Cutting the GRASS: Threshold GRoup Action Signature Schemes

Michele Battagliola[1,4], Giacomo Borin[3], Alessio Meneghetti[1], and Edoardo Persichetti[2]

[1] Università di Trento, Italy
[2] Florida Atlantic University, Boca Raton, USA and Sapienza University, Rome, Italy,
[3] IBM Research, Zurich and University of Zurich
[4] Università Politecnica delle Marche
`m.battagliola@staff.univpm.it`, `giacomo.borin@ibm.com`, `alessio.meneghetti@unitn.it`, `epersichetti@fau.edu`

Abstract. Group actions are 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, other group actions, such as isogeny-based ones, received interest from the cryptographic community, attracted by the possibility of translating old discrete logarithm-based functionalities.
Usually, research focuses on abelian group actions; however in this work we show that isomorphism problems which stem from non-abelian 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 two different flavors of code-based cryptographic group actions, respectively at the basis of the LESS and MEDS signature schemes, two of NIST's candidates in the recent call for post-quantum standardization.

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 [57] 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 [58]. 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 [24].

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 [4,5,2], with two of them expected to be added to the current list of standards (which, for KEMs, includes only Kyber [60]). 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 [50,42,41].

The work of LESS [19], which started in 2020 and continued with various follow-ups [7,8,9], 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 [30] 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 [27,18].

Non commutativity of code-based group actions has advantages from a security viewpoint since it prevents quantum attacks on commutative group actions like Kuperberg's algorithm for the dihedral hidden subgroup problem [52]. However this clearly reduce the possible cryptographic primitives based on them, for example we cannot build a Diffie-Hellmann like key exchange or use Linear Secret Sharing.

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 [46]. 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 [47,48]. Since 2001, several authors started working first on two-party variants of digital signatures [55,56] and then on ECDSA [37,53]. The first general

(T, N)-threshold scheme was proposed in 2016 [45], improved first in 2017 [21], and then again in 2018 [44]. In 2019, the work of [37] has been generalized by the same authors to the multi-party case [38]. 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 [14], where the key generation algorithm involves only 2 out the 3 parties.

As noted in [23], 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 [14] has successively been extended to a $(2, 3)$-threshold EdDSA instantiation [13]. In [22], 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 [13] suitable for Schnorr signatures has been proposed [11] and then generalized to a ZKP-based (T, N)-threshold Schnorr digital signature scheme whose key generation algorithm does not involve any trusted party [12].

Recently, driven by both the NIST call for Post-Quantum Standardization [57] and the call for Multi-Party Threshold Schemes [24], 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 [32], 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 schemes based on the Unbalanced Oil and Vinegar (UOV) framework 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 MPC protocols using Linear Secret Sharing Schemes (LSSS), 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 [36], where they proposed a way to apply a group actions in a threshold like way by using the classical Shamir Secret sharing on a group action induced by a cyclic group. They showed how to apply this for an El Gamal like encryption schemes and a signature based on Σ-protocols proving their simulatability, however this schemes are only secure in the honest-but-curious model and miss a distributed key generation mechanisms. In [33] they showed a way to combine the use of zero-knowledge proofs and replicated secret sharing to obtain a secure threshold signature scheme from isogeny assumptions. The work is an important step for the research and can be extended to more general group actions, but the main drawbacks are the number of shares necessary to implement replicated secret sharing and the important slow down caused by the additional ZKPs required. In [16] they showed how to define a distributed key generation algorithm by using a new primitive called *piecewise verifiable proofs*; proving their security in the quantum random oracle model. All previous techniques are then incorporated in [26] to have actively secure attributed based encryption and signature schemes, in which threshold signature are a particular case.

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 with a distributed key generation mechanism. The core idea is to split both the secret key and the ephemeral map as a product of N group elements, i.e. as $g = g_1 \cdots g_N$, so that thanks to this shared knowledge the users are able to prove the knowledge of secret the key. We then prove its security via a reduction to the original centralized signature[5] $\underline{\text{without}}$ relying on additional ZKPs during the signature phase, but instead relying on a securely generated salt. The details of the construction, as well as the security proof, are given in Section 3.

Bootstrapping using Replicated Secret Sharing. Our second contribution is the (T, N) version of scheme. 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.

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. We~~ conclude in Section 6.

[5] This is a generic signature scheme that is simply an abstraction, but has appeared in literature when instantiated in various works, such as LESS [8] and MEDS [30].

2 Preliminaries

We begin by laying down our notation. Throughout the paper we will denote with capital letters objects such as sets and groups, and with lowercase letters their elements. We will use instead boldface letters to denote vectors and matrices. We indicate by $\mathbb{F}_q$ the finite field of cardinality q, and by $\mathbb{F}_q^{k\times n}$ the set of $k \times n$ matrices with entries in $\mathbb{F}_q$; when $k = 1$, we write simply $\mathbb{F}_q^n$, which denotes the corresponding vector space over $\mathbb{F}_q$. Due to space constraints, we omit standard notions from coding theory; these are included, for completeness, in Appendix 1.

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.

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

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 $h \star (g \star x) = (h \cdot g) \star x$. The orbit of a set element is the set $\mathcal{O}(x) := \{g \star x \mid g \in G\}$. 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 such an element is known as the *vectorization* problem, or sometimes *Group Action Inverse Problem (GAIP).*

Problem 1 (GAIP)**.** *Given x and y in X, compute 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. Observe that GAIP is related to the one-wayness of the group action while the cGAGH is linked to its pseudorandomness, in fact requiring the hardness of the decisional version is implied by the following stronger definition:

Definition 1. *A group action is 2-weakly pseudorandom if no probabilistic polynomial time algorithm that given $(x, g \star x)$ can distinguish with non negligible probability between (x_1, y) and $(x_1, g \star x_1)$ with $x, x_1, y \xleftarrow{\$} X$ and $g \xleftarrow{\$} G$.*

Note that Definition 1 is a weaker assumption than the classical weak pseudorandomness from [3, Definition 3.6]. This new assumption is required since it was recently shown that many cryptographic group actions do not achieve the weakly pseudorandomness property, as per [34].

It is possible to obtain a signature scheme from cryptographic group actions, in full generality; a description is given in Appendix 2. A quick overview of code-based group actions can also be found in the appendix, namely, in Appendix 3.

2.2 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. We assume that each user has access to a secure, reliable and authenticated private channel with each of the other users, without worrying about specific design and peculiarities of the channel.

Usually, threshold signature protocols involve a key-generation protocol that constructs the key pair $(\mathsf{sk}, \mathsf{pk})$ as well as shares of the private key sk_i, and a multiparty signature protocol Thre.Sign, such that any set of T parties who agree on a common message m is able to compute a signature, which is verifiable against the public key via the procedure Verify. KeyGen can be executed by a trusted party or by the N parties alone collaborating. In this "decentralized" case, the parties get access to the additional exchanged information.

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. We refer as the *view* of a user as the probability distribution on the transcripts of all the data available to him during the execution of the multiparty protocol.

The main security property for threshold signature schemes is *Existential Unforgeability under Chosen Message Attacks (EUF-CMA)*:

Definition 2. *A threshold digital signature is secure in the EUF-CMA if for any probabilistic polynomial-time adversary* Evl *that is allowed to:*

1. *Corrupt $T-1$ out of N users;*
2. *Query a key generation oracle for the $T-1$ corrupted users shares and the public key* pk. *In the decentralized case it gets access also to the corrupted users view of* KeyGen *during the shared execution;*
3. *Perform a polynomial number of adaptive queries to a signing oracle that on chosen messages* m_i *obtaining the view of* $\mathsf{Thre.Sign}$*;*

it is not able to obtain a valid signature on a non queried message, i.e.

$$\mathit{Adv}_{CMA}^{\mathit{Evl}} = \mathbb{P}\left[\mathsf{DS.Verify}(\mathsf{pk}, \mathsf{m}^*, \sigma^*) = 1 \,\middle|\, \begin{matrix} \mathsf{m}^*, \sigma^* \leftarrow \mathit{Evl}\ , \\ \mathsf{m}^* \neq \mathsf{m}_i\ \forall i\ . \end{matrix}\right] \leqslant \mathsf{negl}(\lambda) \qquad (1)$$

Informally, the idea is that less than T views cannot be combined to obtain a valid signature.

3 The Full Scheme

We start our analysis with the *full threshold* cases, in which all the users are required to produce a signature (i.e. $T = N$).

Decentralized Key Generation Algorithm The goal of this protocol is to produce a common public key $y = g \star x$ with $g = g_1 \cdot \ldots \cdot g_N$, where each party holds one g_i, in the same way of [10,33]. To do so the users sequentially apply a previously committed random group element to the origin x and add the non-interactive Zero-Knowledge proof from [33] (see it also in Figure 4, in the appendix) to show the freshness of the group element. The resulting protocol is shown in Algorithm 1. At line 5, the Zero-Knowledge Proof is sent and tested by the other parties; the protocol is trusted by all of them if and only if all the ZKPs are valid. The main difference with [33] is that our scheme is specialized for non-abelian group actions and we are able to prove the security with only one ZKP per user, compared to the two required by [33].

Algorithm 1 KeyGen

```
Require: $x \in X$ origin.
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 a commitment to it $\mathsf{com}_i^{\mathsf{KG}}$.
2: Set $x_0 = x$.
3: for $i = 1$ to $N$ do
4:     $P_i$ computes $x_i = g_i \star x_{i-1}$
5:     $P_i$ publishes a ZKP as in Appendix 4 proving the consistence of $x_i$ with the
       commitment $\mathsf{com}_i^{\mathsf{KG}}$.
6:     $P_i$ sends $x_i$ to $P_{i+1}$ (if $i < N$)
7: return $y = x_N$. The private key of $P_i$ is $g_i$.
```

A relevant limitation, for the proposed protocol, is that each user P_i needs to receive the set element x_{i-1} by P_{i-1} before starting its computations. Thus, as explained in [36], it is necessary to adopt a *sequential round-robin* communication structure that makes it impossible to parallelize the algorithm; this results in a slowing of the execution time. Moreover, the users need to agree on a precise execution order at the start.

The initial commitment $\mathsf{com}_i^{\mathtt{KG}}$ depends on the group action being pseudo-random or not. More details about it in Appendix 4.

Signing Algorithm The signing protocol generalizes the one presented in [33,36] for non-abelian group actions, by computing the commitment and response phase of the protocol in Figure 3 in a multiparty setting.

In the commitment phase, each user P_i receives x_{i-1}^j, computes $x_i^j = \tilde{g}_i^j \star x_{i-1}^j$ for random $\tilde{g}_i$ and outputs it. During the response phase (lines 17,18) P_i get u_{i-1}^j and outputs $u_i^j = \tilde{g}_i^j u_{i-1}^j g_i^{-\mathsf{ch}_j}$. In line 19 for the challenge $\mathsf{ch}_j = 0$ the parties verify that $\tilde{x}_i^j = u_i^j \star x$, while in the other case they check $\tilde{x}_i^j = u_i^j \star x_i$.
The idea of this multiparty protocol is illustrated in Figure 1.

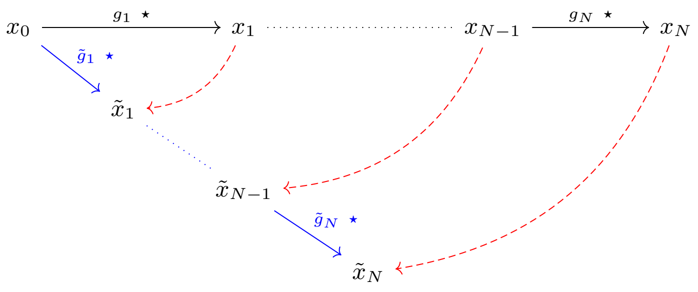

Fig. 1: Scheme representing the idea behind the protocol in Algorithm 2. In blue are the ephemeral group elements revealed on $\mathsf{ch} = 0$, while in red the map reconstructed for $\mathsf{ch} = 1$.

A detailed description of the algorithm is given in Algorithm 2. We also include the verification algorithm Algorithm 3, which is the same as the centralized one, for completeness.

A key feature of Algorithm 2, with respect to the previous literature, is the use of secure salt during the challenge evaluation (line 11), a technique used also in [28]. The salt is crucial to reduce the number of ZKPs in the signing protocol while maintaining security in the presence of malicious users. Indeed, without the salt verification, the scheme can be attacked by a *malicious* adversary opening several concurrent sessions. Intuitively, suppose that the adversary is in control of the N-th user and wants to sign the message m for the public key $y = g \star x$, knowing only g_N. He can proceed in the following way:

1. The adversary starts λ signing sessions for any messages m_1,...,m_λ.

Algorithm 2 Thre.Sign

```
Require: $x \in X$, security parameter $\lambda$, hash function $\mathsf{H}$, public key $(x, y = g \star x)$, secure commitment scheme $\mathsf{COM}$. The party $P_i$ knows the (multiplicative) share $g_i$ of $g = g_1 \cdots g_N$.
Ensure: A valid signature for the message $\mathsf{m}$ under the public key $(x, y)$.
 1: $P_1$ set $x_0^j = x$ for all $j = 1$ to $\lambda$          ▷ Shared commitment generation phase
 2: for $i = 1$ to $N$ do
 3:     Each party pick $\mathsf{salt}_i$ uniformly random and sends $\mathsf{COM}(\mathsf{salt}_i)$
 4: for $i = 1$ to $N$ do
 5:     If $i > 1$ $P_i$ receives $x_{i-1}^j$ from $P_{i-1}$ for all $j = 1$ to $\lambda$
 6:     for $j = 1$ to $\lambda$ do
 7:         $P_i$ chooses $\tilde{g}_i^j \in G$ and computes $x_i^j = \tilde{g}_i^j \star x_{i-1}^j$
 8:         $P_i$ outputs $x_i^j$;
 9: Set $x^j = x_N^j$ for all $j = 1$ to $\lambda$. Party $N$ broadcasts all $x^j$ to all players.
10: Each party publishes $\mathsf{salt}_i$ and checks the consistency of the received data with the initial commitment.
11: $\mathsf{salt} = \sum_i \mathsf{salt}_i$
12: Compute $\mathsf{ch} = \mathsf{H}(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m})$          ▷ Non-iterative challenges evaluation
13: $P_1$ set $u_0^j = e$ for all $j = 1$ to $\lambda$          ▷ Shared response generation phase
14: for $i = 1$ to $N$ do
15:     If $i > 1$ $P_i$ receives $u_{i-1}^j$ from $P_{i-1}$ for all $j = 1, ..., \lambda$
16:     for $j = 1$ to $\lambda$ do
17:         $P_i$ computes $u_i^j = \tilde{g}_i^j u_{i-1}^j g_i^{-\mathsf{ch}_j}$
18:         $P_i$ outputs $u_i^j$
19:         All users verify $u_i^j$ is valid;
20: $\mathsf{resp}_j = u_N^j$ for all $j = 1$ to $\lambda$
21: $\mathsf{sig} = \mathsf{ch} \| \mathsf{salt} \| \mathsf{rsp}_1 \| ... \| \mathsf{rsp}_\lambda$
```

2. For every session s, he receives by P_{N-1} $x^1_{N-1}, ..., x^\lambda_{N-1}$. At this point he evaluates $x^1_N = \tilde{g}^1_N \star x^1_{N-1}$ for each session s as described in the protocol. Let us call this element $\hat{x}^s$ for each session.
3. He evaluates the challenge $\mathsf{ch} = \mathsf{H}(\hat{x}^1 \| ... \| \hat{x}^\lambda \| \mathsf{m})$.
4. For each session s, the adversary then evaluates $x^2_N, ..., x^{\lambda-1}_N$ legitimately, then chooses $\tilde{g}^\lambda_N$ so that the first bit of $\mathsf{H}(x^1_N \| ... \| x^\lambda_N \| \mathsf{m}_i)$ is equal to the s-th bit of ch. This would not be possible if we had a secure salt.
5. Finally, the adversary closes all the concurrent sessions obtaining, for the session s, the response u^1_{N-1} received from P_{N-1}, which is used to evaluate rsp_1. This can be used to answer ch_s and obtain a valid signature $\mathsf{ch} \| \hat{\mathsf{rsp}}_1 \| ... \| \hat{\mathsf{rsp}}_\lambda$.

Algorithm 3 Verify

```
Require: $x \in X$, security parameter $\lambda$, hash function $\mathsf{H}$, public key $(x, y = g \star x)$.
Ensure: Accept if the signature for the message $\mathsf{m}$ is valid under the public key $(x, y)$.
1: Parse $\mathsf{ch}, \mathsf{salt}, \mathsf{rsp}_1, ..., \mathsf{rsp}_\lambda$ from $\mathsf{sig}$
2: for $j = 1$ to $\lambda$ do
3:     if $\mathsf{ch}_j = 0$ then
4:         set $\hat{x}^j = \mathsf{rsp}_j \star x$
5:     else
6:         set $\hat{x}^j = \mathsf{rsp}_j \star y$
7: Accept if $\mathsf{ch} = \mathsf{H}(\hat{x}^1 \| ... \| \hat{x}^\lambda \| \mathsf{salt} \| \mathsf{m})$
```

3.1 Security Proof

Theorem 1. *For a free group action (Definition 1), if the centralized signature is unforgeable in the quantum random oracle model, then the full-threshold signature scheme composed by* KeyGen, Thre.Sign *(Algorithms 1 and 2) and the verification* Verify *is EUF-CMA secure in the quantum random oracle model.*

Lemma 1. *For a 2-weakly pseudorandom free group action (Definition 1), the protocol* KeyGen *can be simulated in the quantum random oracle model in polynomial time so that any probabilistic polynomial-time adversary is convinced that the public key is any fixed pair $x, y \in X$.*

The main idea of the proof is to use the ZKPs to recover their secret shares and simulate a view of the protocol. Unlike [33], here we only have one ZKP for any user, thus we rely in rewinding the tape to change the set element sent in line 6. This proof works in the quantum random oracle model since the protocol in Figure 4 is a non-interactive zero-knowledge quantum proof of knowledge in the quantum random oracle for a free group action [16, Theorem 1] .

Algorithm 4 Sim.KeyGen (Simulation of KeyGen)

```
Require: $x, y \in X$, a non corrupted user $P_{i_0}$.
1: Send to Evl a random $x'_{i_0}$ generated from $x$ (as normal);
2: Checks all the ZKP for $i < i_0$ (as normal);
3: Send to Evl a random $x_{i_0}$;
4: Send a ZKP for $x_{i_0}$ and $x'_{i_0}$.
5: Continue the protocol and estranct $g_i$ from the ZKPs for all $i > i_0$;
6: Rewind the tape of the adversary up to the same state as in line 3;
7: Send $x_{i_0} = (g^{-1}_{i_0+1} \cdots g^{-1}_N) \star y$;
8: Simulate again ZKP for $x_{i_0}$ and $x'_{i_0}$.
9: The protocol is executed normally leading to $x, y$ as public key.
```

Proof of Lemma 1. Algorithm 4 shows the simulation strategy for a probabilistic polynomial-timeadversary Evl. We now need to prove that the simulation terminates in expected polynomial time, it is indistinguishable from a real execution, and outputs y.

The simulation terminates in polynomial time with non-negligible probability if also Evl is a probabilistic polynomial-time algorithm; in fact we have to carry over:

– one rewind of Evl in line 6;
– at most $N - 1$ extractions of secrets from the ZKPs, that can be carried over in polynomial time using the Forking Lemma ([15]) on the single ZKP. The probability for the adversary to fake the ZKP where a share does not exists is negligible, assuming the one-wayness of the group action.

Note that the rewinding can be performed since the adversary has already committed to the values g_i before the rewinding phase. In addition, thanks to the ZKPs, these group elements must exist, and the adversary is forced to apply them on $x_{i_0} = (g^{-1}_{i_0+1} \cdots g^{-1}_N) \star y$, so that the output of the simulation is the public key x, y as desired.

To send the crafted element x_{i_0} and simulate the ZKPs in lines 7 and 8, we need the 2-weakly pseudorandom property (Definition 1). This is because a common group element g_{i_0} such that $x'_{i_0} = g_{i_0} \star x \wedge x_{i_0} = g_{i_0} \star x_{i_0 - 1}$ does not exist anymore. The simulation can be carried over in the quatum random oracle since the protocol in Figure 4 is a non-interactive zero-knowledge quantum proof of knowledge (see Proposition 3). □

The proof of Theorem 1 follows the game-based argument proposed in [49, Theorem 3]. The key idea is to reduce the security of the full threshold signature to the security of the centralized one. We need 3 games (Algorithm 5), and we need to reprogram the random oracle, thanks to [49, Proposition 1].

Proof Theorem 1. Consider a probabilistic polynomial-time adversary Evl that make up to q_s sign queries and q_h quantum call to the random oracle H. By in Lemma 1 we can simulate the KeyGen on any public key x, y, so we will not discuss it here again.

Consider the games from Algorithm 5. Since the protocol $\mathsf{Thre.Sign}$ and KeyGen are executed in multiparty, if by any reason the protocol is aborted because of Evl misbehaviour, the game ends and returns 0.

Game G_0. This game is the same one played for the EUF-CMA security in Definition 2, thus $\mathbb{P}[G_0^{\mathsf{Evl}} \to 1] = \mathsf{Adv}_{CMA}^{\mathsf{Evl}}$ by definition.

Game G_1. In this game nothing is changed but we set ch at random and we reprogram the random oracle. We can observe that any statistical difference between the games can be used to build a distinguisher for the reprogramming of the oracle; in particular we can adapt the distinguisher from the proof of [49, Theorem 3]. In total, we reprogram the oracle q_s times (one for every signature) and Evl performs q_h quantum calls. Moreover, note that $x^1, ..., x^\lambda, \mathsf{m}$ are (at least partially) controlled by the adversary, while salt is randomly sampled thanks to the initial commitments and the secure aggregation. Thus, by [49, Proposition 1] we have:

$$|\mathbb{P}[G_0^{\mathsf{Evl}} \to 1] - \mathbb{P}[G_1^{\mathsf{Evl}} \to 1]| \leqslant \frac{3g_s}{2^{1+\lambda}}\sqrt{q_h} \tag{2}$$

Game G_2. First of all, note that during the computation of the response, it is possible to check whether the received u_i^j is correct or not, if the user $i+1$ saved all the x_i during the key generation step. We exploit this property in our simulation. Indeed, to simulate a signature, the simulator first acts honestly and follows the protocol. Upon receiving all the responses u_i^j of $P_1, ..., P_{i_0-1}$, it checks the correctness of all of them. If they are all correct, it rewinds the adversary up until receiving $\tilde{x}_{i_0-1}$ and chooses $\tilde{x}_{i_0}$ according to challenge ch_j (Figure 2 shows schematically of how the simulation strategy works). In particular:

- linking $\tilde{x}_{i_0-1}$ and $\tilde{x}_{i_0}$ on challenge $\mathsf{ch}_j = 0$;
- linking x_{i_0} and $\tilde{x}_{i_0}$ on challenge $\mathsf{ch}_j = 1$;

The idea is that every time the adversary acts honestly until P_{i_0}, the simulator produces an indistinguishable transcript that will not be rejected during the response computation. When, instead, the adversary sends something wrong before P_{i_0}, the simulation is perfect. Indeed, even if P_{i_0} is not able to answer to the challenge, the error spotted allows for an early abort and the simulation is indistinguishable.

We have shown that G_2 simulates the multiparty signature protocol $\mathsf{Thre.Sign}$, thus we need to bound the distance between the two last games. We are able to prove that the two views have the same distribution, implying null game distance.

If the simulator spots an error and aborts, the simulation is correct and indistinguishable from the real execution, since P_{i_0} followed the protocol normally. If the simulator rewind the adversary, then the view is given by salt_{i_0}, $x_{i_0}^j$, $\tilde{g}_{i_0}^j$ for all $j = 1, ..., \lambda$. The salt and the group elements are uniformly distributed

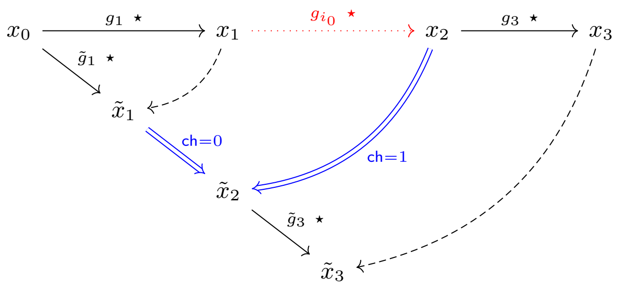

Fig. 2: Example of simulation for $N = 3$ and $i_0 = 2$, in red the missing link, while in blue the elements used to generate x_{i_0} and to answer the challenge.

both in the signature and in the simulation, so they are indistinguishable even for an unbounded adversary. Also for j with $\mathsf{ch}_j = 0$ the set elements $x^j_{i_0}$ are indistinguishable since the simulator is just following the protocol $\mathsf{Thre.Sign}$.

For j with $\mathsf{ch}_j = 1$ we consider the tuples $(\tilde{x}^j_{i_0-1}, \tilde{x}^j_{i_0})$ with $\tilde{x}^j_{i_0} = \tilde{g}^j_{i_0} \star \tilde{x}^j_{i_0-1}$ in the honest execution and $\tilde{x}^j_{i_0} = \tilde{g}^j_{i_0} \star x_{i_0}$ in the simulated ones.

We have rewound Evl, so we know that $\tilde{x}^j_{i_0-1} = u^j_{i_0-1} \star x_{i_0-1} \in \mathcal{O}(x_{i_0-1}) = \mathcal{O}(x)$. Since the group action is free, there exists a unique $\tilde{h}$ with $\tilde{x}^j_{i_0} = \tilde{h} \star \tilde{x}^j_{i_0-1}$. The element $\tilde{h}$ has the same distribution as $\tilde{g}^j_{i_0}$ thanks to the uniqueness of the solution; it follows that these pairs are again indistinguishable.

Finally, we observe that game G_2 is executed entirely without the use of the secret share g_{i_0}, thanks to the simulation, and so succeeding in the game implies being able to forge a signature for the centralized scheme in the quantum random oracle. Since we assumed quantum unforgebility for the centralized signature, this probability is negligible. Combining all the game distances we prove the desired reduction by the resulting equivalence:

$$\mathsf{Adv}^{\mathsf{Evl}}_{CMA} \leqslant \frac{3g_s}{2^{1+\lambda}}\sqrt{q_h} + \mathsf{negl}(\lambda)\ .$$

□

4 Threshold via Replicated Secret Sharing

In this section, we explain how to modify the full threshold scheme, to obtain a T-out-of-N scheme, via replicated secret sharing[6] [51]. Our approach was first proposed in [33].

Definition 3. *A* monotone access structure $\mathcal{A}$ *for the parties* $\mathcal{P} := \{P_1, .., P_N\}$ *is a family of subsets* $S \subset \mathcal{P}$ *that are authorized (to sign a message) such that*

[6] Unfortunately, while standard linear secret sharing would be more efficient, it is difficult to use in a non-abelian setting.

Algorithm 5 Threshold Signature Simulation

```
1: procedure GAMES $G_0 - G_1 - G_2$
2:     $\mathsf{Evl}$ chose at least a non corrupted user $P_{i_0}$;
3:     Execute $\mathsf{KeyGen}$ with $\mathsf{Evl}$;
4:     $\mathsf{m}^*, \sigma^* \leftarrow \mathsf{Evl}^{\mathsf{Sign}, |\mathsf{H}\rangle}$;
5:     return $\mathsf{Verify}((x, y), \sigma^*, \mathsf{m}^*) \wedge \mathsf{m}^* \notin S_M$.
6: procedure $\mathsf{Sign}(\mathsf{m})$
7:     $S_M \leftarrow S_M \cup \{\mathsf{m}\}$;
8:     Run $\mathsf{Thre.Sign}(\mathsf{m}, g_{i_0})$ up to line 11;          ▷ $G_0 - G_1$
9:     $\mathsf{ch} \leftarrow \mathsf{H}(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m})$;          ▷ $G_0$
10:    Get $\mathsf{ch} \xleftarrow{\$} \{0, 1\}^\lambda$;          ▷ $G_1$
11:    $\mathsf{H} \leftarrow \mathsf{H}^{(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m}) \mapsto \mathsf{ch}}$;          ▷ $G_1$
12:    Run $\mathsf{Thre.Sign}(\mathsf{m}, g_{i_0})$ to the end;          ▷ $G_0 - G_1$
13:    Run $\mathsf{Sim.Thre.Sign}(\mathsf{m})$;          ▷ $G_2$
14:    return $\mathsf{salt}_{i_0}, \tilde{x}^j_{i_0}, u^j_{i_0}$ for all $j$.
15: procedure SIM.$\mathsf{Thre.Sign}(\mathsf{m}, g_i$ for $i \neq i_0)$
16:    Run $\mathsf{Thre.Sign}(\mathsf{m}, g_{i_0})$ until line 15.
17:    Check all the $u^j_{i_0-1}$ received.
18:    if At least one $u^j_{i_0-1}$ is not correct then:
19:        return 0          ▷ Abortion in $\mathsf{Thre.Sign}$
20:    else
21:        Rewind $\mathsf{Evl}$ to line 4 after having received $x^j_{i_0-1}$
22:        for $j = 1, ..., \lambda$ do
23:            Get $\tilde{g}^j_{i_0} \leftarrow G$;
24:            Set $\tilde{g}^j_{i_0} \leftarrow \tilde{g}^j_{i_0} \cdot (g_N \cdots g_{i_0+1})^{-\mathsf{ch}_j}$;
25:            if $\mathsf{ch}_j = 0$ then
26:                output $x^j_{i_0} = \tilde{g}^j_{i_0} \star x^j_{i_0-1}$;
27:            else
28:                output $x^j_{i_0} = \tilde{g}^j_{i_0} \star y$;
29:        After receiving $x^j_N$, open $\mathsf{salt}_{i_0}$;
30:        if $\mathsf{salt}_i$ are correct then
31:            compute $\mathsf{salt} = \sum_i \mathsf{salt}_i$;
32:        else return 0          ▷ Abortion in $\mathsf{Thre.Sign}$
33:        $\mathsf{H} \leftarrow \mathsf{H}^{(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m}) \mapsto \mathsf{ch}}$;
34:        Output $u^j_{i_0} = \tilde{g}^j_{i_0}$ for all $j$;
```

given any $S \in \mathcal{A}$ and $S' \supset S$ then $S' \in \mathcal{A}$. To each access structure we can associate a family of unqualified sets $\mathcal{U}$ that satisfies that for all $S \in \mathcal{A}$, $U \in \mathcal{U}$ then $S \cap U = \varnothing$. For all the section we will define the unqualified sets in the canonical way as $\mathcal{U} = 2^{\mathcal{P}} \backslash \mathcal{A}$.

If we want to share a secret s in a group G for a monotone access structure $\mathcal{A}$, we need to consider the family $\mathcal{U}^+$ of the maximal unqualified set with respect to inclusion and define $\mathcal{I}$ as the family of complements for $\mathcal{U}^+$, i.e.

$$\mathcal{I} := \{I \in \mathcal{A} \mid \forall U \in \mathcal{U} \,.\, U \supseteq \mathcal{P} \backslash I \implies U = \mathcal{P} \backslash I\} \ .$$

Having fixed $M = \#\mathcal{I}$, we sort the elements in $\mathcal{I}$ as $I_1, I_2, \ldots$ and for each $l \in \{1, \ldots, M\}$ we define the shares s_l so that $s = s_1 \cdots s_M$; each party P_i is then given access to s_l if and only if $I_l \ni i$. This leads to the following (already known) result.

Proposition 1. *Any authorized subset $J \in \mathcal{A}$ of users can get the secret s, whilst any non-authorized set $A \in \mathcal{U}$ of users cannot retrieve at least one share.*

Proof. We prove that it is possible to recover the share by proving that any share s_I for $I \in \mathcal{I}$ is known by at least one user in J. In fact, suppose that there exists $I \in \mathcal{I}$ so that no user in J has access to it. This means that $I \not\ni P_i$ for all $P_i \in J$, so we have $I \cap J = \varnothing$. This implies that $S \subseteq I^c$. Since $\mathcal{A}$ is monotone, we have $I^c \in \mathcal{A}$, but I^c lies also in $\mathcal{U}^+$ (because of the definition of $\mathcal{I}$), so $I^c \in \mathcal{A} \cap \mathcal{U}$, which is impossible due to Definition 3.

For any $A \in \mathcal{U}$, we know that there exists a maximal element $B \in \mathcal{U}^+$ such that $B \supseteq A$. This implies $B^c \subseteq A^c$ and $B^c \cap A = \varnothing$. In addition, we have that $B^c \in \mathcal{I}$ by definition, but no $P_i \in A$ can have access to s_{B^c} since otherwise there would be an intersection. $\square$

By using this proposition, the parties in the authorized set J can recover the secret just by agreeing on which one of them should be the one sharing each share, i.e. by agreeing on a turn function $\tau(J, i)$ such that $\tau(J, i) \in I_i$ (i.e. $P_{\tau(J,i)}$ knows I_i).

For the T-out-of-N scheme, the authorized sets are the ones having cardinality at least T. In this way, $\mathcal{U}^+$ are all the subsets with at most $T-1$ element, $\mathcal{I}$ the ones of cardinality $N - T + 1$ and $M = \#\mathcal{I} = \binom{N}{T-1}$. The final protocol is depicted in Algorithm 6.

Distributed key generation. The distributed key generation protocol in Algorithm 1 can be used also in this threshold case. The central point is that during the generation each share g_i is known to several users, so to apply it on x_{i-1} they can:

1. jointly generate a shard of it and then combine the shard, essentially repeating a protocol similar to the key generation;

Algorithm 6 $\mathsf{Thre.Sign}_{T,N}$

```
Require: $x \in X$, a security parameter $\lambda$, a hash function $\mathsf{H}$, a public key $(x, y = g \star x)$,
    a secure commitment scheme $\mathsf{COM}$, a set $J$ of $T$ parties and the turn function $\tau$.
    Observe that the party $P_i$ knows all the (multiplicative) shares $g_{I_j}$ of $g = g_{I_1} \cdots g_{I_N}$
    so that $I_j \ni i$.
Ensure: A valid signature for the message $\mathsf{m}$ under the public key $(x, y)$.
 1: for $t \in J$ do
 2:     $P_t$ pick $\mathsf{salt}_t$ uniformly random and sends $\mathsf{COM}(\mathsf{salt}_t)$
 3: $P_{\tau(J,1)}$ set $x_0^j = x$ for all $j = 1$ to $\lambda$        ▷ Shared commitment generation phase
 4: for $i = 1$ to $M$ do
 5:     If $i > 1$ $P_{\tau(J,i)}$ receives $x_{i-1}^j$ from $P_{\tau(J,i-1)}$ for all $j = 1$ to $\lambda$
 6:     for $j = 1$ to $\lambda$ do
 7:         $P_{\tau(J,i)}$ chooses $\tilde{g}_i^j \in G$ and computes $x_i^j = \tilde{g}_i^j \star x_{i-1}^j$
 8:         $P_i$ outputs $x_i^j$
 9: Set $x^j = x_N^j$ for all $j = 1$ to $\lambda$. Party $\tau(J, N)$ broadcast all $x^j$ to all players.
10: Each party publish $\mathsf{salt}_t$ and checks the consistency of the received data with the
    initial commitment.
11: $\mathsf{salt} = \sum_t \mathsf{salt}_t$
12: Compute $\mathsf{ch} = \mathsf{H}(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m})$        ▷ Non-iterative challenges evaluation
13: $P_{\tau(J,1)}$ set $u_0^j = e$ for all $j = 1$ to $\lambda$        ▷ Shared response generation phase
14: for $i = 1$ to $M$ do
15:     If $i > 1$ $P_{\tau(J,i)}$ receives $u_{i-1}^j$ from $P_{\tau(J,i-1)}$ for all $j = 1, ..., \lambda$
16:     for $j = 1$ to $\lambda$ do
17:         $P_{\tau(J,i)}$ computes $u_i^j = \tilde{g}_i^j u_{i-1}^j g_i^{-\mathsf{ch}_j}$
18:         $P_{\tau(J,i)}$ outputs $u_i^j$
19:         All users verify $u_i^j$ is valid;
20: $\mathsf{resp}_j = u_N^j$ for all $j = 1$ to $\lambda$
21: $\mathsf{sig} = \mathsf{ch} \| \mathsf{salt} \| \mathsf{rsp}_1 \| ... \| \mathsf{rsp}_\lambda$
```

2. delegate one of the users that should know a share to apply it; said user can then share it with the others.

We prefer the second option since it has a lower latency for the non-abelian case, but still achieves the same security, assuming that all the users take part to at least one generation round, thanks to the zero-knowledge proofs.

The signature algorithm is also performed in the same way as the full threshold scheme, using the turn function τ to determine which party sends which messages at each round. 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.

Theorem 2. *For a 2-weakly pseudorandom free group action, if the centralized signature is unforgeable in the quantum random oracle model, then the (T, N)-threshold signature scheme composed by* KeyGen, $\mathsf{Thre.Sign}_{T,N}$ *adjoined* *with replicated secret sharing and the verification* Verify *is EUF-CMA secure in the quantum random oracle model.*

Sketch. The proof is very similar to that of the full threshold case (Theorem 1). First of all, note that, since the adversary controls at most $T-1$ players, there must be at least a set $I_{\mathsf{ho}} \in \mathcal{I}$ composed only by honest players on which the adversary has no control, as showed in the proof of Proposition 1. Thus we just use the strategies from Algorithm 4 and Algorithm 5 using as non corrupted user $P_{\tau(J,\mathsf{ho})}$. □

Usability of replicated secret sharing. The main drawback of replicated secret sharing is that the number of shares grows proportionally to the cardinality of $\mathcal{U}^+$, which is usually exponential in the number of parties. In particular, in the threshold case, there are $\binom{N}{T-1}$ shares in total, and each party needs to save $\binom{N}{T}$ shares. Since the group is non-abelian, the number of rounds cannot be reduced and is equal to the total number of shares.

All of this means that 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. Nevertheless, we would like to point out that for the most used combinations of (T, N) such as $(2, 3)$ or $(3, 5)$, the number of shares (and rounds) is manageable and the protocol maintains an acceptable level of efficiency.

5 Concrete Instantiations

In this section, we show how several optimizations used in literature for generic group actions can also be used for this multiparty protocol. We will then present concrete instantiations of our protocols, based on the LESS and MEDS signature schemes [8,30], and discuss tailored optimizations. We will denote by ξ the bit-weight of an element of X, and γ to denote that of an element of G.

5.1 Multi-bit Challenges

Multi-bit challenges are a way to reduce the computational time at the price of bigger keys and are widely used in signature design (e.g. [35]). In a nutshell, the optimization consists of replacing the binary challenge space of the verifier with one of cardinality $r > 1$, where each challenge value corresponds to a different public key. Note that the case $r = 2$ corresponds to the original protocol. In this way, it is possible to amplify soundness, at the cost of an increase in public key size. Security is then based on a new problem:

Problem 3 (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 $x_{j'} = g \star x_j$.

It is folklore that this problem is equivalent to the one-wayness of the group action, e.g. see Theorem 3 from [8]. We can then consider $r-1$ public keys $x_1, ..., x_{r-1}$ generated from the initial element x_0 by $r-1$ shared keys $g^{(1)}, ..., g^{(r-1)}$ (with the notation $g^{(0)} = e$). At this point the challenge is generated as an integer $\mathtt{ch} \in \{0, ..., r-1\}$, thus to evaluate the response (line 17) P_i computes $u_i^j = \tilde{g}_i^j u_{i-1}^j (g_i^{(\mathsf{ch}_j)})^{-1}$. As mentioned above, the soundness error is reduced to r^{-1}, thus in the signing algorithm we only need to execute $\lceil \frac{\lambda}{\log_2(r)} \rceil$ rounds, reducing both signature size and computational cost, but increasing the public key size.

5.2 Fixed-weight challenges

Another possible optimization is to use fixed-weight challenge strings, as shown for instance in [17,8]. Indeed, while $\mathtt{ch} = 1$ requires to send a group element, in the case $\mathtt{ch} = 0$ the Prover can simply send the PRNG seed used to generate the random group element $\tilde{g}$. This consists usually of only λ bits, thus is usually much shorter than a a group element. To exploit this, we can use a hash function H that returns a vector of fixed weight ω and length t.

To avoid a security loss we need to have a *preimage security* (the difficulty of guessing in the challenge space) of still λ bits, thus t, ω are such that: $\binom{t}{\omega} \geqslant 2^\lambda$. In this way, for carefully selected parameters, we can obtain shorter signature size at the price of an higher number of rounds.

To further reduce the signature size, it is possible to send multiple seeds at the same time by using a *seed tree*. This primitive uses a secret master seed to generate t seeds recursively exploiting a binary structure: 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 [50,30]. In [28], the author noted that, to avoid collisions attacks, a fresh salt should be used in combination of the seed tree structure. Since salt_i is already needed to achieve the security of the threshold construction, the parties could use it also for the PRNG call.

Applying this optimization to a threshold signature is not straightforward and requires particular parameters to be used. Indeed, the parties can not 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 $M \cdot \lambda$. So, for this strategy to make sense, we need $M\lambda$ to be smaller than the weight of the group element. Moreover, in some applications, it can be desirable to not disclose the parameters T and N, and thus the fixed-weight challenge should not be used.

5.3 Scheme Parameters

When the two approaches are combined, the final signature weight result is $(N_{\mathsf{seeds}}M + 2)\lambda + \omega\gamma + t$ with t the number of rounds ($\#\mathsf{rounds}$) satisfying

$$\binom{t}{\omega}(r-1)^{\omega} \geqslant 2^{\lambda} \ .$$

In our signing algorithm, for each of the $\binom{N}{T-1}$ 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,
- $\#\mathsf{rounds} \cdot \gamma + 2\lambda$ bits in general and $(N_{\mathsf{seeds}}M + 2)\lambda + \omega\gamma$ when using fixed-weight challenges.

At this point, we can see specific choices for LESS and MEDS. In our analysis, we choose 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.

Instantiations with LESS. From [6] we have taken the secure balanced LESS parameters for the NIST Security Category 1 $n = 252, k = 126$ (length and dimension of the code), $q = 127$ (the field size). We obtain that the size of a single code in systematic form is given by $(n-k)k\lceil\log_2(q)\rceil$ bits, so $\xi = 13.7\mathsf{KiB}$. Instead, to send a monomial map, we can use the IS-LEP technique from [59]. This recent optimization requires the use of a new canonical representation of the generator matrices via information sets. In this way, the equality can be verified using only the monomial map, truncated on the preimage of the information set, thus nearly halving the communication cost to $k(\lceil\log_2(q-1)\rceil + \lceil\log_2(n)\rceil)$ bits for each group element. This optimization (and any other possible new optimization based leveraging modified canonical forms, such as [31]) can be used also for the threshold protocol since:

- for the commitment phase, the last user can simply commit using the modified canonical form, then store the additional information received (the information set used);
- for the response phase, when the monomial map $g^{-1}\tilde{g}$ is recovered, it can be truncated again by the last user by using the additional information from the commitment.

For the cases in which fixed-weight cannot be used, we simply send all the truncated monomial maps. In this case, we can cut the signature size without enlarging too much the public key, by decreasing the code dimension to $k = 50$. Clearly, this requires to increase the code length up to $n = 440$ for $q = 127$ leading to a public key size of $17.1\mathsf{KiB}$ and truncated monomial map size of $100\mathsf{B}$. Numbers are reported in Table 1, where we report, in the last column, also the total amount of exchanged data.

Case	Variant	t	ω	|pk| (KiB)	|sig| (KiB)	Exc. (MiB)
centralized	Fixed	247	30	13.7	8.4	-
(2,3)	Fixed	333	26	13.7	10.59	13.30
(3,5)	Fixed	333	26	13.7	21.09	44.43
(N,T)	$[440, 50]_{127}$	-	-	16.68	12.55	$\binom{N}{T-1}2.19$

Table 1: Parameters for the threshold version of LESS

Instantiations with MEDS. From [29] we have taken the secure parameters for the matrix code equivalence problem: $n = m = k = 14$ (matrix sizes and dimension of the code), $q = 4093$ (the field size). Thus we obtain that the size of a single code in systematic form is given by $(nm - k)k\lceil\log_2(q)\rceil$ bits, so $\xi = 3.84$KiB. Observe that in the distributed key generation case we cannot use the public key compression mechanism from [30, Section 5]. A group element is instead composed by two invertible matrices, so it has size $(n^2 + m^2)\lceil\log_2(q)\rceil$ bits and we have $\gamma = 588$B.

Numbers are reported in Table 2; as above, in the last column we report the total amount of exchanged data.

Case	Variant	t	ω	r	|pk| (KiB)	|sig| (KiB)	Exc. (MiB)
MEDS-13220	F+M	192	20	5	13.2	13.0	-
(2,3)	F+M	291	19	4	11.26	14.49	3.24
(3,5)	F+M	113	22	6	18.76	20.80	4.34
(*,*)	M	-	-	8	26.24	24.74	$\binom{N}{T-1}0.182$
[29, Section 8]	M	-	-	3	7.50	3.37	$\binom{N}{T-1}0.342$

Table 2: Parameters for the threshold version of MEDS

To reduce signature size, another compression technique for group elements is proposed in [29, Section 8], and we briefly recall it here. Consider two equivalent $[m \times n, k]$ matrix codes $\mathcal{C}$, $\mathcal{C}' = \mathbf{A}\mathcal{C}\mathbf{B}$; the core idea is that, using two pairs of independent codewords $(\mathbf{C}_i, \mathbf{C}'_i) \in \mathcal{C} \times \mathcal{C}'$ satisfying $\mathbf{A}\mathbf{C}_i\mathbf{B} = \mathbf{C}'_i$ for $i = 1, 2$, the two invertible matrices $\mathbf{A}, \mathbf{B}$ can be recovered in polynomial time just by solving the system:

$$\begin{cases} \mathbf{A}\mathbf{C}_1 = \mathbf{C}'_1\mathbf{B}^{-1} \\ \mathbf{A}\mathbf{C}_2 = \mathbf{C}'_2\mathbf{B}^{-1} \end{cases} . \tag{3}$$

Note that this is the same process used for key compression in [29, Section 3.2]. To see how it is implemented for the MEDS signature, it is enough to see [29, Section 8]; in here, instead, we propose a slightly less efficient version which is however more suitable for the multiparty calculations (in which the last user modifies its execution).

- **Commitment**: the last user generates via a public seed a full-rank matrix $\mathbf{R} \in \mathbb{F}_q^{2\times mn}$, i.e. random independent codewords, and takes two random codewords in the code received by the previous user. Finally he solves Equation (3) to get $\tilde{\mathbf{A}}_M, \tilde{\mathbf{B}}_M$ and evaluate the final code as usual.
- **Response**: At the end of the response phase, the last user has access (for each round) to $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ such that $\mathrm{SF}(\mathbf{G}_{\mathsf{ch}}(\tilde{\mathbf{A}}^\top \otimes \tilde{\mathbf{B}})) = \tilde{\mathbf{G}}$, thus from $\mathbf{R}$ he can find the two associated codewords that can be used to recover the group element as $\mathbf{R}(\tilde{\mathbf{A}}^\top \otimes \tilde{\mathbf{B}})^{-1}$. Since these codewords are in the code $\mathcal{C}_{\mathsf{ch}}$, they can be represented as linear combinations of the $\mathbf{G}_{\mathsf{ch}}$ rows, i.e. as a $2 \times k$ matrix $\mathbf{M}$ such that
$$\mathbf{R}(\tilde{\mathbf{A}}^\top \otimes \tilde{\mathbf{B}})^{-1} = \mathbf{M}\mathbf{G}_{\mathsf{ch}} \ .$$
From $\mathbf{M}$, the verifier can recover the group element as explained in [29, Section 8]; thus, the communication cost per round is cut down to $2k\lceil\log_2(q)\rceil$ bits.

Remark 1. Unlike the original optimization, in this case we do not know the change-of-basis matrix used in the public key, implying that:

- there are additional linear systems to be solved since we need to invert $(\tilde{\mathbf{A}}^\top \otimes \tilde{\mathbf{B}})$ and find $\mathbf{M}$;
- in the case $\mathsf{ch} = 0$, we cannot save space by sending only the seed used to sample the codewords. To be precise, we could send it together with the seeds used for the previous ephemeral elements, but in most cases it would not save space since seeds and $2 \times k$ matrices have comparable sizes.

5.4 Latency

Because of the *sequential round-robin* structure each party must wait for the previous one results to start its execution, both during the commitment and the response phase, thus increasing the latency of the protocol. Usually the most expensive computation is the group action evaluation, so we can estimate the latency per round as t group actions for the commitment phase and t group actions for the response, since each of the users (in particular the one responsible to publish u_i^j) verify the previous user responses. Thus, we have $2Mt$ group actions, where M is the number of shares equal to $\binom{N}{T-1}$ while t is the number of repetitions for the basic identification protocol. Note that these are already much less then the group actions estimated for Sashimi in Section 4.1 [33].

The latency can be lowered by observing that, if several consecutive rounds are assigned to the same user, the required group actions can be reduced to only one by previously multiplying the group elements, both during commitment and verification phase. For example, in the 2-out of-N case where for each user misses only one of the secret shares, we can always chose the turn function τ so that the rounds can be divided in two consecutive series assigned to the two parties, thus compressing the latency to just $4t$ group actions per user. An estimate and comparison of the latency for the different protocols can be seen in Table 3.

Per party:	Sashimi	T-LESS	T-MEDS
# gr. actions	55377	1332	1164
time	283 s	279 ms	230 ms

Table 3: Comparison of the estimated latency for the 2-out of-N case for Sashimi (from [33]) and the threshold version of LESS and MEDS proposed in this work. We assumed a latency per group actions of 0.21 ms for LESS [6] and 0.24 ms for MEDS [29].

6 Conclusions

We introduced a threshold signature scheme based on the Group Action Inverse Problem that is agnostic about which particular group action is used, and works without any further hypotheses. Our schemes are similar to well-known abelian group action threshold schemes such as the one presented in [33,36,26], 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 prove the security of the key generation algorithm using fewer ZKPs than in [33]. Differently from [36,26], we use the jointly generated salt to reduce the security of the scheme to that of the centralized one without relying on intensive use of ZKPs, cutting by a lot communication cost and overhead computations.

When instantiated, our proposed schemes benefit from optimizations in use, eventually adapted to the multiparty scenario, and 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.

7 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 authors acknowledge support from Ripple's University Blockchain Research Initiative. 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.
The core of this work is contained also in the second author's M.Sc. thesis.

References

1. M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT 2002*. Springer Berlin Heidelberg.
2. C. Aguilar Melchor, N. Aragon, S. Bettaieb, L. ïc Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, E. Persichetti, G. Zémor, and J. Bos. HQC. NIST PQC Submission, 2020.
3. N. Alamati, L. De Feo, H. Montgomery, and S. Patranabis. Cryptographic group actions and applications. In *ASIACRYPT 2020*. Springer.
4. M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, K. G. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C. J. Tjhai, M. Tomlinson, and W. Wang. Classic McEliece. NIST PQC Submission, 2020.
5. N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. Aguilar Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, G. Zémor, V. Vasseur, and S. Ghosh. BIKE. NIST PQC Submission, 2020.
6. M. Baldi, A. Barenghi, L. Beckwith, J.-F. Biasse, A. Esser, K. Gaj, K. Mohajerani, G. Pelosi, E. Persichetti, M.-J. O. Saarinen, P. Santini, and R. Wallace. Matrix equivalence digital signature. `https://www.less-project.com/LESS-2023-08-18.pdf`, 2023. Accessed: 2023-09-15.
7. A. Barenghi, J.-F. Biasse, T. Ngo, E. Persichetti, and P. Santini. Advanced signature functionalities from the code equivalence problem. Cryptology ePrint Archive, Paper 2022/710, 2022. `https://eprint.iacr.org/2022/710`.
8. A. Barenghi, J.-F. Biasse, E. Persichetti, and P. 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.
9. A. Barenghi, J.-F. Biasse, E. Persichetti, and P. Santini. On the computational hardness of the code equivalence problem in cryptography. *Advances in Mathematics of Communications*, 17(1):23–55, 2023.
10. A. Basso, G. Codogni, D. Connolly, L. De Feo, T. B. Fouotsa, G. M. Lido, T. Morrison, L. Panny, S. Patranabis, and B. Wesolowski. Supersingular curves you can trust. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 405–437. Springer, 2023.
11. M. Battagliola, A. Galli, R. Longo, and A. Meneghetti. A provably-unforgeable threshold schnorr signature with an offline recovery party. In *DLT2022 at Itasec 2022, CEUR Workshop Proceedings*, 2022.
12. M. Battagliola, R. Longo, and A. Meneghetti. Extensible decentralized secret sharing and application to schnorr signatures. preprint: `https://eprint.iacr.org/2022/1551`, 2022.
13. M. Battagliola, R. Longo, A. Meneghetti, and M. Sala. A provably-unforgeable threshold EdDSA with an offline recovery party. preprint: `https://arxiv.org/abs/2009.01631`, 2020.

14. M. Battagliola, R. Longo, A. Meneghetti, and M. Sala. Threshold ECDSA with an offline recovery party. *Mediterranean Journal of Mathematics*, 19(4), 2022.
15. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, page 390–399, New York, NY, USA, 2006. Association for Computing Machinery.
16. W. Beullens, L. Disson, R. Pedersen, and F. Vercauteren. Csi-rashi: distributed key generation for csidh. In *International Conference on Post-Quantum Cryptography*, pages 257–276. Springer, 2021.
17. W. Beullens, S. Katsumata, and F. 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`.
18. W. Beullens, T. Kleinjung, and F. Vercauteren. Csi-fish: efficient isogeny based signatures through class group computations. In *ASIACRYPT 2019*. Springer, 2019.
19. J.-F. Biasse, G. Micheli, E. Persichetti, and P. Santini. Less is more: Code-based signatures without syndromes. In *AFRICACRYPT 2020*. Springer International Publishing, 2020.
20. M. Bläser, Z. Chen, D. H. Duong, A. Joux, N. T. Nguyen, T. Plantard, Y. Qiao, W. Susilo, and G. Tang. On digital signatures based on isomorphism problems: Qrom security, ring signatures, and applications. *Cryptology ePrint Archive*, 2022.
21. D. Boneh, R. Gennaro, and S. 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.
22. C. Bonte, N. P. Smart, and T. Tanguy. Thresholdizing hasheddsa: Mpc to the rescue. *International Journal of Information Security*, 20:879 – 894, 2021.
23. L. T. A. N. Brandão and M. Davidson. Notes on threshold eddsa/schnorr signatures. Accessed: 2023-05-01.
24. L. T. A. N. Brandão, M. Davidson, and A. Vassilev. Nist roadmap toward criteria for threshold schemes for cryptographic primitives. Accessed: 2020-08-27.
25. A. Budroni, J.-J. Chi-Domínguez, G. D'Alconzo, A. J. D. Scala, and M. Kulkarni. Don't use it twice! solving relaxed linear code equivalence problems. Cryptology ePrint Archive, Paper 2024/244, 2024. `https://eprint.iacr.org/2024/244`.
26. F. Campos and P. Muth. On actively secure fine-grained access structures from isogeny assumptions. In *International Conference on Post-Quantum Cryptography*, pages 375–398. Springer, 2022.
27. W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. Csidh: an efficient post-quantum commutative group action. In *ASIACRYPT 2018*. Springer.
28. A. Chailloux. On the (in) security of optimized stern-like signature schemes. WCC, 2022.
29. T. Chou, R. Niederhagen, E. Persichetti, L. Ran, T. H. Randrianarisoa, K. Reijnders, S. Samardjiska, and M. Trimoska. Matrix equivalence digital signature. `https://meds-pqc.org/spec/MEDS-2023-05-31.pdf`, 2023. Accessed: 2023-09-12.
30. T. Chou, R. Niederhagen, E. Persichetti, T. H. Randrianarisoa, K. Reijnders, S. Samardjiska, and M. Trimoska. Take your meds: Digital signatures from matrix code equivalence. In *International Conference on Cryptology in Africa*. Springer, 2023.
31. T. Chou, E. Persichetti, and P. Santini. On linear equivalence, canonical forms, and digital signatures. `https://tungchou.github.io/papers/leq.pdf`, 2023. Accessed: 2023-09-20.

32. D. Cozzo and N. P. Smart. Sharing the luov: threshold post-quantum signatures. In *IMA International Conference on Cryptography and Coding*, pages 128–153. Springer, 2019.
33. D. Cozzo and N. P. Smart. Sashimi: Cutting up csi-fish secret keys to produce an actively secure distributed signing protocol. In J. Ding and J.-P. Tillich, editors, *Post-Quantum Cryptography*, pages 169–186, Cham, 2020. Springer International Publishing.
34. G. D'Alconzo and A. J. D. Scala. Representations of group actions and their applications in cryptography. Cryptology ePrint Archive, Paper 2023/1247, 2023.
35. L. De Feo and S. D. Galbraith. Seasign: compact isogeny signatures from class group actions. In *EUROCRYPT 2019*. Springer, 2019.
36. L. De Feo and M. Meyer. Threshold schemes from isogeny assumptions. In *PKC 2020*. Springer, 2020.
37. J. Doerner, Y. Kondi, E. Lee, and A. Shelat. Secure two-party threshold ecdsa from ecdsa assumptions. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 980–997. IEEE, 2018.
38. J. Doerner, Y. Kondi, E. Lee, and A. Shelat. Threshold ecdsa from ecdsa assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1051–1066. IEEE, 2019.
39. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In *CRYPTO 2019*, 2019.
40. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In *CRYPTO 20199*. Springer, 2019.
41. T. Feneuil, A. Joux, and M. Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In *CRYPTO 2022*. Springer, 2022.
42. T. Feneuil, A. Joux, and M. Rivain. Shared permutation for syndrome decoding: New zero-knowledge protocol and code-based signature. *Designs, Codes and Cryptography*, 91(2):563–608, 2023.
43. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO' 86*. Springer Berlin Heidelberg, 1987.
44. R. Gennaro and S. 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.
45. R. Gennaro, S. Goldfeder, and A. 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.
46. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold dss signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 354–371. Springer, 1996.
47. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT'99*. Springer, 1999.
48. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20:51–83, 2007.
49. A. B. Grilo, K. Hövelmanns, A. Hülsing, and C. Majenz. Tight adaptive reprogramming in the qrom. In *ASIACRYPT 2021*. Springer, 2021.
50. S. Gueron, E. Persichetti, and P. Santini. Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. *Cryptography*, 6(1):5, 2022.

51. M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan*, 1989.
52. G. Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In S. Severini and F. G. S. L. Brandão, editors, *TQC 2013*, volume 22 of *LIPIcs*. Schloss Dagstuhl, 2013.
53. Y. Lindell. Fast secure two-party ecdsa signing. In *CRYPTO 2017*. Springer, 2017.
54. Q. Liu and M. Zhandry. Revisiting post-quantum fiat-shamir. In *CRYPTO 2019*, 2019.
55. P. MacKenzie and M. K. Reiter. Two-party generation of dsa signatures. In *CRYPTO 2001*. Springer, 2001.
56. P. MacKenzie and M. K. Reiter. Two-party generation of dsa signatures. *International Journal of Information Security*, 2004.
57. NIST. Post-Quantum Cryptography Standardization, 2017. URL: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`.
58. 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`.
59. E. Persichetti and P. Santini. A new formulation of the linear equivalence problem and shorter less signatures. *Cryptology ePrint Archive*, 2023.
60. P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler, and D. Stehlé. CRYSTALS-KYBER. NIST PQC Submission, 2020.
61. D. Unruh. Post-quantum security of fiat-shamir. In *Advances in Cryptology–ASIACRYPT 2017*. Springer, 2017.

1 Coding Theory Notions

A linear code $\mathcal{C}$ is a vector subspace $\mathcal{C} \subseteq \mathbb{F}_q^n$ of dimension k, and it is usually referred to as an $[n, k]$ linear code. It follows that a basis for $\mathcal{C}$ 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 $\mathbf{G}$, this is known as a *generator matrix* for the code, as it can generate each vector of $\mathcal{C}$ (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 $\mathbf{S}$ 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 entries. It follows, then, that an *isometry* (i.e. a map preserving the weight) is given by any $n \times n$ permutation matrix $\mathbf{P}$ acting on each word, or indeed, on the columns of $\mathbf{G}$ (since every codeword can be generated as a linear combination of the rows of $\mathbf{G}$). 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 $\mathbf{Q}$; it can be seen as a product $\boldsymbol{D} \cdot \mathbf{P}$ 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 $\mathcal{C}$ 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 $\mathbf{A} \in \mathrm{GL}_m$ and $\mathbf{B} \in \mathrm{GL}_n$ 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 $\mathcal{C}$ and $\mathcal{C}'$ are *linearly equivalent* if $\mathcal{C}' = \mathcal{C}\mathbf{Q}$, and two matrix codes $\mathcal{C}$ and $\mathcal{C}'$ are *matrix equivalent* if $\mathcal{C}' = \mathbf{A}\mathcal{C}\mathbf{B}$. 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 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 3.

Public Data : Group G acting on X via $\star$, element $x \in X$ and hash function H.
Private Key : Group element g with $g_i \in G$.
Public Key : $y = g \star x$.

PROVER		**VERIFIER**
Get $\tilde{g} \xleftarrow{\$} G$, send $\mathsf{com} = \mathsf{H}(\tilde{g} \star x)$	$\xrightarrow{\mathsf{com}}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \xleftarrow{\$} \{0,1\}$.
If $\mathsf{ch} = 0$ then $\mathsf{rsp} \leftarrow \tilde{g}$.	$\xrightarrow{\mathsf{rsp}}$	Accept if $\mathsf{H}(\mathsf{rsp} \star x) = \mathsf{com}$.
If $\mathsf{ch} = 1$ then $\mathsf{rsp} \leftarrow \tilde{g}g^{-1}$.		Accept if $\mathsf{H}(\mathsf{rsp} \star y) = \mathsf{com}$.

Fig. 3: 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 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 [43], 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 [35,8]. For further discussions on Fiat-Shamir, and its security in the ROM and QROM, we point instead the reader to [43,1,39,54].

Proposition 2. *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 difficulty of GAIP.

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 $G = \mathsf{M}_n$, by which we denote the group of monomial matrices. Note that this group is isomorphic to $(\mathbb{F}_q^*)^n \rtimes \mathsf{S}_n$ if we decompose each monomial matrix $\mathbf{Q} \in \mathsf{M}_n$ into a product $\boldsymbol{D} \cdot \mathbf{P}$. The group operation can be then seen simply as multiplication, and the group action is given by

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

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

Remark 2. 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 $\mathbf{G} \in \mathbb{F}_q^{k\times mn}$. Then, the action of the group $G = \mathrm{GL}_m \times \mathrm{GL}_n$ on this set can be described compactly as follows:

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

Note that this is equivalent to applying the matrices $\mathbf{A}$ and $\mathbf{B}$ to each codeword $\mathbf{C}$ in the matrix code as $\mathbf{ACB}$; 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 [52].

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

Problem 4 (Linear Equivalence (LEP))**.** Given two k-dimensional linear codes $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$, find, if any, $\mathbf{Q} \in M_n$ such that $\mathcal{C}' = \mathcal{C}\mathbf{Q}$.

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 $\mathbf{Q}$ is a permutation.

Problem 5 (Matrix Code Equivalence (MCE))**.** Given two k-dimensional matrix codes $\mathcal{C}, \mathcal{C}'$, find, if any, $\mathbf{A} \in \mathrm{GL}_m, \mathbf{B} \in \mathrm{GL}_n$ such that $\mathcal{C}' = \mathbf{A}\mathcal{C}\mathbf{B}$.

Note that both of the above problems are traditionally formulated as decisional problems. Extensive discussion of their hardness is given, for instance, in [9,30].

4 Zero-Knowledge Proof for Action Equality

In the Distributed Key Generation given in Algorithm 1, we need that each party commitm to its onw shard g_i and then prove the the consistence of the commitment with the its output data. When the action is pseudorandom, a possible idea is to define $\mathsf{com}_i^{\mathtt{KG}}$ as $y_i = g_i \star x$ and then, when sending $x_i = g_i \star x_{i-1}$, 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 straightforward generalization of the one presented in Section 3.1 of [33], for a general group action.

Public Data : $x_a, x_b \in X$ and hash function 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 $\tilde{g} \overset{\$}{\leftarrow} G$ and set:		
$\tilde{x}_a = \tilde{g} \star x_a,\ \tilde{x}_b = \tilde{g} \star x_b.$	$\xrightarrow{\mathsf{com}}$	
Set $\mathsf{com} = \mathsf{H}(\tilde{x}_a \Vert \tilde{x}_b)$.		
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \overset{\$}{\leftarrow} \{0,1\}$.
If $\mathsf{ch} = 0$ then $\mathsf{rsp} = \tilde{g}$.	$\xrightarrow{\mathsf{rsp}}$	Accept if $\mathsf{H}(\mathsf{rsp} \star x_a \Vert \mathsf{rsp} \star x_b) = \mathsf{com}$.
If $\mathsf{ch} = 1$ then $\mathsf{rsp} = \tilde{g}g^{-1}$.		Accept if $\mathsf{H}(\mathsf{rsp} \star y_a \Vert \mathsf{rsp} \star y_b) = \mathsf{com}$.

Fig. 4: One round of the identification protocol prove that the Private Key is used for the calculation.

For completeness we report here the proof of security for the non interactive version of the protocol, contained in [33] and [16].

Proposition 3. *The protocol in Figure 4 can be rendered to a non interactive computationally zero-knowledge quantum proof of knowledge for a free 2-weakly pseudorandom group actions in the QROM.*

Proof. First we prove that the underlying protocol is complete, sound and computationally zero-knowledge. The completeness 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 $u_1^{-1}u_0$ against the one wayness of the group action (thus also against 2-weakly pseudorandomness) and having that the public keys are generated by the same group elements.
- **Zero Knowledge:** to simulate the protocol without knowing the secret g and for any pairs of elements (x_a, y_a), (x_b, y_b) 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 a distinguisher between the honestly generated transcripts and the simulated one can be used to distinguish pairs $(\bar{x}, g \star \bar{a})$ from random ones, against the 2-weakly pseudorandomness.

For the quantum resistance we can observe that since the automorphisms are all trivial the sigma protocol has perfect unique responses (see [20, Lemma 1]) then by [40, Theorem 25] the protocol is a quantum proof of knowledge. Then the protocol has completeness, high min entropy[7] and HVZK and is zero-knowledge against quantum adversaries thanks to [61]. □

4.1 Alternative Protocol Using Weaker Assumptions

The problem of the above protocol is that it requires a 2-weakly pseudorandom group action. It was recently shown that many commonly used non commutative group actions are not weakly pseudorandom, in particular for linear code equivalence with code rate 1/2 is not even 2-weakly pseudorandom, i.e. by having access to $(x, g \star x, y, g \star y)$ allows to recover g in polynomial time [25]. So for LESS based distributed key generation [8] we need a different and less efficient zero knowledge proof. The strategy is basically to prepare the whole proof before the key generation, committing to both the possible response. So instead of committing to a set element $y_i = g_i \star x$ they commit to the group elements used for the protocol hidden in the merkle tree root, opening only the relevant responses to perform the classical protocol from Figure 3 during the key generation phase.

[7] i.e. the probability of guessing the commitment is negligible

This way the user, against the binding property of the merkle tree, is forced to apply a fixed group element.

Formally, P_i samples uniformly the secret $g_i \in G$ and the ephemeral elements $\tilde{g}_k \in G$ for $k = 1, ..., \lambda$, then commits to

$$\mathsf{com}_i^{\mathsf{KG}} = \left(\mathsf{Merkle}\left((\tilde{g}_k)_{k=1,\dots,\lambda}\right), \mathsf{Merkle}\left((\tilde{g}_k g_i^{-1})_{k=1,\dots,\lambda}\right)\right) \tag{4}$$

where the function $\mathsf{Merkle}(\mathbf{v})$ computes the root of the Merkle Tree have as leafs the entries of the vector $\mathbf{v}$. Later P_i performs the protocol of Figure 5 to prove the knowledge of g_i.

```
Public Data : $x_{i-1}$, $x_i = g_i \star x_{i-1}$, $\mathsf{com}_i^{\mathsf{KG}}$ as in (4)
Private Key : $g_i \in G$ and $\tilde{g}_k \in G$ for all $k = 1, ..., \lambda$.

PROVER                                                   VERIFIER
for $k = 1, ..., \lambda$ do:
    $\tilde{x}_k \leftarrow \tilde{g}_k \star x_{i-1}$
Set $\mathsf{com} \leftarrow \tilde{x}_1 || ... || \tilde{x}_\lambda$      $\xrightarrow{\mathsf{com}}$
                                      $\xleftarrow{\mathsf{ch}}$            $\mathsf{ch} \xleftarrow{\$} \{0,1\}^\lambda$.
for $k = 1, ..., \lambda$ do:
    $\mathsf{rsp}_k \leftarrow \tilde{g}_k g_i^{-\mathsf{ch}_k}$.
    $\mathsf{cov}_k \leftarrow \mathsf{H}(\tilde{g}_k g_i^{\mathsf{ch}_k - 1})$.
$\mathsf{rsp} \leftarrow \mathsf{rsp}_1 || ... || \mathsf{rsp}_\lambda$
$\mathsf{cov} \leftarrow \mathsf{cov}_1 || ... || \mathsf{cov}_\lambda$     $\xrightarrow{\mathsf{rsp},\mathsf{cov}}$ Parse $\mathsf{rsp}$ and $\mathsf{cov}$
                                              Check $\mathsf{com}_i^{\mathsf{KG}}$ as in (4)
                                              for $k = 1, ..., \lambda$ do:
                                                  if $\mathsf{ch}_k = 0$ then:
                                                      Check $\mathsf{rsp}_k \star x_{i-1} = \tilde{x}_k$
                                                  else:
                                                      Check $\mathsf{rsp}_k \star x_i = \tilde{x}_k$
```

Fig. 5: Identification protocol for a non pseudorandom group action

Sketch of the Security Properties The completeness and zero knowledge property are trivial to prove, the simulator can simply adopt the strategy used for the previous protocol, given that the hash function H is hiding.

The soundness is more tricky. It is immediate to see that the protocol is sound for the GAIP relation (Problem 1) on (x_{i-1}, x_i) using the same arguments used for the protocol in Figure 3.

However, we also need now to show that, for a free group action, the witness is uniquely determined by $\mathsf{com}_i^{\mathsf{KG}}$. The problem is that the protocol does not ensure that all the leaf of root_i are correctly formed, indeed suppose that an adversary cheat on one leaf, then probability of being detected is only $\frac{1}{2}$, since half of the time that leaf is not open. What the protocol ensure is that, against the binding properties of the hash function H, all the group elements involved in the in the proof are the same, so given two pairs (x_{i-1}, x_i) and (x^*_{i-1}, x^*_i) they share the same witness, since the extraction depends only on the group elements involved in the protocol.