Cutting the GRASS: Threshold GRoup Action Signature Schemes

Michele Battagliola[1], Giacomo Borin[1], Alessio Meneghetti[1], and Edoardo Persichetti[2]

[1] Università di Trento, Italy
[2] Florida Atlantic University, Boca Raton, USA and Sapienza University, Rome, Italy
michele.battagliola@unitn.it, giacomo.borin@studenti.unitn.it, alessio.meneghetti@unitn.it, epersichetti@fau.edu

Abstract. Group actions are a fundamental mathematical tools, with a long history of use in cryptography. Indeed, the action of finite groups at the basis of the discrete logarithm problem is behind a very large portion of modern cryptographic systems. With the advent of post-quantum cryptography, however, the method for building protocols shifted towards a different paradigm, centered on the difficulty of discerning "noisy" objects, as is the case for lattices, codes, and multivariate systems. This method yields promising results for "core" primitives such as encryption or signature, but can be less than ideal in the case when more advanced functionalities are required. In this work, we show that isomorphism problems which stem from cryptographic group actions, can be viable building blocks for threshold signature schemes. In particular, we construct a full N-out-of-N threshold signature scheme, and discuss the efficiency issues arising from extending it to the generic T-out-of-N case. To give a practical outlook on our constructions, we instantiate them with the LESS and MEDS frameworks, which are two flavors of code-based cryptographic group actions. Finally, we highlight some ideas that would allow for a more efficient and compact (T, N) threshold variant of LESS, whose security relies on new hardness assumptions.

1 Introduction

With the threat of quantum computers looming ever closer, the community has stirred to produce alternative cryptographic solutions, that will be resistant to attackers equipped with such technology. Indeed, considering the timeline expected to design, standardize, implement and deliver such solutions, initiatives such as NIST's [48] are definitely timely. To be sure, NIST's standardization effort can be considered a first step, with more to follow. For instance, while the first standards are about to be drafted, covering key encapsulation and signatures, the situation with the latter is considered not fully satisfactory, to the point that NIST launched an "on-ramp" process to standardize new signature designs [49]. Furthermore, there is a scarcity of threshold-friendly schemes among the current solutions, which is prompting more research in this area, and will lead to its own standardization process [20].

Code-based cryptography, which makes use of problems and techniques coming from coding theory, is the second largest area within the post-quantum realm, capable of offering interesting solutions, particularly in the context of key establishment. Indeed, all three candidates in NIST's 4th round of standardization are code-based [4,5,3], with two of them expected to be added to the current list of standards (which, for KEMs, includes only Kyber [52]). On the other hand, this area has historically struggled to produce efficient signature schemes: as a litmus test, none of those presented to NIST in 2017 made it past the first round. This steered the community towards experimenting with different paradigms, such as, for instance MPC-in-the-head [40,32,31].

The work of LESS [16], which started in 2020 and continued with various follow-ups [7,6,9], uses a different approach, stepping away from the traditional decoding problem, and focusing instead on the difficulty of finding an *isometry* between two linear codes. In fact, the security of LESS relies solely on the well-known *code equivalence problem*. This idea was recently extended [23] to the class of *matrix codes*, which are measured in the rank metric, and yields the parallel notion of *matrix equivalence*. Interestingly, the action of isometries on the respective types of codes can be formulated as a (non-commutative) group action, which gives a new perspective on the field, and opens the way to other constructions beyond plain signatures. Indeed, the use of group actions in cryptography dates back all the way to Diffie and Hellman, and has found new vigor as a post-quantum method, thanks to the recent developments on isogenies [22,15].

1.1 Related Work

A (T, N)-threshold digital signature scheme is a protocol designed to distribute the right to sign messages to any subset of at least T out of N key owners, with the restriction that none of the N players can repudiate a valid signature. A key point in most threshold digital signature schemes is compatibility with existing schemes: even though the key generation and signing algorithms are multi-party protocols (MPC), in fact, the verification algorithm is identical to that of an existing signature scheme, usually referred to as the "centralized" scheme.

In 1996, a first $(T + 1, 2T + 1)$-threshold digital signature scheme was proposed [37]. A few years later, the same authors discuss the security of distributed key generation for the case of schemes based on the Discrete Logarithm Problem [38,39]. Since 2001, several authors started working first on two-party variants of digital signatures [46,47] and then on ECDSA [28,43]. The first general (T, N)-threshold scheme was proposed in 2016 [36], improved first in 2017 [17], and then again in 2018 [34]. In 2019, the work of [28] has been generalized by the same authors to the multi-party case [29]. While the signing algorithm requires the participation of at least T players to take part in a multi-party protocol, the key generation algorithm requires the involvement of a Trusted Authority or the active participation of all N players. This requirement has been relaxed in a recent $(2, 3)$ threshold ECDSA version [13], where the key generation algorithm involves only 2 out the 3 parties.

As noted in [19], a challenging task in designing a threshold version of the EdDSA signature scheme is the distribution among the parties of the deterministic nonce generation, a task that can be carried out either with MPC techniques or with zero-knowledge proofs (ZKP). Following the latter approach, the work presented in [13] has successively been extended to a $(2, 3)$-threshold EdDSA instantiation [12]. In [18], the authors propose instead an MPC-based threshold scheme for HashEdDSA. In the latter, T is bounded to be less than $\frac{N}{2} + 1$. Finally, in 2022, a variant of [12] suitable for Schnorr signatures has been proposed [10] and then generalized to a ZKP-based (T, N)-threshold Schnorr digital signature scheme whose key generation algorithm does not involve any trusted party [11].

Recently, driven by both the NIST call for Post-Quantum Standardization [48] and the call for Multi-Party Threshold Schemes [20], many researchers have started to wonder whether it could be possible to design post-quantum versions of threshold digital signature schemes. Since most of the existing literature for threshold schemes focuses on trapdoors that rely on the difficulty of the Discrete Logarithm Problem, new methods have to be investigated, likely starting with tools already utilized to design plain signatures, such as lattices, codes, multivariate equations etc. In [24], the (round 2) proposals of the standardization process were analyzed in order to determine ways to define threshold variants, eventually identifying multivariate schemes as the most suitable starting point, with UOV-based schemes being the most promising. Even though, from a theoretical point of view, it appears to be indeed possible to obtain a threshold version of UOV by exploiting LSSS-based MPC protocols, this approach remains, at the present time, only theoretical. Notably, threshold signature schemes for cryptographic cyclic group actions have been already discussed in 2020 and applied to isogeny-based schemes [27].

1.2 Our Contribution

In this work, we investigate constructions for post-quantum threshold signature schemes, using cryptographic group actions as the main building block. However, our goal is to take a step back, and keep requirements to a minimum, without needing additional properties such as, for instance, commutativity. This will allow our frameworks to be instantiated with a wider variety of candidates, such as the aforementioned code-based signature schemes.

A full threshold scheme. As a first contribution, we present a construction for a "full" (N, N)-threshold signature scheme. We then prove its security via a reduction to the original centralized signature; this is a generic signature scheme that is simply an abstraction, but has appeared in literature when instantiated in various works, such as LESS signature scheme [7], and MEDS signature scheme [23]. The core idea is to split both the secret key and the ephemeral map as a product of N group elements, i.e. as $g = g_1 \cdots g_N$.

Thanks to this shared knowledge the users are able to prove the knowledge of secret the key. The details of the construction, as well as the security proof, are given in Section 3.

Construction using subsets. Our second contribution is the (T, N) version of scheme, with $T \leq N$. Since we cannot assume any properties on the groups (except the security of the group actions), our construction is quite inefficient in terms of memory required. This is because we need to distribute multiple keys to each user. We illustrate this by presenting some performance figures in the selected scenario, namely, the code-based setting. Nevertheless, our construction remains practical for certain use cases, especially for low values of T and N, or whenever T and N are very close.

Tailoring the solution. Finally, in light of the previous considerations, we present a dedicated framework for the case of LESS signatures. In this setting, in fact, we are able to build an optimized secret sharing scheme, that allows for a better memory management and makes the size of the secret key independent from the values of T and N. This approach requires a slight change in perspective in the formulation of the linear equivalence problem. Indeed, we exploit the fact that the group element is composed of two pieces, a monomial matrix and a non-singular change-of-basis map, which are both necessary to guarantee security; we then focus our attention on the change-of-basis map, and show that it is possible to produce an efficient secret sharing scheme by selecting it among the set of invertible maps with a particular structure. We sketch a security analysis for this new proposal, although a full discussion is intended to be the focus of future studies.

1.3 Outline

We begin in Section 2, where we provide all the necessary preliminary definitions and notions used in the paper. Then, in Section 3 we present the full threshold version of the signature, together with a security proof. In Section 4 we show how to construct a possible solution to obtain a general (T, N)-version, adapting the previous framework as well as its proof. To provide a practical outlook, we present a concrete instantiation of both protocols, in Section 5, utilizing the code equivalence group actions at the basis of the LESS and MEDS signature schemes. Lastly, in Section 6 we present the tailored version of our protocol to the LESS setting, introducing a new security assumption to back the rationale of the construction. We conclude in Section 7.

2 Preliminaries

We begin by laying down our notation. Throughout the paper we will use denote with capital letters object such as sets and groups, and with lowercase letters their elements. We will use instead boldface letters to denote vectors and matrices.

Since the chosen setting for our particular instantiation concerns linear codes, we briefly report here the related notation and notions. First of all, we indicate by $\mathbb{F}_q$ the finite field of cardinality q, and by $\mathbb{F}_q^{k\times n}$ the set matrices of dimension $k \times n$, with elements in $\mathbb{F}_q$; when $k = 1$, we write simply $\mathbb{F}_q^n$, which denotes the corresponding vector space over $\mathbb{F}_q$. A linear code $\mathscr{C}$ is a vector subspace $\mathscr{C} \subseteq \mathbb{F}_q^n$ of dimension k, and it is usually referred to as an $[n, k]$ linear code. It follows that a basis for $\mathscr{C}$ is given by a set of k linearly independent vectors in $\mathbb{F}_q^n$. When these vectors are put as rows of a matrix $\boldsymbol{G}$, this is known as a *generator matrix* for the code, as it can generate each vector of $\mathscr{C}$ (i.e. a *codeword*) as a linear combination of its rows. Note that such a generator is not unique, and any invertible $k \times k$ matrix $\boldsymbol{S}$ yields another generator via a change of basis; however, it is always possible to utilize a "standard" form simply performing a Gaussian elimination on the left-hand side. This is usually called *systematic* if the result is the identity matrix (i.e. if the leftmost $k \times k$ block is invertible); we denote this by SF.

Linear codes are traditionally measured with the Hamming metric, which associates a *weight* to each codeword by simply counting the number of its non-zero positions. It follows, then, that an *isometry* (i.e. a map preserving the weight) is given by any $n \times n$ permutation matrix $\boldsymbol{P}$ acting on each word, or indeed, on the columns of $\boldsymbol{G}$ (since every codeword can be generated as a linear combination of the rows of $\boldsymbol{G}$). Moreover, it is possible to generalize this notion by adding some non-zero scaling factors from $\mathbb{F}_q$ to each column. Such a matrix is commonly known as a *monomial* matrix, and we denote it by $\boldsymbol{Q}$; it can be seen as a product $\boldsymbol{D} \cdot \boldsymbol{P}$ between a permutation matrix and a diagonal matrix with non-zero components.

The notion of linear codes can be generalized to the case where each codeword is a matrix, instead of a vector; more precisely, $m \times n$ matrices over $\mathbb{F}_q$. We talk then about $[m \times n, k]$ *matrix code*, which can be seen as a k-dimensional subspace $\mathscr{C}$ of $\mathbb{F}_q^{m\times n}$. These objects are usually measured with a different metric, known as *rank* metric, where the weight of each codeword corresponds to its rank as a matrix. In this case, then, isometries are maps which preserve the rank of a matrix, and are thus identified by two non-singular matrices $\boldsymbol{A} \in \mathrm{GL}_m$ and $\boldsymbol{B} \in \mathrm{GL}_n$ acting respectively on the left and on the right of each codeword, by multiplication.

In both of the metrics defined above, we are able to formulate a notion of *equivalence* in the same way, by saying that two codes are equivalent if they are connected by an isometry. In other words, with a slight abuse of notation, we say that two linear codes $\mathscr{C}$ and $\mathscr{C}'$ are *linearly equivalent* if $\mathscr{C}' = \mathscr{C}\boldsymbol{Q}$, and two matrix codes $\mathscr{C}$ and $\mathscr{C}'$ are *matrix equivalent* if $\mathscr{C}' = \boldsymbol{A}\mathscr{C}\boldsymbol{B}$. Note that the notion of *permutation equivalence* is just a special case of linear equivalence (with the diagonal matrix $\boldsymbol{D}$ being the identity matrix), yet is often treated separately for a variety of reasons of both historical and practical nature (for instance, certain solvers behave quite differently).

2.1 Cryptographic Group Actions

A *group action* is a well-known object in mathematics. It can be described as a function, as shown below, where X is a set and G a group.

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

A group action's only requirement is to be *compatible* with the group; using multiplicative notation for G, and denoting with e its identity element, this means that for all $x \in X$ we have $e \star x = x$ and that moreover for all $g, h \in G$, it holds that $h \star (g \star x) = (h \cdot g) \star x$. A group action is also said to be:

- *Transitive*, if for every $x, y \in X$, there exists $g \in G$ such that $y = g \star x$;
- *Faithful*, if there does not exist a $g \in G$ such that $x = g \star x$ for all $x \in X$, other than the identity;
- *Free*, if an element $g \in G$ is equal to identity whenever there exists an $x \in X$ such that $x = g \star x$;
- *Regular*, if it is free and transitive.

The adjective *cryptographic* is added to indicate that the group action in question has additional properties that are relevant to cryptography. For instance, a cryptographic group action should be *one-way*, i.e. given randomly chosen $x, y \in X$, it should be hard to find $g \in G$ such that $g \star x = y$ (if such a g exists). Indeed, the problem of finding this element is known as the *vectorization* problem, or sometimes *Group Action Inverse Problem (GAIP)*, as defined below.

Problem 1 (GAIP)**.** Given x and y in X, find, if any, an element $g \in G$ such that $y = g \star x$.

A related problem asks to compute the action of the product of two group elements, given the result of the individual actions on a fixed element. This is known as the *parallelization* problem, and it corresponds to, essentially, the computational version of the Diffie-Hellman problem, formulated for generic group actions. A definition is given next.

Problem 2 (cGADH)**.** Given x, $g \star x$ and $h \star x$, for $g, h \in G$, compute $(g \cdot h) \star x$.

In fact, the analogy to the case of discrete logarithms is easily drawn, once one realizes that this is simply the group action given by the exponentiation map on finite cyclic groups. Then GAIP corresponds to DLP and cGADH to the CDH problem.

Finally, other useful properties for group actions include those that make it *effective*, such as, for instance, the existence of efficient (probabilistic polynomial-time) algorithms for membership testing, sampling, computation (of both the group operation and $\star$) etc.

2.2 Signatures from Generic Group Actions

We summarize here briefly how to design a signature scheme from generic group actions. To begin, we formulate the Sigma protocol described in Figure 1.

Public Data : Group G acting on X via $\star$, element $x \in X$ and hash function $\mathtt{H}$.	
Private Key : Group element g with $g_i \in G$.	
Public Key : $y = g \star x$.	
PROVER	**VERIFIER**
Get $\tilde{g} \xleftarrow{\$} G$, set $\tilde{x} \leftarrow \tilde{g} \star x$, send $h = \mathtt{H}(\tilde{x})$ $\xrightarrow{h}$	
$\xleftarrow{b}$	$b \xleftarrow{\$} \{0,1\}$.
If $b = 0$ then $u \leftarrow \tilde{g}$.	Accept if $\mathtt{H}(u \star x) = h$.
If $b = 1$ then $u \leftarrow \tilde{g}g^{-1}$. $\xrightarrow{u}$	Accept if $\mathtt{H}(u \star y) = h$.

Fig. 1. Identification protocol for the knowledge of the private key.

The protocol above intuitively provides a soundness error of $1/2$; it is in fact trivial to prove that an adversary who could solve answer both challenges simultaneuosly, would be able to recover a witness, i.e. a solution to GAIP. It is then necessary to amplify soundness, in order to reach the desired authentication level. This is accomplished, in the simplest way, by parallel repetition; in practice, several optimizations can be applied, as we will see in Section 5, without impacting security. At this point, a signature scheme can be obtained using the Fiat-Shamir transformation [33], which guarantees EUF-CMA security in the (Quantum) Random Oracle Model. The next result is intentionally a little vague, since it is well-known in literature, and we do not want to overly expand this section. Proofs tailored to the specific instantiations can be found, for example, in [26,8]. For further discussions on Fiat-Shamir, and its security in the ROM and QROM, we point instead the reader to [33,1,30,45].

Proposition 1. *Let* I *be the identification protocol described above, and* $\mathbf{S}$ *be the signature scheme obtained by iterating* I *and then applying Fiat-Shamir. Then* $\mathbf{S}$ *is existentially unforgeable against chosen-message attacks, based on the hardness of GAIP.*

Note that the protocol does not require any specific property from the group action in use, except those connected to efficient sampling and computation. Indeed, even though the action could in principle be non-transitive, as is the case for code-based group actions, the construction makes it so that we operate on a single orbit (i.e. it is transitive by design in this specific use case). It is however advisable to utilize a free group action, since this could have an impact on the security of GAIP.

2.3 Code-based Group Actions

We now present the group action associated to code equivalence, according to the definitions given in the previous sections. 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. We then set $G = M_n$, by which we denote the group of monomial matrices. Note that this group is isomorphic to $(\mathbb{F}_q^*)^n \rtimes S_n$ if we decompose each monomial matrix $\boldsymbol{Q} \in M_n$ into a product $\boldsymbol{D} \cdot \boldsymbol{P}$. The group operation can be then seen simply as multiplication, and the group action is given by

$$\begin{aligned} \star : G \times X &\to X \\ (\boldsymbol{G}, \boldsymbol{Q}) &\to \mathrm{SF}(\boldsymbol{GQ}) \end{aligned}$$

It is easy to see that the action is well-formed, with the identity element being $\boldsymbol{I}_n$, and compatible with respect to (right) multiplication.

Remark 1. The definition above considers a standardized choice of representative by utilizing the systematic form SF. This simplifies the definition and makes sure to avoid cases where multiple generators for the same code could be chosen. Indeed, since the systematic form uniquely identifies linear codes, this allows us to see our group action as effectively acting on linear codes, rather than on their representatives (generator matrices).

The case of matrix code equivalence can be framed analogously. In this case, the set X is formed by the k-dimensional matrix codes of size $m \times n$ over some base field $\mathbb{F}_q$; similarly to linear codes, matrix codes can be represented via generator matrices $\boldsymbol{G} \in \mathbb{F}_q^{k\times mn}$. Then, the action of the group $G = \mathrm{GL}_m \times \mathrm{GL}_n$ on this set can be described compactly as follows:

$$\begin{aligned} \star : G \times X &\to X \\ ((\boldsymbol{A}, \boldsymbol{B}), \boldsymbol{G}) &\to \mathrm{SF}(\boldsymbol{G}(\boldsymbol{A}^\top \otimes \boldsymbol{B})) \end{aligned}$$

Note that this is equivalent to applying the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ to each codeword $\boldsymbol{C}$ in the matrix code as $\boldsymbol{ACB}$; indeed this is often the most convenient notation.

Note that, in both cases, the action is not commutative and in general neither transitive nor free. It is however possible to restrict the set X to a single well-chosen orbit to make the group action both transitive and free. In fact, picking any orbit generated from some starting code ensures transitivity, and the group action is free if the chosen code has a trivial automorphism group, where trivial means up to scalars in $\mathbb{F}_q$. The non-commutativity is both a positive and negative feature: although it limits the cryptographical design possibilities, e.g. key exchange becomes hard, it prevents quantum attacks to which commutative cryptographic group actions are vulnerable, such as Kuperberg's algorithm for the dihedral Hidden Subgroup Problem [42].

The vectorization problems for the code-based group actions are well-known problems in coding theory. We report them below.

Problem 3 (Linear Equivalence (LEP))**.** Given two k-dimensional linear codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$, find, if any, $\boldsymbol{Q} \in M_n$ such that $\mathscr{C}' = \mathscr{C}\boldsymbol{Q}$.

We have not defined explicitly here the *Permutation Equivalence Problem (PEP)*, since we will not use it directly; this can be seen as just a special case of LEP, where the monomial matrix $\boldsymbol{Q}$ is simply a permutation.

Problem 4 (Matrix Code Equivalence (MCE))**.** Given two k-dimensional matrix codes $\mathscr{C}, \mathscr{C}'$, find, if any, $\boldsymbol{A} \in \mathrm{GL}_m, \boldsymbol{B} \in \mathrm{GL}_n$ such that $\mathscr{C}' = \boldsymbol{A}\mathscr{C}\boldsymbol{B}$.

Note that both of the above problems are traditionally formulated as decisional problems, but in cryptographic applications we are most often interested in their computational version. Rather extensive treatments of their hardness is given, for instance, in [9,23].

2.4 Threshold Signatures

We briefly summarize here the relevant notions for threshold signature schemes. In a nutshell, a (T, N)-threshold signature is a multi-party protocol that allows any T parties out of a total of N to compute a signature that may be verified against a common public key. Usually, threshold signature protocols involve a key-generation phase where the N parties collaborate to construct the key pair $(\mathtt{sk}, \mathtt{pk})$ as well as shares of the private key $\mathtt{sk}_i$. These are distributed to every party, so that the share $\mathtt{sk}_i$ is known only by party i, and the private key $\mathtt{sk}$ is never actually computed explicitly.

After this initial phase, any set of T parties who agree on a common message $\mathtt{m}$ is able to perform a signature phase in order to sign it. The resulting signature is verifiable against the public key $\mathtt{pk}$. Often, threshold signature protocols are obtained by adapting "plain" signature schemes, which are then referred to as "centralized", for obvious reasons. In this case, a common requested property is that signatures produced by the threshold protocol are indistinguishable from signatures produced by the centralized one.

The main security property for threshold signature schemes, which is the one that we are able to prove for our signature protocol, is *Existential Unforgeability under Chosen Message Attacks (EUF-CMA)*, and is related to the analogue notion for centralized schemes. For completeness, we present it below.

Definition 5. *We say that a (T, N)-threshold signature scheme is existentially unforgeable under chosen message attacks if no malicious adversary who corrupts at most $T-1$ players can produce with non-negligible probability the signature on a new message $\mathtt{m}$, given the view of $\mathtt{Threshold-Sign}$ on input a polynomially large number of messages $\mathtt{m}_1, ..., \mathtt{m}_{q_s}$ (which the adversary chooses adaptively).*

3 The Full Scheme

In this section we give all the details of our construction for a full threshold scheme. As mentioned in Section 1, we start by designing a zero-knowledge identification protocol, similar to that of Figure 1, but specific to the threshold case. The signature scheme is obtained by applying Fiat-Shamir to it, as usual. The resulting algorithms for (decentralized) key generation, signature, and verification are presented next. We begin with the former.

Algorithm 1 `KeyGen`

Require: $x \in X$.
Ensure: Public key $y = g \star x$, each participant holds g_i such that $\prod g_i = g$.
1: Each participant P_i chooses $g_i \in G$ and publishes $x'_i = g_i \star x$.
2: Set $x_0 = x$.
3: **for** $i = 1$ to N **do**
4: P_i computes $x_i = x_{i-1} \star g_i$
5: P_i produces a ZKP proving the honest use of g_i and publishes it
6: P_i sends x_i to P_{i+1}
7: **end for**
8: **return** $y = x_N$. The private key of P_i is g_i.

Note that, to define a secure distributed key generation (Algorithm 1) it was necessary to add a Zero-Knowledge Proof in line 5, to prove that the element g_i, generated at the start of the protocol, was actually used.

As far as signing is concerned, it is possible to observe that, when sending the set and group elements (lines 6 and 20 of Algorithm 2), these can be grouped all together in a single message. Finally, note that the output of the signature algorithm is a valid signature for the centralized scheme; it follows that the verification procedure is equal to the one in the original scheme. Nevertheless, we reported it here for completeness.

3.1 Security Proof

We now prove the security of the full threshold protocol, according to Definition 5. In particular we have the following theorem:

Theorem 1. *Under the hardness of GAIP, the full threshold signature scheme is Existentially Unforgeable under adaptive Chosen-Message Attacks (EUF-CMA) in the Random Oracle Model.*

The proof follows a standard game-based argument. In particular we show that if an adversary $\mathcal{A}$ is able to forge the signature scheme controlling $N - 1$ players with non negligible probability ϵ, then it is possible to build a simulator $\mathcal{S}$ that, interacting with $\mathcal{A}$, is able to forge the original one user signature with non negligible probability. We now show how to simulate the key generation.

Algorithm 2 $\texttt{Threshold} - \texttt{Sign}$

Require: $x \in X$, a security parameter λ, a hash function $\texttt{H}$, a public key $(x, y = g \star x)$. The party P_i knows the (multiplicative) share g_i of $g = g_1 \cdots g_N$.
Ensure: A valid signature for the message $\texttt{m}$ under the public key (x, y).

Set $x_0^j = x$ for all $j = 1$ to λ ▷ Shared commitment generation phase
for $i = 1$ to N **do**
  If $i > 1$ P_i receives x_{i-1}^j from P_{i-1} for all $j = 1$ to λ
  for $j = 1$ to λ **do**
    P_i chooses $\tilde{g}_i^j \in G$ and computes $x_i^j = \tilde{g}_i^j \star x_{i-1}^j$
    If $i < N$ P_i sends x_i^j to P_{i+1}
  end for
end for
Set $x^j = x_N^j$ for all $j = 1$ to λ
Compute $\texttt{ch} = \texttt{H}(x^1||...||x^\lambda||\texttt{m})$ ▷ Non-iterative challenges evaluation
Set $u_0^j = e$ for all $j = 1$ to λ ▷ Shared response generation phase
for $i = 1$ to N **do**
  If $i > 1$ P_i receives u_{i-1}^j from P_{i-1} for all $j = 1$ to λ
  for $j = 1$ to λ **do**
    if $\texttt{ch}_j = 0$ **then**
      P_i computes $u_i^j = \tilde{g}_i^j u_{i-1}^j$ ▷ $\tilde{g}_i^j$ can also be disclosed via the seed
    else
      P_i computes $u_i^j = \tilde{g}_i^j u_{i-1}^j g_i^{-1}$
    end if
    If $i < N$ P_i sends u_i^j to $Pi + 1$
  end for
end for
$\texttt{resp}_j = u_N^j$ for all $j = 1$ to λ
$\texttt{sig} = \texttt{ch}||\texttt{resp}_1||...||\texttt{resp}_\lambda$

Simulation of the Key Generation Initially the simulator $\mathcal{S}$ receives a challenge for the signature algorithm, i.e. a public key (x, y). The goal of this step is to simulate an honest execution of the Key-Generation with the aim of having public key y. We need to distinguish two cases, based on whether $\mathcal{A}$ controls P_N or not. First we show a simulation for $\mathcal{A}$ not controlling P_N:

- The simulator $\mathcal{S}$, playing the role of P_N starts the adversary $\mathcal{A}$.
- $\mathcal{S}$ choose a random g_N and commits to it by publishing $g_N \star x$.
- $\mathcal{A}$ sends $g_{\mathcal{A}} \star x$ to $\mathcal{S}$, where $g_{\mathcal{A}}$ is $g_1 g_2 ... g_{N-1}$ all the secret key controlled by $\mathcal{A}$
- $\mathcal{S}$ extracts all the g_i, saves them and sends y to $\mathcal{A}$.
- $\mathcal{S}$ simulates the ZKP of Appendix A with public data $g \star x$ and y.
- The public key is y.

Now we need to show a simulation when P_N is controlled by $\mathcal{A}$. Let $P_{\mathcal{S}}$ be the single player controlled by $\mathcal{S}$. With an abuse of notation we use $\mathcal{S}$ to also denote the position of $P_{\mathcal{S}}$ in the protocol (i.e. $P_{\mathcal{S}+1}$ is the party after $P_{\mathcal{S}}$).

Algorithm 3 `Verify`

Require: $x \in X$, the security parameter λ, a hash function $\mathtt{H}$, a public key $(x, y = g \star x)$.
Ensure: Accept if the signature for the message $\mathtt{m}$ is valid under the public key (x, y).

Parse $\mathtt{ch}, \mathtt{resp}_1, ..., \mathtt{resp}_\lambda$ from $\mathtt{sig}$
for $j = 1$ to λ **do**
  if $\mathtt{ch}_j = 0$ **then**
    set $\hat{x}^j = \mathtt{resp}_j \star x$
  else
    set $\hat{x}^j = \mathtt{resp}_j \star y$
  end if
end for
Accept if $\mathtt{ch} = \mathtt{H}(\hat{x}^1 || ... || \hat{x}^\lambda || \mathtt{m})$

- $\mathcal{S}$ starts $\mathcal{A}$ and picks randomly $g_{\mathcal{S}}$.
- $\mathcal{S}$ receive from $\mathcal{A}$ the commitments $g_i \star x$ for all P_i controlled by $\mathcal{A}$.
- $\mathcal{S}$ follows the protocol normally, extracting all the g_i from $\mathcal{A}$.
- $\mathcal{S}$ rewinds $\mathcal{A}$, changes the message it sends to $g_{\mathcal{S}+1}^{-1} ... g_N^{-1} \star y$ and sends it to $P_{\mathcal{S}+1}$ (which is controlled by $\mathcal{A}$).

Lemma 1. *The simulation terminates in expected polynomial time, is indistinguishable from a real execution and outputs y.*

Proof. The proof for the simulation in case the adversary $\mathcal{A}$ does not control P_N is immediate: the simulator $\mathcal{S}$ does not need to rewind $\mathcal{A}$ and can simply output y. The only difference between the simulation and a real execution is that $\mathcal{S}$ does not know its private key; however, it is still able to simulate the final ZKP as shown in Appendix A.

In the case of $\mathcal{A}$ controlling P_N, $\mathcal{S}$ rewinds the adversary after seeing the output of the first iteration and, thanks to the ZKP in Appendix A, it is also able to extract the all the secret keys g_i from the adversary. In this way, $\mathcal{S}$ is able to fix its own private values in order to force the desired output y. Due to the initial commitment, $\mathcal{A}$ chooses always the same group elements $g_{\mathcal{S}}, ..., g_N$ in both execution except with negligible probability and thus the simulation ends in expected polynomial time with the desired output y.

The only difference is that $\mathcal{S}$ does not know its private key, however it is able to simulate the ZKP as shown in Appendix A. □

Simulation of the Signature After the key generation $\mathcal{A}$ is allowed to choose messages $\mathtt{m}_1, ..., \mathtt{m}_{q_s}$ adaptively and ask $\mathcal{S}$ to perform the $\mathtt{Threshold} - \mathtt{Sign}$ algorithm on them. Since $\mathcal{S}$ does not know its private key, it needs to simulate the interaction.

At this point, we do not need to distinguish different cases based on the which is the honest party. Indeed, the simulator can needs reprogram the random

oracle. This technique, which is at the basis for the proof of the Fiat-Shamir transformation [2], allows the simulator to choose the challenge for the message ahead of time, allowing $\mathcal{S}$ to simulate the computation of `resp` in the same way it simulates the ZKP in Figure 1. Let $P_{\mathcal{S}}$ be the party controlled by $\mathcal{S}$. The simulation works as follows:

- $\mathcal{A}$ chooses a message `m` to sign and sends it to the simulator $\mathcal{S}$.
- $\mathcal{S}$ generates a random string $\mathtt{ch}'$.
- Depending on each bit $\mathtt{ch}'_j$ of `ch`, $\mathcal{S}$ computes x^j in this way:
  - if $\mathtt{ch}'_j = 0$ it follows the protocol normally;
  - if $\mathtt{ch}'_j = 1$ picks a random $\tilde{g}^j_{\mathcal{S}}$ and sends $\prod_{i=1}^{\mathcal{S}-1} g_i \tilde{g}^j_{\mathcal{S}}$, where the g_i are the values it received from the adversary during the key generation phase.
- The adversary computes $x^1, ..., x^\lambda$ and, when computing `ch`, the simulator reprograms the random oracle so that $H(x^1||...||x^\lambda||\mathtt{m}) = \mathtt{ch}'$.
- $\mathcal{S}$ performs the computation of `resp` normally, sending $\tilde{g}^j_{\mathcal{S}}$ when $\mathtt{ch}'_j = 1$.

Lemma 2. *The simulation above terminates in expected polynomial time, it is indistinguishable from a real execution of the protocol and either outputs a valid signature.*

Proof. The only difference is that between the simulated execution and a real execution is that $\mathcal{S}$ does not know it public key and thus it fixes the challenge in advance and to be able to compute `resp`. Note that, since the number of queries q_s is polynomially bounded then the probability of collision is negligible, thus the reprogramming step is indistinguishable from a real execution.

Thanks to the zero knowledge properties of the identification protocol (Figure 1), the above simulation is indistinguishable from a real execution, except if $\mathcal{A}$ decides to send to $\mathcal{S}$ a wrong value u_i to $\mathcal{S}$. To avoid this problem, $\mathcal{S}$ chooses a random index $h \in [0, ..., q_s]$, where q_s is the maximum number of signature query the adversary is allowed to perform.

- if $h = 0$ we assume that all executions are correct, i.e. $\mathcal{A}$ always sends the correct u_i. In this case we can always simulates as described.
- Otherwise we assume that the first $h-1$ executions are correct, but at the h one $\mathcal{A}$ sends a wrong u_i. In this case $\mathcal{S}$ does not use the previous simulation and instead outputs a random `resp` and aborts.

With non-negligible probability $\frac{1}{q_s+1}$ the simulator is able to guess the correct index h.

□

As a corollary we obtain the proof of Theorem 1.

Proof. The simulator $\mathcal{S}$ described above produces an indistinguishable view for the adversary $\mathcal{A}$, and therefore, $\mathcal{A}$ will produce a forgery with the same probability as in a real execution. Then success probability of $\mathcal{S}$ is at least $\frac{\epsilon}{q_s+1}$, given by the probability ϵ of $\mathcal{A}$ producing a valid forgery and the probability $\frac{1}{q_s+1}$ of guessing the correct index h. □

3.2 Toward Identifiable Abort and Simulation-Based Security

Before we conclude the security discussion, we note that our protocol detects the presence of a malicious adversary, by noticing that the signature does not verify. As pointed out by Lindell in [44], this strategy is not simulatable against a malicious adversary. Luckily, this is not an issue, as shown in the previous proof, where we utilize the same trick as [35]. Moreover, it is important to note that, at the end of a failed execution, it is impossible to pinpoint the malicious party. To do so, we would need to add ZKPs about the correct computation of all u_i^j with respect to x_i^j, $i = 1, ..., N$ and $j = 1, ..., \lambda$. This would drastically reduce the efficiency of the signature, since it would require the inclusion of multiple ZKPs and commitment, similar to the one presented in Appendix A.

4 Construction using Subsets

In this section, we explain how to modify the construction for the full threshold scheme, to obtain a T-out-of-N scheme. We proceed in the following way: given a pair of parameters (T, N), set $M = \binom{N}{T-1}$ and consider the family $\mathcal{I}$ containing all the M subsets of $\{1, ..., N\}$ of cardinality $N - T + 1$. After labeling $\mathcal{I}$ as $\{I_1, ..., I_M\}$, we split the secret key $g \in G$ as a product $g_{I_1} \cdots g_{I_M}$, where $g_i \in G$. Then each user P_i gets the knowledge of all the shares g_I such that $I \ni i$.

Proposition 2. *Any subset $J \subset \{1, ..., N\}$ of T users can get the secret key g, whilst any adversarial group $A \subset \{1, ..., N\}$ of $T - 1$ users cannot retrieve at least one share.*

Proof. For the first part we will prove that it is possible to recover g_I for any I of cardinality $N - T + 1$. By the inclusion-exclusion principle, $|J \cap I| = |J| + |I| - |I \cup J| \geq T + N - T + 1 - N = 1$, so the intersection is non-empty and contains at least an integer j. Thus, the user P_j has the knowledge of g_I since $j \in I$ and it belongs to the set J. For the second part, note that the complement set $A^C = \{1, ..., N\} \setminus A$, obviously, does not intersect A, and so the share g_{A^C} cannot be retrieved by an adversarial group. □

Thanks to this proposition, we obtain a multiplicative threshold secret sharing scheme for $g \in G$ that can be leveraged to get a threshold signature scheme. Using this secret sharing we are able to build a generic $(T, N)-$threshold signature with the same structure of the full threshold explained in Section 3. In particular, the structure of the protocol will be the same, with the only difference that some participants are required to send multiple messages, such that all the g_{I_M} shares are used once. Since each g_{I_i} is shared among multiple parties, the users need to agree on a common "turn" function τ that, on input the set of participants J and the current round, allows to consistently choose which user will carry on the operations during the current turn. A possible example of τ is the function given by $\min J \cap I_i$, that is clearly unique, and returns a user in J who knows g_{I_i} (since $\min J \cap I_i \in I_i$).

Public Data : Group G acting on X via $\star$, element $x \in X$ and hash function $\mathtt{H}$.
Private Key : Group element $g = g_{I_1} \cdots g_{I_M}$ with $g_I \in \mathcal{G}$.
Shares for P_i : all group elements g_I such that $j \in I$.
Public Key : $y = g \star x$.

PROVERS		**VERIFIER**
Set $\tilde{x} \leftarrow x$ and for $i = 1, ..., M$ do :		
$P_{\tau(J,i)}$ get $\tilde{g}_i \xleftarrow{\$} X$ and set $\tilde{x} \leftarrow \tilde{g}_i \star \tilde{x}$	$\xrightarrow{h}$	
Set $h = \mathtt{H}(\tilde{x})$.		
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0, 1\}$.
If $b = 0$ they opens all $\tilde{g}_I$ and		
set $u \leftarrow \tilde{g}_{I_1} \cdots \tilde{g}_{I_M}$.		Accept if $\mathtt{H}(u \star x) = h$.
If $b = 1$ then $u \leftarrow e$.	$\xrightarrow{u}$	
for $i = 1, ..., M$ do :		
$P_{\tau(J,i)}$ evaluate and send $u \leftarrow \tilde{g}_{I_i} \cdot u \cdot g_{I_i}^{-1}$.		Accept if $\mathtt{H}(u \star y) = h$.

Fig. 2. Threshold identification protocol for the shared knowledge of the Private Key using the subset technique, executed by a set J of at leat T honest users

It is worth mentioning that there is the possibility of decentralizing the Key Generation, after making some important considerations. Indeed, there are two possibilities to realize this:

1. All the users that should know a share, generate a shard of it and then combine the shard;
2. One of the users that should know a share generates it and shares it securely to the other users that are allowed to know it.

At this point, the key generation is performed as before, where each party sends messages according to the function τ. The signature algorithm is also performed in the same way as the full threshold scheme, using the function τ to determine which party sends which messages at each round. To ease the readability and the comprehension of the protocol, we explicitly reported the signature protocol in Algorithm 4.

The proof of security for this scheme is practically equal to the full threshold one: in fact, one can imagine that, after an initial phase to see who has the required shares, the scheme is essentially an (M, M)-threshold scheme. Unfortunately, the number of shares required for each user and the number of rounds become respectively $\binom{N}{T}$ and $\binom{N}{T-1}$, which are exponential in $\min(T, N-T)$, and thus the scheme is practical only in certain scenarios; for example for $T = N$ (full threshold) or N small. For the case $T = N - 1$ and $N > 3$, the size of the shares is already linear in N and the rounds are quadratic in N. The main issue here is the non-commutativity of the group, which precludes the usage of traditional techniques for secret sharing.

Algorithm 4 $\texttt{Subset} - \texttt{Threshold} - \texttt{Sign}$

Require: $x \in X$, a security parameter λ, a hash function $\texttt{H}$, a public key $(x, y = g \star x)$, a set J of T party and the turn function τ. The party P_i knows the (multiplicative) shares g_i of $g = g_1 \cdots g_M$, for all $i \in I$.

Ensure: A valid signature for the message $\texttt{m}$ under the public key (x, y). ▷ Shared commitment generation phase

for $j = 1$ to λ **do**
  Set $x_0^j = x$
  for $i = 1$ to M **do**
    $\texttt{cp} = \tau(J, i)$
    $P_{\texttt{cp}}$ chooses $\tilde{g}_i^j \in G$ and sends $x_i^j = \tilde{g}_i^j \star x_{i-1}^j$ to $P_{\tau(J,i+1)}$
  end for
  They set $x^j = x_M^j = \tilde{g}^j \star x = (\tilde{g}_N^j \cdots \tilde{g}_1^j) \star x$
end for ▷ Non-iterative challenges evaluation
Compute $\texttt{ch} = \texttt{H}(x^1||...||x^\lambda||\texttt{m})$ ▷ Shared response generation phase
for $j = 1$ to λ **do**
  if $\texttt{ch}_j = 0$ **then** P_i discloses all $\tilde{g}_l^j$ generated
    $\texttt{resp}_j = \tilde{g}_M^j \cdots \tilde{g}_1^j$ is then published
  else set $u_0 = e$
    for $i = 1$ to M **do**
      $\texttt{cp} = \tau(J, i)$
      $P_{\texttt{cp}}$ computes $u_i = \tilde{g}_i^j u_{i-1} g_i^{-1}$ and sends u_i to $P_{\tau(J,i+1)}$
    end for
    Get $u_M = \tilde{g}^j g^{-1} = \tilde{g}_M^j \cdots \tilde{g}_1^j g_1^{-1} \cdots g_M^{-1}$
    $\texttt{resp}_j = u_i$ is published
  end if
end for
$\texttt{sig} = \texttt{ch}||\texttt{resp}_1||...||\texttt{resp}_\lambda$

Theorem 2. *Under the hardness of GAIP, the* (T, N)*) threshold signature scheme is Existentially Unforgeable under adaptive Chosen-Message Attacks (EUF-CMA) in the Random Oracle Model.*

Sketch. The proof is very similar to that of the full threshold case. First of all, note that, since the adversary controls at most $T - 1$ players, there must be at least a set $I_{\texttt{ho}} \in \mathcal{I}$ composed only by honest players. Moreover, in every execution of $\texttt{Threshold} - \texttt{Sign}$ there must be at least one player $P_{\texttt{ho}}$ with $\texttt{ho} \in I_{\texttt{ho}}$, holding the share $g_{\texttt{ho}}$.

During the simulation of the key generation, $\mathcal{A}$ follows the protocol normally, except when it takes the role of $P_{\texttt{ho}}$ applying $g_{\texttt{ho}}$. In this round, it uses the same simulation technique as before and sends the appropriate value, in order to have the final output that matches the challenger's key. $\mathcal{A}$ will play the role of all the honest parties and will act following the protocol normally except when it takes the role of $P_{\texttt{ho}}$ during round $\texttt{ho}$. In this round, $\mathcal{A}$ would use the same simulation strategy and simulate the ZKP. The rest of the proof is identical.

□

5 Concrete Instantiations

In this section, we will present concrete instantiations of our protocols, using the code equivalence group actions behind the LESS and MEDS signature schemes [7,23]; note that, however, our protocol is very general, and it is in principle possible to utilize other groups and group actions instead. We begin by showing that several optimizations embedded in the schemes' design can be adapted in order to be applied to our work too. We will discuss these again using the generic group action notation, since they can work in general.

5.1 Multi-bit Challenges

This optimization is applied to LESS in Section 5.1 of [7] and to MEDS in Section 5 of [23]. In a nutshell, the technique proposes to replace the binary challenges of the verifier with multi-bit ones, where each challenge value corresponds to a different public key. In this way, it is possible to amplify soundness, at the cost of an increase in public key size. To be precise, we will fix a non-negative integer l, set $r = 2^l$ and consider a challenge space of cardinality r. Note that the case $l = 1$ corresponds to the original protocol. With this optimization, the security of the scheme relies on a different, but related problem:

Problem 6 (mGAIP: Multiple Group Action Inverse Problem)**.** Given a collection $x_0, ..., x_{r-1}$ in X, find, if any, an element $g \in G$ and two different indices $j \neq j'$ such that $x_{j'} = g \star x_j$.

We proceed to prove the equivalence of hardness between this problem and Problem 1, by generalizing the proof of Theorem 3 from [7].

Proposition 3. *Given an algorithm to solve mGAIP, that runs in time T and succeeds with probability ϵ, it is possible to solve GAIP (Problem 1), in time approximately equal to $T + O(\mathsf{poly}(n))$, with probability of success equal to $\epsilon/2$.*

Proof. Let $\mathcal{A}$ be an adversary for mGAIP. We now show how to construct an adversary $\mathcal{A}'$ that is able to solve GAIP using $\mathcal{A}$ as a subroutine. From a GAIP instance $(x, y = g \star x)$, $\mathcal{A}'$ samples uniformly at random $g_i^{(0)}, ..., g_i^{(r-1)}$ for $r = 2^l$. Then, it computes (in polynomial time) half of the elements starting from x, and half starting from y; wlog, we can imagine that $x_i = g_i^{(i)} \star x$ for $i \in [0; r/2 - 1]$, while $x_i = g_i^{(i)} \star y$ for $i \in [r/2; r-1]$ are generated from y. Since the new instances are randomly generated, they are indistinguishable from the original one. At this point, $\mathcal{A}'$ runs $\mathcal{A}$ on input $x_0, ..., x_{r-1}$, and this will output, with probability ϵ, a response g^*, j, j' such that $x_{j'} = g^* \star x_j$. Now, if the two indices lie in the two different halves of $[0, r]$, it is possible to use the random group element to get g; for example if $j < r/2 < j'$ then $g = \left(g_i^{(j')}\right)^{-1} \cdot g^* \cdot g_i^{(j)}$. Since this happens with probability $1/2$, we get the thesis. □

As a consequence, we will consider $r-1$ public keys $x_1, ..., x_{r-1}$ generated from the initial element x_0 by $r-1$ shared keys $g^{(1)}, ..., g^{(r)}$ (with the notation $g^{(0)} = e$). To do this, we modify the `KeyGen` algorithm in order to generate $r-1$ shared secret keys $g^{(1)}, ..., g^{(r)}$ and public keys $x_1, ..., x_m$ (using the notation $g^{(0)} = e$ and $x_0 = x$). Essentially, the process consists of repeating $r-1$ times the classical `KeyGen` algorithm, as shown in Algorithm 5; to avoid burdening the reader, we have only inserted the modified protocol for the full-threshold case, since the other one follows immediately.

Algorithm 5 `KeyGen` for the l-multibit

Require: $x \in X$.
Ensure: Public keys $x_j = g^{(j)} \star x_0$ with each participant P_i holding $g_i^{(j)}$ such that $\prod g^{(j)} = g^{(j)}$, for all $j = 1, ..., r-1$.

1: Each participant P_i chooses $g_i^{(j)} \in G$ and publishes $\bar{x}_{j,i} = g_i^{(j)} \star x_0$, for all $j = 1, ..., r-1$.
2: Set $x_{j,0} = x$, for all $j = 1, ..., r-1$.
3: **for** $i = 1$ to N **do**
4: **for** $j = 1$ to $r-1$ **do**
5: P_i computes $x_{j,i} = g_i^{(j)} \star x_{j,i-1}$.
6: If $i < N$ P_i sends $x_{j,i}$ to P_{i+1}.
7: P_i produces a ZKP proving the honest use of $g_i^{(j)}$ (using $\bar{x}_{j,i}$).
8: **end for**
9: **end for**
10: **return** $x_j = x_{j,N}$ for all $j = 1, ..., r-1$. The private key of P_i is $g_i^{(j)}$.

At this point we can modify the protocol in Figure 2 to produce challenges in a larger space, as shown in Figure 3.

With this protocol the soundness error is reduced to $\frac{1}{2^l}$, thus in the signing algorithm we only need to execute $\lceil \frac{\lambda}{l} \rceil$ rounds, reducing both the signature size and the computational cost, but increasing the public key size. It is possible to obtain a security reduction to mGAIP (Problem 6), in the random oracle model, for the multibit version of the threshold scheme, in the same way as for the proof of Theorem 1. We avoid do not include it, since it is a well-known optimization in literature (see for example Section 5.1 of [7] and Section 4 of [26]).

5.2 Fixed-weight challenges

Another possible variant for group action-based signature schemes is the use of fixed-weight challenge strings, as shown for instance in [7, section 5.2]. The idea is simple: since on the challenge bit 0 the Prover's response consists of a random group element $\tilde{g}$, he can simply transmit the PRNG seed used to generate it. This consists usually of only λ bits, thus it makes sense to have the challenge 0 occur (much) more often than 1. To do that, one has to use the hash function H to return a vector of fixed weight ω and length t.

Public Data : Group G acting on X via $\star$, element $x_0 \in X$ and hash function $\mathtt{H}$.
Private Key : Group elements $g^{(j)} = g_{I_1}^{(j)} \cdots g_{I_M}^{(j)}$ with ${}^{(j)}g_I \in \mathcal{G}$, for all $j = 1, ..., r-1$.
Shares for P_i : all group elements $g_I^{(j)}$ such that $j \in I$, for all $j = 1, ..., r-1$.
Public Key : $x^{(j)} = g^{(j)} \star x_0$, for all $j = 1, ..., r-1$.

PROVERS		**VERIFIER**
Set $\tilde{x} \leftarrow x$ and for $i = 1, ..., M$ do :		
$P_{\tau(i)}$ get $\tilde{g}_i \xleftarrow{\$} X$ and set $\tilde{x} \leftarrow \tilde{g}_i \star \tilde{x}$	$\xrightarrow{h}$	
Set $h = \mathtt{H}(\tilde{x})$.		
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0, ..., r-1\}$.
If $b = 0$ they opens all $\tilde{g}_I$ and		
set $u \leftarrow \tilde{g}_{I_1} \cdots \tilde{g}_{I_M}$.		Accept if $\mathtt{H}(u \star x_0) = h$.
If $b \neq 0$ then $u \leftarrow e$.	$\xrightarrow{u}$	
for $i = 1, ..., M$ do :		
$P_{\tau(i)}$ evaluate and send $u \leftarrow \tilde{g}_{I_i} \cdot u \cdot \left(g_{I_i}^{(b)}\right)^{-1}$.		Accept if $\mathtt{H}(u \star x_b) = h$.

Fig. 3. Threshold identification protocol with soundness error $\frac{1}{2^l}$.

To avoid security loss we need to have a *preimage security* (the difficulty of guessing in the challenge space) of still λ bits, thus we need to choose t, ω such that:

$$\binom{t}{\omega} \geq 2^{\lambda} \ .$$

In this way, for carefully selected parameters, we can decrease the signature size (even if the number of rounds is increased). The security of the scheme remains unchanged. More on this can be read, for instance, in section 5.2 of [7], but also in [50] and in the original Fiat-Shamir paper [33].

When one tries to apply this optimization to the threshold case, a new obstacle arises. In fact, the parties are not able to share a single seed used for the generation of the ephemeral map $\tilde{g}$, but have to share $M = \binom{N}{T-1}$ of them. Thus, if the challenge bit is 0, the parties need to send all the M bits, and the total communication cost becomes $M\lambda$. This can be a problem in two ways:

- For this strategy to make sense, we need $M\lambda$ to be smaller than the weight of the group element.
- In some applications, it can be desirable to not disclose the parameters T and N.

In these cases, the fixed-weight optimization should not be used, and the signers should just send the group element instead.

Note that the two optimizations can be combined together in a rather intuitive way, as explained again in [7].

Seed tree. If seeds are indeed used to represent random objects, it makes sense to transmit them efficiently, rather than individually; this can be done via a so-called *seed tree*. This primitive uses a secret master seed to generate t seeds recursively exploiting a binary structure, but in the "other direction": each parent node is used to generate two child nodes via a PRNG. When a subset of $t-\omega$ seeds is requested for the signature, we only need to send the appropriate nodes, reducing the space required for the seeds from $\lambda(t-\omega)$ to a value bounded above by $\lambda N_{\mathsf{seeds}}$, where

$$N_{\mathsf{seeds}} = 2^{\lceil \log(\omega) \rceil} + \omega(\lceil \log(t) \rceil - \lceil \log(\omega) \rceil - 1) \ ,$$

as shown in [41,23]. More details are given, for instance, in Section 2.7 of [14].

5.3 Scheme Parameters

In Table 1, we compare the public key size and signature size of the different variants, both alone and combined, with respect also to T and N (setting $M = \binom{N}{T-1}$). We will use ξ to denote the weight in bits of an element of the set X, and γ to denote that of a group element of G.

Version	$\#\mathsf{rounds}$	$\lvert\mathsf{pk}\rvert$	$\lvert\sigma\rvert$
Classic	λ	ξ	$\lambda\gamma + \lambda$
Multibit	$\lceil \frac{\lambda}{l} \rceil$	$(2^l - 1)\xi$	$\lceil \frac{\lambda}{l} \rceil (\gamma + l)$
Fixed	t s.t. $\binom{t}{\omega} \geq 2^\lambda$	ξ	$N_{\mathsf{seeds}} M\lambda + \omega\gamma + t$
Both	t s.t. $\binom{t}{\omega}(2^l - 1)^\omega \geq 2^\lambda$	$(2^l - 1)\xi$	$N_{\mathsf{seeds}} M\lambda + \omega\gamma + t$

Table 1. Overview of the sizes for different variants.

In our signing algorithm, for each iteration of the for loop over $1, ..., M$, each user needs to send the following quantities to the next user:

- $\#\mathsf{rounds} \cdot \xi$ bits for the commitment phase,
- $|\sigma|$ bits for the response phase.

It follows that the overall cost, for the information exchanged by the users, is given by:

$$\binom{N}{T-1}(\#\mathsf{rounds} \cdot \xi + |\sigma|) \tag{1}$$

At this point, we can see specific choices for LESS and MEDS. We will select the public parameters that satisfy the requirement of 128 bits of classical security and at least 64 bits of quantum security, and evaluate ξ and γ accordingly. We

include here the data for the original signature schemes, as well as parameters that we found in order to optimize the sum $|\mathsf{pk}| + |\sigma|$ for the cases $(2, 3)$, $(3, 5)$ and the case without fixed-weight challenges to hide T and N (that must be used also if $M\lambda \geq \gamma$).

Instantiations with LESS. From Table 7 of [7], we have taken the secure parameters for the linear equivalence problem: $n = 198, k = 94$ (length and dimension of the code), $q = 251$ (the field size). We obtain that the size of a single code in systematic form is given by $(n-k)k\log_2(q)$ bits, so $\xi = 9742$ bytes, while the size of monomial map is $\gamma = 194$ bytes. To evaluate it we have used an optimization presented by the LESS team at CBCrypto 2023 that allows to cut in half the cost of the monomial map. Numbers are reported in Table 2, where we report, in the last column, also the total amount of exchanged data.

Case	Variant	t	ω	l	$\lvert\mathsf{pk}\rvert$ (kB)	$\lvert\sigma\rvert$ (kB)	Exc. (MB)
centralized	F+M	333	26	1	9.74	6.85	-
(2,3)	F+M	247	30	1	9.74	10.27	7.28
(3,5)	F+M	247	30	1	9.74	20.57	11.83
(*,*)	M	-	-	1	9.74	24.85	$\binom{N}{T-1}$1.27

Table 2. Parameters for the threshold version of LESS

Instantiations with MEDS. From Section 7.2 [23] we have taken the secure parameters for the matrix code equivalence problem: $n = 12, m = 12, k = 10$ (matrix sizes and dimension of the code), $q = 65521$ (the field size). Thus we obtain that the size of a single code in systematic form is given by $(nm - k)k\log_2(q)$ bites, so $\xi = 2696$ bytes, while the size of group element is given by $(n^2 + m^2)\log_2(q)$ bits, corresponding to $\gamma = 576$ bytes. Numbers are reported in Table 3; as above, in the last column we report the total amount of exchanged data.

Case	Variant	t	ω	l	$\lvert\mathsf{pk}\rvert$ (kB)	$\lvert\sigma\rvert$ (kB)	Exc. (MB)
centralized	F+M	333	26	1	2.70	16.78	-
(2,3)	F+M	333	26	1	2.70	20.30	2.75
(3,5)	F+M	244	20	2	8.09	23.07	6.81
(*,*)	M	-	-	3	18.9	24.78	$\binom{N}{T-1}$0.369

Table 3. Parameters for the threshold version of MEDS

6 A Construction Tailored for LESS

The scheme proposed in Figure 3 becomes impractical for large values of T, N since the number of round and shares necessaries is $\binom{N}{T-1}$. In this section, we propose an alternative construction, tailored to LESS signature scheme. First of all, we reformulate the vectorization problem (Problem 3) in the context of code equivalence:

Problem 7 (Linear Code Equivalence)**.** Given two generator matrices $\boldsymbol{G}, \boldsymbol{G'} \in \mathbb{F}_q^{k\times n}$ for two k-dimensional linear codes $\mathscr{C}$ and $\mathscr{C}'$, find, if any, an invertible matrix $\boldsymbol{S} \in \mathrm{GL}_k$ and a monomial matrix $\boldsymbol{Q} \in M_n$ such that $\boldsymbol{G'} = \boldsymbol{SGQ}$.

While the monomial map is usually considered the "true" representative of the action, the hardness of LEP relies both on the monomial matrix $\boldsymbol{Q}$ and the change of basis matrix $\boldsymbol{G}$. Indeed, if either one of the two matrices is public, it is possible to efficiently retrieve the other one . In particular:

Proposition 4. *Let $\boldsymbol{G}$ and $\boldsymbol{G'} = \boldsymbol{SGQ}$ be the generator matrices for two linearly equivalent codes. If $\boldsymbol{S}$ is known then it is possible to recover $\boldsymbol{Q}$ in polynomial time by using sorting algorithms.*

Proof. Knowing the linear map $\boldsymbol{S}$, we can have the two codes written with respect to the same basis. In particular we can consider the matrix $\boldsymbol{S}^{-1}\boldsymbol{G'} = \boldsymbol{GQ}$. Note that this matrix and $\boldsymbol{G}$ have the same columns vectors, only permuted and multiplied by a constant. So we can write a first naive algorithm to recover the monomial map:

1. Go through the columns of $\boldsymbol{G}$, set them in standard form by multiplying each column by the inverse of the first non-zero entry, and sort them using any order. Memorize the map $\boldsymbol{Q}_1$ used for the sorting.
2. Go through the columns of $\boldsymbol{G'}$, set them in standard form by multiplying each column by the inverse of the first non-zero entry, and sort them with respect to the same order used previously. Memorize the map $\boldsymbol{Q}_2$ used for the sorting.
3. The permutation underlying the monomial map $\boldsymbol{Q}$ is the permutation $\boldsymbol{Q} = \boldsymbol{Q}_1\boldsymbol{Q}_2^{-1}$.
4. Use the previous permutation applied to $\boldsymbol{G}$ to find the coefficients of the monomial map $\boldsymbol{Q}$.

□

Thanks to this observation, we can use the linear map as private key and share it instead of the monomial map. In particular we use particular invertible matrices, that allow to design a multiplicative threshold sharing.

6.1 Multiplicative to Additive Secret Sharing

In the case where k is an even number is possible to have a threshold secret sharing scheme based on matrix multiplication working in the multiplicative abelian subgroup $U \subseteq GL_k(\mathbb{F}_q)$ defined as follows:

$$U = \left\{ \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \mid \boldsymbol{A} \in \mathbb{F}_q^{\frac{k}{2} \times \frac{k}{2}} \right\} . \tag{2}$$

The group U has the interesting property that the multiplication of matrices corresponds to the addition of the submatrices:

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{I} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A}+\boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{3}$$

We can use this property to define a threshold secret sharing scheme for matrix multiplication by simply using Shamir's Secret Sharing [21, Section 1.5] for a secret $\frac{k}{2} \times \frac{k}{2}$ matrix $\boldsymbol{S}$ (it is enough to do it component-wise), so that it can be recovered by T parties by adding the share matrices multiplied by the Lagrange coefficients, i.e. $\boldsymbol{S} = \delta_1 \boldsymbol{S}_1 + ... + \delta_T \boldsymbol{S}_T$. At this point we have that:

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{S} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \delta_1 \boldsymbol{S}_1 \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \cdots \begin{bmatrix} \boldsymbol{I} & \delta_T \boldsymbol{S}_T \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$$

Using this secret sharing scheme reaches clearly the necessary goal of being multiplicative, but has the problem that the secret matrix is in triangular form, so clearly cannot be used alone, since it would leak the private permutation. To solve this problem we use the group U', that is the transpose of the previous group:

$$U' = \left\{ \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ A & \boldsymbol{I} \end{bmatrix} \mid \boldsymbol{A} \in \mathbb{F}_q^{\frac{k}{2} \times \frac{k}{2}} \right\} . \tag{4}$$

Notation. If $\boldsymbol{A}$ is a $\frac{k}{2} \times \frac{k}{2}$ matrix we will indicate by $\hat{\boldsymbol{A}}$ the corresponding matrix $\left(\begin{smallmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{0} & \boldsymbol{I} \end{smallmatrix}\right) \in U$ and by $\hat{\boldsymbol{A}}'$ the corresponding matrix $\left(\begin{smallmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{A} & \boldsymbol{I} \end{smallmatrix}\right) \in U'$.

We can now consider matrices in the product group $U \times U'$ in which all the elements are the product of two matrices $\hat{\boldsymbol{S}}_1 \in U, \hat{\boldsymbol{S}}_2' \in U'$ which, when multiplied, have the form:

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{S}_1 \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{S}_2 & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_1\boldsymbol{S}_2 + \boldsymbol{I} & \boldsymbol{S}_1 \\ \boldsymbol{S}_2 & \boldsymbol{I} \end{bmatrix} . \tag{5}$$

At this point the secret is a $k \times k$ matrix that should not have any structure to allow the recovery of information on the equivalence between the codes. We will discuss the security of this approach, as well as the possible use of more than two matrices, in Section 6.3.

6.2 A New Protocol Exploiting Invertible Matrices

In Algorithm 6 we present our new protocol. The threshold secret sharing of the secret key and the public key generation requires a trusted authority, while the verification function remains unchanged. The core idea use the aforementioned group to achieve a (T, N) secret sharing of the linear map and Proposition 4 to be able to compute the monomial map during the signature. In Appendix B we show a ZKP from which is possible to design a centralized signature for this approach.

Without loss of generality we have assumed that the subset of users trying to sign is $\{1, ..., T\}$. By $\hat{\boldsymbol{S}}_{\mathrm{TSS}(i),r}$ we mean the additive share for $\hat{\boldsymbol{S}}_r$ already modified using Lagrange coefficients so that $\hat{\boldsymbol{S}}_{\mathrm{TSS}(1),r} \cdots \hat{\boldsymbol{S}}_{\mathrm{TSS}(T),r} = \hat{\boldsymbol{S}}_r$.

Algorithm 6 $\mathtt{Threshold-Sign-Triangular}$

Require: $q, n, k \in \mathbb{N}$, $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$, a hash function $\mathtt{H}$. Each party holds additive shares of $\hat{\boldsymbol{S}}_{\mathrm{TSS}(i),1}, \hat{\boldsymbol{S}}_{\mathrm{TSS}(i),2} \in U \times U'$ as previously defined. A public key $(\boldsymbol{G}, \boldsymbol{G}' = \hat{\boldsymbol{S}}_1 \hat{\boldsymbol{S}}'_2 \boldsymbol{G} \boldsymbol{Q})$.

Ensure: A valid LESS signature for the message $\mathtt{m}$ under the public key $(\boldsymbol{G}, \hat{\boldsymbol{S}}_1 \hat{\boldsymbol{S}}'_2 \boldsymbol{G} \boldsymbol{Q})$.

Set $\tilde{\boldsymbol{G}}^j \leftarrow \boldsymbol{G}'$ for $j = 1, ..., \lambda$.
for $r = 1, 2$, $i = 1$ to T **do**
  for $j = 1$ to λ **do**
    P_i samples randomly $\tilde{\boldsymbol{S}}^j_{i,r} \in GL_k(\mathbb{F}_q)$ and $\tilde{\boldsymbol{Q}}^j_{i,r} \in M_n(q)$
    P_i sends $\tilde{\boldsymbol{G}}^j \leftarrow \tilde{\boldsymbol{S}}^j_{i,r} \tilde{\boldsymbol{G}}^j \tilde{\boldsymbol{Q}}^j_{i,r}$ to P_{i+1}. ▷ Assuming $P_{T+1} = P_1$
  end for
end for
Compute $\mathtt{ch} = \mathtt{H}(\mathrm{SF}(\tilde{\boldsymbol{G}}^1)||...||\mathrm{SF}(\tilde{\boldsymbol{G}}^\lambda)||\mathtt{m})$
Set $\boldsymbol{R}^j \leftarrow \mathcal{I}$ for all $j = 1$ to λ
for $r = 1, 2$, $i = 1$ to T **do**
  for $j = 1$ to λ **do**
    if $\mathtt{ch}_j = 0$ **then**
      P_i computes $\boldsymbol{R}^j \leftarrow \boldsymbol{R}^j \tilde{\boldsymbol{Q}}^j_{i,r}$ ▷ $\boldsymbol{R}^j$ is a monomial matrix
    else
      P_i computes $\boldsymbol{R}^j \leftarrow \tilde{\boldsymbol{S}}^j_{i,r} \boldsymbol{R}^j \hat{\boldsymbol{S}}_{\mathrm{TSS}(i),r}$ ▷ $\boldsymbol{R}^j$ is a matrix $k \times k$
    end if
    P_i sends $\boldsymbol{R}^j$ to P_{i+1} ▷ Assuming $P_{T+1} = P_1$
  end for
end for
for $j = 1$ to λ **do**
  If $\mathtt{ch}_j = 1$ all parties use $\boldsymbol{R}^j$ to retrieve $\boldsymbol{Q}\tilde{\boldsymbol{Q}}$.
  Set $\boldsymbol{R}^j = \boldsymbol{Q}\tilde{\boldsymbol{Q}}$.
end for
$\mathtt{resp}_j = \boldsymbol{R}^j$ for all $j = 1$ to λ
$\mathtt{sig} = \mathtt{ch}||\mathtt{resp}_1||...||\mathtt{resp}_\lambda$

Algorithm 7 `Verify`

Require: $q, n, k \in \mathbb{N}$, $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$, a hash function $\mathtt{H}$. A public key $(\boldsymbol{G}, \boldsymbol{G}' = \mathrm{SF}(\boldsymbol{GQ}))$.
Ensure: Accept if the signature for the message $\mathtt{m}$ is valid under the public key $(\boldsymbol{G}, \boldsymbol{G}')$.
1: Parse $\mathtt{ch}, \mathtt{resp}_1, ..., \mathtt{resp}_\lambda$ from $\mathtt{sig}$
2: **for** $j = 1$ to λ **do**
3: **if** $\mathtt{ch}_j = 0$ **then**
4: set $\tilde{\boldsymbol{G}}^j = \mathrm{SF}(\boldsymbol{G}' \cdot \mathtt{resp}_j)$
5: **else**
6: set $\tilde{\boldsymbol{G}}^j = \mathrm{SF}(\boldsymbol{G} \cdot \mathtt{resp}_j)$
7: **end if**
8: **end for**
9: Accept if $\mathtt{ch} = \mathtt{H}(\tilde{\boldsymbol{G}}^1 || ... || \tilde{\boldsymbol{G}}^\lambda || \mathtt{m})$

Proposition 5. *When executed by an honest subset of users, Algorithm 6 produces a valid signature, that can be verified from Algorithm 7.*

Proof. We avoid using the j apex to ease the notation. After executing the first **for** loops, we have that $\tilde{\boldsymbol{G}} = \tilde{\boldsymbol{S}}\boldsymbol{G}'\tilde{\boldsymbol{Q}}$ with $\tilde{\boldsymbol{S}} = \tilde{\boldsymbol{S}}_{T,2} \cdots \tilde{\boldsymbol{S}}_{1,2} \cdot \tilde{\boldsymbol{S}}_{T,1} \cdots \tilde{\boldsymbol{S}}_{1,1}$ and $\tilde{\boldsymbol{Q}} = \tilde{\boldsymbol{Q}}_{1,1} \cdots \tilde{\boldsymbol{Q}}_{T,1} \cdot \tilde{\boldsymbol{Q}}_{1,2} \cdots \tilde{\boldsymbol{Q}}_{T,2}$. Now:

- When $\mathtt{ch} = 0$ in line 13 we are clearly evaluating $\tilde{\boldsymbol{Q}}$ recursively.
- When $\mathtt{ch} = 1$ in line 15 we are evaluating:

$$\tilde{\boldsymbol{S}}_{T,2} \cdots \tilde{\boldsymbol{S}}_{1,2} \cdot \tilde{\boldsymbol{S}}_{T,1} \cdots \tilde{\boldsymbol{S}}_{1,1} \cdot \hat{\boldsymbol{S}}_{\mathrm{TSS}(1),1} \cdots \hat{\boldsymbol{S}}_{\mathrm{TSS}(T),1} \cdot \hat{\boldsymbol{S}}_{\mathrm{TSS}(1),2} \cdots \hat{\boldsymbol{S}}_{\mathrm{TSS}(T),2} = \tilde{\boldsymbol{S}}_{T,2} \cdots \tilde{\boldsymbol{S}}_{1,2} \cdot \tilde{\boldsymbol{S}}_{T,1} \cdots \tilde{\boldsymbol{S}}_{1,1} \cdot \hat{\boldsymbol{S}}_1 \cdot \hat{\boldsymbol{S}}'_2 = \tilde{\boldsymbol{S}} \cdot \hat{\boldsymbol{S}}_1 \cdot \hat{\boldsymbol{S}}'_2 \ .$$

Since $\tilde{\boldsymbol{G}} = \tilde{\boldsymbol{S}}\boldsymbol{G}'\tilde{\boldsymbol{Q}} = \tilde{\boldsymbol{S}}\hat{\boldsymbol{S}}_1\hat{\boldsymbol{S}}'_2\boldsymbol{G}\boldsymbol{Q}\tilde{\boldsymbol{Q}}$, thanks to Proposition 4 it is possible to evaluate $\boldsymbol{Q}\tilde{\boldsymbol{Q}}$ and pass the verification. □

6.3 Security

The main difference between the new protocol and the (N, N)-threshold one is that we are limiting the choice for the linear map $\boldsymbol{S}$. The only other relevant difference is that instead of having a secret sharing of the monomial map $\boldsymbol{Q}$, we have a secret sharing of the linear map $\boldsymbol{S}$, but both approach are equivalent as noted in Proposition 4, as long as $\boldsymbol{Q}$ remains secret. Indeed, parties only recover $\boldsymbol{Q}\tilde{\boldsymbol{Q}}$ during the signature phase, with $\tilde{\boldsymbol{Q}}$ being an ephemeral monomial matrix, whose knowledge is shared among all the P_i.

We briefly analyze the security implication of limiting the choice of $\boldsymbol{S}$, and we suggest a plausible assumption to justify it.

Hardness of the structured problem. First of all, we need to guarantee that the secret matrix $\boldsymbol{S} \in U \times U'$ does not leak information on the monomial map. In other words, it should still be hard to find $\boldsymbol{Q}$ given $\boldsymbol{G}$ and $\boldsymbol{SGQ}$ even for $\boldsymbol{S} \in U \times U'$. Indeed, using a particularly crafted matrix can potentially leak some information on the equivalence map. For example, using a single matrix in U we would have:

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{G}_1 \\ \boldsymbol{G}_2 \end{bmatrix} \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{G}_1\boldsymbol{Q} + \boldsymbol{A}\boldsymbol{G}_2\boldsymbol{Q} \\ \boldsymbol{G}_2\boldsymbol{Q} \end{bmatrix} \tag{6}$$

Since we know both $\boldsymbol{G}_2$ and $\boldsymbol{G}_2\boldsymbol{Q}$, we can use Proposition 4 to retrieve part of the permutation with high probability.

Speaking heuristically, we expect that a matrix in $U \times U'$, that is, of the form given in Equation (5), is enough to scramble the rows and hide the monomial map used. To justify this intuition we introduce the following proposition.

Proposition 6. *Consider the matrix $\boldsymbol{G} \in \mathbb{F}_q^{n \times k}$, a monomial matrix $\boldsymbol{Q} \in M_n(q)$, an invertible matrix $\boldsymbol{S} = \left[\begin{smallmatrix} \boldsymbol{S}_{11} & \boldsymbol{S}_{12} \\ \boldsymbol{S}_{21} & \boldsymbol{S}_{22} \end{smallmatrix}\right] \in GL_k(q)$ and the product $\boldsymbol{G}' = \boldsymbol{SGQ}$. If an adversary $\boldsymbol{A}$ is able to retrieve the secret $\boldsymbol{Q}$ from any instance $(\boldsymbol{G}, \boldsymbol{G}', \boldsymbol{S}_{11}, \boldsymbol{S}_{21})$ (i.e. knowing half of the columns), then it can be used to solve LEP with parameters $[n, k/2]$.*

Proof. Given an instance $(\boldsymbol{H}, \boldsymbol{H}')$ of dimension $k/2$, we can forge an input for $\mathcal{A}$ as $\boldsymbol{G} = \left[\begin{smallmatrix} \boldsymbol{0} \\ \boldsymbol{H} \end{smallmatrix}\right]$ and $\boldsymbol{G}' = \left[\begin{smallmatrix} \boldsymbol{R}\boldsymbol{H}' \\ \boldsymbol{H}' \end{smallmatrix}\right]$ where $\boldsymbol{R}$ is a random invertible matrix. Observe in fact that if $\boldsymbol{H}' = S\boldsymbol{H}\boldsymbol{P}$ then we have that:

$$\begin{bmatrix} \boldsymbol{S}_{11} & \boldsymbol{R}\boldsymbol{S} \\ \boldsymbol{S}_{21} & \boldsymbol{S} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{H} \end{bmatrix} \cdot \boldsymbol{P} = \begin{bmatrix} \boldsymbol{R}\boldsymbol{S}\boldsymbol{H}\boldsymbol{P} \\ \boldsymbol{S}\boldsymbol{H}\boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}\boldsymbol{H}' \\ \boldsymbol{H}' \end{bmatrix} .$$

Thus the algorithm $\mathcal{A}$ on input $\boldsymbol{G}, \boldsymbol{G}'$ would output the secret map for the instance $\boldsymbol{H}, \boldsymbol{H}'$. □

This proposition is not an actual reduction (as for example Proposition 3), since to break the $[n, k/2]$ instance it is necessary to be able to attack the particular instance of a matrix where half of the entries are zeros. Still, it can be used for heuristic reasons, but also two possible ideas to actually exploit it would be to use the matrix $\boldsymbol{G} = \left[\begin{smallmatrix} \boldsymbol{0} \\ \boldsymbol{H} \end{smallmatrix}\right]$ for the users at the start of the protocol (since, while committing, they can still use the systematic form) or to consider codes of dimension $k = 2\frac{n}{3}$ (where n is the code length as usual), so that the codes with parameters $[n, k/2]$ have the same level of security.

Another possible way to improve the security would be to additionally consider matrices which are the product of l triangular matrices from U (where one every two matrices is transposed). Thus, it makes sense to formalize a new security assumption.

Problem 8 (l-Modified Linear Code Equivalence)**.** Let $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$ be the generator matrix of a $[n,k]$ linear code. Consider a monomial matrix $\boldsymbol{Q} \in M_n$, an invertible matrix $\boldsymbol{S} \in U \times U' \cdots$, where l is the number of groups used in the direct product, and the generator matrix $\boldsymbol{G}' = \boldsymbol{SGQ}$. The problem consist of solving the linear equivalence problem for the codes given by $\boldsymbol{G}$ and $\boldsymbol{G}'$, i.e. find $\boldsymbol{Q}$.

At the moment, we are using this assumption for $l = 2$, since for $l = 1$ the problem is easy and we cannot see meaningful advantages in using $l > 2$.

Hiding during calculations. During the combination phases, each participant has access to the following information from the user P_i (note that we simplify the notation to avoid cluttering):

1. $\tilde{\boldsymbol{S}}\tilde{\boldsymbol{G}}\tilde{\boldsymbol{Q}}$ with $\tilde{\boldsymbol{G}}$ from the commitment phase;
2. $\tilde{\boldsymbol{S}}\boldsymbol{R}\hat{\boldsymbol{S}}$ with $\boldsymbol{R}$ from the response to the challenge $b = 1$.

Ideally, we would like it to be infeasible to use these informations, in order to retrieve the share $\hat{\boldsymbol{S}}$ or the ephemeral map $\tilde{\boldsymbol{Q}}$.

First of all, we can see that we have information-theoretic security for the share $\hat{\boldsymbol{S}}$. In fact, by assigning $\bar{\boldsymbol{R}} = \tilde{\boldsymbol{S}}\boldsymbol{R}$ and $\boldsymbol{P} = \boldsymbol{R}\hat{\boldsymbol{S}}$, we can observe that:

$$\begin{bmatrix} \boldsymbol{P}_{11} & \boldsymbol{P}_{12} \\ \boldsymbol{P}_{21} & \boldsymbol{P}_{22} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{R}}_{11} & \bar{\boldsymbol{R}}_{12} \\ \bar{\boldsymbol{R}}_{21} & \bar{\boldsymbol{R}}_{22} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{I} & \boldsymbol{S} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{R}}_{11} & \bar{\boldsymbol{R}}_{11}\boldsymbol{S} + \bar{\boldsymbol{R}}_{12} \\ \bar{\boldsymbol{R}}_{21} & \bar{\boldsymbol{R}}_{21}\boldsymbol{S} + \bar{\boldsymbol{R}}_{22} \end{bmatrix} \tag{7}$$

Thus, we can see that $\boldsymbol{S}$ must satisfy the following system.

$$\begin{cases} \bar{\boldsymbol{R}}_{11}\boldsymbol{S} + \bar{\boldsymbol{R}}_{12} = \boldsymbol{P}_{12} \\ \bar{\boldsymbol{R}}_{21}\boldsymbol{S} + \bar{\boldsymbol{R}}_{22} = \boldsymbol{P}_{22} \end{cases} \tag{8}$$

The system is underdetermined, and so any possible matrix $\boldsymbol{S}$ can be an acceptable solution via the right choices for $\bar{\boldsymbol{R}}_{12}, \bar{\boldsymbol{R}}_{22}$. However, there is still a possible issue here. Indeed, considering the equations $\boldsymbol{P}_{i1} = \tilde{\boldsymbol{S}}_{i1}\boldsymbol{R}_{11} + \tilde{\boldsymbol{S}}_{i2}\boldsymbol{R}_{21}$, for $i = 1, 2$ in principle we have additional information that can be exploited, and used to solve the code equivalence problem in 1 to obtain $\tilde{\boldsymbol{Q}}$ (for example, in an algebraic attack we would have additional equations).

When $\boldsymbol{R} = \boldsymbol{I}$, we are essentially again in the case of Proposition 6, that can be considered safe in some sense. We expect that similar results can be obtained also when $\boldsymbol{R}$ is different from the identity, but, as mentioned before, further cryptanalysis is necessary.

Overall, we summarize our analysis, with the assumption that the following problem is hard.

Problem 9. (Linear Equivalence with Additional Equations) Consider the matrices $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$,$\boldsymbol{R}_1, \boldsymbol{R}_2 \in \mathbb{F}_q^{\frac{k}{2}\times\frac{k}{2}}$, an invertible matrix $\boldsymbol{S} = \left[\begin{smallmatrix} \boldsymbol{S}_{11} & \boldsymbol{S}_{12} \\ \boldsymbol{S}_{21} & \boldsymbol{S}_{22} \end{smallmatrix}\right] \in \mathrm{GL}_k$, a monomial matrix $\boldsymbol{Q} \in M_n$ and define $\boldsymbol{G}' = \boldsymbol{SGQ}$. Recover $\boldsymbol{Q}$ from the knowledge of $\boldsymbol{G}, \boldsymbol{G}'$ and the additional equations:

$$\boldsymbol{S}_{11}\boldsymbol{R}_1 + \boldsymbol{S}_{12}\boldsymbol{R}_2 \ , \ \boldsymbol{S}_{11}\boldsymbol{R}_1 + \boldsymbol{S}_{12}\boldsymbol{R}_2. \tag{9}$$

7 Conclusions

We introduced two protocols for threshold signature schemes based on the Group Action Inverse Problem that are agnostic about which particular group action is used, and works without any further hypotheses. Our schemes are similar to well-known group action threshold schemes such as the one presented in [25,27], and share the strictly sequential round-robin communication sequence. Unfortunately, this structure seems to be unavoidable due to the inherent properties of group action computation. Additionally, we were able to define a decentralized key generation algorithm and to reduce the security of the threshold signature scheme to that of the centralized scheme, which is easily proved to be unforgeable under the hardness of the group action problem.

Our proposed schemes are practical for several real-world instances, such as $(2, 3)$ or $(3, 5)$ sharing, but cannot be used for arbitrary (T, N) since the number of shares required grows as a binomial coefficient. To circumvent this issue, in Section 6 we presented a tailor-made construction for the LESS signature [16], and its improved version [8], that only requires a number of shares which is linear in T. We observe that the new scheme security relies on novel cryptographic assumptions (Proposition 3), that require further study and cryptanalytic scrutiny. This would, incidentally, enable us to select a secure set of parameters for a practical instantiation, as well as a possible implementation. Another future direction for our work would be to define a distributed key generation and to apply the same ideas for MEDS, since for matrix code equivalence we have a result equivalent to Proposition 4 (See Problem 4, Remark 16 of [51]).

8 Acknowledgement

This publication was created with the co-financing of the European Union FSE-REACT-EU, PON Research and Innovation 2014-2020 DM1062/2021. The first author acknowledges support from TIM S.p.A. through the Ph.D. scholarship. The second author acknowledges support from Telsy S.p.A. and De Componendis Cifris through the M.Sc. scholarship and Collegio Clesio. The third author is a member of the INdAM Research Group GNSAGA. The fourth author acknowledges support from NSF through grant 1906360 and NSA through grant H98230-22-1-0328.
All the authors would like to thank Giuseppe D'Alconzo and Leonardo Errati for their comments and suggestions.

References

1. Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT*, pages 418–433. Springer, 2002.

2. Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, pages 418–433, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
3. Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and Jurjen Bos. HQC. NIST PQC Submission, 2020.
4. Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. NIST PQC Submission, 2020.
5. Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zémor, Valentin Vasseur, and Santosh Ghosh. BIKE. NIST PQC Submission, 2020.
6. Alessandro Barenghi, Jean-Francois Biasse, Tran Ngo, Edoardo Persichetti, and Paolo Santini. Advanced signature functionalities from the code equivalence problem. Cryptology ePrint Archive, Paper 2022/710, 2022. `https://eprint.iacr.org/2022/710`.
7. Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. Less-fm: fine-tuning signatures from the code equivalence problem. In *Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12*, pages 23–43. Springer, 2021.
8. Alessandro Barenghi, Jean-Francois Biasse, Edoardo Persichetti, and Paolo Santini. Less-fm: Fine-tuning signatures from the code equivalence problem (full version). Cryptology ePrint Archive, Paper 2021/396, 2021. `https://eprint.iacr.org/2021/396`.
9. Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. On the computational hardness of the code equivalence problem in cryptography. *Advances in Mathematics of Communications*, 17(1):23–55, 2023.
10. Michele Battagliola, Alessio Galli, Riccardo Longo, and Alessio Meneghetti. A provably-unforgeable threshold schnorr signature with an offline recovery party. In *DLT2022 at Itasec 2022, CEUR Workshop Proceedings*, 2022.
11. Michele Battagliola, Riccardo Longo, and Alessio Meneghetti. Extensible decentralized secret sharing and application to schnorr signatures. preprint: `https://eprint.iacr.org/2022/1551`, 2022.
12. Michele Battagliola, Riccardo Longo, Alessio Meneghetti, and Massimiliano Sala. A provably-unforgeable threshold EdDSA with an offline recovery party. preprint: `https://arxiv.org/abs/2009.01631`, 2020.
13. Michele Battagliola, Riccardo Longo, Alessio Meneghetti, and Massimiliano Sala. Threshold ECDSA with an offline recovery party. *Mediterranean Journal of Mathematics*, 19(4), 2022.
14. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. Cryptology ePrint Archive, Paper 2020/646, 2020. `https://eprint.iacr.org/2020/646`.
15. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. *IACR Cryptol. ePrint Arch.*, page 498, 2019.

16. Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. Less is more: Code-based signatures without syndromes. In Abderrahmane Nitaj and Amr Youssef, editors, *Progress in Cryptology - AFRICACRYPT 2020*, pages 45–65, Cham, 2020. Springer International Publishing.
17. Dan Boneh, Rosario Gennaro, and Steven Goldfeder. Using level-1 homomorphic encryption to improve threshold dsa signatures for bitcoin wallet security. In *International Conference on Cryptology and Information Security in Latin America*, pages 352–377. Springer, 2017.
18. Charlotte Bonte, Nigel P. Smart, and Titouan Tanguy. Thresholdizing hasheddsa: Mpc to the rescue. *International Journal of Information Security*, 20:879 – 894, 2021.
19. Luís T. A. N. Brandão and Michael Davidson. Notes on threshold eddsa/schnorr signatures. Accessed: 2023-05-01.
20. Luís T. A. N. Brandão, Michael Davidson, and Apostol Vassilev. Nist roadmap toward criteria for threshold schemes for cryptographic primitives. Accessed: 2020-08-27.
21. Selda Çalkavur, Alexis Bonnecaze, Romar dela Cruz, and Patrick Solé. *Code Based Secret Sharing Schemes: Applied Combinatorial Coding Theory*. World Scientific, 2022.
22. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24*, pages 395–427. Springer, 2018.
23. Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. Take your meds: Digital signatures from matrix code equivalence. *Cryptology ePrint Archive*, 2022.
24. Daniele Cozzo and Nigel P Smart. Sharing the luov: threshold post-quantum signatures. In *IMA International Conference on Cryptography and Coding*, pages 128–153. Springer, 2019.
25. Daniele Cozzo and Nigel P. Smart. Sashimi: Cutting up csi-fish secret keys to produce an actively secure distributed signing protocol. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography*, pages 169–186, Cham, 2020. Springer International Publishing.
26. Luca De Feo and Steven D Galbraith. Seasign: compact isogeny signatures from class group actions. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*, pages 759–789. Springer, 2019.
27. Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography – PKC 2020*, pages 187–212, Cham, 2020. Springer International Publishing.
28. Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Secure two-party threshold ecdsa from ecdsa assumptions. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 980–997. IEEE, 2018.
29. Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Threshold ecdsa from ecdsa assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1051–1066. IEEE, 2019.

30. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In *CRYPTO 2019*, pages 356–383, 2019.
31. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In *Advances in Cryptology – CRYPTO 2022*, volume 13508 of *LNCS*, pages 541–572. Springer, 2022.
32. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Shared permutation for syndrome decoding: New zero-knowledge protocol and code-based signature. *Designs, Codes and Cryptography*, 91(2):563–608, 2023.
33. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.
34. Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194, 2018.
35. Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1179–1194, New York, NY, USA, 2018. Association for Computing Machinery.
36. Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In *International Conference on Applied Cryptography and Network Security*, pages 156–174. Springer, 2016.
37. Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 354–371. Springer, 1996.
38. Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*, pages 295–310. Springer, 1999.
39. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20:51–83, 2007.
40. Shay Gueron, Edoardo Persichetti, and Paolo Santini. Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. *Cryptography*, 6(1):5, 2022.
41. Shay Gueron, Edoardo Persichetti, and Paolo Santini. Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. *Cryptography*, 6(1):5, 2022.
42. Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In Simone Severini and Fernando G. S. L. Brandão, editors, *TQC 2013*, volume 22 of *LIPIcs*, pages 20–34. Schloss Dagstuhl, 2013.
43. Yehuda Lindell. Fast secure two-party ecdsa signing. In *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II 37*, pages 613–644. Springer, 2017.
44. Yehuda Lindell. Fast secure two-party ecdsa signing. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017 - 37th Annual*

International Cryptology Conference, Proceedings, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 613–644. Springer Verlag, 2017. Publisher Copyright: © 2017, International Association for Cryptologic Research.; 37th Annual International Cryptology Conference, CRYPTO 2017 ; Conference date: 20-08-2017 Through 24-08-2017.

45. Qipeng Liu and Mark Zhandry. Revisiting post-quantum fiat-shamir. In *Advances in Cryptology - CRYPTO 2019*, pages 326–355, 2019.
46. Philip MacKenzie and Michael K Reiter. Two-party generation of dsa signatures. In *Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings 21*, pages 137–154. Springer, 2001.
47. Philip MacKenzie and Michael K Reiter. Two-party generation of dsa signatures. *International Journal of Information Security*, 2:218–239, 2004.
48. NIST. Post-Quantum Cryptography Standardization, 2017. URL: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`.
49. NIST. Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process, 2023. URL: `https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals`.
50. Robert Ransom. Constant-time verification for cut-and-choose-based signatures. Cryptology ePrint Archive, Paper 2020/1184, 2020. `https://eprint.iacr.org/2020/1184`.
51. Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. Hardness estimates of the code equivalence problem in the rank metric. *Cryptology ePrint Archive*, 2022.
52. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. NIST PQC Submission, 2020.

A Zero-Knowledge Proof for Action Equality

In the Distributed Key Generation given in Algorithm 1, we need a proof for the knowledge of a set element g_i such that the following relation holds:

$$y_i = g_i \star x \wedge x_i = g_i \star x_{i-1} \ .$$

The protocol presented below is a generalization of the one presented in Section 3.1 of [25], for a general group action.

Public Data : $x_a, x_b \in X$ and hash function $\mathtt{H}$.
Private Key : Group element $g \in G$.
Public Key : $y_a = g \star x_a$ and $y_b = g \star x_b$.

PROVER		**VERIFIER**
Choose $\tilde{g} \xleftarrow{\$} G$ and set: $\tilde{x}_a = \tilde{g} \star x_a$, $\tilde{x}_b = \tilde{g} \star x_b$.	$\xrightarrow{h}$	
Set $h = \mathtt{H}(\tilde{x}_a \| \tilde{x}_b))$.		
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0,1\}$.
If $b = 0$ then $u = \tilde{g}$.	$\xrightarrow{u}$	Accept if $\mathtt{H}(u \star x_a \| u \star x_b) = h$.
If $b = 1$ then $u = \tilde{g}g^{-1}$.		Accept if $\mathtt{H}(u \star y_a \| u \star y_b) = h$.

Fig. 4. Identification protocol to prove that the Private Key is used for the calculation.

Proposition 7. *The interactive proof in Figure 4 is correct, has soundness error $\frac{1}{2}$ and is computationally zero-knowledge assuming decisional-GAIP and the collision resistance of the hash function.*

Proof. The completeness of the protocol is straightforward. We need to prove soundness and zero knowledge.

- **Soundness:** suppose that the Prover is able to answer both the challenges with u_0 and u_1, by the collision resistance of the hash function at this point we would retrieve g as $u_1^{-1}u_0$ solving GAIP (Problem 1) and having that the public keys are generated by the same group elements.
- **Zero Knowledge:** to simulate the protocol without knowing the secret g the Prover flips a coin c. If $c = 0$, the Prover follows the protocol normally and is able to answer the challenge if $b = 0$. If $c = 1$, it computes $\bar{x}_a = \bar{g}y_a$ and $\bar{x}_b = \bar{g}y_b$ and sends them in place of $\tilde{x}_a$ and $\tilde{x}_b$. In this way it is able to answer to the challenge $b = 1$. Thus, if $c = b$ the prover can convince the verifier, otherwise it rewind the verifier and try again. Since at every iteration the prover has probability $\frac{1}{2}$ of guessing the correct c the simulation ends in expected polynomial time. Note that this transcript is indistinguishable from the honestly-obtained one, because the distribution of c is the same as the ones from the challenges and also $\bar{g}$, $\tilde{g}g^{-1}$ have the same distribution.

□

B Zero-Knowledge Proof for LESS Version

Since, in the literature, the signatures based on the Fiat-Shamir transformation start from a zero-knowledge identification protocol, we have included it here for completeness. Note that, at the moment, there is no clear framework for identification protocols executed with multiparty computations, and introducing one is out of the scope of this work.

Public Data : $q, n, k \in \mathbb{N}$, matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ and hash function $\mathtt{H}$.
Private Key : Invertible matrix $\hat{\boldsymbol{S}}_1 \hat{\boldsymbol{S}}_2' \in U \times U'$ and monomial matrix $\boldsymbol{Q}$.
Public Key : $\boldsymbol{G}' = \hat{\boldsymbol{S}}_1 \hat{\boldsymbol{S}}_2' \boldsymbol{G} \boldsymbol{Q}$.

PROVER		**VERIFIER**
Set $\tilde{\boldsymbol{G}} \leftarrow \boldsymbol{G}'$ and for $r = 1, 2$ do:		
get $\tilde{\boldsymbol{S}}_r \xleftarrow{\$} GL_k(\mathbb{F}_q)$ and $\tilde{\boldsymbol{Q}}_r \xleftarrow{\$} M_n(q)$	$\xrightarrow{h}$	
set $\tilde{\boldsymbol{G}} \leftarrow \tilde{\boldsymbol{S}}_r \tilde{\boldsymbol{G}} \tilde{\boldsymbol{Q}}_r$.		
Set $h = \mathtt{H}(\mathrm{SF}(\tilde{\boldsymbol{G}}))$.		
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0, 1\}$.
If $b = 0$ then $u \leftarrow \tilde{\boldsymbol{Q}}_1 \tilde{\boldsymbol{Q}}_2$		Accept if $\mathtt{H}(\mathrm{SF}(\boldsymbol{G}' u)) = h$.
If $b = 1$ then $\nu \leftarrow \boldsymbol{I}$.	$\xrightarrow{u}$	
for $r = 1, 2$ do :		
$\nu \leftarrow \tilde{\boldsymbol{S}}_r \cdot \nu \cdot \hat{\boldsymbol{S}}_i$.		
Use ν to retrieve the map and set $u \leftarrow \boldsymbol{Q} \tilde{\boldsymbol{Q}}$		Accept if $\mathtt{H}(\mathrm{SF}(\boldsymbol{G} u)) = h$.

Fig. 5. Identification protocol for the optimized version of LESS.

Proposition 8. *The interactive proof in Figure 5 is correct, has soundness error $\frac{1}{2}$ and is computationally zero-knowledge assuming decisional-GAIP and the collision resistance of the hash function.*

The protocol's soundness and zero-knowledge follow immediately from the original protocol, since the view of the protocol is unchanged. The completeness can be verified as in Proposition 5. Again, the protocol can be transformed into a signature scheme using the Fiat-Shamir.