We show that the existence of an efficient adversary $\mathcal{A}$ against the non-frameability game with advantage ϵ and running time t implies the existence of an efficient adversary $\mathcal{B}$ against either the frame one-way property or the collision resistance of $\mathsf{H}_{\mathsf{Coll}}$. Consider the following sequence of games, where E_i denotes the event that $\mathcal{A}$ wins Game_i.

- Game_1 is the non-frameability game. Hence $\Pr[\mathsf{E}_1] = \epsilon$.
- In Game_2, when $\mathcal{A}$ performs a signature query $(\mathsf{sign}, i, m, \mathsf{R})$, the challenger programs the random oracle to return a random $\mathsf{H}_{\mathsf{FS}}(\mathsf{msg}, \mathsf{R}, t', \mathsf{com}) = \mathsf{ch}$. It aborts if such inputs were already queried, otherwise, it continues as in Game_1. Hence, we have $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\lambda)$.
- Game_3 differs from Game_2 in how the challenger creates a signature for the query $(\mathsf{Sign}, i, \mathsf{pk}, \mathsf{msg}, \mathsf{R})$, with $\mathsf{pk}_i \in \mathsf{R} = \{x_1, \ldots, x_{|\mathsf{R}|}\}$. It retrieves the secret key $\mathsf{sk}_i = (w, \tilde{w}))$ of pk_i and sets $t' = \tilde{w} \star t$. Then, after sampling a random challenge ch, it runs the simulator $\mathcal{S}$ from the proof of Proposition 4 to obtain $\mathcal{S}((x, x_1, \ldots, x_{|\mathsf{R}|}, t, t'), \mathsf{ch}) = (\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$ and programs the random oracle such that $\mathsf{H}_{\mathrm{FS}}(\mathsf{msg}, \mathsf{R}, t', \mathsf{com}) = \mathsf{ch}$. It returns as a signature the tuple $(t', \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$. By the honest-verifier zero-knowledge of the underlying Sigma protocol, we have that the signature is indistinguishable from a real one and $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\lambda)$.
- In Game_4, the challenger, after generating all the keypairs in the ring $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, with $\mathsf{sk}_i = (w_i, \tilde{w}_i)$, computes and stores the tags $t_i = \tilde{w}_i \star t$. When it receives a signing query, it retrieves the corresponding tag and proceeds as in Game_3, hence $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_3]$.

Now let $\epsilon' = \Pr[\mathsf{E}_4]$ and let Q be the number of queries $\mathcal{A}$ does to the random oracle. We show how to design an adversary $\mathcal{B}$ using $\mathcal{A}$ as a subroutine, with advantage $\mathcal{O}(\epsilon'/N)$ and running in time $\mathcal{O}(tQ/\epsilon')$ breaking the frame one-way property of the canonical lifting or the collision resistance of $\mathsf{H}_{\mathsf{Coll}}$.

The behaviour of $\mathcal{B}$ is as follows. On input a frame one-way challenge $(x, t, x', t') = (x, t, g \star x, \widetilde{g} \star t)$, with $g \leftarrow\!\!\$\ G$ and $\widetilde{g} \leftarrow\!\!\$\ [g]_{\mathsf{Rep}}$, it samples a random j in $[N]$ and generates the keypairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ for $i \in [N] \setminus \{j\}$. Moreover, it sets $\mathsf{pk}_j = x'$ and $t_j = t'$. Now, $\mathcal{B}$ can simulate the challenger of Game_4, since it has all the material needed. When $\mathcal{A}$ wins, it returns $(\mathsf{R}^*, \mathsf{msg}^*, \sigma^*)$ such that $\sigma^* = (t^*, \mathsf{com}^*, \mathsf{ch}^*, \mathsf{rsp}^*)$. From the winning condition, $\mathcal{B}$ must have produced a signature $(\mathsf{ch}', \mathsf{rsp}')$ and a tag $\tilde{t}$ on the signing query $(i, \mathsf{msg}, \mathsf{R})$ where $\mathsf{pk}_i \in PK \setminus CL$ and $\mathsf{LRS.Link}(t^*, t') = \mathsf{accept}$, implying $\mathsf{Can}\left(\tilde{t}\right) = \mathsf{Can}(t^*)$. If $i \neq j$, then $\mathcal{B}$ aborts. Otherwise, we have $\mathsf{Can}\left(t_j\right) = \mathsf{Can}(t^*)$ and $\mathcal{B}$ reruns $\mathcal{A}$ to extract a witness w for the statement $(x, \mathsf{pk}_1, \ldots, \mathsf{pk}_N, t, t_j)$ in the relaxed relation $\tilde{\mathcal{R}}_{\mathsf{CL}}^{\mathsf{OR}-t}$ as done in the proof of Theorem 1. If w is a collision for $\mathsf{H}_{\mathsf{Coll}}$, we are done. Otherwise, if w is not a collision for $\mathsf{H}_{\mathsf{Coll}}$ or, with negligible probability, for the random oracle, the witness is of the form (h, h', j) such that

$\mathsf{Lift}(h, x) = \mathsf{Lift}(h', \mathsf{pk}_j)$ and $\mathsf{Lift}(h, t) = \mathsf{Lift}(h', t_j)$ and, since $\mathsf{pk}_j = x'$, the tuple (x', t_j, h, h') satisfies Definition 10. □

5 CELeReS: Code Equivalence Linkable Ring Signature

In this section, we instantiate the linkable ring signature $\mathsf{LRS}_{\mathsf{CL}}$ using a canonical lifting for the code-equivalence group action. We then analyse the security assumptions for the resulting protocol, CELeReS, and how they relate to known problems from coding theory. We provide the full instantiation of CELeReS in Appendix C and discuss its efficiency in Section 6.

5.1 A Left-compatible Canonical Lifting for LEP

Since the canonical lifting used in LESS is not left-compatible, we cannot adapt it to the OR protocol with tag of Section 3.3. We introduce a different canonical lifting for LEP, which is inspired by constructions proposed in various forms in [33,31,30]. In this instantiation $G = \mathsf{GL}_k(\mathbb{F}_q) \times \mathsf{Mono}_n(\mathbb{F}_q)$, $R = \mathsf{Perm}_n(\mathbb{F}_q)$, X is the set of full-rank matrices in $\mathbb{F}_q^{k\times n}$, $C \subset \mathbb{F}_q^{k\times n}$ is the set of canonical forms, and $(G, X, \star)$ is the monomial action $(\boldsymbol{S}, \boldsymbol{Q}) \star \boldsymbol{G} = \boldsymbol{SGQ}$.

Representative function. We instantiate Rep with the following map

$$\mathsf{PermRep} : \mathsf{GL}_k(\mathbb{F}_q) \times \mathsf{Mono}_n(\mathbb{F}_q) \to \mathsf{Perm}_n(\mathbb{F}_q), \quad (\boldsymbol{S}, \boldsymbol{Q} = \boldsymbol{PD}) \mapsto \boldsymbol{P},$$

that is, returning the underlying permutation of the monomial and ignoring the left-component. The induced equivalence relation identifies monomial matrices that differ only by a diagonal factor. The key idea is to use, for each signer, distinct monomial witnesses for the public key and the tag, but with the same underlying permutation. Namely, we use $\boldsymbol{Q}_i = \boldsymbol{P}_i\boldsymbol{D}_i$ for the public key and $\widetilde{\boldsymbol{Q}}_i = \boldsymbol{P}_i\widetilde{\boldsymbol{D}}_i$ for the tag. Hence, $\mathsf{PermRep}(\boldsymbol{Q}_i) = \mathsf{PermRep}(\widetilde{\boldsymbol{Q}}_i) = \boldsymbol{P}_i$, allowing the same representative to be checked against both the public key and the tag.

Canonical function. The computation of Can consists of three steps.

1. On input $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$, compute $\boldsymbol{R} = \mathsf{RREF}(\boldsymbol{G})$ and return $\perp$ if this step fails.
2. Construct the *bipartite support graph of* R, $\Gamma(\boldsymbol{R})$, as follows. Its row vertices are $u_1, \ldots, u_k$, its column vertices are $v_1, \ldots, v_n$, and (u_i, v_j) is an edge exactly when $R_{i,j} \neq 0$.
3. Determine row-scaling factors $\alpha = (\alpha_1, \ldots, \alpha_k)$ and column-scaling factors $\beta = (\beta_1, \ldots, \beta_n)$. All factors are initially set to 1. In each connected component containing a nonzero row, the smallest-index row vertex u_{i_0} is selected as the root and α_{i_0} is fixed to 1. A deterministic traversal of the component then assigns a factor whenever a new vertex is discovered along an edge (u_i, v_j), choosing the scalars α_i and β_j so that $\alpha_i R_{i,j} \beta_j = 1$.

Finally, the canonical value is

$$\mathsf{Can}(\boldsymbol{G}) = \mathrm{diag}(\alpha)\boldsymbol{R}\,\mathrm{diag}(\beta).$$

The choice of the root fixes the otherwise free common scaling within each connected component, while the traversal propagates this choice to all its row and column vertices. This removes the residual ambiguity arising from left and right diagonal scaling. In particular, we have that $\mathsf{Can}(\boldsymbol{GD}) = \mathsf{Can}(\boldsymbol{G})$ for all $\boldsymbol{D} \in \mathsf{Diag}_n(\mathbb{F}_q)$. We refer to [30, Appendix A] for the full algorithmic description.

Lift function. The map Lift applies the representative permutation before canonisation. For $\boldsymbol{P} \in \mathsf{Perm}_n(\mathbb{F}_q)$ and $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$, it is given by

$$\mathsf{Lift}(\boldsymbol{P}, \boldsymbol{G}) = \mathsf{Can}(\boldsymbol{GP}).$$

Consequently, if $\boldsymbol{Q} = \boldsymbol{PD}$ is monomial, the diagonal factor $\boldsymbol{D}$ is removed by the canonicalisation and $\mathsf{Lift}(\mathsf{PermRep}(\boldsymbol{Q}), \boldsymbol{G}) = \mathsf{Can}(\boldsymbol{GQ})$.

Proposition 6. *The above canonical lifting is effective.*

Proof. For $\boldsymbol{P} \in R$, the group element $(\boldsymbol{I}_k, \boldsymbol{P}) \in G$ satisfies $\mathsf{PermRep}(\boldsymbol{I}_k, \boldsymbol{P}) = \boldsymbol{P}$. Moreover, if $\mathsf{Can}(\boldsymbol{G}) = \mathsf{Can}(\boldsymbol{G}')$, the row-reduction and diagonal-scaling canonicalisation algorithms provide matrices $\boldsymbol{U}, \boldsymbol{U}' \in \mathsf{GL}_k(\mathbb{F}_q)$ and $\boldsymbol{D}, \boldsymbol{D}' \in \mathsf{Diag}_n(\mathbb{F}_q)$ such that $\boldsymbol{UGD} = \mathsf{Can}(\boldsymbol{G}) = \mathsf{Can}(\boldsymbol{G}') = \boldsymbol{U}'\boldsymbol{G}'\boldsymbol{D}'$. Consequently, $\boldsymbol{G}' = (\boldsymbol{U}')^{-1}\boldsymbol{UGD}(\boldsymbol{D}')^{-1}$, and all these matrices are obtained in polynomial time. This establishes both requirements of effectiveness. □

Proposition 7. *The above canonical lifting is left-compatible.*

Proof. Let $(\boldsymbol{U}, \boldsymbol{Q}), (\boldsymbol{U}', \boldsymbol{Q}')$ share representative, i.e., $\boldsymbol{P} = \mathsf{PermRep}\left((\boldsymbol{U}, \boldsymbol{Q})\right) = \mathsf{PermRep}\left((\boldsymbol{U}', \boldsymbol{Q}')\right)$. Then, for every $(\boldsymbol{S}, \boldsymbol{RD}) \in G$, where $\boldsymbol{D}$ is a diagonal matrix and $\boldsymbol{R}$ a permutation matrix, we have

$$\mathsf{PermRep}\left((\boldsymbol{SU}, \boldsymbol{QRD})\right) = \boldsymbol{PR} = \mathsf{PermRep}\left((\boldsymbol{SU}', \boldsymbol{Q}'\boldsymbol{RD})\right).$$

Moreover, given $g = (\boldsymbol{U}, \boldsymbol{Q}) \in G$, one can sample uniformly at random from $[g]_{\mathsf{PermRep}}$ sampling a uniformly random invertible matrix $\boldsymbol{S}$ and a uniformly random diagonal one $\boldsymbol{D}'$, and computing $(\boldsymbol{SU}, \boldsymbol{QD})$. □

5.2 Security Basis for CELeReS

We explicitly state the three computational problems associated with the properties in Definitions 8 to 10 for this canonical lifting. Let X be the set of full-rank matrices in $\mathbb{F}_q^{k\times n}$. Throughout, $\boldsymbol{G}, \boldsymbol{T} \in X$ are the fixed public base matrices, $\boldsymbol{Q} \leftarrow\!\$\ \mathsf{Mono}_n(\mathbb{F}_q)$, and $\widetilde{\boldsymbol{Q}} \leftarrow\!\$\ [\boldsymbol{Q}]_{\mathsf{PermRep}}$. Thus, writing $\boldsymbol{Q} = \boldsymbol{PD}$, the monomial $\widetilde{\boldsymbol{Q}}$ has the same permutation part $\boldsymbol{P}$ and an independently sampled diagonal part. Following the key-generation algorithm of CELeReS, set

$$\boldsymbol{G}' = \mathsf{RREF}(\boldsymbol{GQ}) \qquad \text{and} \qquad \boldsymbol{T}' = \mathsf{RREF}(\boldsymbol{T}\widetilde{\boldsymbol{Q}}).$$

Frame-One-Way. The canonical lifting of Section 5.1 is *frame one-way* if every PPT algorithm solving Problem 3 has success probability negligible in λ.

Problem 3. *Given* $(\boldsymbol{G}, \boldsymbol{T}, \boldsymbol{G}', \boldsymbol{T}')$ *sampled as above, find matrices* $\boldsymbol{Y}, \boldsymbol{Z} \in X$ *and permutation matrices* $\boldsymbol{P}_0, \boldsymbol{P}_1 \in \mathsf{Perm}_n(\mathbb{F}_q)$ *such that*

$$\mathsf{Lift}(\boldsymbol{P}_0, \boldsymbol{G}) = \mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{Y}), \quad \mathsf{Lift}(\boldsymbol{P}_0, \boldsymbol{T}) = \mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{Z}), \quad \mathsf{Can}(\boldsymbol{Z}) = \mathsf{Can}(\boldsymbol{T}').$$

Problem 3 is closely related to the 2-*samples symmetric action underlying monomial code equivalence problem* [31, Problem 3]. In particular, we prove in Appendix D that these problems are polynomial-time equivalent whenever the monomial automorphism group of the code generated by $\boldsymbol{T}$ is trivial which is typically the case for random codes (see Proposition 1 and [41]). Related formulations appeared and were cryptanalysed in [30,2]. We recall below the cryptanalytic efforts in the literature.

First, notice that the formulation of Problem 3 prevents the direct application of the linearization attacks of [17,21] to t-sample code equivalence. There, a linear system is associated to each pair of equivalent codes, and, if the resulting systems share a solution, they can be stacked to reduce the dimension of the solution space to enable polynomial-time recovery. In the case of Problem 3 and [31, Problem 3], however, the distinct diagonals lead to different solution vectors, and the two systems cannot be stacked as equations in a common variable. This obstruction is discussed in [30]. To overcome this fact, an algebraic approach can be used. In [31], the algebraic model uses a set of unknowns for the common permutation matrix, and another set for the two distinct diagonal matrices. This leads to a non-linear system having degree 2. As the experiments done in [31] show, this approach does not seem to lead to an efficient secret recovery.

A second approach is the analysis given in [2], where tools from algebraic geometry are used to construct an equivalent polynomial system to solve the problem, with the advantage that the unknowns are shared by the two samples and there is no need to encode the (distinct) diagonal maskings. Nevertheless, this approach leads to polynomials with an exponential number of monomials, being too dense for an application on cryptographic parameters sets.

Finally, we explored whether the standard approach to solving LEP can be optimized in the setting of Problem 3. Usually, we build a list of fixed-weight codewords for each code, of sizes chosen so that we expect one pair to be equivalent [7,19]. One might hope that, given two pairs of equivalent codes $(\boldsymbol{G}, \boldsymbol{G}')$ and $(\boldsymbol{T}, \boldsymbol{T}')$, the search cost can be distributed between the two instances by balancing the success probability with the cost of building the lists. However, although this allows the individual lists to be shortened, this requires four lists. Indeed, let N_w denote the number of relevant weight-w codewords and let ℓ_i be the size of each list constructed for instance i. The four lists contain $2(\ell_1 + \ell_2)$ codewords in total, while the expected number of matches is $(\ell_1^2 + \ell_2^2)/N_w$. For a fixed list-generation cost, this quantity is maximized by setting either $\ell_1 = 0$ or $\ell_2 = 0$. Thus, under a standard birthday-paradox analysis, the optimal strategy is to concentrate the entire search on one of the two LEP instances.

Non-Splitting. The canonical lifting of Section 5.1 is *non-splitting* if every PPT algorithm solving Problem 4 has success probability negligible in λ.

Problem 4. *Given fixed $\boldsymbol{G}, \boldsymbol{T} \in X$, find permutation matrices $\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{R}_0, \boldsymbol{R}_1 \in \mathsf{Perm}_n(\mathbb{F}_q)$ and matrices $\boldsymbol{G}', \boldsymbol{T}_0, \boldsymbol{T}_1 \in X$ such that $\mathsf{Can}(\boldsymbol{T}_0) \neq \mathsf{Can}(\boldsymbol{T}_1)$ and*

$$\mathsf{Lift}(\boldsymbol{P}_0, \boldsymbol{G}) = \mathsf{Lift}(\boldsymbol{R}_0, \boldsymbol{G}'), \quad \mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{G}) = \mathsf{Lift}(\boldsymbol{R}_1, \boldsymbol{G}'),$$
$$\mathsf{Lift}(\boldsymbol{P}_0, \boldsymbol{T}) = \mathsf{Lift}(\boldsymbol{R}_0, \boldsymbol{T}_0), \quad \mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{T}) = \mathsf{Lift}(\boldsymbol{R}_1, \boldsymbol{T}_1).$$

By Proposition 5, solving Problem 4 is equivalent to finding a non-trivial permutation automorphism of $\langle \boldsymbol{G} \rangle$ that does not preserve $\langle \boldsymbol{T} \rangle$.

If the base elements have trivial automorphisms, which holds with high probability (see Proposition 1 and [41]), non-splitting holds unconditionally. Otherwise, breaking it is as hard as finding a non-trivial one, which is believed to be as hard as solving LEP.

2-Pseudorandom. The canonical lifting of Section 5.1 is *2-pseudorandom* if the distributions of the two pairs in Problem 5 are computationally indistinguishable.

Problem 5. *For fixed $\boldsymbol{G}, \boldsymbol{T} \in X$, distinguish between the following two pairs:*

$$\big(\mathsf{RREF}(\boldsymbol{G}\boldsymbol{Q}), \mathsf{RREF}(\boldsymbol{T}\widetilde{\boldsymbol{Q}})\big) \qquad \text{and} \qquad \big(\mathsf{RREF}(\boldsymbol{G}\boldsymbol{Q}_0), \mathsf{RREF}(\boldsymbol{T}\boldsymbol{Q}_1)\big),$$

where $\boldsymbol{Q}, \boldsymbol{Q}_0, \boldsymbol{Q}_1 \leftarrow\!\!\$\ \mathsf{Mono}_n(\mathbb{F}_q)$, $\widetilde{\boldsymbol{Q}} \leftarrow\!\!\$\ [\boldsymbol{Q}]_{\mathsf{PermRep}}$, and $\boldsymbol{Q}_0, \boldsymbol{Q}_1$ are sampled independently.

This asks to distinguish whether the public key and the tag are generated from monomials $\boldsymbol{Q}, \widetilde{\boldsymbol{Q}}$ sharing permutation or with independent permutations. This can be viewed as a 2-sample version of the weak pseudorandomness [1] of the quotient action induced by $\mathsf{Perm}_n(\mathbb{F}_q)$. While the group action of the monomial matrices on linear codes is not weak pseudorandom (see for instance [17,21]), it is still not known how to attack the same assumption on this quotient action. One approach is to use an algebraic model with the permutation as unknown. Then, one can distinguish between the correlated and uncorrelated pairs by checking if a solution $\boldsymbol{P}$ exists (remark that explicitly recovering $\boldsymbol{P}$ is unnecessary). However, the known algebraic models are not practical [2]. Another approach is attempting to recover, if any, the common permutations, reducing this to 2-one-wayness. Known attempts [31,2] are far from outperforming direct attacks to LEP.

5.3 Keys and Signature Size

For a ring of N members, each public generator matrix $\boldsymbol{G}_i$ is stored in systematic form, requiring $k(n-k)\lceil \log_2 q \rceil$ bits. The public material also contains two seeds of bit-length ℓ_{seed} for the base generator matrix $\boldsymbol{G}$ and the base tag matrix $\boldsymbol{T}$. The secret key is a monomial matrix $\boldsymbol{Q}_i = \boldsymbol{P}_i \boldsymbol{D}_i$. It can be represented by

its underlying permutation, ideally using $\lceil \log_2(n!) \rceil$ bits, and by $n-1$ nonzero coefficients, after normalising one coefficient to 1. Hence, the key sizes in bits are

$$\ell_{\mathsf{pk}} = 2\ell_{\mathsf{seed}} + N\,k\,(n-k)\lceil \log_2 q \rceil, \qquad \ell_{\mathsf{sk}} = \lceil \log_2(n!) \rceil + (n-1)\lceil \log_2(q-1) \rceil.$$

In practice, a slightly longer encoding of the secret permutation may be used for efficiency [10]. The signature contains the tag matrix $\boldsymbol{T}_i$, a digest for H_{FS} of bit-length ℓ_{digest}, a salt of the same length, and n_{nodes} published seed-tree nodes, each of length ℓ_{seed}. It also contains the W responses corresponding to the challenged positions, where W is the Hamming weight of the challenge vector. The challenge vector itself is not serialised, but deterministically recomputed from the digest during verification. Thus,

$$\ell_\sigma = k(n-k)\lceil \log_2(q) \rceil + 2\ell_{\mathsf{digest}} + n_{\mathsf{nodes}}\ell_{\mathsf{seed}} + W\ell_{\mathsf{rsp}},$$

where

$$\ell_{\mathsf{rsp}} = \underbrace{\lceil \log_2(n!) \rceil}_{\text{permutation representative}} + \underbrace{\ell_{\mathsf{seed}}}_{\mathsf{bits}_j} + \underbrace{\lceil \log_2 N \rceil\, \ell_{\mathsf{digest}}}_{\mathsf{path}_j}.$$

Each response consists of a compressed permutation representative, one opening string, and one authentication path in the index-hiding Merkle tree. Since n_{nodes} only depends on the seed tree parameters M and K [32, Prop. 2], the only dependence on N is $\lceil \log_n N \rceil$, therefore the signature size grows logarithmically with the ring size. We discuss in Appendix E an optimised variant of CELeReS that attempts to reduce the size of the tag at the cost of heavier key generation.

6 Signature Size Comparisons and Implementation

In this section, we compare the signature sizes of CELeReS against other relevant post-quantum linkable ring signatures and present our implementations.

Parameter selection. To the best of our knowledge, and given the considerations in Section 5.2, the best known approach to solving any of Problems 3 to 5 is to directly solve one of the underlying LEP instances. We therefore instantiate CELeReS using parameters that account for the most recent cryptanalysis of LEP [20,7,19]. Specifically, we select $n \in \{272, 416, 564\}$, $k = n/2$, and $q = 127$ to achieve target security levels of 128, 192, and 256 bits, respectively.

Comparison vs. other linkable ring signatures. In Table 1, we compare the average signature sizes of CELeReS with other linkable ring signatures proposed in literature that guarantee the same security notions. For a systematic study of the diverse security notions for linkable ring signatures, see [22]. We consider Calamari and Falafl [11] and exclude subsequent proposals for which correctness or security issues have been highlighted [36,17,18,39,42]. As additional baselines, we consider the linkable variant of Raptor [43], which provides only the weaker notion of one-time anonymity, and the recent DualRing-PRF [53], whose anonymity is not established under the same security game as in [11]. Overall, CELeReS compares favourably with the considered schemes. Remark that Calamari, whose claimed security is questionable [16,48,24], also raises efficiency concerns.

Implementation. We provide a proof-of-concept implementation[7] of CELeReS, both in reference C and AVX2 optimised code, and present our benchmarks in Table 2. The optimised code was run on an Intel® Core™ i9-13900K @ 3.00 GHz machine with 64 GB of memory, running Ubuntu 22.04.2 LTS with Hyper-Threading and Turbo Boost disabled. Overall, our benchmarks support the practical applicability of the proposed scheme.

Scheme	Ring size	KeyGen	Sign	Verify
CELeReS	2^1	$1.32\,M$ / $4.5 \times 10^{-4}\,s$	$421\,M$ / $0.141\,s$	$397\,M$ / $0.137\,s$
	2^3		$2419\,M$ / $0.806\,s$	$2035\,M$ / $0.678\,s$
	2^6		$9300\,M$ / $3.090\,s$	$7865\,M$ / $2.631\,s$

Table 2: Performance in average millions of CPU cycles (M) and seconds (s), computed over 100 runs, for different ring sizes and 128 bits of security.

Acknowledgments. AI agents were used to refine the overall quality of the manuscript and the implementation. All calculations and conclusions were independently reviewed and verified by the authors, who accept full responsibility for the integrity of the work. The second and third authors are members of the INdAM Research group GNSAGA and CrypTO, the Cryptography and Number Theory group of Politecnico di Torino. The second author was partially supported by the Italian Ministry of University and Research in the framework of the Call for Proposals for scrolling of final rankings of the PRIN 2022 call - Protocol no. 2022RFAZCJ. The authors thank Jesús-Javier Chi-Domínguez for insightful discussions and the anonymous reviewers of a previous version of this manuscript who provided us with helpful comments and recommendations.

References

1. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14
2. Alecci, G., D'Alconzo, G.: Linear Code Equivalence via Plücker Coordinates. Cryptology ePrint Archive, Paper 2026/495 (2026), https://eprint.iacr.org/2026/495
3. Backes, M., Döttling, N., Hanzlik, L., Kluczniak, K., Schneider, J.: Ring signatures: Logarithmic-size, no setup - from standard assumptions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 281–311. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17659-4_10
4. Baldi, M., Barenghi, A., Beckwith, L., Biasse, J., Chou, T., Esser, A., Gaj, K., Karl, P., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M.O., Santini, P., Wallace, R., Zweydinger, F.: LESS — Linear Equivalence Signature Scheme. Tech.

[7] Available at https://anonymous.4open.science/r/celeres-DBB9.

rep., National Institute of Standards and Technology (2024), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures
5. Baldi, M., Battagliola, M., Borin, G., Di Crescenzo, G., El Mechri, R., Meneghetti, A., Persichetti, E., Santini, P., Zweydinger, F.: LEAST: Linear Equivalence Action Threshold Signature (Preview Writeup). https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/LEAST-PW01.pdf (Jan 20 2026), preliminary version submitted to the NIST First Call for Multi-Party Threshold Schemes
6. Baldi, M., Mechri, R.E., Santini, P., Schiavoni, R.: On the representation of self-orthogonal codes and applications to cryptography. Cryptology ePrint Archive, Paper 2025/2279 (2025), https://eprint.iacr.org/2025/2279
7. Bardet, M., Brion, C., Otmani, A., Saeed, M., Sendrier, N.: Solving the linear equivalence problem from single codeword matching. In: Heninger, N., Rosulek, M. (eds.) Advances in Cryptology – CRYPTO 2026. pp. 256–288. Springer Nature Switzerland, Cham (2026)
8. 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)
9. Benčina, B., Budroni, A., Chi-Domínguez, J.J., Kulkarni, M.: Properties of lattice isomorphism as a cryptographic group action. In: Saarinen, M.J., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I. pp. 170–201. Springer, Cham (Jun 2024). https://doi.org/10.1007/978-3-031-62743-9_6
10. Bettaieb, S., Budroni, A., Palumbi, M., Filho, D.L.G.: Quasi-optimal permutation ranking and applications to PERK. In: Vaudenay, S., Petit, C. (eds.) AFRICACRYPT 24. LNCS, vol. 14861, pp. 49–65. Springer, Cham (Jul 2024). https://doi.org/10.1007/978-3-031-64381-1_3
11. 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, Part II. LNCS, vol. 12492, pp. 464–492. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_16
12. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_9
13. Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: LESS is more: Code-based signatures without syndromes. In: Nitaj, A., Youssef, A.M. (eds.) AFRICACRYPT 20. LNCS, vol. 12174, pp. 45–65. Springer, Cham (Jul 2020). https://doi.org/10.1007/978-3-030-51938-4_3
14. Bläser, M., Chen, Z., Duong, D.H., Joux, A., Nguyen, T.N., Plantard, T., Qiao, Y., Susilo, W., Tang, G.: On digital signatures based on group actions: QROM security and ring signatures. In: Saarinen, M.J., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I. pp. 227–261. Springer, Cham (Jun 2024). https://doi.org/10.1007/978-3-031-62743-9_8
15. Blom, D., Lamorgese, G., Niederhagen, R., Ran, L., Samardjiska, S.: miniMEDS - short GMW signatures from isometry hints. Cryptology ePrint Archive, Paper 2026/1323 (2026), https://eprint.iacr.org/2026/1323
16. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 493–522. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45724-2_17

17. Budroni, A., Chi-Domínguez, J.J., D'Alconzo, G., Scala, A.J.D., Kulkarni, M.: Don't use it twice! Solving relaxed linear equivalence problems. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part VIII. LNCS, vol. 15491, pp. 35–65. Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0944-4_2
18. Budroni, A., Chi-Domínguez, J.J., Franch, E.: Don't use it twice: Reloaded! On the lattice isomorphism group action. CiC **2**(2), 9 (2025). https://doi.org/10.62056/ay76chdj
19. Budroni, A., Esser, A.: One pair to rule them all: Towards an optimal algorithm for solving code equivalence via codeword search. In: Heninger, N., Rosulek, M. (eds.) Advances in Cryptology – CRYPTO 2026. pp. 319–346. Springer Nature Switzerland, Cham (2026)
20. Budroni, A., Esser, A., Franch, E., Natale, A.: Two is all it takes: Asymptotic and concrete improvements for solving code equivalence. In: Bai, S., Persichetti, E. (eds.) Public-Key Cryptography – PKC 2026. pp. 395–427. Springer Nature Switzerland, Cham (2026)
21. Budroni, A., Natale, A.: On the sample complexity of linear code equivalence for all code rates. Cryptography and Communications pp. 1–23 (2025). https://doi.org/10.1007/s12095-025-00790-x
22. Bultel, X., Olivier-Anclin, C.: On the anonymity of linkable ring signatures. In: Kohlweiss, M., Di Pietro, R., Beresford, A. (eds.) Cryptology and Network Security. pp. 212–235. Springer Nature Singapore, Singapore (2025)
23. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
24. Chávez-Saab, J., Chi-Domínguez, J.J., Jaques, S., Rodríguez-Henríquez, F.: The SQALE of CSIDH: Sublinear Vélu Quantum-Resistant Isogeny Action with Low Exponents. Journal of Cryptographic Engineering **12**, 349–368 (2022). https://doi.org/10.1007/s13389-021-00271-w
25. 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: El Mrabet, N., De Feo, L., Duquesne, S. (eds.) AFRICACRYPT 23. LNCS, vol. 14064, pp. 28–52. Springer, Cham (Jul 2023). https://doi.org/10.1007/978-3-031-37679-5_2
26. Chou, T., Niederhagen, R., Ran, L., Samardjiska, S.: Reducing signature size of matrix-code-based signature schemes. In: Saarinen, M.J., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I. pp. 107–134. Springer, Cham (Jun 2024). https://doi.org/10.1007/978-3-031-62743-9_4
27. Chou, T., Persichetti, E., Santini, P.: On linear equivalence, canonical forms, and digital signatures. DCC **93**(7), 2415–2457 (2025). https://doi.org/10.1007/s10623-025-01576-1
28. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
29. Ducas, L., van Woerden, W.P.J.: On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 643–673. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_23
30. Duong, D.H., Khuc, X.T., Qiao, Y., Susilo, W., Zhang, C.: (Partially) Blind Signatures from Cryptographic Group Actions. In: Applied Cryptography and

Network Security – 24th International Conference, ACNS 2026. Lecture Notes in Computer Science, Springer, Stony Brook, USA (2026). https://doi.org/10.1007/978-3-032-32560-0_6
31. Duong, D.H., Qiao, Y., Zhang, C.: Diffie-Hellman key exchange from commutativity to group laws. In: ITCS. LIPIcs, vol. 362, pp. 52:1–52:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Saarland, Germany (2026). https://doi.org/10.4230/LIPIcs.ITCS.2026.52
32. D'Alconzo, G., Gangemi, A.: TRIFORS: LINKable trilinear forms ring signature. Journal of Algebra and Its Applications **0**(0), 2650284 (2025). https://doi.org/10.1142/S0219498826502841, https://doi.org/10.1142/S0219498826502841
33. D'Alconzo, G., Meneghetti, A., Signorini, E.: Group factorisation for smaller signatures from cryptographic group actions: D'alconzo et al. Designs, Codes and Cryptography **94**(2), 42 (2026)
34. 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, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
35. Hanzlik, L., Lai, Y.F., Paracucchi, E., Persichetti, E.: Post-quantum blind signature from standard group action assumptions and more. In: Daemen, J., Thomé, E. (eds.) Advances in Cryptology – EUROCRYPT 2026. pp. 277–307. Springer Nature Switzerland, Cham (2026)
36. Iribarnegaray, I.D., Gregor, V., Leal, F.L.: A note on the unsuitability of LIGA for linkable ring signatures: The perils of non-commutativity. Cryptology ePrint Archive, Paper 2026/671 (2026), https://eprint.iacr.org/2026/671
37. Joint Technical Committee ISO/IEC 1: Information technology – Security techniques – Anonymous digital signatures – Part 1: General. International Standard ISO/IEC 20008-1:2013, International Organization for Standardization (Dec 2013)
38. Khuc, X.T., Ta, A.T., Susilo, W., Duong, D.H., Guo, F., Fukushima, K., Kiyomoto, S.: Logarithmic-size (linkable) ring signatures from lattice isomorphism problems. In: Regazzoni, F., Mazumdar, B., Parameswaran, S. (eds.) Security, Privacy, and Applied Cryptography Engineering. pp. 214–241. Springer Nature Switzerland, Cham (2024)
39. Kuchta, V., Sica, F.: (Mis)using the Lattice Isomorphism Problem. Cryptanalysis of the double-LIP and Construction of LIP-Based Blind Signatures. Cryptology ePrint Archive, Paper 2026/722 (2026), https://eprint.iacr.org/2026/722
40. Lai, Y.F., Persichetti, E.: LEAF: Compact and efficient blind signature from code-based assumptions. Cryptology ePrint Archive, Report 2025/1585 (2025), https://eprint.iacr.org/2025/1585
41. Li, X., Wang, Q., Zhou, Y.: The automorphism groups of random linear codes (2026), https://arxiv.org/abs/2607.23936
42. London, C., Gardham, D., Dragan, C.C.: RingSLIP: Ring signatures from the lattice isomorphism problem. Cryptology ePrint Archive, Paper 2026/889 (2026), https://eprint.iacr.org/2026/889
43. Lu, X., Au, M.H., Zhang, Z.: Raptor: A practical lattice-based (linkable) ring signature. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 19International Conference on Applied Cryptography and Network Security. LNCS, vol. 11464, pp. 110–130. Springer, Cham (Jun 2019). https://doi.org/10.1007/978-3-030-21568-2_6
44. Micali, S., Reyzin, L.: Improving the exact security of digital signature schemes. Journal of Cryptology **15**(1), 1–18 (Jan 2002). https://doi.org/10.1007/s00145-001-0005-8

45. National Institute of Standards and Technology: Post-quantum cryptography: Digital signature schemes. Round 2 Additional Signatures (2023), https://csrc.nist.gov/projects/pqc-dig-sig/round-2-additional-signatures
46. National Institute of Standards and Technology: Multi-Party Threshold Cryptography Project and NIST First Call for Multi-Party Threshold Schemes. https://csrc.nist.gov/projects/threshold-cryptography (2026), call for Multi-Party Threshold Schemes (NIST IR 8214C)
47. Oral, H., Phelps, K.T.: Almost all self-dual codes are rigid. Journal of Combinatorial Theory, Series A **60**(2), 264–276 (1992). https://doi.org/https://doi.org/10.1016/0097-3165(92)90007-H, https://www.sciencedirect.com/science/article/pii/009731659290007H
48. Peikert, C.: He gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 463–492. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45724-2_16
49. Perera, M.N.S., Nakamura, T., Hashimoto, M., Yokoyama, H., Cheng, C.M., Sakurai, K.: A Survey on Group Signatures and Ring Signatures: Traceability vs. Anonymity. Cryptography **6**(1), 3 (2022). https://doi.org/10.3390/cryptography6010003
50. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Berlin, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_32
51. Tang, G., Duong, D.H., Joux, A., Plantard, T., Qiao, Y., Susilo, W.: Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 582–612. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_21
52. Tsang, P.P., Wei, V.K.: Short linkable ring signatures for e-voting, e-cash and attestation. In: Deng, R.H., Bao, F., Pang, H., Zhou, J. (eds.) Information Security Practice and Experience. pp. 48–60. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
53. Zhang, X., Steinfeld, R., Liu, J.K., Esgin, M.F., Liu, D., Ruj, S.: DualRing-PRF: Post-quantum (linkable) ring signatures from Legendre and power residue PRFs. In: Tianqing Zhu, Y.L. (ed.) ACISP 24, Part II. LNCS, vol. 14896, pp. 124–143. Springer, Singapore (Jul 2024). https://doi.org/10.1007/978-981-97-5028-3_7

A Optimisation on Ephemeral Secrets

In this section, we present an optimisation that applies to the signature scheme LESS. To do so, we present a variant of the Sigma protocol Σ_{CL} from Figure 3.

Let $(R, C, \mathsf{Rep}, \mathsf{Can}, \mathsf{Lift})$ be an effective canonical lifting for a group action $((G, \circ), X, \star)$. Assume that R is efficiently samplable and that there exists an efficiently computable operation

$$\odot : R \times G \to R$$

such that, for every $r \in R$ and $g \in G$, it holds that

$$r \odot g = \mathsf{Rep}(g_r \circ g),$$

for any $g_r \in G$ satisfying $\mathsf{Rep}(g_r) = r$. Equivalently, $\odot$ is the operation induced on representatives by right composition in G.

With this operation, the prover can sample a representative directly during the commitment phase, instead of sampling a full group element as in Figure 3. The commitment is computed as $\mathsf{Lift}(r, x')$. For challenge $\mathsf{ch} = 0$, the prover reveals r and the verifier checks the commitment using $\mathsf{Lift}(r, x')$. For challenge $\mathsf{ch} = 1$, the prover returns $r \odot g$, which allows the verifier to compute the same canonical value from x. The resulting Sigma protocol is presented in Figure 7. Notice that the properties of Σ_{CL} are preserved. Correctness follows directly from the definition of $\odot$. Special soundness is unchanged, since two accepting transcripts yield representatives $r, h \in R$ satisfying $\mathsf{Lift}(r, x') = \mathsf{Lift}(h, x)$. Honest-verifier zero-knowledge is also preserved, provided that sampling $r \leftarrow\!\!\$\, R$ gives the same distribution as $\mathsf{Rep}(\rho)$ for $\rho \leftarrow\!\!\$\, G$; then $r \odot g$ is distributed as $\mathsf{Rep}(\rho \circ g)$, and therefore as a freshly sampled representative.

Setup
1. Let $x \in X$ be public
2. Sample $g \leftarrow\!\!\$\, G$
3. Set $x' = g \star x$

PROVER	**VERIFIER**
$\mathsf{Commit}(x')$ 1. Sample $r \leftarrow\!\!\$\, R$ 2. Set $\mathsf{com} = \mathsf{Lift}(r, x')$ 3. Send com to the verifier	$\mathsf{Challenge}$ 1. Sample $\mathsf{ch} \leftarrow\!\!\$\, \{0, 1\}$ 2. Send ch to the prover
$\mathsf{Response}(\mathsf{ch}, g)$ 1. If $\mathsf{ch} = 0$: i. Set $\mathsf{rsp} = r$ 2. Else: i. Set $\mathsf{rsp} = r \odot g$ 3. Send rsp to the verifier	$\mathsf{Verify}(x, x', \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$ 1. If $\mathsf{ch} = 0$: i. parse rsp as r ii. $\mathsf{com}' = \mathsf{Lift}(r, x')$ 2. Else: i. parse rsp as h ii. $\mathsf{com}' = \mathsf{Lift}(h, x)$ 3. Return accept iff $\mathsf{com}' = \mathsf{com}$

Fig. 7: Variant of Σ_{CL} (Figure 3) with compressed ephemeral secrets.

In the canonical lifting used in LESS, the representative r encodes an information set, while the secret key is a monomial matrix. The operation $\odot$ can therefore be defined by applying the permutation underlying the secret key to the information set. Although this saves some operations, the impact on the overall signing time complexity is modest. Experimentally, we observed a 5–8% improvement across all security levels of LESS.

B Trees and Optimizations

B.1 Soundness Amplification, Seed Trees, and Domain Separators

We report here the optimisations and techniques that we consider in our ring and linkable ring signatures presented in Section 4.

Soundness amplification. To amplify the soundness of $\Sigma_{\mathsf{CL}}^{\mathsf{OR}}$, we apply a standard optimisation, that is, instead of sampling λ independent challenge bits, one samples a challenge vector $c \in \mathcal{C}_{M,K}$, where $\mathcal{C}_{M,K} \subseteq \{0,1\}^M$ is the set of length-M vectors of fixed Hamming weight K; the K positions correspond to the more expensive branch. Since $\mathsf{ch} = 0$ which reveals the full randomizer ρ together with all commitment openings, this is substantially more costly than $\mathsf{ch} = 1$. Hence, one chooses parameters M and K so that $\binom{M}{K} \geq 2^\lambda$, and runs M parallel instances while forcing only K of them to use the expensive branch. This preserves the target soundness level while reducing the expected communication.

Seed tree. Another standard technique is to derive the randomness for all M parallel instances from a single master seed using a *seed tree*, a triple of algorithms $\mathsf{ST} = (\mathsf{ST.GetSeeds}, \mathsf{ST.ReleaseSeeds}, \mathsf{ST.RecoverLeaves})$ that behave as follows. Instead of sampling independent random coins for each instance, the prover expands one root seed into M leaves via $\mathsf{ST.GetSeeds}(\mathsf{seed}) \to (\mathsf{leaves}, \mathsf{tree})$, and uses the i-th leaf to generate the randomness of the i-th execution. In the response, rather than transmitting all seeds for the opened instances, the prover sends only the internal nodes selected by $\mathsf{ST.ReleaseSeeds}(\mathsf{tree}, S) \to \mathsf{nodes}$, where S is a list of K selected nodes, from which the verifier recovers the required leaves via $\mathsf{ST.RecoverLeaves}(\mathsf{nodes}, S) \to \{\mathsf{seed}_i\}_{i \in S}$. In our Sigma protocols, seed trees are used for the randomness generating commitments and opening strings in repeated executions, reducing the communication overhead of seed-based openings.

Salt and domain separators. We make use of a fresh salt in each transcript and domain-separate all random-oracle calls by a prefix. We stress that these ingredients are not specific to our canonical-lifting setting and are indeed standard folklore techniques for turning repeated Sigma protocols into practical signature schemes.

B.2 Index-Hiding Merkle Trees

Definition 11. *A* Merkle tree *is a triple of algorithms* $\mathsf{MT} = (\mathsf{MT.GetTree}, \mathsf{MT.GetPath}, \mathsf{MT.ReconstructRoot})$ *with access to some ordering* ord *and hash function*

$$\mathsf{H}_{\mathsf{coll}} : \{0,1\}^* \to \{0,1\}^{2\lambda}$$

such that

- $\mathsf{MT.GetTree}(A) \to (\mathsf{root}, \mathsf{tree})$. *On input a list* $A = (a_1, \ldots, a_{2^k})$ *of* 2^k *elements, where* $k \in \mathbb{N}$*, it constructs a binary tree of height* k *and outputs: its root* root*, a description* tree *of it.*

– $\mathsf{MT.GetPath}(\mathsf{tree}, i) \to \mathsf{path}$. *On input a tree description* tree *and an index* $i \in [2^k]$*, it outputs the authentication path* $\mathsf{path} = (n_1, \ldots, n_k)$*, consisting of the siblings of* l_i *and of each ancestor of* l_i*, ordered by decreasing height.*
– $\mathsf{MT.ReconstructRoot}(a, \mathsf{path}) \to \mathsf{root}'$. *On input an element* a *and a path* $\mathsf{path} = (n_1, \ldots, n_k)$*, it outputs a reconstructed root* $\mathsf{root}' = h_k$*, computed by setting* $h_0 = \mathsf{H}_{\mathsf{coll}}(a)$ *and* $h_i = \mathsf{H}_{\mathsf{coll}}((h_{i-1}, n_i)_{\mathsf{ord}})$ *for* $i \in [k]$*, recursively.*

We require the following properties.

– Correctness. *For every list* A *and index* $i \in [2^k]$*, an honest execution reconstructs the correct root with probability* 1*, i.e.,*

$$\Pr\left[\mathsf{MT.ReconstructRoot}(a_i, \mathsf{path}) = \mathsf{root} \;\middle|\; \begin{array}{c}(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MT.GetTree}(A) \\ \mathsf{path} \leftarrow \mathsf{MT.GetPath}(\mathsf{tree}, i)\end{array}\right] = 1.$$

– Binding. *There is a PPT extractor that, given* tree *and* (b, path) *such that* $b \notin A$ *and* $\mathsf{MT.ReconstructRoot}(b, \mathsf{path}) = \mathsf{root}$*, outputs a collision for* $\mathsf{H}_{\mathsf{coll}}$.

Further, if it also satisfies the following, we call it index-hiding Merkle tree.

– Index Hiding. *Given any two indices* $I_0, I_1 \in [2^k]$ *and two distributions* D, D' *over* $\{0,1\}^*$*, the distributions* $\mathcal{D}_{I_0}$ *and* $\mathcal{D}_{I_1}$ *are identical, where*

$$\mathcal{D}_{I_b} = \left\{ (a_{I_b}, \mathsf{path}, \mathsf{root}) \;\middle|\; \begin{array}{c} a_{I_b} \leftarrow D,\ a_i \leftarrow D',\ \forall\, i \neq I_b, \\ (\mathsf{tree}, \mathsf{root}) \leftarrow \mathsf{MT.GetTree}(A), \\ \mathsf{path} \leftarrow \mathsf{MT.GetPath}(\mathsf{tree}, I_b) \end{array} \right\} \text{ for } b = 0, 1.$$

Lemma 2 ([11], Section 2.6). *Assume* $\mathsf{H}_{\mathsf{coll}}$ *is collision-resistant and* ord *is the lexicographical order. Then the Merkle Tree is binding and index-hiding.*

C Instantiation of CELeReS

In this section, we explicit the construction of CELeReS in Figure 8. We recall that $\mathsf{RREF} : \mathbb{F}_q^{k \times n} \to \mathbb{F}_q^{k \times n}$ denotes the function that returns the row-reduced Echelon form of an input generator matrix. By $\mathsf{Com}(\cdot)$ we denote an instantiaiton of the random oracle $\mathcal{O}$ of Figure 6. For simplicity, we restrict the domain of $\mathsf{PermRep}$ only to $\mathsf{Mono}_n(\mathbb{F}_q)$ since the component $\mathsf{GL}_k(\mathbb{F}_q)$ of the group $G = \mathsf{Mono}_n(\mathbb{F}_q) \times \mathsf{Mono}_n(\mathbb{F}_q)$ is anyway discarded by the map.

D Discussion on Frame One-way for CELeReS

In this section we present a reduction of Problem 3 to the *t-samples symmetric action underlying monomial equivalence problem* [31] for $t = 2$. First, let us reformulate this problem with the notation of Section 5. While [31] states this problem (and its variants) in decision-and-search form, we consider its promise-search version, i.e., a search version whose instances are guaranteed to have a solution.

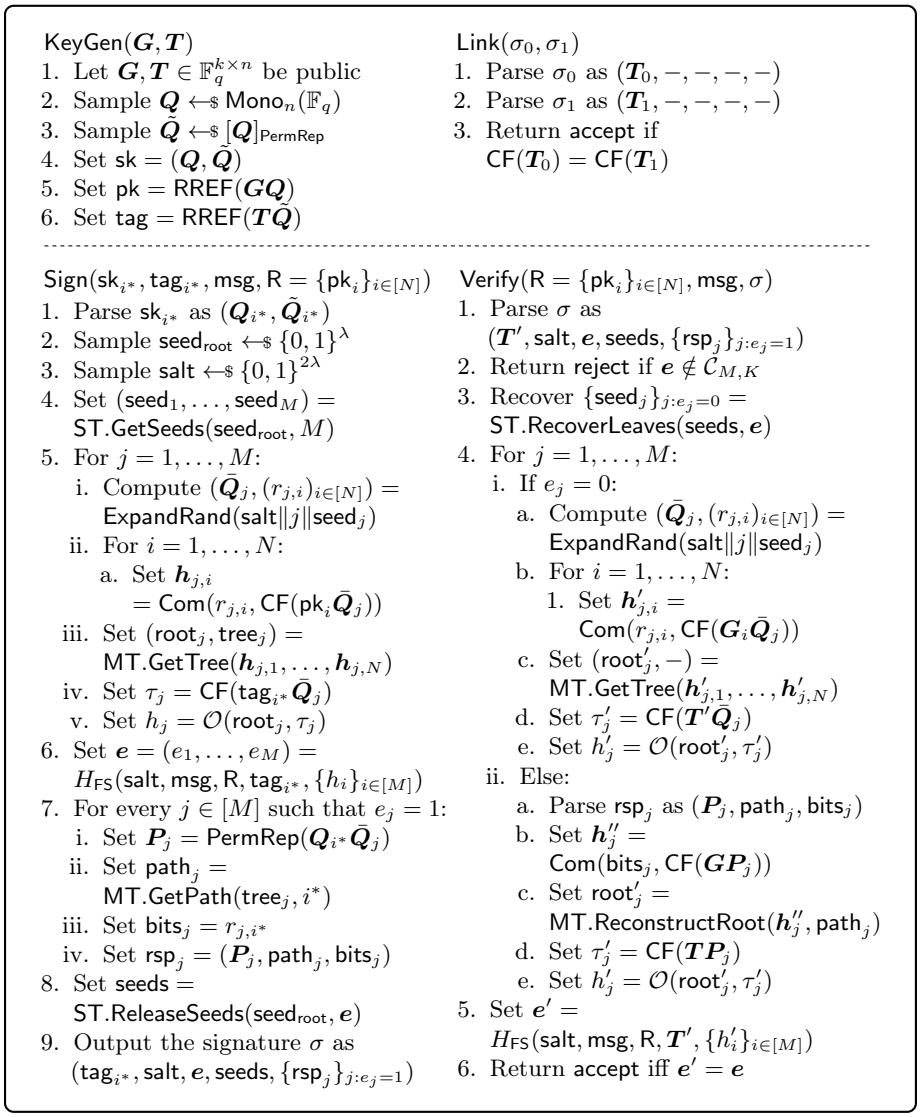

Fig. 8: CELeReS linkable ring signature.

Problem 6 ([31], Problem 3 for $t = 2$)**.** *Given four fixed full-rank matrices* $\boldsymbol{G}, \boldsymbol{T}, \boldsymbol{G}', \boldsymbol{T}' \in \mathbb{F}_q^{k \times n}$ *such that there exist* $\boldsymbol{A}_0, \boldsymbol{A}_1 \in \mathsf{GL}_k(\mathbb{F}_q)$*,* $\boldsymbol{D}_0, \boldsymbol{D}_1 \in \mathsf{Diag}_n(\mathbb{F}_q)$*, and a single permutation matrix* $\boldsymbol{P} \in \mathsf{Perm}_n(\mathbb{F}_q)$ *satisfying*

$$\boldsymbol{A}_0 \boldsymbol{G} \boldsymbol{P} \boldsymbol{D}_0 = \boldsymbol{G}', \qquad \boldsymbol{A}_1 \boldsymbol{T} \boldsymbol{P} \boldsymbol{D}_1 = \boldsymbol{T}',$$

then return $\boldsymbol{P}$*.*

Since $\mathsf{Diag}_n(\mathbb{F}_q)$ is normalised by $\mathsf{Perm}_n(\mathbb{F}_q)$, writing monomial matrices with a diagonal matrix on the right ($\boldsymbol{Q} = \boldsymbol{P}\boldsymbol{D}$) as above rather than on the left ($\boldsymbol{Q} = \boldsymbol{E}\boldsymbol{P}$ for $\boldsymbol{E} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}$) as in [31] leaves Problem 6 unchanged.

Proposition 8. *Suppose that the automorphism group of the code generated by* $\boldsymbol{T}$ *is trivial. Then the* frame one-way problem for CELeReS *(Problem 3) and the* two-sample problem *(Problem 6) are polynomial-time equivalent.*

Proof. Both problems are defined on the same promised tuples $(\boldsymbol{G}, \boldsymbol{T}, \boldsymbol{G}', \boldsymbol{T}')$. We give PPT reductions between their solutions on the same instance $(\boldsymbol{G}, \boldsymbol{G}', \boldsymbol{T}, \boldsymbol{T}')$.

Consider a solution $(\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{P}_0, \boldsymbol{P}_1)$ of Problem 3, which satisfies

$$\mathsf{Lift}(\boldsymbol{P}_0, \boldsymbol{G}) = \mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{Y}), \quad \mathsf{Lift}(\boldsymbol{P}_0, \boldsymbol{T}) = \mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{Z}), \quad \mathsf{Can}(\boldsymbol{Z}) = \mathsf{Can}(\boldsymbol{T}').$$

The last equation proves that there exist $\boldsymbol{A} \in \mathsf{GL}_k(\mathbb{F}_q)$ and $\boldsymbol{E} \in \mathsf{Diag}_n(\mathbb{F}_q)$ with $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{T}'\boldsymbol{E}$. In particular, $\boldsymbol{E}\boldsymbol{P}_1 = \boldsymbol{P}_1\boldsymbol{E}^*$ for $\boldsymbol{E}^* = \boldsymbol{P}_1^{-1}\boldsymbol{E}\boldsymbol{P}_1$, which itself lies in $\mathsf{Diag}_n(\mathbb{F}_q)$. Thus,

$$\begin{aligned}\mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{Z}) &= \mathsf{Can}(\boldsymbol{Z}\boldsymbol{P}_1) = \mathsf{Can}(\boldsymbol{A}\boldsymbol{T}'\boldsymbol{E}\boldsymbol{P}_1) = \mathsf{Can}(\boldsymbol{A}\boldsymbol{T}'\boldsymbol{P}_1\boldsymbol{E}^*) \\ &= \mathsf{Can}(\boldsymbol{T}'\boldsymbol{P}_1) = \mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{T}'),\end{aligned}$$

so the tag equation above implies

$$\mathsf{Lift}(\boldsymbol{P}_0, \boldsymbol{T}) = \mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{T}'). \tag{3}$$

Since $\boldsymbol{T}' = \boldsymbol{A}_1\boldsymbol{T}\boldsymbol{P}\boldsymbol{D}_1$ for some $\boldsymbol{A}_1, \boldsymbol{P}, \boldsymbol{D}_1$, using again the normality of $\mathsf{Diag}_n(\mathbb{F}_q)$ we have $\mathsf{Lift}(\boldsymbol{P}_1, \boldsymbol{T}') = \mathsf{Can}(\boldsymbol{T}\boldsymbol{P}\boldsymbol{P}_1)$, which with Equation (3) becomes

$$\mathsf{Can}(\boldsymbol{T}) = \mathsf{Can}(\boldsymbol{T}\boldsymbol{P}\boldsymbol{P}_1\boldsymbol{P}_0^{-1}).$$

therefore, $\boldsymbol{P}\boldsymbol{P}_1\boldsymbol{P}_0^{-1}$ is the permutation part of a monomial automorphism of the code generated by $\boldsymbol{T}$. By hypothesis this can only be the trivial automorphism, so $\boldsymbol{P}$ is given as $\boldsymbol{P} = \boldsymbol{P}_0\boldsymbol{P}_1^{-1}$.

Let now $\boldsymbol{P} \in \mathsf{Perm}_k(\mathbb{F}_q)$ be a solution of Problem 6. It satisfies

$$\boldsymbol{A}_0 \boldsymbol{G} \boldsymbol{P} \boldsymbol{D}_0 = \boldsymbol{G}', \qquad \boldsymbol{A}_1 \boldsymbol{T} \boldsymbol{P} \boldsymbol{D}_1 = \boldsymbol{T}',$$

for some $\boldsymbol{A}_0, \boldsymbol{A}_1 \in \mathsf{GL}_k(\mathbb{F}_q)$, $\boldsymbol{D}_0, \boldsymbol{D}_1 \in \mathsf{Diag}_n(\mathbb{F}_q)$. We show that $(\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{P}_0, \boldsymbol{P}_1) = (\boldsymbol{G}', \boldsymbol{T}', \boldsymbol{P}, \boldsymbol{I}_n)$ is a solution to Problem 3. Indeed,

$$\begin{aligned}\mathsf{Lift}(\boldsymbol{P}, \boldsymbol{G}) = \mathsf{Can}(\boldsymbol{G}\boldsymbol{P}) = \mathsf{Can}(\boldsymbol{G}') = \mathsf{Lift}(\boldsymbol{I}_n, \boldsymbol{G}'), \\ \mathsf{Lift}(\boldsymbol{P}, \boldsymbol{T}) = \mathsf{Can}(\boldsymbol{T}\boldsymbol{P}) = \mathsf{Can}(\boldsymbol{T}') = \mathsf{Lift}(\boldsymbol{I}_n, \boldsymbol{T}'),\end{aligned}$$

while $\mathsf{Can}(\boldsymbol{Z}) = \mathsf{Can}(\boldsymbol{T}')$ holds trivially since $\boldsymbol{Z} = \boldsymbol{T}'$. □

E CELeReS-o: Optimisation via Self-Orthogonal Codes

A substantial part of the signature size of CELeReS, as detailed in Section 5.3, comes from the tag matrix $\boldsymbol{T}_i$. Here, we introduce CELeReS-o, a variant adopting the optimisation of Baldi et al. [6] where the base tag matrix $\boldsymbol{T}$ is chosen to be self-orthogonal and, for each $i \in [N]$, we set $\boldsymbol{T}_i = \mathsf{RREF}(\boldsymbol{T}\boldsymbol{P}_i)$, where $\boldsymbol{P}_i$ is the permutation underlying the secret-key monomial $\boldsymbol{Q}_i = \boldsymbol{P}_i\boldsymbol{D}_i$. The diagonal factor $\boldsymbol{D}_i$ is retained in the public key but omitted from the tag key. Since permutations and row operations preserve self-orthogonality, each $\boldsymbol{T}_i$ is also self-orthogonal, enabling the following compression.

Instead of storing the full matrix $\boldsymbol{T}_i$, we store its RREF pivot mask and exploit self-orthogonality to serialise only about half of the non-pivot block. More precisely, after fixing an invertible column ordering, each new row is stored only partially, while its missing prefix is recovered from the orthogonality equations with the previous rows. This yields an encoding of size

$$\ell_{\mathsf{tag}} = n + k\lceil \log_2 k \rceil + \frac{k(k+1)}{2}\lceil \log_2 q \rceil$$

bits, and reduces the signature size to

$$\ell_\sigma = \ell_{\mathsf{tag}} + 2\ell_{\mathsf{digest}} + \ell_{\mathsf{count}} + n_{\mathsf{nodes}} \cdot \ell_{\mathsf{seed}} + W\ell_{\mathsf{rsp}}.$$

Security Problems for CELeReS-o. The optimisation does not change the relations underlying the three problems in Problems 3 to 5 but it changes the distribution of the base tag and of its secret transformation. Let

$$X_{\mathsf{SO}} = \{\boldsymbol{T} \in X : \boldsymbol{T}\boldsymbol{T}^{\mathsf{T}} = \boldsymbol{0}\}.$$

The base tag is now chosen in X_{SO}. Moreover, writing the public-key monomial as $\boldsymbol{Q} = \boldsymbol{P}\boldsymbol{D}$, key generation sets

$$\boldsymbol{G}' = \mathsf{RREF}(\boldsymbol{G}\boldsymbol{P}\boldsymbol{D}) \qquad \text{and} \qquad \boldsymbol{T}' = \mathsf{RREF}(\boldsymbol{T}\boldsymbol{P}).$$

Thus, compared with the distribution used above, the public-key component is unchanged, while the independent diagonal factor on the tag component is fixed to the identity.

For 2-one-wayness, the adversary receives $(\boldsymbol{G}, \boldsymbol{T}, \boldsymbol{G}', \boldsymbol{T}')$ from this modified distribution, but its goal remains exactly the one in Problem 3. For non-splitting, the task in Problem 4 is unchanged; only the fixed base tag $\boldsymbol{T}$ is restricted to X_{SO}, while the adversarially produced $\boldsymbol{T}_0, \boldsymbol{T}_1$ remain arbitrary elements of X. Finally, in the 2-pseudorandomness problem, the correlated and independent distributions in Problem 5 become, respectively,

$$\big(\mathsf{RREF}(\boldsymbol{G}\boldsymbol{P}\boldsymbol{D}), \mathsf{RREF}(\boldsymbol{T}\boldsymbol{P})\big) \qquad \text{and} \qquad \big(\mathsf{RREF}(\boldsymbol{G}\boldsymbol{P}_0\boldsymbol{D}_0), \mathsf{RREF}(\boldsymbol{T}\boldsymbol{P}_1)\big),$$

where $\boldsymbol{P}, \boldsymbol{P}_0, \boldsymbol{P}_1 \leftarrow\!\!\$\ \mathsf{Perm}_n(\mathbb{F}_q)$ and $\boldsymbol{D}, \boldsymbol{D}_0 \leftarrow\!\!\$\ \mathsf{Diag}_n(\mathbb{F}_q)$ are sampled independently.

Because $\boldsymbol{T}$ generates a self-orthogonal code, not all security considerations done for CELeReS in Section 5.2 hold for CELeReS-o. In particular, Proposition 1 is specific to random codes only. Random binary self-orthogonal codes are known to have trivial automorphism group with high probability [47], but no analogous result is known over odd prime fields. We therefore recommend further cryptanalysis of this setting before assigning this variant the same security confidence as CELeReS. Nevertheless, for the sake of evaluating the optimization, we instantiate CELeReS-o with the same parameters of CELeReS and we present in Table 3 the signature sizes and performances to be compared with Tables 1 and 2. Overall, CELeReS-o presents substantial savings in terms of signature size especially for rings of small size. On the other hand, its performance degrades slightly in signing and verification, and more significantly in key generation.

Signature sizes

Security	2^1	2^2	2^6	2^{10}	2^{12}
128 bit	18.9	19.9	24.5	28.9	31.2
192 bit	50.8	54.0	66.7	79.5	85.9
256 bit	85.4	90.1	108.8	127.6	136.9

Performance at 128-bit security

Ring size	KeyGen	Sign	Verify
2^1		511 M/0.171 s	477 M/0.159 s
2^3	55.0 M/0.018 s	2508 M/0.834 s	2090 M/0.697 s
2^6		9836 M/3.255 s	7647 M/2.543 s

Table 3: Signature sizes (KB) and performance in average millions of CPU cycles (M) and seconds (s) of CELeReS-o over 100 runs.