Advanced Signature Functionalities from the Code Equivalence Problem

Alessandro Barenghi[1], Jean-François Biasse [2], Tran Ngo[3], Edoardo Persichetti[3] and Paolo Santini[4]

[1]Politecnico di Milano, Italy; [2]University of South Florida, USA; [3] Florida Atlantic University, USA; [4] Università Politecnica delle Marche, Italy

ARTICLE HISTORY
Compiled May 30, 2022

ABSTRACT
The LESS signature scheme, introduced in 2020, represented a fresh start for code-based signatures. In this paper we explore advanced functionalities for signature schemes, stemming from the work of LESS. First, we adapt a recent protocol of Beullens et al. to obtain a construction for (linkable) ring signatures. Then, we realize an identity-based signature scheme following the traditional approach by Joye and Neven. Our performance numbers confirm that signature schemes based on the code equivalence problem have considerable potential for practical applications.

KEYWORDS
Post-Quantum; Signatures; Identity-Based; Ring Signatures; Code Equivalence

1. Introduction

Digital signatures are arguably one of the most important cryptographic primitives in the modern communication world, providing a powerful means of authentication in several crucial contexts such as software distribution, blockchain, electronic voting and many others. It stands to reason, then, that a considerable amount of effort is being made to establish new designs for digital signature schemes, that are resistant to quantum attackers. As is well known, in fact, the vast majority of cryptographic protocols currently in use is threatened by algorithms such as Shor's [30], which would render them insecure. As a consequence, important initiatives are taking place, aimed at creating new standards for so-called Post-Quantum Cryptography (PQC), such as the one led by the National Institute of Standards and Technology (NIST) [1]. The process, started in 2017, now arrived to its third round, after which certain designs will already be standardized; for digital signatures, these are most likely designs based on hard problems from lattices, such as Dilithium [20] or Falcon [22]. Besides these, NIST is considering the standardization of alternative schemes, whose hardness relies on different mathematical assumptions; in this regard, the current range of candidates is somewhat limited, to the point of NIST announcing a partial reopening of the call [2], with the goal of attracting new proposals.

CONTACT ?? Email: ??

Code-based cryptography is one of the major players in the PQC scenario, with schemes such as Classic McEliece, based on McEliece's seminal work from 1978 [25], being among the most credible candidates for the key establishment functionality. The history of code-based signatures is somewhat more complicated, with several schemes being proposed over the years (e.g. [16, 18, 32]), which only recently started to get closer to a desirable level of performance [5, 7, 19]. The situation is even more sketchy when considering advanced functionalities, such as ring signatures, group signatures, identity-based signatures etc. In this case, only few proposals exist in literature, which are all based on either Stern's zero-knowledge identification scheme [32] or the CFS scheme [18], leading to very impractical solutions. It is therefore of extreme interest to investigate alternative code-based approaches to achieve such functionalities.

Our Contribution. In this work, we introduce two new designs for code-based signatures with advanced functionalities. To do so, we leverage the LESS signature scheme [13], a newcomer in the code-based ecosystem. Indeed, LESS is based on a security notion, the code equivalence problem, which is only indirectly related to the hardness of decoding; this makes it possible to exploit the associated group action, and construct protocols which are substantially different from the rest of the literature. To be precise, such protocols are similar in spirit to those based, for example, on the Discrete Logarithm Problem, and this is what makes LESS particularly suitable to construct the protocols in the first place.

As a first contribution, we give a construction for a ring signature scheme. This kind of primitive has found increasing importance in recent times, thanks to its anonymity guarantees. To build our protocol, we leverage the general framework described in [12], which utilizes a simple cryptographic group action; applying it to our case, we are able to construct a ring signature scheme whose security relies entirely on the code equivalence problem. To the best of our knowledge, this is the first time a code-based ring signature scheme is built using a problem other than Syndrome Decoding (SDP). The construction incorporates a variety of computational optimizations that are common in this kind of protocols (but not in syndrome-based schemes), such as Merkle trees and unbalanced challenge space. We then show how, in principle, it would be possible to extend our construction to obtain a linkable scheme, by using a *pair* of group actions that link nicely with each other. To do so, however, one would need to rely on a new, untested computational assumption, and we thus feel that, at this stage, it is best to avoid an explicit construction, until further study is conducted, and more confidence is acquired.

Our second contribution, is the design of an identity-based signature scheme. Such schemes have the major benefit of eliminating the need for certificates, and find use in applications such as smart cars [4] and many others. Our protocol follows the construction of [23], which utilizes two signature instances "nested" within each other, where one serves as a certificate for the other. Once more, this is the first such protocol built on code equivalence, and the construction again features certain computational optimizations, that bring it to the front of the line when compared to previous code-based solutions.

In terms of performance, our calculations show how both proposals are at the cutting edge, when compared to the state of the art in code-based cryptography. For ring signatures, our construction exhibits a signature size that grows only logarithmically with the number of users in the ring; this allows to outpace existing code-based solutions and compares very well even with other post-quantum schemes (e.g. [11]).

For identity-based signatures, our certificate-based protocol is far superior to the alternatives, which are largely based on either CFS or Stern, and thus suffer from the respective flaws (large keys and signatures). In both cases, we expect the schemes to provide good timings, as the computational overhead is minimal (essentially, the same as LESS).

2. Background

2.1. Notation

We will use the following conventions throughout the rest of the paper:

a	a scalar
A	a set
$\mathbf{a}$	a vector
$\mathbf{A}$	a matrix
a	a protocol object (keys, identities etc.)
A	a function or relation
$\mathcal{A}$	an algorithm
$\mathbf{I}_n$	the $n \times n$ identity matrix
λ	a security parameter

We denote with $\mathbb{Z}_q$ the ring of integers modulo q, and with $\mathbb{F}_q$ the finite field of order q. The multiplicative group of $\mathbb{F}_q$ is indicated as $\mathbb{F}_q^*$. We denote with $\mathsf{Aut}(\mathbb{F}_q)$ the group of automorphisms of the field $\mathbb{F}_q$. The sets of vectors and matrices with elements in $\mathbb{Z}_q$ (resp. $\mathbb{F}_q$) are denoted by $\mathbb{Z}_q^n$ and $\mathbb{Z}_q^{m\times n}$ (resp. $\mathbb{F}_q^n$ and $\mathbb{F}_q^{m\times n}$). We also write $\mathbb{Z}_{q,w}^n$ (resp. $\mathbb{F}_{q,w}^n$) to indicate the set of vectors with components in $\mathbb{Z}_q$ (resp. $\mathbb{F}_q$) with length n and Hamming weight w. For our purposes, we will sometimes interpret bit strings as integers: for instance, an ℓ-bit string, can be seen as an integer between 0 and $2^\ell - 1$. To simplify notation and avoid confusion, we will use boldface to distinguish between the bit string and the corresponding integer (e.g. writing $\mathbf{h}_i$ for the bit string, and h_i for the corresponding integer).

We write $\mathsf{GL}_k(q)$ for the set of invertible $k\times k$ matrices with elements in $\mathbb{F}_q$. Let Sym_n be the set of permutations over n elements. For a vector $\mathbf{x} \in \mathbb{F}_q^n$ and a permutation $\pi \in \mathsf{Sym}_n$, we write the action of π on $\mathbf{x}$ as $\pi(\mathbf{x})$. A permutation can equivalently be described as an $n \times n$ matrix $\mathbf{P}$ with exactly one 1 per row and column. Analogously, for *linear isometries*, i.e. transformations $\mu = (\mathbf{v}; \pi) \in \mathbb{F}_q^{*n} \rtimes \mathsf{Sym}_n$, we write the action on a vector $\mathbf{x}$ as $\mu(\mathbf{x})$. We can also describe these in matrix form as a product $\mathbf{Q} = \mathbf{DP}$ where $\mathbf{P}$ is an $n\times n$ permutation matrix and $\mathbf{D}$ is an $n\times n$ diagonal matrix with entries in $\mathbb{F}_q^*$. We denote with Mono_n the set of such matrices, usually called *monomial*.

2.2. Technical Definitions

In this section, we briefly present the technical tools and definitions we need to realize our constructions for advanced functionalities. We begin with a notion that is central to this work.

Definition 2.1. Let X be a set and $(G, \circ)$ be a group. A group action is a mapping

$$\begin{array}{rccc} \star: & X \times G & \to & X \\ & (x, g) & \to & x \star g \end{array}$$

such that, for all $x \in X$ and $g_1, g_2 \in G$, it holds that $(x \star g_1) \star g_2 = x \star (g_1 \circ g_2)$.

A group action is usually called *cryptographic* if it satisfies some additional properties that make it interesting in a cryptographic context. For instance, in the first place, and besides efficient sampling, computation, and membership testing, a cryptographic group action should certainly be *one-way*, i.e. given randomly chosen $x_1, x_2 \in X$, it should be hard to find $g \in G$ such that $x_1 \star g = x_2$ (if such a g exists).

The first part of this work is about designing ring signature schemes. For this purpose, the authors in [12] define an additional flavor of group action called *admissible*, to enable the option of aborting during the signing process. As we will see, our group action, based on the code equivalence problem and monomial matrices, requires no rejection sampling, and thus such a notion is not necessary for this work. We instead proceed to lay out the basic definitions for the schemes that we will define. As the definition of zero-knowledge identification schemes is rather standard, we have chosen to include it in the appendix (see Appendix A), and thus, the first definition that we will give is that of ring signature. Note that, from this point onwards, when describing our schemes, we always assume that any sort of public information (public data, system parameters, public keys etc.) is available as input to every party, and thus, for ease of readability, is sometimes omitted from an explicit input definition.

Definition 2.2. A *Ring Signature (RS) scheme* is a protocol between $r+1$ parties: (potential) signers $\mathcal{S}_1, \ldots, \mathcal{S}_r$ and a verifier $\mathcal{V}$. The protocol is composed of the following procedures:

I. Setup: on input the public data (including system parameters), output a secret key sk_i and a public key pk_i for each signer $\mathcal{S}_i$.

II. Sign: on input a set of public keys $R = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_r\}$, a secret key sk_{i^*} and a message msg, output a signature σ.

III. Verify: on input a set of public keys $R = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_r\}$, a message msg and a signature σ, $\mathcal{V}$ outputs either 1 (accept) if the signature is valid, or 0 (reject) otherwise.

Ring signatures owe their name to the fact that the group of people comprising the set R is usually called a "ring", even though this may be slightly confusing to the non-specialist, as it does not correspond to the mathematical understanding of the word. A crucial feature of such a scheme is that anyone in the ring can produce a valid signature (on behalf on the entire ring), and the verifier can only check validity but will not learn who was the signer. This is guaranteed by requiring that the scheme satisfies the following list of properties. *Correctness*, as usual, means that honestly generated signatures are always accepted. More significantly, *anonymity* captures the idea of protecting the identity of the signer, i.e. it should be impossible for a verifier to tell which secret key was used to sign. Finally, *unforgeability* corresponds to the familiar security notion for signature schemes, extended to include all ring participants, i.e. it should be infeasible to forge a valid signature without knowing at least one of the secret keys in the ring. For further details, we refer the reader to [12].

Linkable ring signature schemes are enhanced with additional security guarantees. These protocols are composed of the same three procedures described in Definition 2.2, plus a fourth one, given below:

IV. Link: on input two signatures σ and σ', outputs either 1 if the signatures were produced with the same secret key, or 0 otherwise.

A linkable RS scheme is required to satisfy the following properties, besides correctness. *Linkability* means that an adversary should not be able to produce more than r unlinked valid signatures, even if some or all of the r public keys are malformed; in other words, if such a number of valid signatures is produced, then the Link algorithm will output 1 on at least two of them. *Linkable anonymity* is a stronger version of the anonymity property, so that even if an adversary obtains multiple signatures from the same signer, it is still unable to determine which secret key was used. Finally, *non-frameability* guarantees that an adversary cannot produce a valid signature that is linked to one produced by an honest party. Note that, as pointed out in [12, Remark 2.4], unforgeability is directly implied by linkability plus non-frameability.

We now recall the basics of identity-based signatures.

Definition 2.3. An *Identity-Based Signature (IBS) scheme* is a protocol between three parties: a trusted key generation center $\mathcal{C}$, a signer $\mathcal{S}$ corresponding to identity id and a verifier $\mathcal{V}$. The protocol is composed of the following procedures:

I. Setup: on input the public data (including system parameters), $\mathcal{C}$ returns a master secret key msk and a master public key mpk.

II. $\mathsf{Extract}$: on input a master secret key msk and a user identity id, $\mathcal{C}$ generates and assigns a user secret key usk to $\mathcal{S}$.

III. Sign: on input a user identity id, a user secret key usk and a message msg, $\mathcal{S}$ outputs a signature σ.

IV. Verify: on input a user identity id, a master public key mpk, a message msg and a signature σ, $\mathcal{V}$ outputs either 1 (accept) if the signature is valid, or 0 (reject) otherwise.

This definition comes together with the usual correctness requirement. The desirable security notion for an IBS scheme is Existential Unforgeability under Chosen Message Attack (EUF-CMA), as is standard for signature schemes, with the exception that, in addition to signing queries, an adversary has access to *extract queries* as well, which return the user secret key for arbitrary identities. The adversary's goal is to produce a valid message/signature pair $(\mathsf{msg}^*, \sigma^*)$ for a certain identity id^*. For further details, including precise definitions, we refer the interested reader to the literature (e.g. [23]).

3. Code Equivalence

An $[n,k]$-*linear code* $\mathfrak{C}$ of length n and dimension k over $\mathbb{F}_q$ is a k-dimensional vector subspace of $\mathbb{F}_q^n$. It can be represented by a matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$, called *generator matrix*, whose rows form a basis for the vector space, i.e., $\mathfrak{C} = \{\mathbf{uG},\ \ \mathbf{u} \in \mathbb{F}_q^k\}$. There exists a standard choice of basis, called *systematic form*, which is given by $\mathbf{G} = (\mathbf{I}_k \mid \mathbf{M})$. This can be obtained by calculating the row-reduced echelon form, starting from any other generator matrix, and corresponds to the action of multiplying on the left by an invertible matrix. We denote such a procedure by SF. Finally, we recall the concept of *equivalence* between two codes which, in its most general formulation, is defined as follows.

Definition 3.1 (Code Equivalence)**.** We say that two linear codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *equivalent*, and write $\mathfrak{C} \sim \mathfrak{C}'$, if there exists a field automorphism $\alpha \in \mathsf{Aut}(\mathbb{F}_q)$ and a linear isometry $\mu = (\mathbf{v}; \pi) \in \mathbb{F}_q^{*n} \rtimes \mathsf{Sym}_n$ that maps $\mathfrak{C}$ into $\mathfrak{C}'$, i.e. such that $\mathfrak{C}' = \mu(\alpha(\mathfrak{C})) = \{\mathbf{y} \in \mathbb{F}_q^n \ : \mathbf{y} = \mu(\alpha(\mathbf{x})),\ \mathbf{x} \in \mathfrak{C}\}$.

Clearly, if $\mathfrak{C}$ and $\mathfrak{C}'$ are two codes with generator matrices $\mathbf{G}$ and $\mathbf{G}'$, it holds that

$$\mathfrak{C} \sim \mathfrak{C}' \iff \exists(\mathbf{S}; (\alpha, \mathbf{Q})) \in \mathsf{GL}_k(q) \rtimes (\mathsf{Aut}(\mathbb{F}_q) \times \mathsf{Mono}_n) \text{ s.t. } \mathbf{G}' = \mathbf{S}\alpha(\mathbf{G}\mathbf{Q}).$$

The notion we just presented is usually known as *semilinear equivalence* and it is the most generic. If the field automorphism is the trivial one (i.e. $\alpha = id$), then the notion is simply known as *linear equivalence*. If, furthermore, the monomial matrix is a permutation (i.e. $\mathbf{Q} = \mathbf{D}\mathbf{P}$ with $\mathbf{D} = \mathbf{I}_n$), then the notion is known as *permutation equivalence*. Note that in this work, as in previous literature [7, 13], we only consider the last two notions. Finally, we state the following decisional problem.

Problem 3.2 (Code Equivalence)**.** *Let $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$ be two generator matrices for, respectively, linear codes $\mathfrak{C}$ and $\mathfrak{C}'$. Determine whether the two codes are equivalent, i.e. if there exist two matrices $\mathbf{S} \in \mathsf{GL}_k(q)$ and $\mathbf{Q} \in \mathsf{Mono}_n$ such that $\mathbf{G}' = \mathbf{S}\mathbf{G}\mathbf{Q}$.*

Note that, although this problem is traditionally formulated as a decisional problem in literature, for our purposes we will often consider the search version. We will refer, respectively, to *linear* or *permutation equivalence problem*, according to what is the notion of code equivalence considered, or simply to the *code equivalence problem* if such a distinction is not important.

We now present the group action associated to code equivalence, as defined in [7]. First, consider the set $X \subseteq \mathbb{F}_q^{k\times n}$ of all full-rank $k \times n$ matrices, i.e. the set of generator matrices of $[n, k]$-linear codes. Then, let $G = \mathsf{GL}_k(q) \rtimes \mathsf{Mono}_n$. Note that this group is isomorphic to $(\mathsf{GL}_k(q) \times (\mathbb{F}_q^*)^n) \rtimes \mathsf{Sym}_n$ if we decompose each monomial matrix $\mathbf{Q} \in \mathsf{Mono}_n$ as a product $\mathbf{D} \cdot \mathbf{P} \in (\mathbb{F}_q^*)^n \rtimes \mathsf{Mono}_n$. The group operation $\circ$ is defined as

$$((\mathbf{S}, \mathbf{D}); \mathbf{P}) \circ ((\mathbf{S}', \mathbf{D}'); \mathbf{P}') = (\mathbf{S}\mathbf{S}', \mathbf{D}\mathbf{D}'\mathbf{P}); \mathbf{P}\mathbf{P}').$$

The group action is given by

$$\begin{array}{rccc} \star: & X \times G & \to & X \\ & (\mathbf{G}, (\mathbf{S}; \mathbf{Q})) & \to & \mathbf{S}\mathbf{G}\mathbf{Q} \end{array}$$

It is easy to see that the action is well-formed, with the identity element being $(\mathbf{I}_k; \mathbf{I}_n)$. The action satisfies the usual computation and samplability requirements, and is also *one-way*, based on the hardness of the code equivalence problem.

Remark 1. While the definition above is given in full generality, it makes sense to consider a simplified version. In fact, to avoid trivial instances, in which the action returns a different generator matrix for the same code, one can assume that X contains only the (full-rank) generator matrices in systematic form. Keeping in mind, also, that the semilinear case is not considered in this (or previous) work, one can then simplify the group action to the computation of the systematic form $\mathsf{SF}(\mathbf{S}\mathbf{G}\mathbf{Q})$ or, in fact, to just $\mathsf{SF}(\mathbf{G}\mathbf{Q})$.

It is worth noting that, unlike other group actions (e.g. based on lattices), our definition benefits from the fact that, in the group of monomial matrices, it is very easy to produce uniform random elements. In fact, we have the following trivial result (the proof is omitted in the interest of space).

Lemma 3.3. *Let Mono_n be the set of monomial matrices as defined in Section 2. If $\mathbf{A} \in \mathsf{Mono}_n$ is fixed and $\mathbf{B}$ is picked uniformly at random in Mono_n, then $\mathbf{A}\mathbf{B}$ is uniformly distributed over Mono_n.*

An important consequence of this result is that, as we will see, no rejection sampling is necessary, which greatly simplifies the protocol definition, and contributes to its efficiency. In the next section, we will briefly summarize the LESS scheme and its derivatives, which are based on the group action we just presented.

4. The LESS Signature Scheme

The LESS signature scheme was introduced in [13], where the authors present a zero-knowledge identification scheme, and then mention how this can be converted to a full-fledged signature scheme using the Fiat-Shamir transformation. The work was later revisited in [7], which introduces some protocol variants aimed at optimizing performance, and reducing signature size. More precisely, two variants are presented: the -M variant, which exploits the idea of multi-bit challenges using multiple public keys, and the -F variant, which utilizes challenge strings of fixed Hamming weight. Below, we recall the description of the LESS-FM scheme, which can be seen as the most generic, high-level description of the framework, of which the LESS scheme is a particular case.

Public Data
Parameters $q, n, k, \lambda, \ell, t, \omega \in \mathbb{N}$, with $r = 2^\ell - 1$. Generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$.
Weight-restricted hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_{2^\ell,\omega}^t$.

I. Setup
Input: -

1. Set $\mathbf{Q}_0 = \mathbf{I}_n$ and $\mathbf{G}_0 = \mathsf{SF}(\mathbf{G})$.
2. For all $i = 1, \ldots, r$:
   i. Select uniformly at random $\mathbf{Q}_i \in \mathsf{Mono}_n$.
   ii. Compute $\mathbf{G}_i = \mathsf{SF}(\mathbf{G}\mathbf{Q}_i)$.
3. Set $\mathsf{sk} = \{\mathbf{Q}_0, \ldots, \mathbf{Q}_r\}$.
4. Set $\mathsf{pk} = \{\mathbf{G}_0, \ldots, \mathbf{G}_r\}$.

II. Sign
Input: sk and msg.

1. For all $i = 1, \ldots, t$:
   i. Select uniformly at random $\tilde{\mathbf{Q}}_i \in \mathsf{Mono}_n$.
   ii. Compute $\tilde{\mathbf{G}}_i = \mathsf{SF}(\mathbf{G}\tilde{\mathbf{Q}}_i)$.
2. Compute $\mathbf{h} = \mathsf{H}(\tilde{\mathbf{G}}_1, \ldots, \tilde{\mathbf{G}}_t, \mathsf{msg})$.
3. Parse $\mathbf{h}$ as $(\mathbf{h}_1, \ldots \mathbf{h}_t)$, for $\mathbf{h}_i \in \{0,1\}^\ell$.
4. For all $i = 1, \ldots, t$:
   i. Compute $\mu_i = \mathbf{Q}_{h_i}^{-1}\tilde{\mathbf{Q}}_i$.
5. Set $\sigma = (\mu_1, \ldots \mu_t, \mathbf{h})$.

III. Verify
Input: $\mathsf{pk}, \mathsf{msg}$ and σ.

1. Parse $\mathbf{h}$ as $(\mathbf{h}_1, \ldots \mathbf{h}_t)$, for $\mathbf{h}_i \in \{0,1\}^\ell$.
2. For all $i = 1, \ldots, t$:
   i. Compute $\hat{\mathbf{G}}_i = \mathsf{SF}(\mathbf{G}_{h_i}\mu_i)$.
3. Compute $\mathbf{h}' = \mathsf{H}(\hat{\mathbf{G}}_1, \ldots, \hat{\mathbf{G}}_t, \mathsf{msg})$.
4. Accept if $\mathbf{h}' = \mathbf{h}$ or reject otherwise.

Figure 1.: The LESS-FM scheme

The EUF-CMA security of the scheme was discussed in [7], where the impact of each variant is analyzed, and proofs are provided. As far as the security of the code equivalence problem itself, we can say that the problem has been extensively studied in literature, with different techniques leading to different solvers [10, 24, 29]. A full analysis of such attacks is out of the scope of this paper, and we refer the reader to [8], where the subject is given an extensive treatment.

5. Ring Signatures

We present here our construction for an RS scheme. This will rely on a basic building block, which is closely related to the LESS identification scheme [13]. The scheme described below is to be intended as a single round (with soundness error equal to 1/2), within the context of an iterated protocol. The scheme makes use of a dedicated construction called *index-hiding Merkle tree* [12], which we will recall in Section 7.

Public Data
Parameters $q, n, k, \lambda, r \in \mathbb{N}$. Generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$.
Commitment scheme $\mathsf{Com} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{2\lambda}$.

I. Setup
Input: -

1. For all $i = 1, \ldots, r$:
   i. Select uniformly at random $\mathbf{Q}_i \in \mathsf{Mono}_n$.
   ii. Compute $\mathbf{G}_i = \mathsf{SF}(\mathbf{G}\mathbf{Q}_i)$.
   iii. Set $\mathsf{sk}_i = \mathbf{Q}_i$.
   iv. Set $\mathsf{pk}_i = \mathbf{G}_i$.

II. Commit
Input: $\mathsf{pk}_1, \ldots, \mathsf{pk}_r$.

1. Select uniformly at random $\tilde{\mathbf{Q}} \in \mathsf{Mono}_n$.
2. For all $i = 1, \ldots, r$:
   i. Sample uniformly at random $\mathsf{r}_i \in \{0,1\}^\lambda$.
   ii. Compute $\tilde{\mathbf{G}}_i = \mathsf{SF}(\mathbf{G}_i\tilde{\mathbf{Q}})$.
   iii. Compute $\mathbf{h}_i = \mathsf{Com}(\mathsf{r}_i, \tilde{\mathbf{G}}_i)$.
3. Compute $\mathsf{root} \xleftarrow{\mathsf{tree}} (\mathbf{h}_1, \ldots, \mathbf{h}_r)$.
4. Send root to verifier.

III. Challenge
Input: -

1. Select uniformly at random a bit $b \in \{0,1\}$.
2. Send b to prover.

IV. Response
Input: b and sk_{i^*}, for $i^* \in \{1, \ldots, r\}$.

1. If b = 0 then:
   i. Compute $\mu = \mathbf{Q}_{i^*}\tilde{\mathbf{Q}}$.
   ii. Compute $\mathsf{path} \xleftarrow{\mathsf{tree}} (\mathbf{h}_1, \ldots, \mathbf{h}_r, i^*)$.
   iii. Set $\mathsf{bits} = \mathsf{r}_{i^*}$.
   iv. Set $\mathsf{rsp} = (\mu, \mathsf{path}, \mathsf{bits})$.
2. Else:
   i. Set $\mu = \tilde{\mathbf{Q}}$.
   ii. Set $\mathsf{bits} = (\mathsf{r}_1, \ldots, \mathsf{r}_r)$.
   iii. Set $\mathsf{rsp} = (\mu, \mathsf{bits})$.
3. Send rsp to verifier.

V. Verify

Input: $\mathsf{pk}_1, \ldots, \mathsf{pk}_r, \mathsf{root}$ and rsp.

1. If b $= 0$ then:
   i. Compute $\hat{\mathbf{G}} = \mathsf{SF}(\mathbf{G}\mu)$.
   ii. Compute $\mathbf{h}' = \mathsf{Com}(\mathsf{bits}, \hat{\mathbf{G}})$.
   iii. Compute $\mathsf{root}' \xleftarrow{\mathsf{tree}} (\mathbf{h}', \mathsf{path})$.
   iv. Accept if $\mathsf{root}' = \mathsf{root}$ or reject otherwise.
2. Else:
   i. For all $i = 1, \ldots, r$:
      † Compute $\hat{\mathbf{G}}_i = \mathsf{SF}(\mathbf{G}_i\tilde{\mathbf{Q}})$.
      ‡ Compute $\mathbf{h}'_i = \mathsf{Com}(\mathsf{r}_i, \hat{\mathbf{G}}_i)$.
   ii. Compute $\mathsf{root}' \xleftarrow{\mathsf{tree}} (\mathbf{h}'_1, \ldots, \mathbf{h}'_r)$.
   iii. Accept if $\mathsf{root}' = \mathsf{root}$ or reject otherwise.

Figure 2.: The basic identification scheme for ring R.

Essentially, the scheme above is a simple variation of LESS, with multiple public keys (corresponding to each user in the ring) and a single challenge bit. The scheme already includes some formal modifications: for instance, we have replaced the hash function H with a commitment scheme, and we utilized a Merkle tree for the protocol commitments. The latter is not only an optimization, as is commonly treated, but, due to the index-hiding property, represents a crucial factor in providing anonymity for the scheme. Note that, as opposed to the original LESS formulation, the scheme follows what could be seen as a "dual" framework, in which the commitment is formed from the public key (rather than from the public data $\mathbf{G}$) and the response is of the form $\mathbf{Q}_i\tilde{\mathbf{Q}}$ (rather than $\mathbf{Q}_i^{-1}\tilde{\mathbf{Q}}$). Moreover, the protocol in [12] swaps the role of the challenge bit, assigning to the 1 the uniform random response, and to 0 the response containing the private key. Both modifications, leading to the new formulation, are of course completely transparent in terms of security.

It is easy to show that the scheme presented above verifies the required security properties; to be sure, it is possible to reproduce the result given in Theorems 3.2 and 3.3 of [12], since this applies to the generic construction, by simply replacing the underlying hard problem. Indeed, in our case, the analysis is even simpler, and several redundant restrictions can be removed, since rejection sampling is not needed (thanks to Lemma 3.3). This eliminates all notions of aborting, together with the corresponding steps in the security proofs of the aforementioned theorems. For instance, we do not need to employ the concepts of *relaxed* sigma protocol, or an *admissible* group action. The identification scheme is then transformed into a full-fledged RS scheme via Fiat-Shamir in the standard way, incorporating a variety of additional optimizations. We will present these in Section 7, before discussing parameter choices. Below, then, we proceed to discuss the variations that would be necessary to build a linkable RS scheme.

The authors in [12] introduce a new object called an *admissible pair of group actions*. This gives continuity to the idea of admissible group actions, which, remember, is designed to allow for rejection sampling (and therefore not needed in our case). Consider two cryptographic group actions $\star : X \times G \to X$ and $* : Y \times G \to Y$ as in Definition 2.1, and a function $\mathsf{Link} : Y \times Y \to \{0, 1\}$. The pair of group actions needs to satisfy the following properties:

a) For all $y \in Y$, $\mathsf{Link}(y, y) = 1$.

b) Given $(x, y) \in X \times Y$, it is hard to output $g, g' \in G$ with $x \star g = x \star g'$ and $\mathsf{Link}(y * g, y * g') = 0$.

c) Given $(x, y) \in X \times Y$, the pair $(x \star g, y * g)$ is indistinguishable from (x', y'), where g and (x', y') are sampled uniformly at random from G and $X \times Y$, respectively.

d) Given $(x, y) \in X \times Y$, $x' = x \star g$ and $y' = y * g$, for g sampled uniformly at random from G, it is hard to output $g' \in G$ with $\mathsf{Link}(y * g', y') = 1$.

Note how the last three properties recall the linkability, linkable anonymity and non-frameability properties given in Section 2, respectively. As noted before, one could define unforgeability in a manner similar to the last property (by asking to output $g' \in G$ with $x \star g' = x'$) but this is not necessary, since this is a direct consequence of linkability and non-frameability. Informally, then, one can treat elements $y \in Y$ as "tags", that can be checked to establish the link.

In [12], for the isogeny-based construction, the second group action is obtained via composition, i.e. $* = \star^2$, so that $x * g = x \star^2 g = (x \star g) \star g$. The security then relies on an ad-hoc problem, named "Squaring Decisional CSIDH" (or sdCSIDH), which relates to the usual CSIDH in a manner that is similar to the discrete logarithm case; namely, a variant of the decisional Diffie-Hellman problem, called "Square decisional Diffie-Hellman", or SDDH, which is relatively well-known (see for example [6]). In our case, we could use the same approach, which would yield $x * g = \mathsf{SF}(\mathbf{G}\mathbf{Q}^2)$, and an analogue notion of "Squared Code Equivalence"; however, this definition is not the best in our case. For example, it is easy to see that, given monomial matrices $\mathbf{Q}$ and $\mathbf{Q}'$, the condition $\mathbf{Q}^2 = (\mathbf{Q}')^2$ does not imply that $\mathbf{Q} = \mathbf{Q}'$; this becomes an issue when considering the previous properties.

Despite the claim given in [12, Section 2.4], there is no hardness guarantee stemming from the parallel with discrete logarithms. In fact, as shown in [6], it is in general unknown whether SDDH is equivalent to DDH, as it is only possible to prove that SDDH $\leq$ DDH, but not viceversa.With this in mind, one could instead decide to leverage one of the other variations of Diffie-Hellman. For instance, it would be relatively easy to define a notion of "Inverse Code Equivalence", analogue to the InvDDH notion given in [6], by setting $x * g = \mathsf{SF}(\mathbf{G}\mathbf{Q}^{-1})$.

Problem 5.1 (Inverse Code Equivalence (ICE))**.** *Let* $\mathbf{G}, \mathbf{G}', \mathbf{G}'' \in \mathbb{F}_q^{k \times n}$ *be generator matrices, in systematic form, for, respectively, linear codes* $\mathfrak{C}, \mathfrak{C}'$ *and* $\mathfrak{C}''$*. Consider a matrix* $\mathbf{Q} \in \mathsf{Mono}_n$ *sampled uniformly at random. Determine whether* $(\mathbf{G}', \mathbf{G}'') = (\mathsf{SF}(\mathbf{G}\mathbf{Q}), \mathsf{SF}(\mathbf{G}\mathbf{Q}^{-1}))$.

The above definition would be a better choice from both a complexity theory point of view, as well as with regards to the previous properties. However, the computational hardness of this new security notion is still untested. For instance, attacks based on algebraic systems such as [28] could benefit from the additional equations provided by the matrix $\mathbf{Q}^{-1}$, and gain a significant advantage. Such an analysis is, at the moment, beyond the scope of this paper and so, in the interest of preserving the security of the construction, we choose to not flesh out the details of the linkable scheme. Suffice to say that, if the ICE problem proved to be safe in practice, it would be possible to satisfy all the necessary properties, and thus build a linkable ring signature scheme. We leave this task for future work, and proceed instead to the identity-based construction.

6. Identity-based Signatures

For our IBS scheme, we follow the generic approach of [23], since this yields the most efficient instantiation. In a nutshell, this approach uses a canonical signature scheme nested within itself, where the "inner" instantiation produces an ephemeral keypair, as well as a signature produced, to be used as a certificate for the public key which is part of the user's signing key. The signer can then use the ephemeral secret key to sign, and includes the public key, along with the certificate, in the scheme's signature. In our case, the underlying signature scheme is LESS-FM (Figure 1), to which, for brevity, we refer as L in the description below.

Public Data

Two sets of parameters $q_1, n_1, k_1, \ell_1, t_1, \omega_1 \in \mathbb{N}$ and $q_2, n_2, k_2, \ell_2, t_2, \omega_2 \in \mathbb{N}$ as in L, with $r_1 = 2^{\ell_1} - 1$, $r_2 = 2^{\ell_2} - 1$.
Matrices $\mathbf{G}' \in \mathbb{F}_q^{k_1 \times n_1}$ and $\mathbf{G}'' \in \mathbb{F}_q^{k_2 \times n_2}$.

I. Setup

Input: -

1. Set $\mathbf{Q}'_0 = \mathbf{I}_{n_1}$ and $\mathbf{G}'_0 = \mathsf{SF}(\mathbf{G}')$.
2. For all $i = 1, \ldots, r_1$:
   i. Select uniformly at random $\mathbf{Q}'_i \in \mathsf{Mono}_n$.
   ii. Compute $\mathbf{G}'_i = \mathsf{SF}(\mathbf{G}'\mathbf{Q}'_i)$.
3. Set $\mathsf{msk} = \{\mathbf{Q}'_0, \ldots, \mathbf{Q}'_{r_1}\}$.
4. Set $\mathsf{mpk} = \{\mathbf{G}'_0, \ldots, \mathbf{G}'_{r_1}\}$.

II. Extract

Input: msk and id.

1. Set $\mathbf{Q}''_0 = \mathbf{I}_{n_2}$ and $\mathbf{G}''_0 = \mathsf{SF}(\mathbf{G}'')$.
2. For all $i = 1, \ldots, r_2$:
   i. Select uniformly at random $\mathbf{Q}''_i \in \mathsf{Mono}_n$.
   ii. Compute $\mathbf{G}''_i = \mathsf{SF}(\mathbf{G}''\mathbf{Q}''_i)$.
3. Set $\mathsf{sk} = \{\mathbf{Q}''_0, \ldots, \mathbf{Q}''_{r_2}\}$.
4. Set $\mathsf{pk} = \{\mathbf{G}''_0, \ldots, \mathbf{G}''_{r_2}\}$.
5. Run algorithm II. of L on input msk and $\overline{\mathsf{msg}} = (\mathsf{pk}, \mathsf{id})$ to get signature $\bar{\sigma}$.
6. Set $\mathsf{usk} = \{\mathsf{sk}, \mathsf{pk}, \bar{\sigma}\}$.

III. Sign

Input: $\mathsf{id}, \mathsf{usk}$ and msg.

1. Parse usk as $\mathsf{usk} = \{\mathsf{sk}, \mathsf{pk}, \bar{\sigma}\}$.
2. Run algorithm II. of L on input sk and msg to get signature $\hat{\sigma}$.
3. Set $\sigma = \{\mathsf{pk}, \bar{\sigma}, \hat{\sigma}\}$.

IV. Verify

Input: $\mathsf{id}, \mathsf{mpk}, \mathsf{msg}$ and σ.

1. Parse σ as $\{\mathsf{pk}, \bar{\sigma}, \hat{\sigma}\}$.
2. Run algorithm IV. of L on input $\mathsf{pk}, \mathsf{msg}$ and $\hat{\sigma}$. Set $b = 1$ if the output is "Accept" and $b = 0$ if "Reject".
3. Run algorithm IV. of L on input $\mathsf{mpk}, \overline{\mathsf{msg}} = (\mathsf{pk}, \mathsf{id})$ and $\bar{\sigma}$. Set $b' = 1$ if the output is "Accept" and $b' = 0$ if "Reject".
4. Accept if $b \oplus b' = 1$ or reject otherwise.

Figure 3.: Our proposed IBS scheme.

The correctness of the scheme is immediate. As far as its security argument, note that this is standard, and follows directly from the EUF-CMA security of the underlying scheme (see for example [9]). It is interesting to note that, thanks to the inherent flexibility of the LESS-FM scheme, we are able to modulate the framework to accommodate different needs. In fact, we can use two flavors of LESS-FM for the nested schemes, corresponding to the parameters r_1 and r_2 (or actually, ℓ_1 and ℓ_2). The choice of these two parameters will be an important factor to optimize the scheme and obtain the maximum efficiency, as we will discuss in the next section.

7. Performance

7.1. Optimizations

Merkle Trees. Merkle trees [26] are a very well-known cryptographic primitive. In a nutshell, the goal of a Merkle tree is to provide an efficient way to verify that a certain element x^* is part of a list $(x_0, \ldots, x_r)$. To do that, the elements of the list are assigned to the leaves of a binary tree[1], say T, via a hash computation; in other words, the leaves are set as $T_{d,l} = \mathsf{H}(x_l)$, for all $l = 0, \ldots, r$, where $d = \lceil \log(r+1) \rceil$ is the tree depth. Then, the internal nodes are computed, at each level, as the hash digest of the concatenation of the two children, i.e. $T_{u,l} = \mathsf{H}(T_{u+1,2l} || T_{u+1,2l+1})$, for $u = 0, \ldots, d$ and $l = 0, \ldots, 2^u - 1$. One then has $\mathsf{root} = T_{0,0}$. To verify that x^* is part of the list, the element is matched to a position $i \in \{0, \ldots, r\}$ and the root is reconstructed using the *authentication path* corresponding to the position. This contains all the "companion" elements necessary for the computation; thus, path includes the sibling leaf, the sibling of their parent, etc. A pictorial representation is given below. Obviously, the verification is successful only if the reconstructed root is equal to the original one.

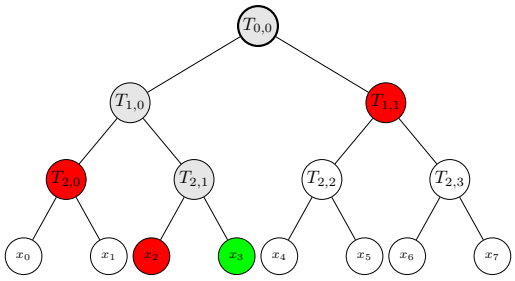

Figure 4.: Example of binary tree for $r = 7$. The element to verify is in green. The authentication path consists of the red nodes. The reconstructed nodes are marked in gray. The goal is to reconstruct the root (thick line).

The advantage of using Merkle trees is evident, as it allows to verify membership using only a logarithmic number of tree nodes (in exchange for the computational overhead of recomputing the root). Note that, as we mentioned in Section 5, the construction for a ring signature scheme uses a special variant of Merkle trees that are *index-hiding*, that is, they do not reveal the position of the element in the ring; this is crucial to ensure anonymity. Such a variant can be accomplished by specifying a different method to construct the tree, based on an alternative ordering (e.g. lexicographic), as explained in [12].

[1]For this reason, one usually has a number of elements equal to a power of two, i.e. $r = 2^\ell - 1$, which yields a complete tree.

Seeds. The idea of generating random objects via a PRNG is staple in cryptography. In the context of certain protocols, this becomes a tool to optimize communication cost as well, as is commonly the case with ZKID schemes. Our proposal for a ring signature scheme makes no exception; in fact, this was already done for the underlying LESS framework. The thing to observe is that, for a particular type of challenge, the object transmitted is a purely random monomial matrix $\tilde{\mathbf{Q}}$, for which it is sufficient to transmit the corresponding seed for the PRNG, instead. This was the case $b = 0$ in the original LESS, and $b = 1$ in the new scheme (as we have mentioned before, the roles are reversed, without impact). The verifier can then regenerate the object himself and proceed with verification. This alone saves a very significant amount in signature size (monomial matrices require $n(\lceil \log n \rceil + \lceil \log q \rceil)$ bits, whereas a seed is only λ bits). However, the usage of seeds can lead to further optimizations, as detailed next.

- Since the protocol uses a commitment scheme Com, which requires prefix randomness strings r_i, communication can be reduced by having the seed generate both $\tilde{\mathbf{Q}}$ and the r_i. Then, it is enough to set $\mathsf{rsp} = \mathsf{seed}$, and the verifier can use seed to regenerate the entire ensemble of objects.
- Given that the difference in cost between the seed and a monomial matrix is so large, it makes sense to try and minimize the number of the latter over the span of the protocol repetitions. Thus, instead of iterating the scheme for $t = \lambda$ times, in which each b is chosen uniformly at random, it is preferable to use an "unbalanced" distribution. To be precise, one can set $t > \lambda$ and then choose a combination of ω instances to have $b = 0$. This achieves the same security level provided that $\binom{t}{\omega} \geq 2^{\lambda}$, and has the additional advantage of yielding constant signature size and verification time. Note that this optimization was already present in [7], for instance.
- Considering that, over the course of the iterations, one needs to send multiple seeds (in fact, exactly $t - \omega$ if using the above optimization), it is possible to further reduce the space allocated towards seeds. The authors in [12] formalize the definition of *seed tree*, which was already informally utilized in literature (e.g. [11]). The main idea here, is to use but a single "master" seed, to generate an ensemble of t seeds, via a binary tree, and then reveal only[2] the $t - \omega$ that are requested. Unlike Merkle trees, a seed tree is generated in reverse order, that is from root to leaves; in other words, one sets $\mathsf{seed} = T_{0,0}$, then computes $(T_{u+1,2l}||T_{u+1,2l+1}) = \mathsf{PRNG}(T_{u,l})$, for $0 \leq u \leq d-1$ and $0 \leq l \leq 2^u - 1$, where $d = \lceil \log t \rceil$ as before. Each of the leaves will then constitute a separate seed, to be utilized in the respective round. To reveal the requested $t - \omega$ seeds, it is then enough to release the appropriate sequence of internal nodes[3], as depicted in Figure 5. For further details, we refer the reader to Section 2.7 of [12].

Remark 2. As suggested in [12], it is recommended to use "salt", by including a 2λ-bit prefix string in the random oracle computations. The impact of such a modification in practice is negligible, but it has the advantage of providing a tighter security proof, and preventing multi-target attacks.

[2] In fact, it is crucial that the ω seeds corresponding to instances $b = 0$ are *not* revealed, or security would be compromised.

[3] This number is variable, and depends on the actual position of the desired leaves.

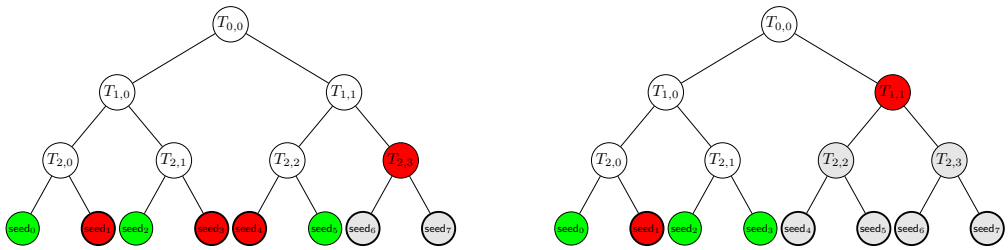

Figure 5.: Example of two binary trees for $t = 8, \omega = 3$. Color codes are similar to Figure 4, with the transmitted nodes marked in red, the reconstructed ones in gray and the target ones with the thick line. The green nodes here represent the seeds that should not be revealed. It is necessary to transmit 4 nodes for tree (a), and only 2 for tree (b).

7.2. Parameter Choice

We begin by providing an estimate of the various costs in the two protocols. As we will see, there will be very important differences between the two cases, leading to different priorities in terms of cost optimization.

Ring signatures. The computational costs of our RS schemes are closely related to those of LESS-FM. Indeed, the public key (per user) consists of exactly one generator matrix (in systematic form), and thus amounts to $k(n-k)\lceil \log q \rceil$ bits. Now, when considering signature size, we have to keep in mind the optimizations described in the first part of this section. Indeed, the base protocol needs to be repeated t times, out of which we have ω instances corresponding to the case $b = 0$. Furthermore, we use a seed tree to generate all the t seeds necessary for the various instances. Finally, we obtain our signature via Fiat-Shamir (thus, it is not necessary to transmit the commitments). In the end, the signature is composed by the following objects:

- the fixed-weight challenge string ch, of length t and Hamming weight $t - \omega$;
- ω responses for the case $b = 0$, consisting of a monomial matrix μ, an authentication path, and the randomness bits;
- the internal nodes necessary to reconstruct the $t - \omega$ seeds for the case $b = 1$.

This leads to the following computational cost (in bits):

$$\underbrace{\omega\lceil \log t \rceil}_{\mathsf{ch}} + \underbrace{\omega(n\lceil \log n \rceil + n\lceil \log q \rceil + 2\lambda\lceil \log r \rceil + \lambda)}_{\{\mathsf{rsp}_i\}} + \underbrace{\lambda\lceil \log t \rceil}_{\{seeds\}} \tag{1}$$

Identity-based signatures. The analysis of the computational costs in the IBS scheme is quite simple; yet, as we anticipated, it will be important to choose the optimal values for the LESS-FM instances, depending on their role within the framework. We begin by calculating the size of the master public key, and user secret key; these depend on the specific choice of parameters for the two instances, and are given as follows. The former consists of r_1 copies of generator matrices $\mathbf{G}'_i$, and thus amounts to $r_1 k_1 (n_1 - k_1)\lceil \log q_1 \rceil$ bits. The latter, contains the ephemeral keys sk and pk as well as the certificate signature $\bar{\sigma}$; thus, the user secret key is given (in bits) by:

$$\underbrace{r_2 n_2(\lceil \log n_2 \rceil + \lceil \log q_2 \rceil)}_{\mathsf{sk}} + \underbrace{r_2 k_2 (n_2 - k_2)\lceil \log q_2 \rceil}_{\mathsf{pk}} + \underbrace{\ell_1 t_1 + (t_1 - \omega_1)\lambda + \omega_1 n_1(\lceil \log n_1 \rceil + \lceil \log q_1 \rceil)}_{\bar{\sigma}}. \quad (2)$$

As far as the signature is concerned, this is composed by the following objects:

- the ephemeral public key pk;
- the certificate signature $\bar{\sigma}$;
- the message signature $\hat{\sigma}$.

Note that we have already calculated the first two items, as part of usk. This leads to the following formula for the signature length (in bits):

$$\underbrace{r_2 k_2 (n_2 - k_2)\lceil \log q_2 \rceil}_{\mathsf{pk}} + \underbrace{\ell_1 t_1 + (t_1 - \omega_1)\lambda + \omega_1 n_1(\lceil \log n_1 \rceil + \lceil \log q_1 \rceil)}_{\bar{\sigma}} + \underbrace{\ell_2 t_2 + (t_2 - \omega_2)\lambda + \omega_2 n_2(\lceil \log n_2 \rceil + \lceil \log q_2 \rceil)}_{\hat{\sigma}}. \quad (3)$$

We now proceed to select parameters for our scheme. To do this, we can draw from the recommendations given in [7] for the underlying LESS-FM scheme, according to our needs. For instance, for the case of ring signatures, it makes sense to minimize the size of the signature, and therefore select the scheme version based on the permutation equivalence problem; this would eliminate the factor of $n\lceil \log q \rceil$ associated to the monomial matrices, which is a considerable factor when determining signature size. On the other hand, the identity-based scheme presents different priorities. In the "outer" scheme (parameters q_1, n_1 etc.), the signature should be short, since this is part of the user secret key, while the size of the (master) public key is not a major concern. This allows us, for example, to leverage the tradeoff provided by the -M variant, which yields the shortest signatures. At the contrary, for the "inner" scheme, it is crucial that the public key is as small as possible, since this will be transmitted as part of the signature; for this task, using monomial matrices is optimal as it minimizes the size of the generator matrix. With all this in mind, we choose the appropriate parameter sets from [7], and report them below.

Set	Type	q	n	k	t	ω	r	**pk**	**sig**
I	Perm	127	230	115	233	31	2^3	11.57	10.76
II							2^6		13.74
III							2^{12}		19.69
IV							2^{21}		28.61

Table 1.: Different parameter sets for our RS scheme, corresponding to various ring sizes, for a security level of $\lambda = 128$ classical bits. All sizes in Kilobytes (kB).

Scheme	Type	q	n	k	ℓ	t	ω	mpk	usk	sig
IBS - Outer	Perm	251	235	108	4	66	19	205.74	15.43	30.23
IBS - Inner	Mono	251	198	94	1	283	28			

Table 2.: Sample parameters for our IBS scheme, for a security level of $\lambda = 128$ classical bits. All sizes in Kilobytes (kB).

7.3. Performance Analysis

Ring signatures. We first discuss the RS scheme. In this case, the most important aspect to discuss is the scalability of the signature size, according to the number of members of the ring. In fact, due to the construction employed, this size increases only logarithmically, i.e. proportionally to $\log r$. To be precise, the signature is composed by a "fixed" part, which amounts to 7.78 kB, plus a "variable" component (that depends on r), which weighs approximately $\log r$ kB. This provides an immediate advantage when compared with other schemes, as we will explain next.

In terms of code-based constructions for ring signatures, as we suggested in the beginning of this paper, not many solutions exist. The scheme of [33] is, to the best of our knowledge, the first to appear in literature; this is a straightforward adaptation of the CFS scheme, and is thus affected by the usual flaws, namely a very large public key (in excess of 1MB), and a very slow signing time. Moreover, what is usually a good feature of CFS, the short signature size, provides little advantage here as the RS construction scales linearly in r: the authors report a size equal to $144 + 126r$ bits, which yields signatures much larger than ours, when the ring includes at least 1000 users. For instance, the signature size for a ring of 2^{12} users is approximately 64 kB. A follow-up work [17] only realizes a minor improvement, with the signature differing only by a single syndrome (the constant component 144). In 2018, a construction for a linkable RS scheme was proposed [14], utilizing Stern's ZKID, rather than CFS. Once again, such a scheme inherits the traits of its core component, in this case small public keys but extremely large signatures. A later work [27] manages to reduce this number, but the reported figures (nearly 80 kB for $r = 2^6$ users) make it clear that the construction is completely impractical. Note that, while we have not fully fleshed out our own linkable RS design, this would be obtainable with a relatively small overhead (a single "tag" element), and still well below the threshold set by the aforementioned schemes.

Finally, for completeness, we draw a comparison with the Calamari and Falafl schemes [12]. As expected, the former produces smaller signatures, with a reported size of approximately $\log r + 3.5$ kB. Also as expected, such a scheme features a very slow signing time (79s) and moreover, due to the complexity of the class group computation, we do not believe it will be able to provide security levels higher than the listed one (NIST Category 1). On the other hand, our scheme compares very favorably with the lattice-based Falafl, which features larger signatures (in the order of $1/2 \log r + 29$ kB); unfortunately, the lack of an optimized implementation prevents us from a full comparison, with regards to signing times, as Falafl is able to leverage the pre-existing Dilithium code. Nevertheless, we expect our scheme to exhibit speeds of the same order of magnitude.

Identity-based signatures. The scenario of code-based IBS schemes resembles what we have just described for ring signatures. The first scheme appeared in literature in 2007 [15], although it seems this was never published in a venue with formal proceedings. The scheme uses a combination of CFS and a variant of Stern, featuring a huge public key (1 MB) and an even larger signature (1.5 MB), and is thus completely impractical. This was later improved in [3], using Quasi-Dyadic (QD) codes; despite a non-trivial reduction in both sizes, this is still very far from being useful in practice: reported timings are about 5 minutes for producing a single signature, and sizes are still in the order of hundreds of kilobytes. Note also that the parameters considered were only designed to achieve 80-bit (classical) security, so that all sizes would substantially increase if the scheme had to be adapted to fit our intended security target. Follow-up work (e.g. [31]) was unfortunately only able to yield even bigger sizes (with signatures as large as 35 MB), due to the concurrent appearance of a Generalized-Birthday attack by Bleichenbacher (see for example [21]). In the end, it is reasonable to claim that no efficient code-based solution exists to date.

In stark contrast with the state of the art, our protocol is able to obtain practical signature sizes, which are of an order of magnitude of just two signatures in the underlying building block. Using LESS as building block has the noticeable advantage of relying on just one security assumption, yet allows for additional flexibility in the choice of parameters, which can be modulated according to the different purposes, as shown above.

8. Conclusion

Code-based cryptography is one of the main avenues of research to develop quantum-resistant cryptographic primitives. Yet, the state of the art for code-based signatures is still not fully satisfactory, and the landscape of advanced signature schemes even less so. In this context, then, our work represents an important contribution. In fact, to the best of our knowledge, this is the first instance of code-based constructions for both ring and identity-based signatures, based on a problem other than syndrome decoding. More importantly, our construction is the first to offer concretely practical performance figures.

For ring signatures, our protocol fully exploits the logarithmic nature of the underlying framework, to obtain compact signature sizes that are able to beat those of other code-based constructions, as soon as the ring size is reasonably large. Furthermore, this comes with contained computational cost, and much smaller public keys. Remarkably, our protocol compares very well with isogeny-based and lattice-based solutions, providing a credible alternative in both cases. For identity-based signatures, the comparison swings even more heavily in our favor, given the inadequacy of existing proposals. The intrinsic flexibility of the LESS-FM scheme is inherited by our scheme, providing another relevant advantage factor. In conclusion, we see this work as an important witness towards the practicality of code-equivalence-based protocols in post-quantum cryptography.

Acknowledgments

The work of Jean-Francois Biasse is supported by NIST grant 60NANB17D184 and NSF grant 183980, while Edoardo Persichetti is supported by NSF grant 1906360.

References

[1] (2017), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

[2] (2021), https://csrc.nist.gov/Presentations/2021/status-update-on-the-3rd-round

[3] Alaoui, S.M.E.Y., Cayrel, P.L., Mohammed, M.: Improved identity-based identification and signature schemes using quasi-dyadic goppa codes. In: International Conference on Information Security and Assurance. pp. 146–155. Springer (2011)

[4] Ali, I., Lawrence, T., Li, F.: An efficient identity-based signature scheme without bilinear pairing for vehicle-to-vehicle communication in vanets. Journal of Systems Architecture **103**, 101692 (2020)

[5] Aragon, N., Blazy, O., Gaborit, P., Hauteville, A., Zémor, G.: Durandal: A rank metric based signature scheme. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. pp. 728–758. Springer International Publishing, Cham (2019)

[6] Bao, F., Deng, R.H., Zhu, H.: Variations of diffie-hellman problem. In: International conference on information and communications security. pp. 301–312. Springer (2003)

[7] Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: LESS-FM: Fine-tuning Signatures from the Code Equivalence Problem. In: International Conference on Post-Quantum Cryptography. pp. 23–43. Springer (2021)

[8] Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: Less-fm: Fine-tuning signatures from the code equivalence problem. Cryptology ePrint Archive, Report 2021/396 (2021), https://eprint.iacr.org/2021/396

[9] Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. Journal of Cryptology **22**(1), 1–61 (2009)

[10] Beullens, W.: Not enough LESS: An improved algorithm for solving code equivalence problems over $\mathbb{F}_q$. In: International Conference on Selected Areas in Cryptography. pp. 387–403. Springer (2020)

[11] Beullens, W.: Sigma protocols for mq, pkp and sis, and fishy signature schemes. EUROCRYPT (3) **12107**, 183–211 (2020)

[12] Beullens, W., Katsumata, S., Pintore, F.: Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 464–492. Springer (2020)

[13] Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: LESS is more: Code-based signatures without syndromes. In: Nitaj, A., Youssef, A. (eds.) AFRICACRYPT. pp. 45–65. Springer (2020)

[14] Branco, P., Mateus, P.: A code-based linkable ring signature scheme. In: International Conference on Provable Security. pp. 203–219. Springer (2018)

[15] Cayrel, P.L., Gaborit, P., Girault, M.: Identity-based identification and signature schemes using correcting codes. In: WCC. vol. 2007, pp. 69–78 (2007)

[16] Cayrel, P.L., Véron, P., El Yousfi Alaoui, S.M.: A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In: Selected Areas in Cryptography. pp. 171–186. Springer Berlin Heidelberg (2011)

[17] Chen, S., Zeng, P., Choo, K.K.R., Wang, Q.: An efficient ring signature scheme based on syndrome decoding problem. In: 2016 12th International Conference on Computational Intelligence and Security (CIS). pp. 228–232. IEEE (2016)

[18] Courtois, N.T., Finiasz, M., Sendrier, N.: How to achieve a McEliece-based digital signature scheme. Advances in Cryptology - ASIACRYPT 2001, Lecture Notes in

Computer Science **2248**, 157–174 (2001)
[19] Debris-Alazard, T., Sendrier, N., Tillich, J.P.: Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In: ASIACRYPT. pp. 21–51. Springer (2019)
[20] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium (2017), https://pq-crystals.org/dilithium/
[21] Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 88–105. Springer (2009)
[22] Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON: Fast-Fourier Lattice-based Compact Signatures over NTRU (2017), https://falcon-sign.info/
[23] Joye, M., Neven, G.: Identity-based signatures. Identity-based cryptography **2**(31), 75 (2009)
[24] Leon, J.: Computing automorphism groups of error-correcting codes. IEEE Transactions on Information Theory **28**(3), 496–511 (May 1982)
[25] McEliece, R.: A Public-Key Cryptosystem Based On Algebraic Coding Theory. Deep Space Network Progress Report **44**, 114–116 (Jan 1978)
[26] Merkle, R.C.: A digital signature based on a conventional encryption function. In: Conference on the theory and application of cryptographic techniques. pp. 369–378. Springer (1987)
[27] Ren, Y., Zhao, Q., Guan, H., Lin, Z.: On design of single-layer and multilayer code-based linkable ring signatures. IEEE Access **8**, 17854–17862 (2020)
[28] Saeed, M.A.: PhD thesis (2017)
[29] Sendrier, N.: The support splitting algorithm. Information Theory, IEEE Transactions on pp. 1193 – 1203 (08 2000)
[30] Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing **26**(5), 1484–1509 (1997)
[31] Song, B., Zhao, Y.: Provably secure identity-based identification and signature schemes from code assumptions. Plos one **12**(8), e0182894 (2017)
[32] Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) Advances in Cryptology — CRYPTO' 93. pp. 13–21. Springer Berlin Heidelberg (1994)
[33] Zheng, D., Li, X., Chen, K.: Code-based ring signature scheme. Int. J. Netw. Secur. **5**(2), 154–157 (2007)

Appendix A. Zero-Knowledge Identification Schemes

We give a formal definition below, to facilitate the description of the ZKID underlying our proposed RS scheme.

Definition A.1. A *Zero-Knowledge Identification (ZKID) scheme* is a protocol between 2 parties: a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. The protocol is composed of the following procedures:

I. Setup: on input the public data (including system parameters), output a secret key sk and a public key pk.

II. Commit: on input a public key pk, $\mathcal{P}$ computes a commitment c and sends it to $\mathcal{V}$.

III. Challenge: $\mathcal{V}$ selects uniformly at random a challenge b and sends it to $\mathcal{P}$.

IV. Response: on input a challenge b and a private key sk, $\mathcal{P}$ computes a response rsp and sends it to $\mathcal{V}$.

V. Verify: on input a public key pk, a commitment c and a response rsp, $\mathcal{V}$ outputs either 1 (accept) if the response is valid, or 0 (reject) otherwise.

A ZKID scheme should verify the usual, well-known security properties. *Correctness* dictates that an honest prover is always accepted. *Special soundness* defines the success probability of a cheating prover; typically, this is done by constructing an extractor algorithm that is capable to recover a "witness" (e.g. a secret key). Finally, *Zero-Knowledge* guarantees that there is no information leaked by honest executions.