
UNIVERSITÀ
DI TRENTO

DIPARTIMENTO DI MATEMATICA
Laurea Magistrale in Matematica

# *Cryptographic Group Actions and Digital Signatures,*

## *with a focus on Code Equivalence Problems*

Student:
Giacomo Borin

Supervisor:
Alessio Meneghetti

Co-Supervisors:
Edoardo Persichetti
Michele Battagliola

Academic Year 2022-23

In realms of ciphered lore, where secrets dwell,
In realms where quantum linger, truths to tell,
Signatures emerge, like echoes in the night,
Unveiling tales of servers, wrapped in cryptic light.

Cryptographic group actions, intricate and wise,
Weave the fabric of protection, where danger lies,
In graphs they dance, their steps a subtle art,
Guarding realms of knowledge, shielding every part.

Through sigma protocols, our stories intertwine,
In this realm of signatures, where the powers align,
In words simulated, whispered on the wind,
May our isomorphisms be safe, as destinies chime.

And lo, code equivalences, bound in sacred ties,
Unlock the paths to linear codes, where wisdom lies,
They bridge the worlds of algorithms and keys,
Unraveling the mysteries that lie beneath the bits.

Yet behold, threshold signatures, a potent spell,
Where trust is shared, secrets they compel,
A fellowship united, wielding strength untold,
In unity, they conquer, forging futures bold.

O Lord of Groups, in your poetic grace,
Let signatures guide us, in this digital space,
May zero knowledge shield us from the dark,
And preserve our identities, like an elven arc.

Thus, we venture forth, with code as our guide,
Seeking truth and efficiency, side by side,
In this tapestry of cryptographic might,
May our actions shine, forever in the light.

# CONTENTS

# INTRODUCTION

Thanks to the progressive digitalization any angle of the world is interconnected and more digital institution, services, assets are accessible each day. In this digital world the need for secure, reliable and flexible authentication mechanisms has become paramount. In particular digital signatures serve as a vital cryptographic tool to ensure data integrity, authenticity, and non-repudiation.

However, by using the 1994 Shor's algorithm [Sho94], conventional hard problems, factorization and discrete logarithm, used for digital signatures, can be broken in polynomial times using quantum computers.

Post-quantum cryptography tries to counter this by developing cryptographic schemes based on problems resistant to attacks by both classical and quantum computers.

**The NIST Calls** Recognizing the threat that quantum computers advancement represent for classical cryptography, the National Institute of Standards and Technology (NIST) opened a Post-Quantum Cryptography Standardization process in December 2016 [NIS17]. This pioneering effort aims to standardize a new suite of resistant algorithms that will secure the digital infrastructure in a post-quantum era. The process involved extensive research, public submissions, and evaluations by researchers, universities and private companies from all around the world.

After 5 years of proposal and discussions in the cryptographic community NIST has selected four schemes for standardization: three lattice based (1 KEM and 2 signatures) and one hash-based (signature) [NIS22b].

Even if lattices assumptions have been proven resilient and reliable, a secure post-quantum digital infrastructure should not rely only on them. NIST considers the selected schemes alone inadequate for the long term goal of standardization, so it is performing an additional fourth round focused on code-based Public-key Encryption [NIS22a; Ara+20; Agu+20; Alb+20] and has opened an "on ramp" call for signature schemes not based on ideal lattices [NIS23].

**Code-based Cryptography** Coding theory is a branch of mathematics merging algebra and information theory, with the goal of correcting errors happening

in telecommunications. First results on the discipline traces back to the 60's, with Shannon's pivotal work on capacity. Now coding theory applications are transversal, from classical noisy reduction to network coding [SK11], reliable data storage with NANDs or with DNA [EZ17], and also in digital security. Code-based cryptography is the second largest area of research and surely the one with best established cryptanalysis records. The first cryptosystem built on error-correcting codes is due to McElience, still undamaged in its first instantiation based on Goppa codes [Alb+20].

Even if generally secure and efficient, there are some drawbacks for this and other cryptosystems in the same family. They have by far the largest public key sizes and usually rely on trapdoors, that makes difficult the security evaluation of newly proposed schemes. A new approach started in [BMPS20] tries to solve these complications by exploiting the hardness of finding equivalence of codes instead of the decoding random ones. Code equivalence based signatures have best overall public key sizes, does not rely on trapdoors and the operations used can be modeled as group actions.

**Group Actions use for Cryptographic Purposes** Well known by group theorist and used in several branches of algebra (also with applications in theoretical physic), group actions are already in use in cryptography, even if without awareness, in relation to the discrete logarithm problem.

With the incoming threat of quantum computers over classical cryptography, group actions open the way to new secure and efficient primitives. Introduced to the community thanks to isogeny-based schemes, they immediately got traction thanks to the promise of getting back the flexibility of group based constructions.

Recently Code Equivalence and other Isomorphism Problems have also been (re)discoverd as an alternative way to define (non abelian) group actions suitable for cryptographic use, but relying on more established assumptions.

This triggered the interest of the community in studying cryptographic group actions within a more general framework. Not only to improve current schemes and understand their limits, but also to get back functionalities missing in the post-quantum scenario, like ring signatures [BKP20] and, in particular, threshold signatures.

**Threhsold Signatures** 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. Then, any subset of malicious users of size up to the threshold $T$ won't be able to forge a valid signature alone. 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, in fact, the verification algorithm is identical to that of an existing signature scheme, usually referred to as the "centralized" scheme.

The history of threshold signatures dates back to [GJKR96], that propose a scheme for the parameters $(T + 1, 2T + 1)$, and only in 2016 a proper general

$(T, N)$-threshold scheme was proposed in [GGN16].

Recently the design and the standardization of Multi-Party Threshold Schemes received much more interest, even from NIST ([BDV]), thanks to the multiple applications of these schemes. Threshold signatures have the potential to impact several crucial aspects of today's digital landscape, including:

- Enhancing more **fine-grained control of secret assets**: secrets or the rights over an asset can be shared and stored in multiple locations or by different users, avoiding single points of failure.
- Implementation by design of **Access Control** policies: both between administrators granting access or users identifying themselves.
- Thanks to the interchangeability of threshold signatures with centralized one they can achieve both **compatibility** with conventional schemes in use and **privacy** preserving properties.
- Secure **Multi-Party Computation**; the computations on shared data, preserving shares privacy, is closely related with threshold signatures. MPC techniques can be used to design shared signatures, but also these can be required to prove integrity during protocols multiparty executions.
- Implement **Distributed Ledger Technologies**: via threshold signatures multiple parties, such as members of a multi-signature wallet, can collectively sign transactions without exposing sensible information, allowing the creation of distributed wallets on any blockchain network. This is used also, for instance, by users willing to store the rights on their assets in multiple locations to avoid stealing of credentials.

**Post-Quantum Scenario and Group Actions** Recently, driven by both the NIST call for Post-Quantum Standardization [NIS17] and the call for Multi-Party Threshold Schemes [BDV], 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 rely on the difficulty of the Discrete Logarithm Problem, new methods for post-quantum cryptography have to be investigated. In [CS19], the (round 2) proposals (particularly the UOV ones) of the standardization process were analyzed in order to determine ways to define threshold variants, principally exploiting LSSS-based MPC protocols, however, at the present time, this approach remains only theoretical.

Instead, a solution has been found in 2020 for cryptographic *cyclic* group actions and applied to isogeny-based schemes [DM20]. We give here a brief recap of the results that followed that paper.

- In [DM20] 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 [CS20] 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 rounds necessary to implement replicated secret sharing and the important slow down caused by the addition ZKP required.

– In [BDPV21] 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.

– In [CM22] they incorporate the techniques from [CS20; BDPV21; DM20] to have actively secure attributed based encryption and signature schemes, in which threshold signature are a particular case. Again the main drawbacks are the number of ZKPs used.

All these constructions rely on commutative group actions with peculiar structures, so essentially on isogeny based schemes like [DG19; BKV19], that are not under submission to the NIST call [NIS23]. Thus it makes sense to focus on more general group actions, to obtain threshold schemes compatible with signatures based on isomorphism problems submitted to the call [Bal+23b; Cho+23; Bla+23].

In [BBMP23] they leveraged previous ideas to apply this constructions also to the case of non-abelian group actions avoiding excessive us of ZKPs. This last preprint is a result of my thesis work, thus Chapter 4 can be seen as detailed and extended version of that article.

## Contributions and Future Directions

As explained, group actions are a growing trend in post quantum cryptography, so here we give a detailed overview of all the mathematics requirements necessary to instantiate a secure post-quantum digital signature. To do that we bridge between established isogeny based literature and more recent one relative to isomorphism problems.

When going through the optimizations we insert here a new model for the base identification protocol, which uses a general graph. The new framework comprehends both established ones, like the fixed-weight, and new ones, based on multiparty computations, hypercube shaped commitments and faster signature verification. Some of the last ones have also been published, as preprints, in [BPS23] (and concurrently in [Jou23]). Due to the current characteristics of in-use group actions, current signature schemes benefits only partially by the

new constructions derived from the general graph framework, however the work is still open in 3 directions.

1. Searching for new graph topologies that lead to efficient schemes.
2. Categorize the most efficient scheme for any use case with respect to sizes of group and set elements.
3. Finding general bounds, which considers more complex structures, e.g. seed trees, generalizing [BGZ23].

In Chapter 3 we give a general insight on code equivalence problems, a new group action based primitive. Then we group together the relevant literature on the topics, both from Hamming and rank metric, discussing differences and similarity. Then we explain how they are actually rendered to signatures with tailored optimizations (LESS, MEDS [Bal+23b; Cho+23]). These signatures (and also all the other sharing the same design) are then rendered to a threshold version via a construction designed for non-abelian group actions, which does not make extensive use of non-interactive zero-knowledge proofs. Differently for previously proposed distributed schemes cited before these one have direct and practical applications for signatures submitted to the NIST call for post-quantum digital signatures [NIS23], not only LESS and MEDS cited before, but also ALTEQ [Bla+23]. As in [BBMP23] we then instantiate them for code equivalence based signatures, proving them feasible.

We also highlight possible future direction for more efficient threshold signatures based on linear secret sharing. One possibility is the use of the construction for cyclic group actions described in [DM20] combined with the new construction, that requires additional study to properly exploit the efficiency of linear secret sharing. The other one is instead tailored for LESS and use a different representation for the group action. At the moment there is no secure instantiation of this idea, since all the previous trials leak sensible information, still we point out the core idea and possible future directions.

**Outline** We fix the notation in the next section. In Chapter 1 we introduce the concepts of digital signatures, identification protocol and explain how they are related via the Fiat Shamir transform. In Section 1.2 we also give some examples of post quantum signatures derived from identification protocols. In Chapter 2 we describe cryptographic group action properties, then in Section 2.2 we see how to derive a signature from any group action and how to optimize it (Section 2.3), also following the . In Chapter 3 we recall some coding theory concepts, then explain the code equivalence problem for both the Hamming and rank metric. Then we see how LESS and MEDS are instantiated in Section 3.4. Finally in Chapter 4 we explain how to render the signature in Section 2.2 to a threshold signature following [BBMP23]. We also briefly explain a different possible approach for cyclic group actions in Section 4.3.

# Notation

To ease the reading of the thesis we fix here all the notation used. With the goal of having a self contained notation section, we briefly anticipate some elements that be defined later in details. We try to make it consistent with the recent literature on group actions, isogeny-based cryptography and code-based cryptography.

**Elementary Notation** We mean the sets by capital letters $X, G, ...$, the elements by small letters $x, g, ....$ We denote the power set of a set $A$ as $2^A$, i.e.

$$2^A := \{\ B \mid B \subseteq A\}\ \ ;$$

while for any $B \in 2^A$ the complementary is labelled as $B^c := A \backslash B$. The collection of maximal elements of an ordered set $O$ is labeled $O^+$. The set $\mathbb{N}$ represents the positive integers $1, 2, 3, ...$ while for any two integers $a \leqslant b$ we indicate $[a, b]$ as the interval of integers $a \leqslant x \leqslant b : \{\ a, a+1, ..., b-1, b\}$.

**Algebra** When not specified we use the multiplicative notation $\cdot$ for groups and the identity element be indicated by the letter $e$. The group action be indicated by the $\star$ symbol and be applied on the left. The symmetric group on a set of size $n$ is labeled $\mathbb{S}_n := \{\phi : [1, n] \to [1, n] \mid \phi \text{ invertible}\}$. $K$ is a general field and $K^*$ its multiplicative group, while $\mathbb{F}_q$ indicates the finite field with $q$ elements and $\mathbb{Q}$ the field of rationals. We use instead boldface letters $\mathbf{G}, \mathbf{v}, \ldots$ to denote vectors and matrices. Given a matrix $\mathbf{A}$ over $K$, we write $\mathbf{a}_i$ to indicate its $i$-th column. The general linear group formed by the non-singular $k \times k$ matrices over $K$ is indicated as $\mathsf{GL}_k(K)$, if the definition filed is clear by the context we omit it. For an ordered set $J$, we write $\mathbf{A}_J$ to indicate the matrix formed by the columns of $\mathbf{A}$ that are indexed by the elements in $J$; equivalent notation is adopted for vectors. The identity with size $k$ is indicated as $\mathbf{I}_k$, while $\mathbf{0}$ denotes the null-matrix (its dimensions always be clear from the context). A systematic for of a matrix $\mathbf{G}$ be indicated by $\mathrm{SF}(\mathbf{G})$. We denote by $\mathbb{S}_n$ the symmetric group on $n$ elements, and consider its elements as permutations of $n$ objects. We represent permutations as $n$-tuples of the form $\pi := \{i_1, i_2, \cdots, i_n\}$, so that for $j = 1, \cdots, n$, it holds that $\pi(j) = i_j$. Given a matrix $\mathbf{A}$, we write $\pi(\mathbf{A})$ to indicate the matrix resulting from the action of $\pi$ on the columns of $\mathbf{A}$. We denote by $\mathsf{Mono}_n$ the set of monomial transformations, that is, transformations of the form $\mu := (\pi, \mathbf{v})$ with $\pi \in S_n$ and $\mathbf{v} \in \mathbb{F}_q^{*n}$, acting as follows

$$\mu(\mathbf{A}) = \mu\big((\mathbf{a}_1, \cdots, \mathbf{a}_n)\big) = (v_1 \mathbf{a}_{\pi^{-1}(1)}, v_2 \mathbf{a}_{\pi^{-1}(2)}, \cdots, v_n \mathbf{a}_{\pi^{-1}(n)}).$$

**Cryptography** By $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ we mean a negligible function, i.e. a function such that for every positive polynomial $p(\cdot)$ there exists an integer $N_p > 0$ so that for all $x > N_p$:

$$|\mathsf{negl}(x)| < \frac{1}{p(x)}\ .$$

Hash functions, i.e. maps $\{0,1\}^* \to \{0,1\}^*$ easy to evaluate but with strong collision resistance, are labeled $\mathsf{H}$. The letter $E$ means an elliptic curve and the defining field be clear from the context. The pair public and private key are labelled as $(\mathsf{pk}, \mathsf{sk})$, the signature by $\mathsf{sig}$, or $\sigma$. Bytes (8 bits) are labeld as $\mathsf{B}$ or $\mathsf{Bytes}$, while $\mathsf{KiBytes}$ mean $2^{10}$ $\mathsf{Bytes}$. By $\mathsf{Mcyc}$. we mean megacycles. By lexicographic order for sequences of symbols in an ordered set $(X, <)$ we mean the order $<_{lex}$ such that

$$\mathbf{t} <_{lex} \mathbf{s} \iff \exists i \text{ s.t. } t_j = s_j \text{ for all } j < i \text{ and } t_i < s_i \ .$$

A algorithm is labeled as probabilistic polynomial-time if there exists a polynomial $p(\cdot)$ such that for each input $x \in \{\ 0,1\}^*$ the computation terminates in at most $p(|x|)$ times and uses at most $p(|x|)$ independent random bits.

In this thesis, we assume the readers have a foundational understanding of concepts in linear algebra, commutative algebra, algebraic cryptography and group theory. Readers encountering any gaps in their knowledge in these areas are encouraged to refer to the books [Sil09], [Lan12], [LN94].

CHAPTER

# ONE

# SIGNATURES AND FIAT SHAMIR

In this chapter we introduce the basic definitions for signature schemes, identification protocols and their security requirements. Then we show how to render a zero knowledge proof to a secure digital signature scheme, with particular attention to the quantum random oracle model. Then we make some relevant examples of post-quantum primitives derived from $\Sigma$-protocols, all submitted to the NIST call for standardization.

First of all we lie here the definition of digital signature, with its security requirements.

**Definition 1.1** (Signature scheme)**.** A digital signature scheme is a tuple of four probabilistic polynomial-time algorithms $\mathsf{DS} = (\mathsf{DS.Setup}, \mathsf{DS.KeyGen}, \mathsf{DS.Sign}, \mathsf{DS.Verify})$, such that:

- $\mathsf{DS.Setup}(1^\lambda) \to \mathsf{pp}$, on input the security parameter $\lambda$ it returns public parameters for the scheme.
- $\mathsf{DS.KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$, on input the public parameters $\mathsf{pp}$ returns a pair of public and secret keys.
- $\mathsf{DS.Sign}(\mathsf{sk}, \mathsf{m}) \to \mathsf{sig}$, on input the secret key and a message $\mathsf{m}$ outputs a signature $\mathsf{sig}$.
- $\mathsf{DS.Verify}(\mathsf{pk}, \mathsf{m}, \mathsf{sig}) \to 1/0$, on input the public key, the message and the signature accepts (1) or reject (0) them.

A digital signature scheme needs to be correct, i.e. a signature honestly generated by using the 3 initial algorithms should always be accepted with probability 1, and unforgeable. Intuitively unforgeable means that a non honest party should not be able to forge a valid signature, but as for encryption schemes, there are several models for the adversary capabilities and objectives. We consider here the most desirable one: the *Existential Unforgeability under Chosen-Message Attacks* (EUF-CMA) from [GMR88].

**Definition 1.2.** A digital signature $\mathsf{DS}$ is secure in the EUF-CMA if for any probabilistic polynomial-time adversary $\mathsf{Evl}$ that is allowed to:

1. Query a key generation oracle that runs $\mathsf{DS.Setup}$ and $\mathsf{DS.KeyGen}$ for the public parameters $\mathsf{pp}$ and the public key $\mathsf{pk}$ (but not the private key);
2. Perform a polynomial number of query to a signing oracle that on chosen messages $\mathsf{m}_i$ obtaining pairs of valid signatures $(\mathsf{m}_i, \mathsf{sig}_i)$;

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

$$\mathbb{P}\left[\mathsf{DS.Verify}(\mathsf{pk}, \mathsf{m}^*, \mathsf{sig}^*) = 1 \;\middle|\; \begin{array}{l} \mathsf{m}^*, \mathsf{sig}^* \leftarrow \mathsf{Evl}\ , \\ \mathsf{m}^* \neq \mathsf{m}_i\ \forall i\ . \end{array}\right] \leqslant \mathsf{negl}(\lambda) \tag{1.1}$$

There is a stronger variant of this model, called *stronger* Existential Unforgeability under Chosen-Message Attacks (sEUF-CMA) in which we allow the attacker to also output a signature on a previously queried message, as long as the outputted signature $\mathsf{sig}^*$ is different from $\mathsf{sig}_i$ for all $\mathsf{m}_i = \mathsf{m}^*$. Instead the easier security model is the Existential Unforgeability under *No-Message* Attacks (EUF-NMA), where the adversary cannot perform adaptive queries on messages.

## 1.1 Identification Protocols

Modern cryptography does not restrict it self to encryption mechanism and signature schemes, but extends its capabilities to a range of different possible uses. One of them consisting in proving that a particular statement is true, possibly without disclosing any sensible information during the process.

A groundbreaking work in this direction was the article "The knowledge complexity of interactive proof-systems", form Goldwasser, Micali, and Rackoff [GMR19], in which they introduced the concept of Zero-Knowledge Proof, catching the essential ideas behind proofs that disclose only the veridicity of a statement. This notion had in the years an incredible amount of practical applications, from e-voting to digital identity, but also a close relation with Complexity Theory.

We start with a more abstract approach to the definition, following the style of [AB09], then we move to the practical uses and security results of our interest, using a notation more inline with the practical flavor that can be found in nowadays papers.

**Definition 1.3.** The language $L \subset \{0,1\}^*$ have probabilistic identification protocol with $k$ rounds if there exists a probabilistic polynomial time algorithm $V$ that can have a $k$-rounds interaction with the function $P : \{0,1\}^* \to \{0,1\}^*$,

i.e. they can produce a transcript:

$$\begin{aligned} a_1 &= V(x) \\ a_2 &= P(x, a_1) \\ &\dots \\ a_{k-1} &= V(x, a_1, ..., a_{k-2}) \\ a_k &= P(x, a_1, ..., a_{k-1}) \ ; \end{aligned}$$

from which $V$ computes a final binary value $V(x, a_1, ..., a_k)$ to accept or not the transcript. The interaction has the two additional properties:

- (*Completeness*) for all $x \in L$ exists functions $P_1, ..., P_k$ so that $V$ accept the transcripts with probability more than $2/3$.
- (*Soundeness*) for all $x \notin L$ cannot exists functions $P_1, ..., P_k$ so that $V$ accept the transcripts with probability more than $1/3$.

We define as $\mathsf{IP}[k]$ the languages having a probabilistic identification protocol with $k$ rounds, and as $\mathsf{IP}$ the languages having a probabilistic identification protocol with a finite number of rounds, i.e. $\bigcup_{k>0} \mathsf{IP}[k]$.

Here the function $P$ represent the role of a Prover that tries to convince a Verifier $V$ that the string $x$ is in the language $L$. There are a lot of interesting observation that can be done from this definition.

- The value $2/3$ and $1/3$ can be substituted by any other values in the interval $(0, 1)$, since by repeating the same protocol a polynomial number of times the completeness probability increase and the soundness decrease.
- We can replace the constant $2/3$ with 1 in the completeness definition and we would still get the same set $\mathsf{IP}$. This can be seen as a consequence of an important result shown later in Theorem 1.4.
- Instead replacing the constant $1/3$ with 1 in the soundness definition would result in the collapse of the class to $\mathsf{NP}$ since the protocol would be deterministic.
- With this current definition the Verifier has access to a private random tape that the Prover cannot access during its calculation. Allowing the Prover to see them would be equivalent to force the Verifier to only send random bits and nothing more. This protocols in this particular class are called *Arthur-Merlin* proofs or *Public-Coin* proofs. They are indicated as $\mathsf{AM}[k]$.

Doing a detailed discussion on all the rich computational theory of the interactive protocols is out of the scope of this document, however I would like to still give you a glimpse of its most unpredictable results. First of all even if the structure of probabilistic interactive protocol has a very basic definition is extremely powerful, in fact we have that:

**Theorem 1.4** ([Sha92])**.** *The class* IP *is equal to the class* PSPACE*, i.e. all the languages that can be decided by a algorithm in polynomial space (even if with umbounded computational time) have a probabilistic identification protocol with finite rounds.*

With respect to Arthur-Merlin proofs we have a nice property: using any constant $k \geqslant 2$ number of rounds does not add any language to the class, actually:

$$\mathsf{AM}[k] \ = \mathsf{AM}[2] =: \mathsf{AM} \ . \tag{1.2}$$

Moreover it may seem that allowing the Verifier to keep its coins private adds significant power to interactive proofs, but it does not, in fact in [GS86] they proved that:

**Theorem 1.5.** *For every $k : \mathbb{N} \to \mathbb{N}$ with $k(n)$ computable in a polynomial time with respect to $n$ we have:*

$$\mathsf{IP}[k] \ \subseteq \mathsf{AM}[k+2]$$

### 1.1.1 3 Pass Protocols

We focus now in the main tool for our thesis: the 3-pass protocol, a special type of public-coin three move interactive protocol for an NP relation $R \subset \{0,1\}^* \times \{0,1\}^*$. We recall that $R$ is an NP relation if:

- for any $(\mathsf{x}, \mathsf{w})$ there is a polynomial time algorithm that decides if $(\mathsf{x}, \mathsf{w}) \in R$;
- for all relations in $R$ the length of $\mathsf{w}$ is polynomial in the length of $\mathsf{x}$.

We refer to $\mathsf{x}$ as the statement and to $\mathsf{w}$ as the witness. The language $L_R \subset \{0,1\}^*$ is the set of statements $\mathsf{x}$ such that there exists a witness $\mathsf{w}$ with $(\mathsf{x}, \mathsf{w}) \in R$. The decision problem for an NP relation asks, given a statement $\mathsf{x}$, to decide if $\mathsf{x} \in L_R$, while the search one requires to find this witness, if it does exists.

For the definitions concerning this part of the thesis we are using [BKP20] as reference. The algorithms here have black box access to a random oracle $\mathsf{DELPHI}_{\mathsf{rnd}}$ that models an hash function, i.e. an oracle that answer randomly to each new query, but has always the same output given the same input.

**Definition 1.6.** A *3-pass protocol* for the NP relation $R$ is a tuple of four probabilistic polynomial-time algorithms $\Pi = (P_1, P_2, V_1, V_2)$ where $P_1, P_2$ (the Prover) share the states, while $V_2$ (Verifier) is deterministic. The protocol flow goes as following.

- The Prover on input $(\mathsf{x}, \mathsf{w}) \in R$ runs $\mathsf{com} \leftarrow P_1^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{x}, \mathsf{w})$, we call this initial output the *commitment*. The commitment lie in the commitments set COM.
- The Verifier on input com evaluate the random string $\mathsf{ch} \leftarrow V_1^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{com})$ called *challenge*. The challenge lie in the challenges set ch.

– The Prover using the challenge evaluate $\mathsf{resp} \leftarrow P_2^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{ch}, \mathsf{x}, \mathsf{w})$, called the *response*. In some situation it use useful to allow a special character $\perp$ denoting an abortion. The response lie in the responses set $\mathsf{RESP} \cup \{\perp\}$.

– The Verifier finally evaluate a bit output from $V_2(\mathsf{x}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$ used to accept or reject the proof. An accepted tuple $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ is called a *valid transcript* for $\mathsf{x}$.

The protocol need to satisfy the *completeness with abort* property: for all $(\mathsf{x}, \mathsf{w}) \in R$ the last deterministic algorithm $V_2(\mathsf{x}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$ accepts with probability 1 when $\mathsf{com} \leftarrow P_1^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{x}, \mathsf{w})$, $\mathsf{ch} \leftarrow V_1^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{com})$ and $\mathsf{resp} \leftarrow P_2^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{ch}, \mathsf{x}, \mathsf{w})$ (with $\mathsf{resp} \neq \perp$). To have a meaningful protocol we should also ask also that the aborting probability

$$\mathbb{P}\left[\mathsf{resp} = \perp \;\middle|\; \begin{array}{l} \mathsf{com} \leftarrow P_1^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{x}, \mathsf{w}), \\ \mathsf{ch} \leftarrow V_1^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{com}), \\ \mathsf{resp} \leftarrow P_2^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{ch}, \mathsf{x}, \mathsf{w}) \end{array}\right] \tag{1.3}$$

is different from 1, even if it may be non negligible.

*Remark* 1. Analogously we can define a 5-pass identification protocol in which there are two response protocols intercut by two challenges for each commitment. These additional rounds may be useful for some particular protocols to reduce the soundness, like in [CVE11]. Note that the following properties would be defined in the same ways even for the 5-pass case.

First of all we give the first security definition for 3-pass protocols.

**Definition 1.7** (Definition 2.2 of [AABN02])**.** We say that the 3-pass protocol $\Pi = (P_1, P_2, V_1, V_2)$ is *polynomially-secure against impersonation under passive attacks* if for any probabilistic polynomial-time impersonator $\mathsf{Evl}$ with access to a polynomial number of honestly generated transcripts $\mathsf{Tr}(\Pi, \mathsf{x}, \mathsf{w}) = \{(\mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)\}_i$ of the protocol $\Pi$ for $(\mathsf{x}, \mathsf{w}) \in R$ the impersonation probability:

$$\epsilon_{\Pi,\mathsf{Evl}} = \mathbb{P}\left[V_2(\mathsf{x}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = 1 \;\middle|\; \begin{array}{l} (\mathsf{x}, \mathsf{w}) \in R \text{ generated at random}, \\ (\mathsf{com}, st) \leftarrow \mathsf{Evl}^{\mathsf{Tr}(\Pi,\mathsf{x},\mathsf{w})}(\mathsf{x}), \\ \mathsf{ch} \leftarrow V_1^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{com}), \\ \mathsf{resp} \leftarrow \mathsf{Evl}^{\mathsf{Tr}(\Pi,\mathsf{x},\mathsf{w})}(\mathsf{ch}, \mathsf{x}, st) \end{array}\right] \tag{1.4}$$

is negligible in $\lambda$.

An important security property that allow the evaluation of the impersonation probability is the *special soundness*.

**Definition 1.8.** A 3-pass protocol $\Pi$ is *special sound* if there exists a polynomial time algorithm called *extractor* that given as input any statement $\mathsf{x}$ and any two valid transcripts $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$, $(\mathsf{com}, \mathsf{ch}', \mathsf{resp}')$ with $\mathsf{ch} \neq \mathsf{ch}'$ outputs a valid witness $\mathsf{w}$ for the $\mathsf{NP}$ relation $R$.

Eventually we can also consider a relaxed version of the protocol soundness in which there is an additional relation $\tilde{R} \supset R$ such that the extractor algorithm outputs a witness for this presumably weaker relation $\tilde{R}$.

Moreover we need another requirement for the commitment phase: to have an *high min-entropy*. A detailed definition for this concept can be read in Definition 3.2 of [AABN02], but a simple version from [BKP20] say that the protocol has $\alpha$ min-entropy if for any statement-witness pair $(\mathsf{x}, \mathsf{w}) \in R$ and commitment $\mathsf{com}$:

$$\mathbb{P}\left[\mathsf{com} = P_1^{\mathsf{DELPHI}_{\mathsf{rnd}}}(\mathsf{x}, \mathsf{w})\right] \leqslant 2^{-\alpha} \ ;$$

then they say that the protocol has *high* min-entropy if $2^{-\alpha}$ is negligible in $\lambda$. This observation is required to avoid degenerate situations in which there is redundancy in the commitments.

Lastly we would like to have that the protocol does not yield undesired information with respect to the witness $\mathsf{w}$. To achieve this we need an additional important property called *Zero Knowledge* that we define in details in the following section.

### 1.1.2 Zero Knowledge

Consider an $\mathsf{NP}$ decisional problem $L$, to prove that an instance $\mathsf{x}$ is in $L$ with an interactive proof the prover could simply send the certificate at the first round interaction. Even if effective this strategy clearly discloses more information than $\mathsf{x} \in L$ or not.

Instead we would like the possibility for the prover of convincing the verifier without disclosing anything more than the veridicity of the proposition. To define rigorously this property in [GMR19] they use the following definition:

**Definition 1.9.** We say that an interactive protocol $\langle V, P \rangle$ is *Perfect Zero Knowledge* if there is a probabilistic polynomial time algorithm $S$ that for any $\mathsf{x} \in L$ can generate, without any further knowledge, a valid transcript with the same distribution of the ones generated by $\langle V, P \rangle$.

Eventually we may relax the condition to:

- *Statistical Zero Knowledge* if we require only that the actual distribution and the simulated one have negligible statistical distance in length of $\mathsf{x}$;
- *Computational Zero Knowledge* if we require that no probabilistic polynomial-time algorithm can distinguish between original and simulated transcripts with non negligible probability.

The main idea is that if an indistinguishable transcript can be generated from $\mathsf{x}$ in polynomial time then we can recover only information that can be already be obtained from the knowledge of $\mathsf{x}$. With this we can finally proceed in defining the concept of $\Sigma$-protocol, by using a more modern definition.

**Definition 1.10.** A 3-pass protocol $\Pi_\Sigma = (P_1, P_2, V_1, V_2)$ for the NP relation $R$ (from Definition 1.6) is said $\Sigma$-*protocol* if it satisfy the (non aborting) *Honest Verifier Zero Knowledge* property, i.e. if there exists a probabilistic polynomial-time simulator $\mathsf{Sim}$ with access to the same random oracle $\mathsf{DELPHI}_{\mathsf{rdm}}$ that for any $(\mathsf{x}, \mathsf{w}) \in R$, $\mathsf{ch} \in \mathsf{ch}$ produce a valid transcript $(\mathsf{com}', \mathsf{resp}')$ from $\mathsf{x}$, $\mathsf{ch}$.

More formally, for any computationally unbounded adversary $\mathsf{Evl}$ with access to a polynomial number of queries to the common random oracle $\mathsf{DELPHI}_{\mathsf{rnd}}$ the advantage probability

$$\left| \mathbb{P}\left[ \mathsf{Evl}(\mathsf{com}, \mathsf{resp}) = 1 \;\middle|\; \begin{array}{l} \mathsf{com} \leftarrow P_1(\mathsf{x}, \mathsf{w}), \\ \mathsf{resp} \leftarrow P_2(\mathsf{ch}, \mathsf{x}, \mathsf{w}), \\ \mathsf{resp} \neq \perp \end{array} \right] - \mathbb{P}\left[ \mathsf{Evl}(\mathsf{Sim}(\mathsf{x}, \mathsf{ch})) = 1 \right] \right| \tag{1.5}$$

is negligible in $\lambda$.

We see later several examples of identification protocol, but just to give an idea we show now a simple identification protocol for the discrete logarithm problem. Given a cyclic group $G = \langle g \rangle$ of order $N$ the NP relation is $R = \{ (g^a, a) \mid a \in \mathbb{Z}_N \}$. For a pair $(h, a)$ the protocol consists in:

- $P_1$ computes a random integer $k \in \mathbb{Z}_N$ and outputs $\mathsf{com} = g^k$;
- $V_1$ outputs a random challenge $\mathsf{ch} \in \mathbb{Z}_N$;
- $P_2$ computes $\mathsf{resp} = k + \mathsf{ch} \cdot a$;
- $V_2$ accepts if and only if $\mathsf{com} \cdot h^{\mathsf{ch}} = g^{\mathsf{resp}}$.

Since there are later similar and more meaningful proofs we leave to the reader to verify that this is a $\Sigma$-protocol by proving completeness, special soundness and zero knowledge.

In Protocol 1.1.1 you can see a more compact way to represent a $\Sigma$-protocol, that we also use for the rest of the thesis (eventually adding above the used parameters).

| **PROVER** | | **VERIFIER** |
|---|---|---|
| Sample $k \in \mathbb{Z}_N$ and set $\mathsf{com} = g^k$ | $\xrightarrow{\mathsf{com}}$ | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \mathbb{Z}_N$. |
| Set $\mathsf{resp} = k + \mathsf{ch} \cdot a$ | $\xrightarrow{\mathsf{resp}}$ | |
| | | Accept if $\mathsf{com} \cdot h^{\mathsf{ch}} = g^{\mathsf{resp}}$. |

Protocol 1.1.1: $\Sigma$-protocol for the discrete logarithm problem

*Remarks* 2. As said for the AM definition, since the randomness for the first verifier step is public, it is enough for the challenge evaluation to output the randomness. So this step is always a random sampling on the challenge set. Also we can observe that the assumption that $P_1, P_2$ share the state is used here for the secret integer $k$.

### 1.1.3 Fiat-Shamir Transform

At this point we have a protocol that requires the interaction between two parties, that is inconvenient since in realty the verification may happen in a separate moment. To solve this Fiat and Shamir in their seminal work "How To Prove Yourself: Practical Solutions to Identification and Signature Problems" [FS87] showed how to make an identification protocol non-interactive and render it to a digital signature.

The idea is fairly simple: use $\mathsf{H}(\mathsf{com}\|\mathsf{m})$ as challenge, where $\mathsf{H}$ is collision resistant cryptographic hash function, that takes the role of the random oracle.

You can see the resulting digital signature scheme in Algorithm 1, that follows the conventions of Definition 1.1. For example by applying the Fiat-Shamir transform on Protocol 1.1.1 we would get what is called a Schnorr signature scheme, the golden standard for nowadays digital signature schemes [Sch91; Sch90].

**Algorithm 1** Digital signature from a $\Sigma$-protocol

$\mathsf{Setup}(1^\lambda)$:
1: decide public parameters for $R$;
2: decide the hash $\mathsf{H}$;
3: **return** $\mathsf{pp}$ with this info

$\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \mathsf{sig})$:
1: parse $(\mathsf{com}, \mathsf{resp}) \leftarrow \mathsf{sig}$;
2: $\mathsf{ch} \leftarrow \mathsf{H}(\mathsf{com}\|\mathsf{m})$;
3: $\mathsf{out} \leftarrow V_2^{\mathsf{H}}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$;
4: **return** $\mathsf{out}$

$\mathsf{KeyGen}(\mathsf{pp})$:
1: Sample $(\mathsf{x}, \mathsf{w}) \in R$ ;
2: **return** $\mathsf{pk} \leftarrow \mathsf{x}$, $\mathsf{sk} \leftarrow \mathsf{w}$.

$\mathsf{Sign}(\mathsf{sk}, \mathsf{m})$:
1: **do**
2: $\quad \mathsf{com} \leftarrow P_1^{\mathsf{H}}(\mathsf{pk}, \mathsf{sk})$;
3: $\quad \mathsf{ch} \leftarrow \mathsf{H}(\mathsf{com}\|\mathsf{m})$;
4: $\quad \mathsf{resp} \leftarrow P_2^{\mathsf{H}}(\mathsf{ch}, \mathsf{pk}, \mathsf{sk})$
5: **while** $\mathsf{resp} \neq \perp$
6: **return** $\mathsf{sig} = (\mathsf{com}, \mathsf{resp})$

Signatures rendered from a $\Sigma$-protocol have a lot of nice properties, like forward security and efficient instantiations. An important result for the Fiat-Shamir transform is the following reduction:

**Theorem 1.11** (Theorem 3.3 [AABN02])**.** *Consider a $\Sigma$-protocol $\Pi_\Sigma = (P_1, P_2, V_1, V_2)$ with high min-entropy and the associated signature scheme $\mathsf{DS}$ described in Algorithm 1. Then the digital signature scheme is EUF-CMA secure in the random oracle model if and only if the identification protocol is polynomially-secure against impersonation under passive attacks.*

To prove the if direction of Theorem 1.11, i.e. that from a secure identification protocol we can get a secure digital signature (Lemma 3.5 [AABN02]), they used a strategy called code-based game-playing ([BR06]). The core idea is to start from an adversary $\mathsf{Evl}$ able to forge a signature and initiate a game defined by:

– an **Initialize** procedure, executed at the start so that its outputs are the $\mathsf{Evl}$ inputs;

– specific procedures to respond to adversary oracle queries, in our case random oracle queries and signing queries;

– a **Finalize** procedure, executed after the termination of $\mathsf{Evl}$ so that its outputs are the procedure inputs.

The output of the latter, denoted $G^{\mathsf{Evl}}$, is the output of the game. The game allow for the situation in which the execution may fail, thus there is an additional flag $\mathsf{bad}$ initially set to $\mathsf{false}$.

Another approach used to reduce the security of the signature scheme to the one of the identification protocol (introduced in [PS00] for signature schemes) rely in an pivotal result for the study of security assumptions: the *Forking Lemma*. There are several form of this lemma, we consider here the most general one:

**Lemma 1.12** (Forking Lemma, Lemma 1 [BN06])**.** *Fix an integer $q \geqslant 1$ that represents the number of call to a random oracle; a set $H$ of size $h > 1$ that represents the codomain of the random oracle. Let $\mathsf{Evl}$ be a randomized algorithm that on input $x, h_1, ..., h_q$ returns a pair $(I, \mathsf{sig})$. $I$ is an integer in the range $0, ..., q$, that represents the random oracle call in which the forking is happening, while $\mathsf{sig}$ is the side output. Let $\mathsf{IG}$ be a randomized algorithm that we call the input generator. The accepting probability of $\mathsf{Evl}$, denoted $\mathsf{acc}$, is defined as the probability that $J \geqslant 1$ in the experiment*

$$x \leftarrow \mathsf{IG}; \quad h_1, ..., h_q \xleftarrow{\$} H; \quad (J, \mathsf{sig}) \leftarrow \mathsf{Evl}(x, h_1, ..., h_q) \ . \tag{1.6}$$

*The forking algorithm $\mathbf{Fork}^{\mathsf{Evl}}$ associated to $\mathsf{Evl}$ is the randomized algorithm that on input $x$ proceeds as follows in Algorithm 2:*

**Algorithm 2** Forking algorithm $\mathbf{Fork}^{\mathsf{Evl}}$

Pick coins $\rho$ for $\mathsf{Evl}$ at random;
$h_1, \ldots, h_q \xleftarrow{\$} H$;
$(I, \mathsf{sig}) \leftarrow \mathsf{Evl}\,(x, h_1, \ldots, h_q; \rho)$
**if** $I = 0$ **then**
    **return** $(0, \perp, \perp)$
$h'_I, \ldots, h'_q \xleftarrow{\$} H$;
$\left(I', \mathsf{sig}'\right) \leftarrow \mathsf{Evl}\left(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q; \rho\right)$;
**if** $(I = I'$ and $h_I \neq h'_I)$ **then**
    **return** $\left(1, \mathsf{sig}, \mathsf{sig}'\right)$
**else**
    **return** $(0, \perp, \perp)$.

*Now let* $\mathsf{frk}$ *be the probability of returning* $\left(1, \mathsf{sig}, \mathsf{sig}'\right)$*, then we have*

$$\mathsf{frk} \ = \mathsf{acc}\left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right) \ . \tag{1.7}$$

We give an intuition on the use of the lemma by proving:

**Proposition 1.13.** *Consider an high min-entropy $\Sigma$-protocol $\Pi_\Sigma$ for the relation $R$ that is special sound (Definition 1.8). Then, in the random oracle model, if there exists an EUF-CMA adversary that break the signature in Algorithm 1 derived from an $\Pi_\Sigma$ we can extract the witness $\mathsf{w}$ in probabilistic polynomial-time by the knowledge of the statement $\mathsf{x} \in L_R$ with non negligible probability.*

*Proof.* Consider the probabilistic polynomial-time attacker $\mathsf{Evl}$ to the signature in Algorithm 1 instantiated with the statement $\mathsf{x}$ as public key. We would like to use the Forking Lemma to have two different valid signatures on the same message $\mathsf{m}$ and commit $\mathsf{com}$, but different responses $\mathsf{resp} \neq \mathsf{resp}'$. Hence by using the extractor from the special soundness definition we would get the witness.

$\mathsf{Evl}$ use a randomness $\rho$, performs a polynomial number $q$ of call to the random oracle and at some point asks the random oracle the query $\mathsf{com}\|\mathsf{m}$ used for the forged signature. Let $J$ be the index of this call, we modify $\mathsf{Evl}$ to return as firs output $J$.

We need now to define the answers for $\mathsf{Evl}$. For to the random oracle calls we mediate them by the use of a table that stores pairs $(x, h)$. When an input $x$ is not contained in the table we just query the oracle and store the output $h$ in the table with $x$, then return $h$. When the input is contained just return the associated output.

For the signing queries on input $\mathsf{m}_i$ we use the Perfect Zero Knowledge property and we proceed as follows:

**Algorithm 3** Signature simulation

1: Query the random oracle for a random challenge $\mathsf{ch}_i$;
2: Run the simulator $\mathsf{Sim}(\mathsf{x}, \mathsf{ch}_i) \rightarrow (\mathsf{com}_i, \mathsf{resp}_i)$;
3: Add the pair $(\mathsf{com}_i\|\mathsf{m}_i, \mathsf{ch}_i)$ to the table; ▷ the input may be already be asked only with negligible probability
4: **return** the signature $(\mathsf{com}_i, \mathsf{resp}_i)$.

The action performed at line 3 is usually referred as *reprogramming the random oracle* since what we are actually doing is modifying the hash function so that on input $\mathsf{com}_i\|\mathsf{m}_i$ answer $\mathsf{ch}_i$. By the properties of the simulator this is a valid signature undistinguishable from an original one. When the adversary outputs a valid signature $\mathsf{com}, \mathsf{resp}$ for the message $\mathsf{m}$ return $(J, \mathsf{m}, \mathsf{cmt}, \mathsf{resp})$, otherwise return $(0, \perp)$. We see that the $\mathsf{acc}$ probability $(J \geqslant 1)$ in the Forking Lemma is the probability of forgery for the original attack (thus non negligible).

We can now execute Algorithm 2 for our modified adversary $\mathsf{Evl}$. The algorithm succeeds with non negligible probability $\mathsf{frk}$ since $\mathsf{acc}$ is non negligible, $q$

is polynomial and $1/h$ is negligible for the high min-entropy assumption. When the algorithm succeeds:

- We have as output two different message-signature pairs $(\mathsf{m}, (\mathsf{com}, \mathsf{resp}))$ and $(\mathsf{m}', (\mathsf{com}', \mathsf{resp}'))$.
- $I = I'$, this means that the query for $\mathsf{com}\|\mathsf{m}$ and $\mathsf{com}'\|\mathsf{m}'$ has happened at the same time. This implies that $\mathsf{com} = \mathsf{com}'$ and $\mathsf{m} = \mathsf{m}'$ since until that point the executions where the same (same initial randomness and oracle calls).
- $h_I \neq h'_I$, this implies that the challenge associated to the response are different. In fact here we have *forked* the execution, creating a different final result for $\mathsf{Evl}$.

Since the resulting signature are two both valid transcripts with same commitments but different challenges we can use the extractor as described before. □

### 1.1.4 Quantum Random Oracle

As said the ROM (Random Oracle Model) is a valid tool for understanding the security of cryptographic schemes, de facto under the assumption that the adversary cannot tamper with the hash function. However we should point out that in post-quantum cryptography we should consider quantum probabilistic polynomial-time adversaries allowed to perform computations on a superposition $\sum \alpha_x |x\rangle$.

This means that a quantum adversary, if able to compute the hash function, can also learn $\sum \alpha_x |x\rangle|\mathsf{H}(x)\rangle$ in a single Random Oracle call. Thus in the *Quantum Random Oracle Model* the adversary is allowed also to ask query in superposition ([Bon+11]), eventualy measured in a second moment.

This create a problem for our proof strategies that reprogram the oracle at runtime, in fact by the quantum mechanics properties we cannot observe or copy the queries in advance without destroying them.

The problem of carrying out proof in ROM to proof for the QROM is clearly non trivial, even if this does not imply that currently secure schemes are vulnerable in the quantum model. For the specific case of the Fiat-Shamir transform there have been intensive study in recent years, like [Unr17; DFMS19; LZ19]. The main result from this analysis involve the use of an additional property for the Σ-protocol:

**Definition 1.14** (Definition 24 of [DFMS19])**.** A Σ-protocol has *quantum computationally unique responses*, if the verification predicate $V_2(x, \cdot, \cdot, \cdot) : \mathsf{COM} \times \mathsf{CH} \times \mathsf{RESP} \to \{0, 1\}$ seen as a relation between $\mathsf{COM} \times \mathsf{CH}$ and $\mathsf{RESP}$ is collapsing from $\mathsf{RESP}$ to $\mathsf{COM} \times \mathsf{CH}$.

By saying that a relation $R \subset \mathsf{X} \times \mathsf{Y}$ is collapsing $\mathsf{X}$ to $\mathsf{Y}$ we mean that when $(x, y) \in R$ it is unfeasible to distinguish in polynomial time if $x$ has been measured or not. A detailed definition can be read in [DFMS19, Definition 23], that generalized the one introduced from Unruh in [Unr16].

In successive literature like [Blä+22] the *quantum computationally unique response* definition is simplified. It is enough to ask that, given the security parameter $\lambda$, for any probabilistic polynomial-time quantum adversary $\mathsf{Evl}$ the probability

$$\mathbb{P}_{(\mathsf{x},\mathsf{w})\xleftarrow{\$}R}\left[\begin{array}{l} V_2(\mathsf{com},\mathsf{ch},\mathsf{resp})=1,\\ V_2(\mathsf{com},\mathsf{ch},\mathsf{resp}')=1\\ \mathsf{resp}\neq\mathsf{resp}'. \end{array}\middle|\ (\mathsf{com},\mathsf{ch},\mathsf{resp},\mathsf{resp}')\leftarrow\mathsf{Evl}(\mathsf{x})\right] \tag{1.8}$$

is negligible in $\lambda$.

Eventually a stronger requirement is the existence of only one response for any pair of commitment and challenge. In this case we say the protocol has *perfect unique response*.

**Theorem 1.15** (Combination of Theorem 9 [Unr12], Theorem 22 [DFMS19])**.** *Let $\Pi_\Sigma$ be a $\Sigma$-protocol with quantum computationally unique responses for some quantum secure relation $R$[1]; then the signature obtained via the Fiat-Shamir transform is sEUF-CMA secure in the quantum random oracle model.*

This theorem show the relevance of the quantum computationally unique responses property when designing post-quantum primitives. The only lack of this result is that the reduction from sEUF-CMA to EUF-NMA security is not tight.

Tight reductions can be proved using additional assumption, e.g. see *lossy* key generation from [AFLT12; KLS18], or using results from [GHHM21].

There they proved that, if we consider a random oracle model in which the calls are only partially controlled by the adversary, we can randomly reprogram the oracle also in the quantum oracle model. By *partially controlled by the adversary* we mean that the oracle is a random function between finite sets $X_1 \times X_2 \to Y$ and on each call $(x_1\|x_2)$ reprogrammed to $y$ the input $x_2$ may be chosen arbitrarily by the adversary, while the $x_1, y$ values are chosen accordingly to a random distribution (possibly the uniform one).

To be more precise we state here the theorem for the uniform distribution case.

**Theorem 1.16** (Proposition 1 [GHHM21])**.** *Consider finite sets $X_1, X_2, Y$, a quantum probabilistic polynomial-time adversary and the following game. If*

**GAME** $\mathsf{Repro}^{\mathrm{EVL}}(b)$
- $\mathsf{DELPHI}_0 \xleftarrow{\$} X_1 \times X_2 \to Y$;
- $\mathsf{DELPHI}_1 \leftarrow \mathsf{DELPHI}_0$;
- $b' \leftarrow \mathsf{Evl}^{|\mathsf{DELPHI}_b\rangle,\mathsf{Reprogram}}$;
- **return** $b'$.

$\mathsf{Reprogram}(x_2)$
- $(x_1, y) \xleftarrow{\$} X_1 \times Y$;
- $\mathsf{DELPHI}_1 \leftarrow \mathsf{DELPHI}_1^{(x_1\|x_2)\mapsto y}$;
- **return** $x_1$.

[1] i.e. a quantum probabilistic polynomial-time adversary can recover the witness from the statement only with negligible probability

*$R$ is the number of times the oracle is reprogrammed and $q$ is the number of quantum queries of $\mathsf{Evl}$ to $\mathsf{DELPHI}_b$ then distinguishing advantage is bounded as:*

$$\left|\mathbb{P}[\mathsf{Repro}^{\mathsf{Evl}}(1)=1]-\mathbb{P}[\mathsf{Repro}^{\mathsf{Evl}}(0)=1]\right| \leqslant \frac{3R}{2}\sqrt{\frac{q}{\#X_1}}$$

In plain English this theorem implies that the reprogramming of a random oracle is indistinguishable even for quantum queries if we only reprogram on inputs with a "good portion" of randomness to random outputs.

From this theorem the classical Fiat-Shamir reduction from NMA to CMA that simulates signatures via reprogramming the random oracles on random commits can be leveraged to prove this reduction also in the quantum random oracle model ([GHHM21, Theorem 3]).

## 1.2 An Overview of Current Post Quantum Signature Schemes

We now proceed in showing here some signature schemes relevant for the post-quantum cryptography community, all derived from the identification protocol associated to some hard mathematical probel. The protocol is then rendered non-interactive via the Fiat-Shamir construction explained before. We avoid the one derived for group actions like LESS, MEDS and CSI-FiSh, which are explained later in Section 2.2.

**Crystarls-Dilithium** The Crystarls-Dilithium signature scheme [Duc+18] is one of the finalist of the 4 post-quantum algorithms selected by NIST for standardization in 2022. It is part of the so called lattice-based primitives, in particular its security rely on the following NP problem.

**Problem 3** (Module-Learning With Errors (Mod-LWE))**.** Consider the ring of polynomials $R_q = \mathbb{Z}_q[x]/\langle x^k+1\rangle$. Given a lattice $\mathbf{A} \in R_q^{n\times m}$ and a Gaussian distribution $\chi$ on $R_q$ centered on 0 define, from a secret vector $\mathbf{s} \in R_q^{n\times m}$ and an error $\mathbf{e} \leftarrow \chi^m$, the statement $\mathbf{b} = \mathbf{sA} + \mathbf{e}$. The Module-Learning With Errors Problem requires to find the secret $\mathbf{s}$ from $\mathbf{b}, \mathbf{A}, \chi$.

We can use Mod-LWE to define an equivalent of the Schnorr protocol (Protocol 1.1.1). Consider a statement $\mathbf{b}$ and a witness $\mathbf{s}$, as in the problem formulation. To compare two vectors and see if they are equal, minus an error sampled with $\chi$, we use a function $\mathsf{HighBits}$ that selects the highest bits of each vector entry (we do not go into the details of its characteristics). Intuitively the protocols is complete since:

$$\mathsf{resp}\cdot\mathbf{A} - \mathsf{ch}\cdot\mathbf{b} = \underbrace{\mathbf{yA} + \mathsf{ch}\cdot\mathbf{sA}}_{\mathsf{resp}\cdot\mathbf{A}} - \mathsf{ch}\underbrace{(\mathbf{sA}+\mathbf{e})}_{\mathbf{b}} = \mathbf{yA} - \mathsf{ch}\cdot\mathbf{e} \simeq \mathbf{yA}\ .$$

Since we are dealing with random non uniform distribution it is possible that the response $\mathbf{y} + \mathsf{ch}\cdot\mathbf{s}$ may reveal some information about the secret, thus it

necessary to add an additional control before sending the response. If the check is not passed then the round of the protocol is aborted. This observation is crucial for the design of the protocol and the associated signature scheme, as shown in [Lyu09]. Then Protocol 1.2.1 can be converted to a signature with the Fiat-Shamir transform.

| **PROVER** | | **VERIFIER** |
|---|---|---|
| Sample $\mathbf{y} \in R_q^n$, set $\mathsf{com} = \mathsf{HighBits}(\mathbf{yA})$ | $\xrightarrow{\mathsf{com}}$ | |
| Set $\mathsf{com} = \mathsf{HighBits}(\mathbf{yA})$ | | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\phi} R_q$. |
| Set $\mathsf{resp} = \mathbf{y} + \mathsf{ch} \cdot \mathbf{s}$ | $\xrightarrow{\mathsf{resp}}$ | Accept if |
| If $\mathsf{resp} \notin G^m$ then $\mathsf{resp} = \perp$ | | $\mathsf{com} = \mathsf{HighBits}(\mathsf{resp} \cdot \mathbf{A} - \mathsf{ch} \cdot \mathbf{b})$. |

Protocol 1.2.1: sig-protocol for the Mod-LWE Problem

**SqiSign** An isogeny between the elliptic curves $E, E'$ is a surjective (on the algebraic closure of the field) morphism $\phi : E \to E'$, fixing the identity. Two valid references for isogeny based cryptography can be found on [De 17] and [Sil09, Chapter 3]. A separable isogeny may be described by its kernel, when seen as a group homomorphism, thus for any subgroup $G \leqslant E$ we can define and evaluate the isogeny $\phi : E \to E/G$. The degree of an isogeny is both its degree as rational map and the order of the kernel group describing it. A natural problem arising from these objects it to find an isogeny between two random elliptic curves.

Given a curve $E$ the endomorphism ring $\mathrm{End}(E)$ is the ring of isogenies $E \to E$, with point-wise addition and composition as operations. The evaluation of the endomorphism for a random elliptic curve is a conjecturally hard problem well understood by the algebraic geometry community. More interestingly we have that:

- given two elliptic curves with known endomorphism ring, the isogeny between them can be evaluated in polynomial time;
- given two elliptic curves with an isogeny $\phi : E \to E'$ between them, from the knowledge of $\mathrm{End}(E)$ the other endomorphism ring $\mathrm{End}(E')$ can be evaluated in polynomial time.

The *Short Quaternion and Isogeny Signature* (SQISign) is a isogeny based digital signature scheme using a sig-protocol which proves the knowledge of the endomorphism ring of a public curve $E_A$. The curve $E_A$ is generated from $E_0$, a starting curve with known endomorphism ring, via a random isogeny $\tau : E_0 \to E_A$.

The commitment of the scheme is an elliptic curve $E_1$ generated by a random secret isogeny path from $E_0$, i.e. a random isogeny $\psi : E_0 \to E_1$, and the challenge is a random cyclic isogeny $\phi : E_1 \to E_2$ (cyclic mean that the associated kernel is cyclic) of fixed degree. The response is another cyclic isogeny

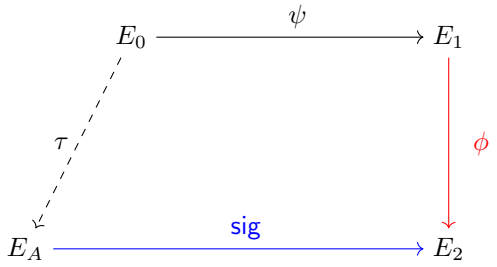



Figure 1.1: Diagram for the SQISign identification protocol.

$\mathsf{sig} : E_A \to E_2$ of fixed degree. See the diagram in Figure 1.1. Thus the verifier only need to check that the diagram commutes and the response is cyclic of the expected degree. Even if simple this protocol requires careful attention and sophisticated algebraic procedures to ensure its security and efficiency.

**Schemes based on Decoding** A natural choice of hard problem is the Syndrome Decoding or Maximum Likelihood Decoding Problem, both for its interdisciplinary and its well understood hardness. More on that in Section 3.1, which is specifically dedicated to coding theory, for now we only insert here the classical formulation for completeness.

**Problem 4** (Syndrome Decoding (MLD))**.** Given an $(n-k) \times n$ matrix $\mathbf{H}$, a vector $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and a positive integer $w$ find a non-zero vector $\mathbf{e} \in \mathbb{F}_q^n$ solving $\mathbf{eH}^\perp = \mathbf{s}$ with number of non-zero entries less than $w$.

The story of Fiat-Shamir signature obtained from coding theory assumptions trace back to 1994 with [Ste94; Vér97]. In these work a 3-pass protocol with 1/3 soundness for the MLD Problem was firstly proposed and improved.

Even if at the time the scheme parameters were impractical new variants were studied. Around 15 years later a new 5-pass protocol for the same problem was proposed in [CVE11], this time with soundness roughly 1/2. This last one is shown in Protocol 1.2.2, where $\mathcal{S} \subset \mathbb{F}_q^n$ is the set of vectors with exactly $w$ non-zero entries. Thanks to the renovated interest for code-based cryptography as a suitable family of quantum resistant problems new improvements where proposed to render the Syndrome Decoding Problem to a functional digital signature.

Much of these were in the direction of using structured codes that allows for compression of the protocol responses, however, notably, two of the more interesting proposal for the new NIST on-ramp call [NIS23] are based on random codes. One uses cutting-edge multiparty computation techniques [Agu+23] and is presented in Section 1.2.1, while the other [Bal+23a] is based on the Restricted Syndrome Decoding Problem (R-SDP), that corresponds to a (conjectured) harder version of the Syndrome Decoding Problem (SDP). R-SDP adds the additional constraint that entries of the solution vector must live in a desired

Public Data : Field $\mathbb{F}_q$, integer $w$, set $\mathcal{S}$ and hash function $\mathsf{H}$.
Private Key : Error vector $\mathbf{e} \in \mathcal{S}$
Public Key : $\mathbf{s} = \mathbf{e}\mathbf{H}^\perp$.

| **PROVER** | | **VERIFIER** |
|---|---|---|
| Get $\mathbf{u} \xleftarrow{\$} \mathbb{F}_q^n$, $\pi \xleftarrow{\$} \mathbb{S}_n$ | $\xrightarrow{\mathsf{com}_0,\mathsf{com}_1}$ | |
| Set $\mathsf{com}_0 = \mathsf{H}(\pi, \mathbf{u}\mathbf{H}^\perp)$ | | |
| Set $\mathsf{com}_1 = \mathsf{H}(\pi(\mathbf{u}), \pi(\mathbf{e}))$ | | |
| | $\xleftarrow{\mathsf{ch}_0}$ | $\mathsf{ch}_0 \xleftarrow{\$} \mathbb{F}_q^*$. |
| Set $\mathsf{resp}_0 \leftarrow \pi(\mathbf{u} + \mathsf{ch}_0 \cdot \mathbf{e})$ | $\xleftarrow{\mathsf{resp}_0}$ | |
| | $\xleftarrow{\mathsf{ch}_1}$ | $\mathsf{ch}_1 \xleftarrow{\$} \{0,1\}$. |
| If $\mathsf{ch}_1 = 0$, set $\mathsf{resp}_1 \leftarrow \pi$ | $\xrightarrow{\mathsf{resp}_1}$ | |
| If $\mathsf{ch}_1 = 1$, set $\mathsf{resp}_1 \leftarrow \pi(\mathbf{e})$ | | If $\mathsf{ch}_1 = 0$, accept if $\mathsf{com}_0 = \mathsf{H}(\pi, \pi^{-1}(\mathbf{y})\mathbf{H}^\perp - \mathsf{ch}_0 \cdot \mathbf{s})$ |
| | | If $\mathsf{ch}_1 = 1$, accept if $\mathsf{resp}_1 \in \mathcal{S}$ and $\mathsf{com}_1 = \mathsf{H}(\mathbf{y} - \mathsf{ch}_0\mathsf{resp}_1, \mathsf{resp}_1)$ |

Protocol 1.2.2: 5-pass identification protocol for the SDP ($\mathcal{S} = \mathcal{S}_w$) and R-SDP ($\mathcal{S} = \mathcal{S}_w^{\mathbb{E}}$) .

subset $\mathbb{E}$ of the finite field.

**Problem 5** (Restricted Syndrome Decoding (R-SDP))**.** Given an $(n-k) \times n$ matrix $\mathbf{H}$, a vector $\mathbf{s} \in \mathbb{F}_q^{n-k}$, a subset $\mathbb{E} \subseteq \mathbb{F}_q^*$ and a positive integer $w$ find a non-zero vector $\mathbf{e} \in \mathbb{F}_q^n$ solving $\mathbf{e}\mathbf{H}^\perp = \mathbf{s}$ with $\mathbf{e} \in \mathcal{S}_w^{\mathbb{E}}$, where

$$\mathcal{S}_w^{\mathbb{E}} := \{\, \mathbf{x} \in (\mathbb{E} \cup \{0\})^n \mid \mathbf{x} \text{ has } w \text{ non-zero entries} \} \,. \tag{1.9}$$

The additional constrain in the set $\mathbb{E}$ is conjectured in [Bal+23a] to notably increase the hardness, allowing for much better parameters choices. The identification protocol used in the signature is the same as for Syndrome Decoding Problem (Protocol 1.2.2), but with $\mathcal{S}_w^{\mathbb{E}}$ instead of $\mathcal{S}_w$, i.e. the response $\mathsf{resp}_1$ should be checked also for the $\mathbb{E}$ set constrain when $\mathsf{ch}_1 = 1$. The scheme obtained via the non-interactive version of the protocol is known as CROSS.

### 1.2.1 Multyparty Computations in the Head

An important ingredient for most of the protocols used today is the *Multyparty Computations in the Head* paradigm, eventually adjoined with the hypercube technique. The core idea is to share the witness $\mathsf{w}$ of an $\mathsf{NP}$ relation additively, i.e. each party $P_i$ get access to $[\![\mathsf{w}]\!]_i$ so that $\mathsf{w} = \sum_{i=1}^N [\![\mathsf{w}]\!]_i$. The set of shares is also labelled $[\![\mathsf{w}]\!]$. Then a protocol evaluating $f(\mathsf{w})$ is run collectively by the parties using classical multiparty computations techniques, like:

- *Addition*: $[\![\mathbf{x} + \mathbf{y}]\!]_i = [\![\mathbf{x}]\!]_i + [\![\mathbf{y}]\!]_i$;
- *Constant addition*: $[\![c + \mathbf{x}]\!]_i = [\![\mathbf{x}]\!]_i + c\mathbb{1}(i = 1)$;

– *Constant multiplication*: $[\![c\mathbf{x}]\!]_i = c[\![\mathbf{x}]\!]_i$;

– *Multiplication*: two shared value can be multiplied using a so called Beaver Triple ($[\![\mathbf{a}]\!], [\![\mathbf{b}]\!], [\![\mathbf{c}]\!]$ with $\mathbf{ab} = \mathbf{c}$) using a well known procedure explained in [Bea92].

This MPC protocol is then rendered to a 5-pass identification protocol with the following structure:

1. Prover generates $N$ shares of the input and commit to them as $\mathsf{com}_1$, ..., $\mathsf{com}_N$;
2. Verifier generates the randomness used for the protocol as first challenge;
3. Prover runs the MPC protocol *in the Head*, i.e. without disclosing it, and commit to the output $[\![f(\mathsf{w})]\!]$;
4. Verifier sends as challenge a set $I \subsetneq [1, N]$ of users, usually $\#I = N - 1$;
5. Prover sends the input shares of all the parties in $I$ plus the shares $[\![f(\mathsf{w})]\!]_i$ for $i \notin I$;
6. Verifier checks the consistency of all the responses by rerunning the protocol for users in $I$.

The identification protocol can be then rendered non-interactive again using the Fiat-Shamir transform. During the step 1 $N - 1$ shares are generated via seeds using pseudorandom generator while the last one is obtained by subtracting them from the original one. The soundness of the protocol is approximatively[2] $N^{-1}$, but sadly the complexity also is linear in $N$.

To render the MPC in the head competitive with respect to the classical repetition of the protocol a geometrical approach in [Agu+23], inspired to the *hypercube*, is used in several schemes. The idea is to carry over the protocol computations on the steps 3, 6 not on all the shares but just on the sum of some selected partitions of $[1, N]$ that allow the computation of the final output $f(\mathsf{w})$.

More precisely, for each partition in $r$ subsets during step 3 we associate $r$ shares, obtained via the sum on the subset of the partition. With this share we perform an MPC computation. Then, during step 6, the verifier can use the $N - 1$ shares to recover the sum of $r - 1$ subsets and it only receives the missing one, so it can check the computations consistency. The soundness of this part of the protocol is approximatively $r^{-1}$. The turning point of this strategy is that, if we consider $n$ pairwise distinct partitions with sizes $r_1, ..., r_n$, the soundness is approximatively $(r_1 \cdots r_n)^{-1}$ (suppose the missing subsets always intersect) while the complexity is $O(r_1 + \cdots + r_n)$.

The geometrical interpretation is necessary to properly define these partitions, fix $N = r^n$ and $r = r_1 = ... = r_n$, then consider an hypercube of dimension

[2] since also the choice of the randomness from step 2 influences the soundness of the protocol

$n$ with edge length $r$. Label each vertex with a schare and for each dimension define a partition considering the $r$ slices of the hypercube. The most used edge length is usually 2. This way we got a complexity $rn$ that is logarithmic in the reciprocal of the soundness ($\simeq r^n$). More on that can be read in [Agu+23].

We see now two schemes based in classical decoding problems that exploits the MPC in the head approach to achieve interesting parameters.

**SDitH** As anticipated before an appealing problem hard problem for post-quantum signatures is the Syndrome Decoding Problem (Problem 16), its multiparty version where studied in [GPS22] and [FJR22], culminating in the hypercube optimized version [Agu+23].

This last protocol uses some polynomials that are null when the constrains on the number of non-zero entries and the relation $\mathbf{s} = \mathbf{e}\mathbf{H}^\perp$ are verified. This way by proving them equal to zero on random points, using hypercube MPC, we get an identification protocol with soundness exponential with respect to the overhead complexity.

**MiRitH** The protocol **Min**Rank **i**n **t**he **H**ead Digital Signature Scheme (MiRitH, [Adj+23]) is based on the upcoming problem:

**Problem 6** (MinRank)**.** Given $k+1$ $m \times n$ matrices $\mathbf{C}^{(0)}, \mathbf{C}^{(1)}, ..., \mathbf{C}^{(k)} \in \mathbb{F}_q^{m \times n}$ and a positive integer $r \leqslant \min(n, m)$ find $k$ coefficients $\mathbf{t} \in \mathbb{F}_q^k$ such that:

$$\text{Rank}\left(\mathbf{C}^{(0)} + \sum_{i=1}^{k} t_i \mathbf{C}^{(i)}\right) \leqslant r \ .$$

The MinRank problem has been introduced in [BFS99] and has been studied for over 20 years, e.g. see [BBBGT23; Bar+20; FLP08; GRS15].

In particular the MiRitH protocol proposed in [Adj+23] focuses on the Knipis-Shamir modelling for Problem 6. This modelling states that, if a vector $\mathbf{t} \in \mathbb{F}_q^k$ and a matrix $\mathbf{K} \in \mathbb{F}_q^{r \times (n-r)}$ satisfy

$$\left(\mathbf{C}_0 + \sum_{i=1}^{k} \alpha_i \mathbf{C}_i\right) \cdot \begin{bmatrix} \mathbf{A} \\ \mathbf{K} \end{bmatrix} = \mathbf{0} \ , \tag{1.10}$$

with $\mathbf{A} \in \mathsf{GL}_{n-r}$; then $\mathbf{t}$ is a solution for the MinRank instace given by $r$ and the matrices $\mathbf{C}_i$ for $i = 0, .., k$. By fixing $\mathbf{A} = \mathbf{I}_{n-r}$ is possible to render Equation (1.10) to an equation of the form $\mathbf{C}_\mathbf{t}^L = \mathbf{C}_\mathbf{t}^R \cdot \mathbf{K}$, which can be verified by a generalization of the multiparty protocol by Baum and Nof for the verification of matrix multiplication of triples of matrices. At this point the MPC protocol can be rendered to a 5-pass protocol and a signature as explained before. The protocol benifit also from several improvements proposed in the literature, e.g. [DS23; Fen22].

| Scheme | pk (B) | sig (B) | KeyGen (Mcyc.) | Sign (Mcyc.) | Verify (Mcyc.) |
|---|---|---|---|---|---|
| Dilitium | 1312 | 2420 | 0.3007 | 0.3007 | 0.3007 |
| CROSS fast | 61 | 12944 | 0.10 | 6.76 | 3.17 |
| CROSS small | 61 | 10304 | 0.10 | 22.00 | 10.28 |
| SqiSIGN | 64 | 177 | 3728 | 5779 | 108 |
| SDitH small | 120 | 8241 | 3.2 | 5.1 | 1.6 |
| SDitH | 120 | 10117 | 1.7 | 4.4 | 0.6 |
| Mirith fast | 129 | 7661 | 0.1091 | 8.904 | 8.309 |
| Mirith small | 129 | 5665 | 0.1091 | 77.911 | 78.181 |

Table 1.1: Comparison of the principal post-quantum digital signatures based on the Fiat-Shamir paradigm.

**Comparison of the Schemes** In Table 1.1 we inserted a comparison of the average parameters and running times for the schemes introduced before. All this schemes are tailored to the NIST first level of security ($\lambda = 128$ bits).

You can see that clearly the lattice-based one has the best parameters overall, in fact it has already been chosen for standardization. However, a wise digital infracture should not rely on a single family of assumptions, thus further research is necessary. All the other schemes but SQISign have small public keys, balanced signing and verification time, but extensive signature sizes. Moreover, being derived from coding theory related problems, their parameters choices for the hardness assumptions are trusted since derived by established algorithms. The only exception is the R-SDP form Cross since, even if convincing, its cryptanalysis is much more recent. The isogeny based signature instead stand out negatively for its performance, even if still practical, but also positively for the very minimal public key and signature sizes, useful for several real world applications.

In Chapter 3 there are other schemes like MEDS and LESS, with again different characteristics, like smaller signature with respect to the public key size.

CHAPTER

# TWO

# CRYPTOGRAPHIC GROUP ACTIONS

In this chapter we describe a relatively new mathematical tool for defining cryptographic primitives: the group action.

We start with an overview of some standard properties, in particular the one relevant for cryptography. The main reference is [AFMP20], still we try to give an overview of different approaches used in the literature. Even if we give some instances of group action, we only discuss their hardness and quantum resistance in the generic case, i.e. without trying to tamper with the action internal structures.

Then in Section 2.2 we show how they can be used to instantiate a signature scheme. In Section 2.3 we then show how to manipulate the scheme to improve it with respect to the particular group action used.

For all the chapter we use multiplicative notation for $G$, and denoting with $e$ its identity element,

**Definition 2.1.** A *group action* is a function, shown below, that allows a group $G$ to act on a set $X$:

$$\begin{aligned} \star : G \times X &\to X \\ (g, x) &\to g \star x \end{aligned} \tag{2.1}$$

It is required to be *compatible* with the group; this means that

- for all $x \in X$ we have $e \star x = x$;
- for all $g, h \in G$, it holds that $h \star (g \star x) = (h \cdot g) \star x$.

We indicate it by the tuple $(G, X, \star)$. When not further specified a group action is always defined on the set $X$ and group $G$ as in Equation (2.1).

**Additional Definitions** There are other properties and definitions classically associated to group actions, in fact we say that a group action is:

- *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 the identity whenever there exists an $x \in X$ such that $x = g \star x$;
- *Regular*, if it is free and transitive.

**Definition 2.2.** Given a group action $(G, X, \star)$ for a set element $x \in X$ we can define its orbit as

$$\mathcal{O}(x) := \{g \star x \mid g \in G\}$$

and its stabilizer as

$$G_x := \{g \in G \mid g \star x = x\} .$$

For finite groups we have a classical result called Orbit-Stabilizer Theorem that implies

$$|G| = |\mathcal{O}(x)| \cdot |G_x| \quad \forall x \in X .$$

Notably, we can always have a transitive group action by restricting $X$ to an orbit $\mathcal{O}(x)$. If the action is also free we get by the theorem the equality $|G| = |X|$, that is trivially implied by the bijection $g \mapsto g \star x$ (this is also a consequence of the Theorem). This bijection implies that for any pair $x, y$ of elements in $X$ there exists one and only one group element $g$ with $g \star x = y$, we label this group element as $\delta_\star(x, y)$.

### 2.0.1 Effective Group Action

While regularity is very reasonable property for group actions, to comply with cryptographic purposes we need specific additional properties. First of all we need to efficiently work on the action, i.e. on the group $G$, the set $X$ and their interactions. By *efficiently* we always mean that there exists a probabilistic polynomial-time algorithm that solves the problem.

In [AFMP20] they characterize them as *Effective Group Actions* (EGA):

**Definition 2.3.** A group action $(G, X, \star)$ is said effective if:

1. It is possible to work efficiently on the group $G$, in particular we can perform efficiently each of these:
   (a) Given $g, h \in G$ compute their product $gh$ and the inverse $g^{-1}$;
   (b) Sample an element $g$ from $G$ using a distribution that is statistically close to the uniform;
   (c) Testing if one string represent a valid group element in $G$;
   (d) Deciding if two group elements $g, h \in G$ are actually equal.
2. It is possible to verify efficiently that a string corresponds to an element in $X$ and compute efficiently a unique representation for it;

3. There exists at least on element $x_0 \in X$ we can represent using a finite length string;
4. Given any $g \in G$ and $x \in X$ we can efficiently compute $g \star x$.

Observe that when using an orbit like set $\mathcal{O}(x)$ the origin described in point 3 is exactly the element $x$. In some cases it may not be achievable in practice to have an effective way to evaluate the group action on any group element (property 4), thus we may restrict ourselves to ask that the action is effective on a generating set $(g_1, ..., g_n)$ of polynomial size, i.e. such that $\langle g_1, ..., g_n \rangle = G$ and $n = \mathsf{poly}(\log(\#G))$. In this case we say the action is *Restricted* Effective Group Actions (REGA).

### 2.0.2 Cryptographic Group Actions

To add the adjective *cryptographic* to an effective group action $(G, X, \star)$ we also need it to define a *one-way* function, i.e. a function that is easy to evaluate but hard to invert.

**Definition 2.4** (Definition 9.4 of [AB09] adapted to [ADMP20])**.**
A polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is a *one-way function* with respect to the distribution $\mathcal{D}_{\mathsf{inp}}$ (indexed by the integer $\lambda$) if, for every probabilistic polynomial-time algorithm $\mathsf{Evl}$, there is a negligible function $\mathsf{negl}$, such that for every $\lambda$:

$$\mathbb{P}_{x \leftarrow \mathcal{D}_\lambda}\left[f(\mathsf{Evl}(f(x))) = f(x)\right] \leqslant \mathsf{negl}(\lambda) \ .$$

Eventually the function $f$ can depend on a set $P$ of public parameters $\mathsf{pp}$, sampled according to the distribution $\mathcal{D}_P$ (still indexed by $\lambda$) and known to the adversarial algorithm $\mathsf{Evl}$. In this case we can speak of a *one-way function family* with respect to the distributions $(\mathcal{D}_{\mathsf{inp}}, \mathcal{D}_P)$. When the distribution is the uniform one we omit it.

The natural problem that arises from group action is known as the *vectorization* problem, or sometimes *Group Action Inverse Problem (GAIP)*.

**Problem 7** (GAIP)**.** Given $x$ and $y$ in $X$, find, if any, an element $g \in G$ such that $y = g \star x$.

Usually we require that $x, y$ lie in the same orbit to be sure about the existence of a solution $g$. In other words, we require that given the public parameter $x \in X$, the functions $f_x : G \to X$ with $f_x(g) := g \star x$ are a one-way family. It can also be useful to define the decisional version of Problem 7:

**Problem 8** (d-GAIP)**.** Given $x$ and $y$ in $X$ decide if they lie in the same orbit, i.e. if there exists an element $g \in G$ such that $y = g \star x$.

Another 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.

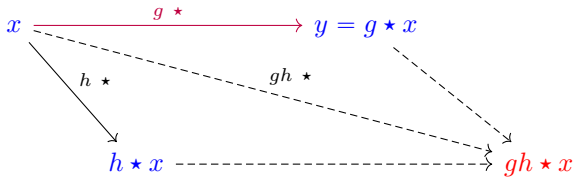



Figure 2.1: Here we can see why the name vectorization is associated to the task of finding the purple arrow (Problem 7), while parallelization to the task of finding the red element to complete the parallelogram composed by the blue one (Problem 9).

**Problem 9** (c-GADH)**.** Given $x$, $g \star x$ and $h \star x$, for $g, h \in G$, compute $(g \cdot h) \star x$.

In general this problem is of particular interest when $G$ is an abelian group, since it is the computational assumption for the group action version of the Diffie-Hellman key exchange.

Another possible problem for the group actions is relative to the search of non trivial stabilizer elements:

**Problem 10** (Stabilizer Computation)**.** Given $x \in X$ find $g \in G$ such that $g \neq e$ and $g \star x = x$, i.e. $g \in G_x \backslash \{ e\}$.

This problem is clearly impossible if the action is free. For several group actions the Stabilizer Computation Problem is also called Group Automorphism problem and has already been studied extensively, e.g. for code automorphisms see [Leo82; BBPS21].

**Hard Homogeneous Spaces** Usually in isogeny-based cryptography they use the terminology from the Couveignes's notes [Cou06]. There the author defines an *Hard Homogeneous Space* (HHS) as a group action $(G, X, \star)$, or an homogeneous space $X$ on $G$ to say it in an old-fashioned way, that is effective, with Problem 7 and Problem 9 difficult to solve.

A *Very Hard Homogeneous Space* is instead a HHS $X$ on $G$ with the additional requirement that the following problem is difficult:

**Problem 11** (Parallelization Testing)**.** Given a polynomial number of tuples $\{(x_i, y_i)\}_{i=0}^{n}$ of elements in $X$ decide if there exists a group element $g \in G$ such that $g \star x_i = y_i$ for all $i = 0, ..., n$.

This problem is equivalent to the decisional version of Problem 9, in which when given $(x, z_1, z_2, z_3)$ it is required to distinguish between the case $z_1 = g \star x, x_2 = h \star x$ and $z_3 = (gh) \star x$ from $z_3 \xleftarrow{\$} X$. In fact, by fixing $n = 1$, $(x_0, y_0) = (x, z_1)$ and $(x_1, y_1) = (z_2, z_3)$, the problems are equivalent for any transitive group action, since in this case there exists $g, h \in G$ with $z_1 = g \star x$ and $z_2 = h \star x$.

In [AFMP20] they propose more adequate definitions to the current state of literature. Consider a randomized oracle $\mathsf{DELPHI}_g$ that, on query set elements $x$ sampled according to $\mathcal{D}_X$, outputs $(x, g \star x)$ for a fixed $g \in G$

**Definition 2.5.** An HHS is an EGA such that for any probabilistic polynomial-time algorithm $\mathsf{Evl}$ with access to $\mathsf{DELPHI}_g$ we always have:

$$\mathbb{P}_{\substack{g \leftarrow \mathcal{D}_G, \\ x^* \leftarrow \mathcal{D}_X}} \left[ \mathsf{Evl}^{\mathsf{DELPHI}_g}(x^*) = g \star x^* \right] \leqslant \mathsf{negl}(\lambda) \ .$$

We also say that a group action with this property is $(\mathcal{D}_G, \mathcal{D}_X)$-*weak unpredictable*.

De facto we are asking that for a random $g \in G$, given a polynomial number of tuples $(x_i, g \star x_i)$ randomly sampled, it is difficult to compute $(x^*, g \star x^*)$ on a given $x^* \in X$. Consider also another randomized oracle $\mathsf{DELPHI}_{\mathcal{D}_X}$ that, when queried a set element $x$ sampled according to $\mathcal{D}_X$, answers $(x, u)$ for $u \leftarrow \mathcal{D}_X$.

**Definition 2.6.** A VHHS is an EGA such that for any probabilistic polynomial-time algorithm $\mathsf{Evl}$ we have

$$\left| \mathbb{P}_{g \leftarrow \mathcal{D}_G} \left[ \mathsf{Evl}^{\mathsf{DELPHI}_g}(1^\lambda) \right] - \mathbb{P}_{g \leftarrow \mathcal{D}_G} \left[ \mathsf{Evl}^{\mathsf{DELPHI}_{\mathcal{D}_X}}(1^\lambda) \right] \right| \leqslant \mathsf{negl}(\lambda) \ .$$

We also say that a group action with this property is $(\mathcal{D}_G, \mathcal{D}_X)$-*weak pseudorandom*.

De facto we are asking that, for a random $g \in G$, we cannot distinguish efficiently between a polynomial number of tuples $(x_i, g \star x_i)$ and a polynomial number of tuples $(x_i, u_i)$ sampled uniformly at random.

### 2.0.3 Examples of Group Actions

Group actions are not new to modern cryptography, the Discrete Logarithm Problem is a celebrated example. The first use of group actions non related to DLOG trace back to 1997 [Cou06][1], with the use of isogenies of elliptic curves. In general for any category the group of isomorphism $\mathsf{Isom}$ (with respect to composition) of such category acts on the objects of the category $x \in X$ in a natural way as $\phi \star x := \phi(x)$. By the properties of morphism of category we immediately get the compatibility with the group structure. We show some examples in the upcoming paragraphs.

**Group actions and the discrete log** From a finite cyclic group $G = \langle g \rangle$ of order $N$ we have a simple and classical example of EGA, the exponentiation:

$$\begin{aligned} \star : \mathbb{Z}_N^* \times G &\to G \\ (a, g) &\to a \star g := g^a \ . \end{aligned} \tag{2.2}$$

[1] The work was submitted and rejected in 1997, then never published until its rediscover

For this specific group action, the vectorization problem (Problem 7) is the well-known Discrete Logarithm Problem, which serves as one of the cornerstones of modern public key cryptography. It can be utilized to instantiate several renowned cryptographic protocols, such as Schnorr signatures, the El Gamal Encryption scheme, Diffie-Hellman key exchange, and others. It's worth noting that the modelization of the group action is reductive because we also have the option to multiply elements in $G$, resulting in:

$$(a \star g) \cdot (b \star g) = g^a \cdot g^b = g^{a+b} = (a+b) \star g \ . \tag{2.3}$$

Thanks to this additional property, more efficient protocols like Schnorr signatures have been developed. However, as we point out in Section 2.1.1, it also opens the door to potential polynomial quantum attacks.

Luckily, part of the nowadays used protocols do not directly rely on the use of multiplication, but just on the action of the group. A nice and useful example is the Diffie-Hellman like key exchange. In the classical scheme starting from the public parameters $G, g, N$ Alice and Bob decides secret keys $a, b \in \mathbb{Z}_N^*$ and publish respectively $A = g^a$ and $B = g^b$. At this point the secret is the group element $g^{ab} = A^b = B^a$. The same can be carried over by an abelian group action, as shown in Protocol 2.0.1.

| Public Data : Abelian group action $(G, X, \star)$ and a shared element $x \in X$ | |
|---|---|
| **Alice** | **Bob** |
| Generate $g_a \xleftarrow{\$} G$; | Generate $g_b \xleftarrow{\$} G$; |
| Set $x_a \leftarrow g_a \star x$; | Set $x_b \leftarrow g_b \star x$; |
| Send $x_a$; | Send $x_b$; |
| $\xleftrightarrow{x_a, x_b}$ | |
| Set $s = g_a \star x_b$; | Set $s = g_b \star x_a$; |
| Use $s = g_a g_b \star x = g_b g_a \star x$ as secret key. | |

Protocol 2.0.1: Diffie-Hellman like key exchange using abelian group action.

**Isogenies of Abelian Varieties** As explained before to introduce SQISign, isogenies are non constant morphism $\phi : E \to E'$ between the elliptic curves, fixing the identity. Also each separable morphism is associated to its kernel. These objects have very interesting properties, first of all the commutativity of the actions (modulo using some shrewdness for the objects used in the implementation), that makes post-quantum Diffie-Hellman like key exchanges (Protocol 2.0.1) feasible.

The hardest obstacle to overcame for isogeny based schemes is the complexity of securely computing the resulting elliptic curve via the Velù formulas [Vél71],[BDLS20a]. This were solved in the Supersingular Isogeny Key Encapsulation scheme (SIKE, [Jao+20]) by using supersingular elliptic curves (i.e.

curves with no non-zero points of order the field characteristic) and additional torsion points.

This points where used as an auxiliary information to speed up the computation, however researchers shown that they can be used also for an efficient key recovery attack, fully breaking the scheme [CD23; MMPPW23; Rob23].

Note that SIKE, even if much appreciated, was just one of the possible instantiation of isogeny-based cryptography and its dawn does not impact the security of other isogeny constructions like [CLMPR18] and [Cha+23]. Still the family is a fertile research field, with several directions open, both for small key signatures [DG19; BKV19], delayed encryption [BD21] and others [AFMP20].

**Isomorphism Problems** As hinted before the task of retrieving the isomorphism, if exists, between two category objects can be seen as a particular instance of GAIP (Problem 7). In general, these classes of problems do not rely on abelian isomorphism groups, but still they can be interesting for cryptographic purposes thanks, to the traversal interest received from the mathematics and computer science communities.

In Chapter 3 isomorphism problem related to coding theory are discussed, while here we can recall the well known Graph Isomorphism Problem from Graph Theory:

**Problem 12** (Graph Isomorphism Problem (GI))**.** Given two graphs of size $n$ $\Gamma = (V, E)$ and $\Gamma' = (V, E')$ find a graph isomorphism $\pi$, i.e. a permutation such that $\{v_i, v_j\} \in E$ if and only if $\{v_{\pi(i)}, v_{\pi(j)}\} \in E'$.

GI is an important reference for isomorphism problems, but it cannot be used in cryptography since it has been solved in quasi-polynomial time [Bab16]. Other equivalence problem can be defined from other mathematical structures, like lattices [DW22], tensors [GQ21] and knots [FGHLS12].

## 2.1 Theoretical Hardness

We go now through some hardness properties for a generic group action. An important one for the group actions derived problems is that they are *hard on average*, in the sense that a random instance is expected to be as hard as the hardest one. More formally:

**Definition 2.7.** A search or decision problem is said *nonadaptively $k$-random self reducible* if there exists two probabilistic polynomial-time algorithms $\phi, \sigma$ using a polynomial size randomness $\mathbf{r}$ (sampled uniformly) such that for any problem instance $\mathsf{x}$ with solution $\mathsf{sol}$:

- the output $\sigma(i, \mathbf{r}, \mathsf{x})$ for $i \in [1, k]$ are random instances of the initial problem and the output distribution is independent of the input instance $\mathsf{x}$;
- let $\mathsf{sol}_i$ be the solution associated to $\sigma(i, \mathbf{r}, \mathsf{x})$ for all $i \in [1, k]$, then

$$\mathbb{P}_{\mathbf{r}}\left[\phi(\mathsf{x}, \mathbf{r}, \mathsf{sol}_1, ..., \mathsf{sol}_k) = \mathsf{sol}\right] \geqslant \frac{2}{3} .$$

What the definition says is that, given any hard instance, we can sample random instances so that, if we are able to solve them, we can solve the search problem for the hard instance with constant probability.

In [DAl22a] they proved that the GAIP is hard on average, more precisely:

**Lemma 2.8** (Theorem 15 [DAl22a])**.** *The GAIP for a regular effective group action is nonadaptively* 1*-random self reducible.*

*Proof.* Given a instance $(x, y) \in X^2$ you can generate another random instance by generating from the randomness $\mathbf{r}$ two group elements $\tilde{g}_1, \tilde{g}_2$ and using them to generate $(\tilde{g}_1 \star x, \tilde{g}_2 \star y)$. This is clearly a valid instance because of the transitivity.

When having a GAIP solution $h$ to the instance we have

$$\tilde{g}_2 \star y = h \star (\tilde{g}_1 \star x) = h\tilde{g}_1 \star x \ .$$

By observing that:

$$y = \tilde{g}_2^{-1} h \tilde{g}_1 \star x \ ,$$

we get the witness $\tilde{g}_2^{-1} h \tilde{g}_1$ for $(x, y) \in X^2$ as required.

Also observe that since the action is regular and the group elements are sampled at random via $\mathbf{r}$ also the output is uniformly distributed, implying also it's independence from the instance $(x, y)$.

□

Clearly this property is very desirable for cryptographic primitives since this way we have a provable security for any instance used as public key.

However there is also a downside regarding its NP-completeness.

**Theorem 2.9** (Theorem 3.5 [FF93])**.** *If an* NP*-complete problem is nonadaptively* $k$*-random self reducible, then the Polynomial Hierarchy collapses at the third level.*

More on the Polynomial Hierarchy can be read in Section 5.2 [AB09], however, the important point is that many computational complexity researchers consider unlikely a collapse of the hierarchy. Thus as an immediate corollary of Theorem 2.9 and Lemma 2.8 is that GAIP is unlikely to be NP-complete for any group action.

### Generic Attacks

As for the Discrte Logarithm Problem, we can explore attacks that does not rely on the particular group action choice, but only on the group action operations. First of all we can do a meet-in-the-middle attack to the GAIP with complexity $O(\sqrt{\#G})$. The attacks is on the same line as the *Baby-step giant-step* algorithm, in fact we can create two lists $\{(\tilde{g}_i \star x, \tilde{g}_i)\}$ and $\{(\tilde{h}_i \star y, \tilde{h}_i)\}$ by sampling random group elements $\tilde{g}_i, \tilde{h}_i$. Then we search for collisions $i, j$ such that $\tilde{g}_i \star x = \tilde{h}_j \star y$, this way the solution is $\tilde{g}_i^{-1} \tilde{h}_j$. By the Birthday Paradox lists of size $\sqrt{\#G}$ are enough to find a collision with non negligible probability. This idea can be

traced back to the study of path finding in graphs from [Poh69]. Moreover, as done with Pollard Rho attack for DLOG, a low memory variant (with similar running time) can be found by performing pseudorandom walk on the graph. It is called GHS attack, in honour of the authors Galbraith, Hess, and Smart [GHS02]. Initially it was designed to find isogeny of elliptic curves, but it works in general and an implementation for a generic group actions can be seen in Section 4.3 of Stolbunov PhD thesis [Sto12].

### 2.1.1 Quantum Attacks

In the pivotal work [Sho94] it was introduced a quantum algorithm able to break the discrete logarithm and factorization problems in polynomial time. The procedure uses the Quantum Fourier Transform to estimate the phases of a unitary operator that lead a solution for the underlying problem.

However, this algorithm strongly relies on being able to multiply two elements as in (2.3). When using it to find an $a \in \mathbb{Z}_N$ such that $g^a = h$, the first step requires to compute the function $f(n, m) = g^n \cdot h^m$ in superposition:

$$\frac{1}{q}\sum_{n=0}^{q-1}\sum_{m=0}^{q-1}|n, m\rangle \quad \rightarrow \quad \frac{1}{q}\sum_{n=0}^{q-1}\sum_{m=0}^{q-1}|n, m, g^n h^m\rangle \ . \tag{2.4}$$

The algorithm proceeds with a measure on the third register on a random state $g^c$, so that the first two registers satisfy $n + ma = c$. Then the Quantum Fourier Transform put them again in another superposition, sieving all the pairs $n', m'$ that are not in phase, i.e. with $n' \neq am'$, so that any measurement lead to a solution with constant probability. The key point of the algorithm is that, thanks to the red term in (2.3), after the first measurement the second register is a function of the first one with phase $a$. When we consider instead a group action on a set $X$, where no compatible multiplication is defined, we cannot anymore tamper with $y = g^a \star x$ to end up with $g^{am+n} \star x$.

**Kuperberg's quantum algorithm** The Shor's breakthrough lead the study of quantum algorithms to solve group related problems. An important step in that direction is the Kuperberg's Algorithm [Kup05], which solves in subexponential time a natural generalization of the problem solved by Shor: the *hidden subgroup problem* for the dihedral group. This problem requires, given black box access on $f : G \rightarrow X$, to find the unique subgroup $H \leqslant G$ such that $f(a) = g(b)$ if and only if $ab^{-1} \in H$ (on the promise of its existence).

Quantum complexity of the hidden subgroup problem is deeply related to the security of lattices and isomorphism problems [Reg04b], justifying the cryptographic interest on it. Moreover there is a variant of the problem more related to *abelian* cryptographic group actions.

**Problem 13** (Abelian Hidden Shift Problem (HSP))**.** Given a finite abelian group $G$, a set $X$ and oracle access to two injective functions $f_1, f_2 : G \rightarrow X$ find $g \in G$ such that $f_1(h) = f_2(hg)$ for all $h \in G$.

The analogy with free abelian group actions and the vectorization problem is immediate, given a group action $(G, X, \star)$ and a Problem 7 instance $x, y = g \star x$ (with $g$ secret) we can define the two functions $f_1, f_2 : G \to X$ as $f_1(h) := h \star y$ and $f_2(h) := h \star x$ such that their shift is $g$:

$$f_1(h) = h \star y = h \star (g \star x) = (hg) \star x = f_2(hg) \ .$$

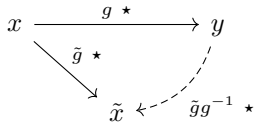
The following observation is used in [CJS14] to show that the vectorization problem for a free abelian group action can be solved by a quantum computer in sub-exponential time $O(2^{\sqrt{\log \# G}})$.

In particular the paper focuses on the problem of finding an isogeny between two elliptic curves, that can be reduced to abelian HSP under the Generalized Riemann Hypothesis. After the reduction the HSP is solved using a slight generalization of Kuperberg's result in [Kup05]. The major limitation for this algorithm is that also a sub-exponential amount of space is required; which, for our current understanding of quantum computations, is an important technical limitation. However they also adapt a slightly slower polynomial space solution from [Reg04a], with complexity bounded by $L_{\#G}(\frac{1}{2}, \sqrt{2})$, where

$$L_N(\alpha, c) := \exp\{(c + o(1))(\log N)^{\alpha}(\log \log N)^{1-\alpha}\} \ . \tag{2.5}$$

## 2.2 Signatures

We can finally how to render any group action to a secure digital signature. Let's start by definining a $\Sigma$-protocol for the $\mathsf{NP}$-relation induced by the GAIP (Problem 7). The statement is the pair $(x, y)$, while the withness is $g \in G$ such that $y = g \star x$. The idea is to commit to a random element $\tilde{x} = \tilde{g} \star x$ via an ephemeral map $\tilde{g} \xleftarrow{\$} G$, then the prover is challenged to disclose the link between $(x, \tilde{x})$ or between $(y, \tilde{x})$. The commutative diagram underlying the protocol, an important tool to manage the protocol, goes as follows:




You can see it also in Protocol 2.2.1. For simplicity we assume that the element $\tilde{g} \xleftarrow{\$} G$ is sampled according to the uniform distribution, even if there is the possibility of considering different distributions.

**Proposition 2.10.** *The 3-pass protocol in 2.2.1 is Complete, Special Sound for the relation induced by the GAIP and Honest Verifier Zero Knowledge under the collision resistence of the hash function* $\mathsf{H}$.

*Proof.* The protocol is clearly complete as it can be seen from the diagram. For the special soundness assume the knowledge of two valid transcripts $(\mathsf{com}, 0, a)$

| | | |
|---|---|---|
| Public Data : Group $G$ acting on $X$ via $\star$, element $x \in X$ and hash function $\mathsf{H}$. | | |
| 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 $\mathsf{com} = \mathsf{H}(\tilde{x})$ | $\xrightarrow{\mathsf{com}}$ | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0,1\}$. |
| If $b = 0$ then $\mathsf{resp} \leftarrow \tilde{g}$. | $\xrightarrow{\mathsf{resp}}$ | Accept if $\mathsf{H}(\mathsf{resp} \star x) = \mathsf{com}$. |
| If $b = 1$ then $\mathsf{resp} \leftarrow \tilde{g}g^{-1}$. | | Accept if $\mathsf{H}(\mathsf{resp} \star y) = \mathsf{com}$. |

Protocol 2.2.1: Identification protocol for the knowledge of the private key.

and $(\mathsf{com}, 1, b)$. Since the hash function is collision resistant we have that $\mathsf{H}(\tilde{x}) = \mathsf{H}(\tilde{x}')$ implies $\tilde{x} = \tilde{x}'$, thus we have:

$$a \star x = \tilde{x} \ = b \star y \ ,$$

thus we can extract the witness $b^{-1}a$ for the GAIP.

For the Honest Verifier Zero Knowledge we consider the simple simulator $\mathsf{Sim}$ that on input $x$ and $\mathsf{ch}$ output the following.

- When $\mathsf{ch} = 0$ sample $\tilde{h} \xleftarrow{\$} G$ and output $\mathsf{com} = \mathsf{H}(\tilde{h} \star x)$ and $\mathsf{resp} = \tilde{h}$. This transcript is exactly the same as in the protocol so it is clearly indistinguishable.

- When $\mathsf{ch} = 1$ sample $\tilde{h} \xleftarrow{\$} G$ and output $\mathsf{com} = \mathsf{H}(\tilde{h} \star y)$ and $\mathsf{resp} = \tilde{h}$. This transcript has the same distribution as one generated by the use of the secret $g$. In fact since the multiplication by a group element is a bijective map then $\tilde{g}$ and $g^{-1}\tilde{g}$ have the same uniform distribution, thus $\tilde{h}$ and $g^{-1}\tilde{g}$ are indistinguishable also to an unbounded adversary.

□

Thanks to the special soundness the above protocol provides a soundness error of $1/2$ (the size of the challenge space), that can be amplified by simply repeating the protocol $\lambda$ times using independent challenges, resulting on a soundness error of $2^{-\lambda}$. Thus we are ready to use the Fiat-Shamir transform (Algorithm 1) as in Section 1.1.3 to render the $\Sigma$-protocol to a digital signature EUF-CMA secure under the hardness of GAIP in the Random Oracle Model, thanks to Theorem 1.11.

The first protocol using this construction, even if only sketched, traces back to [Sto12], while the first practical implementation is the SeaSign scheme [DG19], using the CSIDH group action [CLMPR18]. SeaSign also employ rejection sampling techniques since, in general, it is not possible to sample random ideals in the class group (i.e. get $g \xleftarrow{\$} G$). The rejection sieves all elements with ideal powers over a threshold, similarly as it was done in the Crystarls-Dilithium scheme [Duc+18]. However note that commutativity is not required

**Algorithm 4** Digital signature based on the GAIP

$\mathsf{Setup}(1^\lambda)$:
1: decide a cryptographic group action $(G, X, \star)$;
2: decide the hash $\mathsf{H}$;
3: **return** $\mathsf{pp} = \langle (G, X, \star), \mathsf{H} \rangle$

$\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma)$:
1: parse $(\mathsf{com}, \mathsf{resp}_1, ..., \mathsf{resp}_\lambda) \leftarrow \sigma$;
2: parse $\mathsf{com}$ as $\mathsf{ch}_i$;
3: assign $(x_0, x_1) \leftarrow \mathsf{pk}$;
4: compute $\tilde{y}_i \leftarrow \mathsf{resp}_i \star x_{\mathsf{ch}_i}$;
5: accept if $\mathsf{com} = \mathsf{H}(\tilde{y}_1 \| ... \| \tilde{y}_\lambda \| \mathsf{m})$;

$\mathsf{KeyGen}(\mathsf{pp})$:
1: Sample $x \xleftarrow{\$} X$, $g \xleftarrow{\$} G$ ;
2: **return** $\mathsf{pk} \leftarrow (x, g \star x)$, $\mathsf{sk} \leftarrow g$.

$\mathsf{Sign}(\mathsf{sk}, \mathsf{m})$:
1: $\tilde{g}_i \xleftarrow{\$} G$;
2: $\tilde{x}_i \leftarrow \tilde{g}_i \star x$;
3: $\mathsf{com} \leftarrow \mathsf{H}(\tilde{x}_1 \| ... \| \tilde{x}_\lambda \| \mathsf{m})$;
4: parse $\mathsf{com}$ as $\mathsf{ch}_i$;
5: $\mathsf{resp}_i \leftarrow \tilde{g}_i g^{-\mathsf{ch}_i}$;
6: **return** $\sigma = (\mathsf{com}, \mathsf{resp}_1, ..., \mathsf{resp}_\lambda)$

for this construction, so also each isomorphism problem can be used to render signature schemes. The more intesting one are presented later in Section 3.4.1 and 3.4.2.

### 2.2.1 Security in the Quantum Random Oracle Model

Since this protocol is meant to be quantum resistant we need to investigate its security in the quantum random oracle model. We use the results in Section 1.1.4 integrated with [Blä+22], where they are adapted to the group actions framework.

We focus now on the computationally unique response property for Protocol 2.2.1. Consider a valid transcript $(\mathsf{com}, 0, \tilde{g})$ for Protocol 2.2.1. The only strategy to generate another transcript is to search another element $h \in G \backslash \{\tilde{g}\}$ such that $\mathsf{H}(h \star x) = \mathsf{com}$, that implies, assuming the hash function collision resistance, $h$ satisfies $h \star x = \tilde{g} \star x$. Hence $x = h^{-1}\tilde{g} \star x$, thus we have found a non trivial element in the stabilizer of $x$. On the contrary, from any non trivial element $s \in G_x$ we can find $h$ as $\tilde{g}s$. Said this it makes sense to consider the Stabilizer Computation Problem (Problem 10) that requires, given $x \in X$, to find $g \in G$ non trivial such that $g$ is in the stabilizer $G_x$. So we get the following.

**Lemma 2.11** (Lemma 2 [Blä+22])**.** *The $\Sigma$-protocol in Protocol 2.2.1 has computationally unique response if and only if the Stabilizer Computation Problem (Problem 10) cannot be solved by any quantum probabilistic polynomial-time adversary with non-negligible probability, assuming the collision resistance of the hash function used.*

The proof can be derived directly from the discussion above; eventually it can also be read on [Blä+22]. We point out that perfect unique response property is equivalent to require that the stabilizer of each set element in $X$ is trivial (see Lemma 1 of [Blä+22]). We can finally combine Lemma 1.12 and Theorem 1.15

to immediately have:

**Corollary 2.12.** *For any group action such that the GAIP (Problem 7) and the Stabilizer Computation Problem (Problem 10) are quantum resistant then the signature obtained rendering Protocol 2.2.1 via the Fiat-Shamir transform is sEUF-CMA secure in the quantum random oracle model.*

Since free group actions have only trivial stabilizer they satisfy immediately the perfect unique response, thus we have also:

**Corollary 2.13.** *For any free group action such that the GAIP (Problem 7) is quantum resistant then the signature obtained rendering Protocol 2.2.1 via the Fiat-Shamir transform is sEUF-CMA secure in the quantum random oracle model.*

## 2.3 Optimizations

Assuming that set elements can be represented with $l_X$ bits, while the group elements with $l_G$ bits, the digital signature presented in Algorithm 4 has the following specifics. Note that for the elements generated at random, like $x$ in the public key, we can use instead the seed of size $\lambda$ used for element generation.

- *Public key size*: $\lambda + l_X$ bits.
- *Signature size*: $l_G \cdot \frac{\lambda}{2} + \lambda \cdot \left(\frac{\lambda}{2} + 3\right)$ bits on average. Here the size is non constant since we can use the seed optimized representation only on the challenge 0. The $3\lambda$ additional bits are for the commitment and for the salt (see Remark 14).
- *Signing time*: $\lambda$ group action computations and around $\frac{\lambda}{2}$ group action inversion and multiplications (usually these are neglectable like the hash computations).
- *Verification time*: $\lambda$ group action computations.

*Remark* 14 (A Salty Remark). As pointed out in [Cha22] a straight use of a pseudorandom algorithm, that from $\mathsf{seed}$ recover the elements, opens the possibility for a collision search attack. Thanks to the birthday paradox it cuts the security to half the seed size. However in the same article they provided a lightweight fix which only require the use of a single fresh salt for each signature of length $2\lambda$. The fix works as follows: for each signature, when calling the pseudorandom function for the $i$-th time, use as seed: $\mathsf{seed}\|\mathsf{salt}\|i$. The salt can then simply be inserted in the signature.

Now we go through several techniques used in the literature to improve some of these parameters (eventually at the cost of worsening others). Even if this optimizations are general, a combination of them is each time tailored for the specific characteristics of the group action used.

### 2.3.1 Graph Topology

As said before when on $\mathsf{ch}_i = 0$ the prover is asked for the ephemeral group element $\tilde{g}$ it can be compressed to the $\lambda$ bits seed used for its generation, 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 to return a vector of fixed weight $w$ and length $t$, with $\binom{t}{w} \geqslant 2^{\lambda}$ to ensure the same level of security. This optimization is in fact known in the literature as *fixed-weight challenges*. This idea is widely used in the literature, for example in [BBPS21; Cho+22; BKP20].

Another idea proposed in [Jou23] and [BPS23] tries to optimize the protocol via using a particular commitment strategy that resembles multiparty computations techniques used for other Fiat-Shamir like signatures.

These two optimizations can both be framed as a particular case of a generalized version of Protocol 2.2.1, in which the commitment can be seen as a graph with set elements as nodes. The idea of use a graph abstraction to study the signature induced by group actions is a powerful tool, for example we see it with threshold functionalities, but also in [BGZ23] it was used to model all the possible the commitment strategies and obtain a lower bound on the signature size. In the following, this idea is used in a constructive way, following the path of [BPS23].

The idea is to generate $N$ commitments $\tilde{x}_1, ..., \tilde{x}_N$ using any undirected graph $\Gamma$ of $N + 2$ vertices in which

- two vertices are $x$, $y$;
- for all the other $\tilde{x}_1, ..., \tilde{x}_N$ there exists a unique trail going from $x$ or $y$ to each of them, like in figure 2.2a.

The graph contains exactly $N$ edges. Nodes represent set elements, while edges are ephemeral group elements, i.e. generated from a seed. Clearly if $y = g \star x$ all the set elements in the graph belong to the same orbit.

The intuitive idea behind this construction is that, since we are assuming that $\lambda \ll l_G$, the paths on the graph are a lightweight link between them.

The challenge is a subset $I \subset \{1, ..., N\}$ taken from a family $\mathcal{F}$. To answer it the prover has to send group elements such that he can link $x$ to the elements $\tilde{x}_i$ for $i \in I$ and $y$ to the elements $\tilde{x}_i$ for $i \notin I$. Let's define the protocol in detail, with some examples.

**Commit.** The prover generate at random $N$ group elements $\tilde{g}_1, ..., \tilde{g}_N$ and assign one of them per each edge of $\Gamma$. The he generate $\tilde{x}_1, ..., \tilde{x}_N$ following the trails, as in Figure 2.2a. The prover at this point can commit to the values $(\tilde{x}_1, \ldots, \tilde{x}_N) = \mathsf{com}$.

**Challenge.** The verifier can chose any subset of the family $\mathsf{ch} = I \xleftarrow{\$} \mathcal{F} \subseteq 2^{\{1,\ldots,N\}}$. This is assigned to the element $x$, while the complement $I^c$ is assigned to $y$, as show for example in Figure 2.2b.

**Response.** The response to $\mathsf{ch} = I$ is composed by two sets of group elements corresponding to vertices in the graph.

– The first set of vertices/group elements links all the element in $I$ to $x$

– The second set of vertices/group elements links all the element in $I^c$ to $y$

They can be evaluated via the knowledge of the secret key $g$ and the ephemeral group elements $\tilde{g}_1$, ..., $\tilde{g}_N$, as show for example in Figure 2.2c.

The response is composed always by at most two paths of $N$ group elements starting from $x$ and $y$, that can be choose in different ways. Intuitively to reduce the weight of the response we need to find the paths having the most edges already contained in $\Gamma$. To have a unique response for each challenge we need to force a policy on the sent paths. For instance, since for all the nodes there is unique path from $x$ or $y$ we can use the following policy:

– if two nodes both in $I$ or $I^c$ are already linked send the link;

– if a trial is not all contained in $I$ just send the link between $x$ and the node closest to the origin $x$ or $y$; same with $y$ if it is not all contained in $I^c$.

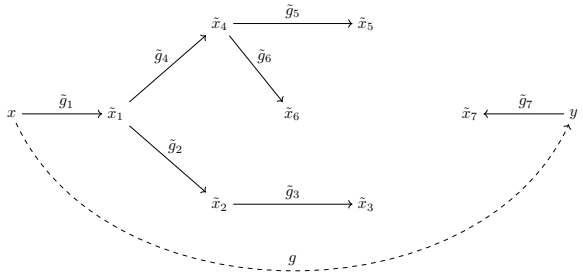
**Verification.** To verify the correctness of the response the verifier need to reach all the nodes $r_i$ and compute them via the received paths starting from $x$ for all $i \in I = \mathsf{ch}$ and from $y$ for all $i \in I^c$. At the end then he checks that $\mathsf{com} = (r_1, \ldots, r_N)$ otherwise he rejects.

**Proposition 2.14.** *Given any integer $N$ and family $\mathcal{F} \subset 2^{\{1,\ldots,N\}}$ the identification protocol for the relation $y = g \star x$ described above is complete, Honest-Verifier Zero Knowledge, has soundness $\frac{1}{\#\mathcal{F}}$ and quantum computationally unique responses under the assumption of the hardness of Problem 10.*

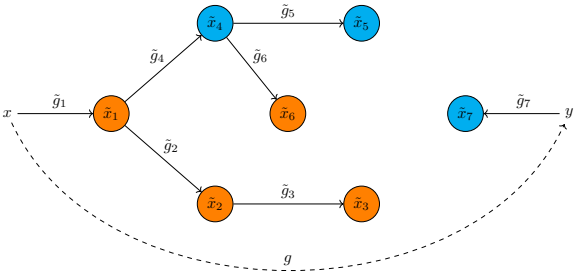
*Proof.* *Completeness.* As seen in Figure 2.2c, by using the knowledge of the link $g$ between $x$ and $y$ and the trails of $\Gamma$ an honest verifier can always link any node to $x$ or $y$.

*Honest-Verifier Zero Knowledge.* A simulator that knows in advance the challenge set $I$ can generate ad-hoc nodes from $x$ (and $y$) without the knowledge of the secret $g$. When possible, he may follow the trails as in the commitment, then for the ones linked to $y$ simply generate a random group element as it is done for the HVZK property in Proposition 2.10. For the same reasons as in Proposition 2.10 the generated nodes and edges (group elements) have the same distribution.

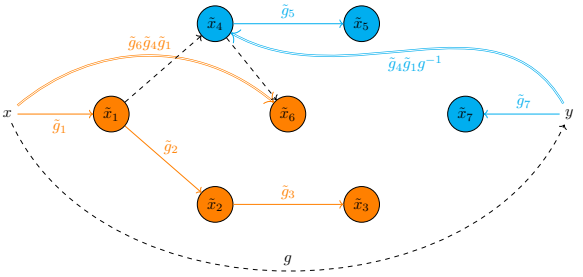
*Special soundness.* Suppose that the prover can answer two different challenges $I$, $J$. Thus it exists at least an index $k$ in the symmetric union $I \Delta J$, wlog assume that $k \in I$ and $k \in J^c$. Thus (assuming the collision resistance) from the response to $I$ they have both a trail linking $x$ to $\tilde{x}_k$, i.e. a product of group elements $\tilde{g}_k^I$ such that $\tilde{x}_k = \tilde{g}_k^I \star x$, and a trail linking $y$ to $\tilde{x}_k$, i.e. a product of group elements $\tilde{g}_k^J$ such that $\tilde{x}_k = \tilde{g}_k^J \star y$. So by considering $\left(\tilde{g}_k^J\right)^{-1} \cdot \tilde{g}_k^I$ we get a solution to the group action inverse problem. From the special soundness we get that the soundness error is $|\mathcal{F}|^{-1}$.




(a) Initial graph




(b) Random subset $I = \{1, 2, 3, 6\}$ as challenge.




(c) Response to the previous challenge. It consists on maps that are able to complete the graphs.

Figure 2.2: Example of action subgraph optimization.

*Quantum CUR.* We use the same idea of Lemma 2.11. Suppose we have two valid different responses from the same pair commitment and challenge. Since we ruled out using different paths for the same challenge, there is at least one link $(r_i, r_j)$ so that the group elements $g_{i\to j}, g'_{i\to j}$ in the responses are different. Hence $r_j = g_{i\to j} \star r_i = g'_{i\to j} \star r_i$, that implies $g^{-1}_{i\to j} g'_{i\to j} \in G_{r_i}$ with $g^{-1}_{i\to j} g'_{i\to j} \neq e$, against the hardness of Problem 10. □

**Uses for the graph** Now we can try several combination for the graph $\Gamma$ and the challenges set $\mathcal{F}$ to improve the protocol. The classical version of the protocol corresponds to fixing $N = \lambda$,

$$\Gamma = (\{x, y, \tilde{x}_1, ..., \tilde{x}_N\}, E) \text{ with } E = \{\{x, \tilde{x}_i\}\}_{i=1}^{N}$$

and $\mathcal{F} = 2^{\{1,...,N\}}$. An example can be seen in Figure 2.3a, where it is shown how to commit to 6 set elements, while in Figure 2.3b there is an example of response to the challenge $I = \{1, 2, 4, 6\}$, where the non-ephemeral group elements are marked with the double arrow.

### 2.3.2 Fixed-Weight Challenges

To obtain the fixed-weight challenges optimization we can consider a graph with the same topology as the classical one, $N = t$ and the challenge set composed by the subsets of cardinality $w$. This way we have a soundness error of:

$$\epsilon = \binom{t}{w}^{-1} \leqslant 2^{-\lambda} .$$

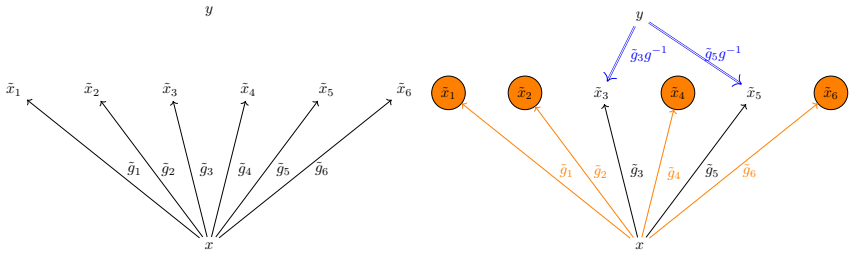
Hence we have a (reduced) signature size of $(t - w + 3)\lambda + wl_G$, but the signing and verification requires the computation of $t > \lambda$ group actions. This way we obtain a trade-off between space and time efficiency, that can be adjusted in relation to the particular use cases.

Only for heuristic we can estimate asymptotically the overhead computation necessary to get a signature of size $l_\sigma$ for a group action, assuming that $\lambda \ll l_G$ and the most expensive procedure during the signature is the group action computation. If $N$ is the total number of rounds then we can estimate it using the Stirling approximation[2] for the binomial coefficient:

$$\underbrace{\lambda \sim \log_2 \binom{N}{t}}_{\text{security req.}} \simeq t \log_2(N) - \log_2(t!) \simeq t \log_2(N) - t \log_2(t) . \tag{2.6}$$

Since $\lambda$ is negligible we can assume that $l_\sigma \simeq l_G t$, with $t$ the number of non-ephemeral rounds. Thus combining the two estimations we get

[2] $n! \simeq (n/e)^n$, we use $\simeq$ to highlight that they are not asymptotic according to the classical definition since there is also a $\sqrt{2\pi n}$ term.



(a) Initial graph (b) Response to the challenge

$$\log_2(N) \simeq \frac{\lambda l_G}{l_\sigma} + \log_2\left(\frac{l_\sigma}{l_G}\right) \implies \underbrace{N \simeq \frac{l_\sigma}{l_G} 2^{\frac{\lambda l_G}{l_\sigma}}}_{\text{approx. complexity}} . \tag{2.7}$$

### 2.3.3 Seed Tree

As said before the ephemeral elements can be recovered efficiently from the seed used in their pseudorandom generation, that usually is much more light then a full group element. In [BKP20] they showed how to improve this compression by the use of a Merkle-like structure: the *Seed tree*.

The idea is to generate the seeds in a recursive way, via the use of a binary complete tree. The tree has on each node a $\lambda$-bit string, while each pair of children nodes is generated from the parent $\mathsf{node}$ parsing the output of a secure pseudo random function $\mathsf{Childs} : \{0,1\}^* \to \{0,1\}^{2\lambda}$, i.e. the first $\lambda$ bits of $\mathsf{Childs}(\mathsf{node})$ are the left child, while the last $\lambda$ ones are the right child. Thus from a root seed $\mathsf{seed}_{\mathsf{root}}$ we have generated $2^M$ leaves $\{\mathsf{leaf}_i\}_{i=1}^{2^M}$ (where $M$ is the height of the tree). Each of this efficiently generated leaves can then be used as seed in the protocol.

Note that to ensure collision resistance of the expander function $\mathsf{Childs}$ we need it to take as input $\mathsf{node}\|\mathsf{salt}\|i$, where $i$ is a unique predetermined index for the call, as explained in Remark 14.

In general strategies like this are already in use to generate the randomness used in cryptographic protocols, but they can also save space for the signature, with negligible increase in complexity. In fact, when you need to reveal $t-w$ out of the $t \leqslant 2^M$ seeds generated[3] you just need to reveal the appropriate sequence of nodes used during the computations. This way instead of using $\lambda(t-w)$ seeds you can use $\lambda N_{\mathsf{seed}}$, where this value can be upperbounded as:

$$N_{\mathsf{seeds}} = 2^{\lceil \log(\omega) \rceil} + \omega(\lceil \log(t) \rceil - \lceil \log(\omega) \rceil - 1) \; ; \tag{2.8}$$

for example for $t = 247$, $w = 30$ and we end up using $91.2\lambda$ bits instead of $247\lambda$.

To use this functionality we need three functions, described in [BKP20]:

[3] We use this $t, w$ notation to complain with the notation used in the literature

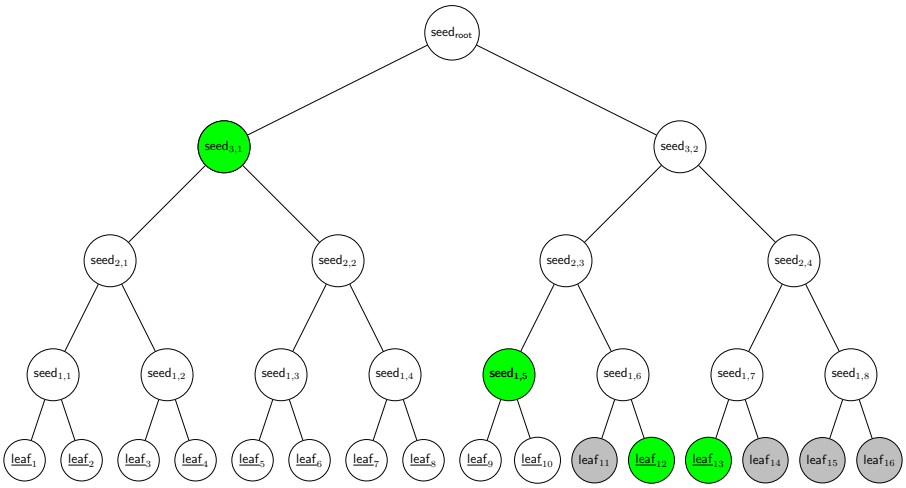



Figure 2.4: Example of seed tree structure, here $\mathsf{seed}_{\mathsf{root}}$ generates 16 leaves. Note how the 4 nodes in green cover all the leaves underlined, without disclosing information on the gray ones.

- $\mathsf{SeedTree}\,(\mathsf{seed}_{\mathsf{root}}, t) \rightarrow \{\mathsf{leaf}_i\}_{i=1}^{t}$ : On input a root seed $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^{\lambda}$ and an integer $t \in \mathbb{N}$, it constructs a complete binary tree with at least $t$ leaves by recursively expanding each seed using the functionality $\mathsf{Childs}$.
- $\mathsf{ReleaseSeeds}\,(\mathsf{seed}_{\mathsf{root}}, \mathsf{ch}) \rightarrow \mathsf{seed}_{\mathsf{internal}}$ : On input a root seed $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^{\lambda}$, and a challenge $\mathsf{ch} \in 0,1^{t}$, it outputs a list of seeds $\mathsf{seed}_{\mathsf{internal}}$ associated to nodes. These nodes covers the set of leaves with index $i$ satisfying $\mathsf{ch}_i = 0$. This means that union of all the leaves contained in subtrees rooted at each of them is equal to the covered set of leaves.
- $\mathsf{RecoverLeaves}\,(\mathsf{seed}_{\mathsf{internal}}, \mathsf{ch}) \rightarrow \{\mathsf{leaf}_i\}_{i \text{ s.t. } \mathsf{ch}_i=0}$ : On input the nodes $\mathsf{seed}_{\mathsf{internal}}$ and a challenge $\mathsf{ch}$, it computes and outputs all the leaves of covered by the nodes, that it is actually $\{\mathsf{leaf}_i\}_{i \text{ s.t. } \mathsf{ch}_i=0}$.

**Security** In Lemma 1 [BKP20] they showed that, when modelling the function $\mathsf{Childs}$ as a random oracle, the seeds obtained by the function $\mathsf{ReleaseSeeds}$ using $\mathsf{seed}_{\mathsf{root}}$ are indistinguishable by the one generated via a simulator without access to $\mathsf{seed}_{\mathsf{root}}$ for any computationally unbounded adversary making up to a polynomial number of calls to the random oracle.

### 2.3.4 MPC for group actions

To obtain the same protocol as in [BPS23; Jou23] we need to consider a different topology:

$$\Gamma = (\{x, y, \tilde{x}_1, ..., \tilde{x}_N\}, E) \text{ with } E = \{(x, \tilde{x}_1)\} \cup \{\{\tilde{x}_i, \tilde{x}_{i+1}\}\}_{i=1}^{N-1} ; \qquad (2.9)$$

while the challenge set is:

$$\mathcal{F} = \{\varnothing\} \cup \{[1, M]\}_{M=1}^{N} .$$

This way the soundness error is $(N+1)^{-1}$, while response requires:

– only ephemeral elements when $\mathsf{ch}$ is the full set $[1, N]$, all generated from a single root seed;

– one non-ephemeral element in the other cases, plus $\log_2(N)$ leaves to recover the ephemeral elements from the seed tree.

An example of this setting for the graph can be seen in Figure 2.5. Note that we can indentify the challenges using only the challenge set size $\#I \in [0, N]$.

This technique is associated with MPC in the Head (Section 1.2.1) because what its happening here can be seen as $N+1$ users evaluating the group action $g \star x$ via passing through set elements $\tilde{x}_i$, obtained by applying in sequence random group elements $\tilde{g}_i$, until the last user applies $g \cdot \tilde{g}_1^{-1} \cdots \tilde{g}_N^{-1}$. Then, on challenge $j$, the computations of all the users, but the $j+1$-th one, are disclosed and verified.

By repeating this MPC version of the protocol $t = \lceil \lambda / \log_2(N+1) \rceil$ times we would get a signature with average size:

$$t \left( \frac{N}{N+1}(l_G + \log_2(N)\lambda) + \frac{1}{N+1}\lambda \right) \text{ bits} , \tag{2.10}$$

that requires the computation of $tN$ group actions both for the signature and the verification. We can estimate the asymptotic complexity here as well, similar to what was done in Equation (2.7), under the assumption that $\lambda \ll l_G$. For a secure parameter set, we can express the relationship as $t \log_2(N) \sim \lambda$, and the signature satisfies $l_\sigma \simeq t l_G$ (ignoring the $\log_2(N)\lambda$ term). Since the most intensive part of the signature algorithm involves the evaluation of $tN$ group actions, we can approximate the complexity as:

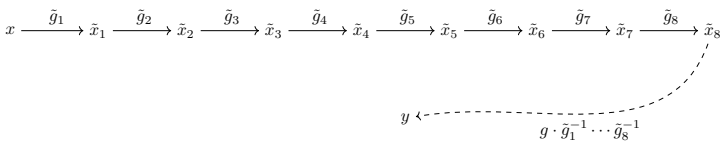
$$tN \simeq \frac{l_\sigma}{l_G} 2^{\frac{\lambda}{t}} \simeq \frac{l_\sigma}{l_G} 2^{\frac{\lambda l_G}{l_\sigma}} . \tag{2.11}$$

While this is a heuristic approximation, it's evident that this different approach doesn't significantly improve the parameters compared to a fixed weight challenge model.

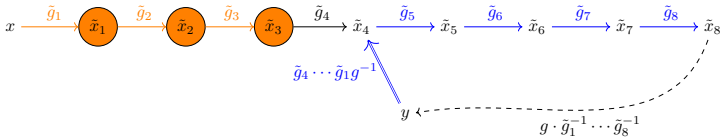
However, a natural progression from this approach would be to unbalance the challenges, as discussed in Section 2.3.2. This involves considering a challenge set $\mathcal{F}$ with $t - w$ full sets and $w$ sets of cardinality $< N$. In this case, the cardinality can be expressed as:

$$\#\mathcal{F} = \underbrace{\binom{t}{w}}_{\mathsf{ch}=N+1} \cdot \underbrace{N^w}_{\mathsf{ch}\neq N+1} . \tag{2.12}$$

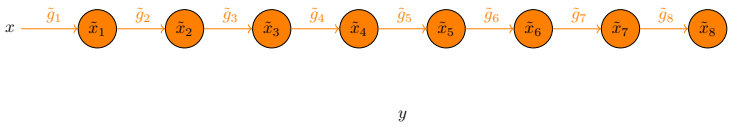
When $\#\mathcal{F} > 2^\lambda$, we obtain a signature of size:



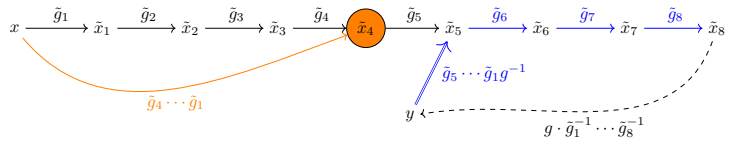

(a) Initial graph




(b) Response to the challenge $I = [1, 3]$ ($\#I = 3$).




(c) Response to the challenge $I = [1, 8]$ ($\#I = 8$).




(d) Optimized response to the challenge $I = [1, 4]$ ($\#I = 4$).

Figure 2.5: Example of action subgraph optimization on different challenges.

$$w(l_G + \log_2(N)\lambda) + (t - w)\lambda \ . \tag{2.13}$$

This approach allows for a more flexible challenge model, which can be advantageous in certain scenarios. However still it cannot improve with respect to the fixed weight challenges, as shown in [BPS23]. They proved this by considering two protocols,

- one using the MPC like protocol with $N > 1$, $t$ rounds and $w$ non-ephemeral rounds and soundness $\epsilon$;
- the other using fized weight challenge with $t' = tN$ rounds, $w' = w$ non-ephemeral rounds and soundness $\bar{\epsilon}$.

The signature size is almost the same for both protocols since they have the same number of non-ephemeral elements in the response, with only slight differences due to the strategies used for seed compression. Clearly both protocols require $tN$ group action computations, so they have almost equal running times.

So we now compare the two soundness errors via the following chain of inequalities:

$$\epsilon^{-1} = \binom{t}{w} N^w = \left(\prod_{i=0}^{w-1} \frac{t-i}{w-i}\right) N^w = \prod_{i=0}^{w-1} \frac{N(t-i)}{w-i} = \\ = \prod_{i=0}^{w-1} \frac{tN - iN}{w-i} < \prod_{i=0}^{w-1} \frac{tN - i}{w-i} = \binom{tN}{w} = \bar{\epsilon}^{-1} \ . \tag{2.14}$$

Thus we have that for same computational complexity and communication cost the one based on fixed weight has smaller soundness error, making it preferable from any point of view. Hence, as showed in [BPS23], the protocol based on MPC does not make sense as it is, but still there is a possible use.

In fact, there is a possibile modification to the verification (and the response), preserving the same soundness, but reducing the group actions computations required, thus decreasing the verification time. The modified rounds goes as:

**Response.** In response to the challenge $i$ send all the group elements, but $\tilde{g}_{i+1}$, as $h_j = \tilde{g}_j$ and $h_{i+1} = \tilde{g}_{i+1}\tilde{g}_i \cdots \tilde{g}_1 g^{-1}$, as usual. Then add $t_j = \tilde{x}_j$ for $j = 1, ..., i-1$.

**Verification.** From the responses $h_1, ..., h_N$ evaluate:

1. $r_i = (\tilde{g}_i \cdots \tilde{g}_1) \star x$ (only one group action);
2. $r_{i+1} = h_{i+1} \star y$;
3. $r_j = h_j \star r_{j-1}$ for $j > i + 1$.

so in the end he verifies $\mathsf{com} = (t_1, ..., t_{i-1}, r_i, r_{i+1}, ..., r_N)$. Note that in real situation the commitment is compacted via hash function and Merkle-like structure, thus also $t_1, ..., t_{i-1}$ would have negligible impact on communication cost.

Note that the verification now requires only $N - i + \mathbb{1}(\mathsf{ch} > 0)$ group actions computations, cutting the verification time to an average of

$$\underbrace{t - w}_{\mathsf{ch}=N} + w \underbrace{(N/2 + 1)}_{\mathsf{ch}<N} = t + w\frac{N}{2} \text{ group actions.} \tag{2.15}$$

### Hypercube MPC for Group Actions

The Hypercube technique explained in Section 1.2.1 can be adapted also for abelian group actions (additive notation). The strategy start again from the graph in Equation (2.9), that match up with the multiparty computation of $y = g \star x = (\tilde{g}_{N+1} + \tilde{g}_N + \cdots \tilde{g}_1) \star x$, for $\tilde{g}_{N+1} = g - (\tilde{g}_N + \cdots + \tilde{g}_1)$. Lets fix $N + 1 = r^n$ and consider the $n$ partition of size $r$ given by the hypercube slices on each dimension.

Suppose that $S_1 \sqcup \cdots \sqcup S_r$ is one of these, thus

$$g = \sum_{i \in S_1} \tilde{g}_i + \cdots + \sum_{i \in S_r} \tilde{g}_i \ ,$$

and the resulting group action evaluation goes as:

$$x \xrightarrow{\sum_{i \in S_1} \tilde{g}_i} x_1 \xrightarrow{\sum_{i \in S_2} \tilde{g}_i} x_2 \text{ --------- } x_{r-1} \xrightarrow{\sum_{i \in S_r} \tilde{g}_i} y \quad .$$

Since the structure is the same as for the classical MPC version of the protocol (where we commit to $x_1, ..., x_{r-1}$) $r - 1$ group actions are required, both for the commitment and the verification phase. Instead the $r - 1$ group elements necessary for the response can be obtained by summing the original group elements.

More formally, the partitions can be defined using the function $\mathsf{dig}_r(n, d)$, that returns the $d$-th digits of $n$ in base $r$, e.g. $\mathsf{dig}_2(10, 2) = 1$. In the $d$-th partition all group elements with equal $d$-th digit are grouped together. So to commit the group elements on dimension $d = 0, ..., n - 1$ Prover start from $\tilde{x}_0^d = x$ and evaluates:

$$\tilde{x}_j^d = \left( \sum_{\mathsf{dig}_r(i-1,d)=j-1} \tilde{g}_i \right) \star \tilde{x}_{j-1}^d \quad \text{for } j = 1, ..., r - 1 \ .$$

The key point now is that, after receiving a challenge for all of them, a unique response composed by $r^n - 1$ group elements can be generated. In fact the union of all the required slices always miss exactly one element. To see this geometrically observe that the complement of the union is the intersection of the missing slice for each dimension, since them are orthogonal there is exactly one point in all of them; the complementary of this point can be used to evaluate all the responses, but never the missing slice sum. At digit level instead we can observe that each challenge on the $d$-th partition misses one possible digit value, say $\mathsf{ch}_d$, corresponding to the missing partition, thus the element $\tilde{g}_c$ with

$c-1 = \mathsf{ch}_0 + \mathsf{ch}_1 r + \cdots \mathsf{ch}_{n-1} r^{n-1}$ is never required. Hence we can identify the challenge with $c = \mathsf{ch}$ and the challenge set with $[1, r^n]$. Hence to verify the commitment the from $\mathsf{resp}_i = \tilde{g}_I$, with $i \neq \mathsf{ch}$ the Verifier, for all $d = 0, ..., n-1$:

1. fix $z_0^d = x$, $z_r^d = y$, $\mathsf{ch}_d = \mathsf{dig}_r(\mathsf{ch} - 1, d)$;
2. for $j = 1, ..., \mathsf{ch}_d$ set $z_j^d = \sum \mathsf{resp}_i \star x_{j-1}^d$ with the sum on the indeces with $\mathsf{dig}_r(i-1, d) = \mathsf{ch}_d$;
3. for $j = r-1, ..., \mathsf{ch}_d + 1$ set $z_j^d = (-\sum \mathsf{resp}_i) \star x_{j+1}^d$ with the sum on indeces with $\mathsf{dig}_r(i-1, d) = \mathsf{ch}_d$.

Also observe that the response contains just one non ephemeral element (the rest are all generated from seeds) by the initial construction of the graph.

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

| **PROVER** | | **VERIFIER** |
|---|---|---|
| Get $\tilde{g}_i \xleftarrow{\$} G$, set $\tilde{g}_{2^n} \leftarrow g - \sum_{i<r^n} \tilde{g}_i$ | | |
| For $d = 0, ..., n-1$: | $\xrightarrow{\mathsf{com}_d}$ | |
| Set $S_j \leftarrow \{i \mid \mathsf{dig}_2(i-1, d) = 0\}$; | | |
| Set $\mathsf{com}_d \leftarrow (\sum_{i \in S_j} \tilde{g}_i) \star x$ ; | | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} [1, r^n]$. |
| Set $\mathsf{resp}_i \leftarrow \tilde{g}_i$ for $i \neq \mathsf{ch}$. | | Accept if, for all $d = 0, ..., n-1$: |
| | | if $\mathsf{ch}_d = 0$: |
| | $\xrightarrow{\mathsf{resp}_d}$ | $\mathsf{com}_d = (\sum_{\mathsf{dig}_2(i-1,d)=0} \tilde{g}_i) \star x$; |
| | | if $\mathsf{ch}_d = 1$: |
| | | $\mathsf{com}_d = (\sum_{\mathsf{dig}_2(i-1,d)=1} \tilde{g}_i) \star y$; |

Protocol 2.3.1: Identification protocol using the Hypercube.

**Proposition 2.15.** *The identification protocol for the relation $y = g \star x$ described before is Complete, Honest-Verifier Zero Knowledge and Special Sound. It has soundness error $r^{-n}$, requires $n(r-1)$ group action computations and the response can be compressed in $l_G + \lambda \log_2(r^n - 1)$ bits.*

*Proof.* The Completeness comes directly from the observation on the challenge, while for the soundness we can observe that the protocol is special sound since for two different challenges $\mathsf{ch} \neq \mathsf{ch}'$ there esist a dimension $d$ with, wlog, $\mathsf{ch}_d \neq \mathsf{ch}'_d$. Since on this dimension we are doing one round of the classical MPC protcol we can use the extractor from Proposition 2.14. Since the protocol is special sound the soundness error is the inverse of $\#[1, r^n]$.

To simulate the protocol on a known challenge $\mathsf{ch}$ is enought to prooceed as usual, generating at random $h_i$ for $i \neq \mathsf{ch}$ and repeating the verification procedure to generate the commitments.

Clearly the group action computations are $r-1$ for each dimension. To compress the response instead note that $\tilde{g}_1, ..., \tilde{g}_{r^n-1}$ can be generated by a seed tree, thus to send them all but one is enought to send the $\log_2(r^n-1)$ nodes covering all the leaves but the missing one.

□

Protocol 2.3.1 contains one round of the protocol with $r = 2$ and soundness equal to $2^{-n}$.

### 2.3.5 Multiple-Bits Challenges

Fix $r > 1$ and suppose that from $x$ you generate $x_i = g_i \star x$ for $i = 0, ..., r-1$, with $g_0 = e$ and $g_i \xleftarrow{\$} G$ for $i > 0$. 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. Thus the scheme become as in Protocol 2.3.2.

| | | |
|---|---|---|
| Public Data : Group $G$ acting on $X$ via $\star$, element $x_0 \in X$ and hash function $\mathsf{H}$. | | |
| Private Key : Group elements $g_i$ with $g_i \in G$. | | |
| Public Key : $x_i = g_i \star x_0$. | | |
| **PROVER** | | **VERIFIER** |
| Get $\tilde{g} \xleftarrow{\$} G$, send $\mathsf{com} = \mathsf{H}(\tilde{g} \star x)$ | $\xrightarrow{\mathsf{com}}$ | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0,1\}$. |
| Set $\mathsf{resp} \leftarrow \tilde{g} g_{\mathsf{ch}}^{-1}$. | $\xrightarrow{\mathsf{resp}}$ | Accept if $\mathsf{H}(\mathsf{resp} \star x_{\mathsf{ch}}) = \mathsf{com}$. |

Protocol 2.3.2: Identification protocol for the knowledge of the private key.

This optimization was already proposed in [DG19] and its security relies on a different, but related problem:

**Problem 15** (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 prove the equivalence of hardness between this problem and Problem 7, by generalizing the proof of Theorem 3 from [BBPS21].

**Proposition 2.16.** *Given an algorithm to solve mGAIP, that runs in time $T$ and succeeds with probability $\epsilon$, it is possible to solve GAIP (Problem 7), 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 $g_i^{(0)}, ..., g_i^{(r-1)}$. 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 outputs, with probability $\epsilon$, 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. □

Thus the protocol in Protocol 2.3.2 is still sound for the GAIP, but has soundness error $r^{-1}$, cutting the number of rounds to $\left\lceil \frac{\lambda}{\log_2(r)} \right\rceil$. Note that while the number of rounds decreases logarithmically the public key size increases linearly, so major advantages can be obtained for group action with efficient bit representations of set elements, like isogeny based ones.

In fact in [BKV19] they go even further, by generating a large number of public key elements and compressing them in a Merkle tree structure. This way only the set elements required for the challenges are actually sent during the verification and verified with Merkle proof.

Multibit challenges can be used in combination with the other optimization listed before, for example with fixed-weight challenges for each of the $w$ non-ephemeral challenges the Verifier can ask the link to an element $x_{\mathsf{ch}}$, for $\mathsf{ch} > 0$. This way the combined soundness is

$$\binom{t}{w}^{-1} (r-1)^{-w} .$$

Instead with the MPC in the head protocol model the Verifier can ask a link between $\tilde{x}_{\mathsf{ch}+1}$ and $x_{\mathsf{ch}'}$, for $\mathsf{ch}' > 0$. So the soundness error goes down to

$$\binom{t}{w}^{-1} [N(r-1)]^{-w} .$$

Other combination of schemes with mutiple public keys are possible, but should be considered with respect to the characteristics of the particular group action.

CHAPTER

# THREE

# CODE EQUIVALENCE

In this chapter we finally go through two coding theory derived isomorphism problems. For a coding theory introduction we use as reference the wonderful book [PWBJ17], in particular Chapter 1 contains all the basic propositions and proofs for linear codes and their isomorphism maps. Then we follow the current literature, e.g. [BMPS20; Cho+22; RST22; BBPS22], for the cryptographic discussion on code equivalence.

## 3.1 Coding Theory

An $[n, k]$-linear code $\mathcal{C}$ of lenght $n$ is linear subspace of $\mathbb{F}_q^n$ of dimension $k$. Any $k \times n$ matrix $\mathbf{G}$ that generate the space is called generator matrix, while any $(n-k) \times n$ matrix $\mathbf{H}$ containing the linear equations of the code is called parity-check matrix. Any pair of these two matrices satisfies:

$$\mathbf{G} \cdot \mathbf{H}^t = \mathbf{0} \ . \tag{3.1}$$

A vector $\mathbf{c}$ contained in $\mathcal{C}$ is also called *codeword*. When codewords are sent on a noise channel errors may happens; we model them as vectors in $\mathbb{F}_q^n$, added on the sent codeword. So the received word has the form $\mathbf{r} = \mathbf{c} + \mathbf{e}$.

The generator matrix can be used to generate each codeword from a row vector in $\mathbf{v} \in \mathbb{F}_q^k$ by a product $\mathbf{vG}$, since it is a combination of the row vectors generating $\mathcal{C}$, also we have that:

$$\mathbf{c} \in \mathcal{C} \iff \mathbf{cH}^\perp = \mathbf{0} \tag{3.2}$$

In some situation the vector space can have an additional structure, for example for matrix linear codes they consider the space of $m \times n$ matrices over $\mathbb{F}_q$, that is isomorphic to $\mathbb{F}_q^{mn}$. In this case we refer to them as $[m \times n, k]$ codes.

**Systematic Form** A set $I \subseteq [1, n]$ is said *information set* for a linear code $\mathcal{C}$ if, given any generator matrix $\mathbf{G}$ of $\mathcal{C}$, its restriction to the columns that belong to $I$ is of full rank, i.e. $\text{Rank}(\mathbf{G}_I) = k$.

When we have an information set $I$ we can apply the change of basis $\mathbf{G}' = \mathbf{G}_I^{-1}\mathbf{G}$, this way $\mathbf{G}'_I = \mathbf{I}_k$ and we can link bijectively each vector $\mathbf{v}$ to a codeword $\mathbf{c} = \mathbf{v}\mathbf{G}'$ such that $\mathbf{c}_I = \mathbf{v}$. Of particular relevance is the case in which $[1, k]$ is an information set:

**Definition 3.1.** A generator matrix for a linear code $\mathbf{G}$ is in *systematic form* if it has the identity matrix in the first $k \times k$ block on the left, i.e. it is of the form

$$\mathbf{G} = (\mathbf{I}_k \mid \mathbf{R})$$

Given a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ with $[1, k]$ has information set we label the systematic form determination as:

$$\text{SF}(\mathbf{G}) = \mathbf{G}_{[1,k]}^{-1}\mathbf{G} \ . \tag{3.3}$$

The evaluation of (3.3) requires the evaluation of the right reduced echelon form via doing Gaussian Elimination on $\mathbf{G}$, that requires $O(k^3)$ operations.

Given a random matrix the probability of failure of SF is the probability of $\mathbf{G}_{[1,k]}$ not having maximum rank, i.e. if $\mathbf{G}$ is a random matrix:

$$1 - (q^k - 1)(q^k - q)\cdots(q^k - q^{k-1})q^{-k^2} \ . \tag{3.4}$$

For example for $[252, 126]_{127}$-codes a random generator matrix systematic form computations fail with probability less than 0.8%. The systematic form is relevant during the encoding procedure, in fact we get $\mathbf{vG} = \mathbf{v}\,(\mathbf{I}_k \mid \mathbf{R}) = (\mathbf{v} \mid \mathbf{vR})$ and we can read the original vector $\mathbf{v}$ in the first $k$ entries.

Also it gives a canonical representation of a code $\mathcal{C}$, but it is not the only possibility. Another choice is the use of a $(\mu, \nu)$-semi-systematic form:

**Definition 3.2.** A matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ of rank $k$ is in $(\mu, \nu)$-semi-systematic form if it is in right row reduced echelon form and:

- the first $k - \mu$ pivots are in the first possible positions $1, 2, ..., k - \mu$;
- the other pivots are contained between $k - \mu + 1$ and $k - \mu + \nu$.

The systematic form is a $(0, 0)$-semi-systematic form.

**Duality** The Equation (3.2) explain why the parity check matrix has an important role in *error detection*, in fact when we recieve a vector $\mathbf{r} = \mathbf{c} + \mathbf{e}$ we can test if it lie in the code, i.e. if there are no errors with high probability, by testing if $\mathbf{0} = \mathbf{r}\mathbf{H}^\perp = (\mathbf{c} + \mathbf{e})\mathbf{H}^\perp = \mathbf{c}\mathbf{H}^\perp + \mathbf{e}\mathbf{H}^\perp = \mathbf{e}\mathbf{H}^\perp$. Moreover the vector $\mathbf{r}\mathbf{H}^\perp = \mathbf{e}\mathbf{H}^\perp$, usually called *syndrome* and labeled $\mathbf{s}$, has an important role for the majority of the decoding algorithms.

**Definition 3.3.** The *dual code* of a linear code $\mathcal{C}$ is the linear space

$$\mathcal{C}^{\perp} := \{\mathbf{x} \in \mathbb{F}_q^n \mid \langle \mathbf{x}, \mathbf{c} \rangle \quad \text{for all } \mathbf{c} \in \mathcal{C}\} \ .$$

If $\mathbf{H}$ is a parity check matrix for $\mathcal{C}$ then it is also a generator matrix for the dual code $\mathcal{C}^{\perp}$. The *hull* of a linear code $\mathcal{C}$ is the intersection:

$$\mathcal{H}(\mathcal{C}) = \mathcal{C} \cap \mathcal{C}^{\perp} \ ;$$

In most situations $\langle \mathbf{x}, \mathbf{y} \rangle$ is the classical inner product $\mathbf{x}^{\perp} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$ for finite dimensional vector spaces, but sometimes we may define it in different ways. For example for matrix codes in [Rav16] they use the trace product:

$$\langle \mathbf{M}, \mathbf{N} \rangle_T = \mathsf{Tr}(\mathbf{M} \cdot \mathbf{N}^{\perp}) \ . \tag{3.5}$$

**Definition 3.4.** A code that satisfies $\mathcal{C} \subseteq \mathcal{C}^{\perp}$ is said *weakly self-dual*, while if $\mathcal{C} = \mathcal{C}^{\perp}$ it is said *self-dual*.

Observe that for a weakly self dual code $\mathcal{H}(\mathcal{C}) = \mathcal{C} \cap \mathcal{C}^{\perp} = \mathcal{C}^{\perp}$.

**Decoding** To actually use a code now we need to define a decoding strategy. The most simple way to do that would be to use the channel model to evaluate the conditional probability:

$$\mathbb{P}(\mathbf{r} \text{ is received} \mid \mathbf{c} \text{ is sent}) \ ;$$

then find the codeword that maximize this probability. This decoding strategy is called *Maximum Likelihood Decoding* and even if precise is far from being efficient for most channels. Another simpler way to quantify the effect of the errors, that can also be used to evaluate the quality of a code, is to define a metric on the vector space.

The most natural and classical way to do that is the *Hamming metric*, that measure the number of different entries in two vectors:

$$d_H(\mathbf{x}, \mathbf{y}) = \#\{i \mid x_i \neq y_i\} \ . \tag{3.6}$$

A metric can also be defined through a weight $w$, that is the equivalent of the norms from classical euclidean spaces, so that $d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y})$. For the Hamming case is:

$$w_H(\mathbf{x}) = \#\{i \mid x_i \neq 0\} \ . \tag{3.7}$$

Other important examples of metrics are:

– The *rank metric*, used for matrix linear codes, in which the weight is the rank of the matrix:

$$w_R(\mathbf{M}) = \mathrm{Rank}(\mathbf{M}) \ .$$

It was introduced by Delsarte in [Del78] and Gabidulin in [Gab85], and got beyond the radar for 30 years. It was rediscovered for its important applications to network coding [SK11]. Also in post quantum cryptography rank-metric codes are used for their optimal parameters, even if at the moment are considered at a relative early stage.

– The *Lee metric*, relevant for finite fields of cardinality $q > 3$, that weights in a different ways the entries of the vectors:

$$w_L(\mathbf{x}) = \sum_{i=1}^{n} \min\{|x_i|, |q - x_i|\} \ . \tag{3.8}$$

– The *edit distance* or *Levenstein distance* is the length of the shortest sequence of substitutions, insertions and deletion from one vector to the other. It is used for channels where there are also synchronization errors, like the one used for the DNA-storage or telecommunications with delays.

More nice example of metrics can be read on a Gabidulin's survey: [Gab12]. We recall now some other useful classical concepts.

**Definition 3.5.** For a linear code $\mathcal{C}$ we can define the *weight enumerator function* $\mathsf{Wef}(\mathcal{C}) = (A_j)_{j \geqslant 0}$ as

$$A_j := \#\{\mathbf{c} \in \mathcal{C} \mid w(\mathbf{c}) = j\} \text{ for all integers } j \leqslant 0 \ .$$

The values $A_j$ are also referred as *weigh distribution* and the minimum $d > 0$ such that $A_d > 0$ is called the *minimum distance*.

**Definition 3.6.** The *support* of a vector $\mathbf{c} \in \mathbb{F}_q^n$ is the set of indices $i \in [1, n]$ corresponding to non zero entries:

$$\mathsf{supp}(\mathbf{c}) = \{i \in [1, n] \mid c_i \neq 0\} \ .$$

For any set $B \subset \mathbb{F}_q^n$ we can also define its support as:

$$\mathsf{supp}(B) = \bigcup_{\mathbf{c} \in B} \mathsf{supp}(\mathbf{c}) \ .$$

For any subset $J \subset [1, n]$ we define as $\mathcal{E}_J$ the subset of vectors with support contained in $J$. We can use this sets to define the puncturing and shortening of a code, as in [Sen99]:

**Definition 3.7.** Let $\mathcal{C}$ be a code in $\mathbb{F}_q^n$ and $J \subset [1, n]$, we can define the *punctured code* $\mathcal{C}_J$ as the projection of $\mathcal{C}$ on the subspace with the indices $J$ equal to zero ($\mathcal{E}_{J^c}$), i.e

$$\mathcal{C}_J := (\mathcal{C} + \mathcal{E}_J) \cap \mathcal{E}_{J^c} \ ;$$

while the *shortened code* $\mathcal{C}_{\setminus J}$ as the intersection with the subspace the same subspace $\mathcal{E}_{J^c}$, i.e.

$$\mathcal{C}_{\setminus J} := \mathcal{C} \cap \mathcal{E}_{J^c} \ .$$

When $J = j$ we use the notation $\mathcal{C}_j, \mathcal{C}_{\setminus j}$.

Classically speaking the punctured code is the code of lenght $n - |J|$ with the indices in $J$ discarded, while here are simply set to zero, but it is clear we are dealing *de facto* with the same objects. Instead the shortened code is the set of codewords with zero entries on $J$. Results that follows this notation can be read in [Sen99], here I want to recall only the classical duality:

$$(\mathcal{C}_J)^\perp = (\mathcal{C}^\perp)_{\setminus J} \text{ and } (\mathcal{C}_{\setminus J})^\perp = (\mathcal{C}^\perp)_J \ . \tag{3.9}$$

**Hard problems for codes** Being coding theory an established research topic for over 60 years the complexity of the problems related to error-correcting codes is mainly well understood. In particular for cryptography we are interested in the NP-complete ones, that are the one related to the decoding of a generic linear code, i.e. a code with no underlying structure.

**Problem 16** (Maximum Likelihood Decoding (MLD))**.** Given a parity check $(n-k) \times n$ matrix $\mathbf{H}$, a vector $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and a positive integer $w$ find a non-zero vector $\mathbf{e} \in \mathbb{F}_q^n$ solving $\mathbf{e}\mathbf{H}^\perp = \mathbf{s}$ with $w_H(\mathbf{e}) = w$.

**Problem 17** (Nearest Codeword Problem (NCP))**.** Given a generator $k \times n$ matrix $\mathbf{G}$, a vector $\mathbf{y} \in \mathbb{F}_q^n$ and a positive integer $w$ find a non-zero vector $\mathbf{x} \in \mathbb{F}_q^k$ such that $d_H(\mathbf{x}\mathbf{G}, \mathbf{y}) = w$.

The high level decoding problem is Problem 17, in fact when we send over a noisy channel a codeword $\mathbf{c} = \mathbf{x}\mathbf{G}$ and we receive a vector $\mathbf{y}$, the most reasonable output for the decoding is the nearest codeword.

**Lemma 3.8.** *Maximum Likelihood Decoding Problem and Nearest Codeword Problem are equivalent.*

*Proof. NCP reduce to MLD in polynomial time*: given an instance $(\mathbf{G}, \mathbf{y}, w)$ fix $\mathbf{s} = \mathbf{y}\mathbf{H}^\perp$ with $\mathbf{H}$ parity-check matrix associated to $\mathbf{G}$, let $\mathbf{e}$ be a solution of the MLD instance $(\mathbf{H}, \mathbf{s}, w)$, then $(\mathbf{y}-\mathbf{e})\mathbf{H}^\perp = \mathbf{0}$ that implies $(\mathbf{y}-\mathbf{e})$ lie in the code generated by $\mathbf{G}$. Thus we have $d_H(\mathbf{y}, \mathbf{y}-\mathbf{e}) = w_H(\mathbf{e}) = w$.
*MLD reduce to NCP in polynomial time*: given an MLD instance $(\mathbf{H}, \mathbf{s}, w)$ consider a generator matrix $\mathbf{G}$ for the code with parity-check matrix $\mathbf{H}$ and any solution $\mathbf{y}$ to the equation $\mathbf{y}\mathbf{H}^\perp = \mathbf{s}$. Solve NCP for $(\mathbf{G}, \mathbf{y}, w)$ obtaining $\mathbf{x}$, then define $\mathbf{e} = \mathbf{y} - \mathbf{x}\mathbf{G}$, then $w_H(\mathbf{e}) = d_H(\mathbf{x}\mathbf{G}, \mathbf{y}) = w$ and $(\mathbf{y} - \mathbf{x}\mathbf{G})\mathbf{H}^\perp = \mathbf{y}\mathbf{H}^\perp - \mathbf{x}\mathbf{G}\mathbf{H}^\perp = \mathbf{s}$. □

Another important observation is that the decoding problem is equivalent to the problem of finding low weight codewords, i.e. solving the MLD instance $(\mathbf{H}, \mathbf{0}, w)$.

**Lemma 3.9.** *Given an algorithm that finds all the codewords of weight $w$ we can solve use it to solve MLD Problem.*

*Proof.* Given the MLD instance $(\mathbf{H}, \mathbf{s}, w-1)$ consider the code $\mathcal{C}$ with parity-check $(-\mathbf{s}^\perp \mid \mathbf{H})$. If $\mathbf{e}$ is a solution of the MLD instance then $(1 \mid \mathbf{e})$ is weight $w$ codeword of $\mathcal{C}$ since $(1 \mid \mathbf{e})(-\mathbf{s}^\perp \mid \mathbf{H})^\perp = -\mathbf{s} + \mathbf{e}\mathbf{H}^\perp = \mathbf{0}$. Thus if we have all the weight $w$ codewords we can solve MLD simply by considering the ones with 1 as first entry. If no such codeword is found then the MLD instance has no solution.

□

Even if the complexity of the decoding problem has been studied by the cryptographic community for more than 50 years there are still some questions

regarding their theoretical complexity, in particular, differently from, for example, lattices, there is no known reduction from the average case to the worst case.

In other words we are only mathematically sure of the existence of some asymptotically hard instances, while a random code may be, in principle, easy to solve.

**ISD** Information Set Decoding (ISD) is an established technique to solve MLD Problem for a random code. It was introduced by Prange in the 60's in [Pra62]. Several techniques and optimizations have been explored extensively both for coding and cryptography for over 40 years. An updated list containing all the literature for ISD can be found in https://isd.mceliece.org/.

The core idea, due to Prange, is to consider an Information Set $I$ such that there exists a $w$-weight codeword with support contained in $I$, consider a permutation $\pi$ that maps $I$ to $[1, k]$ and consider the generator map $\mathbf{G}' = \pi(\mathbf{G})$ in systematic form, thus we have:

$$\mathbf{x}\mathbf{G}' = \mathbf{x}(\mathbf{I}_k \mid \underbrace{\mathbf{R}}_{k\times n-k}) = (\underbrace{\mathbf{e}}_{k} \mid \underbrace{\mathbf{0}}_{n-k}) \; ; \tag{3.10}$$

so $\mathbf{x} = \mathbf{e}$ and, if we are sure about the support assumption, we can find it in the left kernel of $\mathbf{R}$. The solution then is found as $\pi^{-1}(\mathbf{x} \mid \mathbf{0})$.

Observe that finding a set $I$ is equivalent to finding a permutation with $\pi(I) = [1, k]$.

Since the hard part of this algorithm is searching the right permutation Prange approximated the complexity as the inverse of the probability of getting it:

$$O\left(\frac{\binom{n}{w}}{\binom{n-k}{w}}\right) .$$

Essentially the algorithm is composed by two parts: permutation search and a solution of a linear algebra problem. Its cost is completely unbalanced on the search of a proper information set (i.e. disjoined from the support), so the improvements that followed Prange's idea tried to rebalance this gap by changing the procedure, allowing an intersection of size $p$ between $I$ and the support, easing the search for a good permutation, but increasing the difficulty of the second part.

In particular the best know modifications for non-binary fields (our case of interest) is the one from [Pet10]. Suppose that we want to find a codeword of the code generated by $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ of weight $w < \min(k, n-k)$, thus we consider a permutation $\pi$ and the permuted matrix $\pi(\mathbf{G}) = \mathbf{G}'$. The core process of Peters's ISD depends on two parameters $p, l$ with $0 \leqslant l \leqslant n-k$ and $0 \leqslant p \leqslant \lfloor k/2 \rfloor$, that can be adapted to find optimal balancing between different

procedures. Via basic linear algebra $\mathbf{G}'$ can be reduced to:

$$\mathbf{G}' = \left( \begin{array}{cc|c|c} \mathbf{I}_{\lfloor k/2 \rfloor} & \mathbf{0} & \mathbf{G}'_{X,Z} & \\ \mathbf{0} & \mathbf{I}_{\lceil k/2 \rceil} & \underbrace{\mathbf{G}'_{Y,Z}}_{l} & \underbrace{\mathbf{G}'_{V}}_{n-k-l} \end{array} \right),$$

clearly if $[1, k]$ is not an information set just chose another permutation. At this point we proceed by creating two lists:

- $X = \left\{ \left( \mathbf{e}, \mathbf{s} = \mathbf{e}\mathbf{G}'_{X,Z} \right) \right\}$, for all $\mathbf{e} \in \mathbb{F}_q^{\lfloor k/2 \rfloor}$ with weight $p$;
- $Y = \left\{ \left( \mathbf{e}, \mathbf{s} = \mathbf{e}\mathbf{G}'_{Y,Z} \right) \right\}$, for all $\mathbf{e} \in \mathbb{F}_q^{\lceil k/2 \rceil}$ with weight $p$.

A collision between the two lists is found if, given $(\mathbf{x}, \mathbf{s}_x) \in X, (\mathbf{y}, \mathbf{s}_y) \in Y$, we have that $\mathbf{s}_x + \mathbf{s}_y = \mathbf{0}$ and $(\mathbf{x} \mid \mathbf{y})G'_V$ has weight $w - 2p$.

In this case we fix $\mathbf{e} = [(\mathbf{x} \mid \mathbf{y}) \mid (\mathbf{x} \mid \mathbf{y})G'_V]$ and return $\pi^{-1}(\mathbf{e})$, otherwise we chose another permutation and restart the process.

More about this ISD procedure is in [BBPS21; BBPS22] and [Bal+23b], in particular the last one estimate the complexity as:

$$C_{\text{ISD}}(w) = \min_{p,l} \left\{ \frac{C_{\mathsf{ITER}}(w, p, l)}{P_{\mathsf{SUCC}}(w, p, l)} \right\}, \tag{3.11}$$

where

$$C_{\text{ITER}}(w, p, l) = l \left( \left( \frac{k}{2} - p + 1 \right) + (q-1)^p \left( \binom{\lfloor \frac{k}{2} \rfloor}{p} + \binom{\lceil \frac{k}{2} \rceil}{p} \right) \right) + \\ + \frac{\frac{2pq(w-2p+1)}{q-1} \left( 1 + \frac{q-2}{q} \right) \binom{\lfloor \frac{k}{2} \rfloor}{p} \binom{\lceil \frac{k}{2} \rceil}{p} (q-1)^{2p}}{q^l} + \frac{(n-k)^2(n+k)}{2}, \tag{3.12}$$

and

$$P_{\mathsf{SUCC}}(w, p, l) = \frac{\binom{\lfloor \frac{k}{2} \rfloor}{p} \binom{\lceil \frac{k}{2} \rceil}{p} \binom{n-k-u}{w-2p}}{\binom{n}{w}} \cdot A_w.$$

*Remark* 18. When trying to solve MLD aided by a quantum computer Grover's Algorithm [Gro96] can be leveraged to get a quadratic speedup on the Prange's ISD (this quantum procedure is also called Bernstein's algorithm [Ber10]). Another improvement have been proposed in [KT17] using quantum optimized walk algorithm, but the additional price in quantum memory is so excessive to make them a real threat. A new memory efficient ISD procedure have been proposed in [KTT23].

## 3.2 Code Equivalence

In the flavor of the Erlangen program ([Kis12]) we should ask ourselves which group actions act on the geometrical objects we are looking into, in a similar

way as the general linear group act on euclidean spaces. We can start with a general definition for isometry:

**Definition 3.10.** A map $\phi : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is said an *isometry* for the distance $d$ if it leaves the metric invariant, i.e.

$$d(\phi(\mathbf{x}), \phi(\mathbf{y})) = d(\mathbf{x}, \mathbf{y}) \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n \ .$$

If $\phi$ it is also linear we call it *linear isometry*. Two $[n, k]$-codes $\mathcal{C}, \mathcal{C}'$ are said *equivalent* with respect to the metric $d$ if it exists a linear isometry for the relative metric that maps $\mathcal{C}$ to $\mathcal{C}'$.

In analogy to the euclidean case, for metrics induced by a weight, it is also enough to check that the map is linear and

$$w(\phi(\mathbf{x})) = w(\mathbf{x}) \text{ for all } \mathbf{x} \in \mathbb{F}_q^n \ .$$

We have enough ingredients to define a general version of the code equivalence problem:

**Problem 19** (General Code Equivalence)**.** Consider two linear codes $\mathcal{C}$ and $\mathcal{C}'$ equivalent with respect to the metric induced by $d$, find the linear isometry $\phi$ such that $\phi(\mathcal{C}) = \mathcal{C}'$.

We can see this as a particular instance of GAIP (Problem 7). Observe that clearly the identity is an isometry and the composition of two isometries is still an isometry. Also they are invertible since the set $\mathbb{F}_q^n$ is finite and the maps are injective, in fact by the definition of distance:

$$\mathbf{x} \neq \mathbf{y} \iff 0 \neq d(\mathbf{x}, \mathbf{y}) = d(\phi(\mathbf{x}), \phi(\mathbf{y})) \Rightarrow \phi(\mathbf{x}) \neq \phi(\mathbf{y}) \ .$$

The isometries form a group with respect to the composition, let's call it $\mathsf{Isom}_d$. Define $X$ as the set containing $[n, k]$-codes, we can consider the following group action associated to the linear code equivalence:

$$\begin{aligned} \star : \mathsf{Isom}_n \times X &\to X \\ (\phi, \mathcal{C}) &\to \phi \star \mathcal{C} := \phi(\mathcal{C}) \end{aligned} \tag{3.13}$$

Now we proceed in instantiating Problem 19 for the Hamming metric, then for Rank metric in Section 3.2.2. We assume here that the field $\mathbb{F}_q$ is a prime field, i.e. that $q$ is a prime, to avoid dealing with automorphism of the field.

### 3.2.1 Hamming Metric

Let's start with two examples of isometries for the Hamming metric.

*Example* 20. Consider a permutation $\pi$ in the symmetric group $\mathbb{S}_n$, we can use it to define the linear map $\phi_\pi : \mathbb{F}_q^n \to \mathbb{F}_q^n$:

$$\phi_\pi(\mathbf{a}) = \phi_\pi\big(a_1, \cdots, a_n\big) = \big(a_{\pi^{-1}(1)}, a_{\pi^{-1}(2)}, \cdots, a_{\pi^{-1}(n)}\big) \ .$$

The linearity is trivial, while for the isometry we can observe that the Hamming weight is unchanged, in fact the number of non-zero entries of $\mathbf{a}$ does not depend on the order.

*Example* 21. Consider a vector $\mathbf{v} \in \left(\mathbb{F}_q^*\right)^n$, i.e. with only non zero entries, we can use it to define the linear map $\phi_{\mathbf{v}} : \mathbb{F}_q^n \to \mathbb{F}_q^n$:

$$\phi_{\mathbf{v}}(\mathbf{a}) = \phi_{\mathbf{v}}\big(a_1, \cdots, a_n\big) = (v_1 a_1, v_2 a_2, \cdots, v_n a_n) \ .$$

Again the linearity is trivial, while for the isometry we can observe that $a_i = 0 \iff v_i a_i = 0$, thus the Hamming weight is unchanged.

This two examples are particularly relevant for our work since each linear isometry can be seen as a combination of these, also called *monomial map*:

**Theorem 3.11.** *Each linear isometry for the Hamming metric is a monomial map.*

*Proof.* Clearly a monomial map is a linear isometry as shown in Examples 20, 21.

Instead given a linear isometry $\phi$ we can observe that when applied to canonical basis vectors $\mathbf{e}_i$ since they have weight one necessarily also $\phi(\mathbf{e}_i)$ has weight one, thus we have that:

$$\phi(\mathbf{e}_i) = v_i \mathbf{e}_{j_i} \ ;$$

with $v_i \neq 0$ and $j_i \neq j_{i'}$ for all $i \neq i'$ otherwise using linearity $w(\phi(\mathbf{e}_i - \mathbf{e}_{i'})) = w((v_i - v_{i'})\mathbf{e}_{j_i}) < 2$ against the isometry assumption. Define the permutation $\pi$ such that $j_i \mapsto i$ and observe that:

$$\phi_{\mathbf{v}} \circ \phi_\pi(\mathbf{e}_i) = v_i \phi_\pi(\mathbf{e}_i) = v_i \mathbf{e}_{j_i} = \phi(\mathbf{e}_i) \ .$$

Since both $\phi$ and $\phi_{\mathbf{v}} \circ \phi_\pi$ are linear and are equal on a basis we have $\phi = \phi_{\mathbf{v}} \circ \phi_\pi$ as requested. □

With this observation in mind we can instantiate the Problem 19 for the Hamming metric. Let's now define the set of all monomial maps as $\mathsf{Mono}_n$, that can also be seen as the group product $\mathbb{S}_n \times (\mathbb{F}_q^*)^n$. They can also be represented by invertible $n \times n$ matrices with exactly one non zero entries for each row and each column. Since we are studying code equivalence for practical applications we 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. On these applying a monomial map, in matrix form $\mathbf{Q}$, is equivalent to the right multiplication. Also to have a unique representation we need to restrict ourselves to the case in which the generator matrix are in a systematic form as in Definition 3.1, ensuring our group action as effectively acting on linear codes, rather than on their representatives (generator matrices). Combining these two observations the group action becomes:

$$\begin{aligned} \star : \mathsf{Mono}_n \times X &\to X \\ (\mathbf{Q}, \mathbf{G}) &\to \mathbf{Q} \star \mathbf{G} := \mathrm{SF}(\mathbf{G}\mathbf{Q}) \end{aligned} \tag{3.14}$$

It is easy to see that the action is well-formed (since the standard form is just a change of basis), with identity element $\mathbf{I}_n$ and is compatible with respect to (right) multiplication.

To see that it is an Effective Group Action we can go through the points of Definition 2.3. By using the matrix representation for monomial maps we can easily perform all the required computation on the group (item 1). For item 2 and 4 we simply observe that evaluating multiplications, rank and reduced right echelon form (as required in Definition 3.1) are classical problems from linear algebra that can be solved in polynomial times with respect to the matrices dimensions. Item 3 is trivial for this group action.

We can see now that the vectorization problem (problem 7) for this group action is a well-known problem in coding theory:

**Problem 22** (Linear Equivalence (LEP))**.** Given two $k$-dimensional linear codes $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$ , find, if any, $\mathbf{Q} \in \mathsf{Mono}_n$ such that $\mathcal{C}' = \mathcal{C}\mathbf{Q}$.

In fact when representing the codes via generator matrices in standard form the left multiplication by $\mathbf{Q}$ corresponds exactly to the group action computation in Equation (3.14). It is also of interest a special case of LEP, where the monomial matrix $\mathbf{Q}$ is simply a permutation, known as *Permutation Equivalence Problem (PEP)*:

**Problem 23** (Permutation Equivalence (PEP))**.** Given two $k$-dimensional linear codes $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$ , find, if any, $\pi \in \mathbb{S}_n$ such that $\mathcal{C}' = \pi(\mathcal{C})$. In this case we say that $\mathcal{C}$ and $\mathcal{C}'$ are permutation equivalent.

We can consider also a decisional version of LEP:

**Problem 24** (Decisional Linear Equivalence (dLEP))**.** Given two $k$-dimensional random linear codes $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$ say if they are equivalent or not.

In Section 3.3.3 we show that dLEP and LEP are equivalent, and also LEP and PEP. All previous problems are conjectured hard by the cryptographic community, also the group action $(\mathsf{Mono}_n, X, \star)$ (3.14) is conjectured to be a Very Hard Homogeneous Space (Definition 2.6).

**Reduction Between PEP and LEP** In the binary case ($q = 2$) clearly PEP and LEP are the same problem. Instead when $q > 2$ we can observe a permutation is also a monomial map, thus the instances of PEP are just a particular subsets of LEP instances that can be solved in the same time.

To study the other direction between these problems we need to use the *closure of a code* $\mathcal{C}$, i.e. the Kronecker product $\mathcal{C} \otimes \mathbf{a}$ where $\mathbf{a}$ is a vector of length $q-1$ containing all the non-zero elements of $\mathbb{F}_q$ (the order does not matter). If $\mathbf{G}$ is the generator matrix of $\mathcal{C}$ then the closure of the code is generated by $\mathbf{G} \otimes \mathbf{a}$. By using this construction we immediately have the inverse reduction:

**Proposition 3.12** (Theorem 1 of [SS13])**.** *Two linear codes $\mathcal{C}, \mathcal{C}'$ in $\mathbb{F}_q$ are linearly equivalent if and only if their closures $\mathcal{C} \otimes \mathbf{s}$ and $\mathcal{C}' \otimes \mathbf{s}$ are permutation equivalent.*

**The Role of the Change of Basis** In the group action definition (3.14) we have used the standard form SF to have a unique representation of the codes, but this is not necessary in principle since we can consider a general change of basis, this way the group action would become:

$$\begin{aligned} \star : (\mathsf{GL}_k \rtimes \mathsf{Mono}_n) \times X &\to X \\ ((\mathbf{S}, \mathbf{Q}), \mathbf{G}) &\to \mathbf{SGQ} \ . \end{aligned} \tag{3.15}$$

We would like to point out that, even if the monomial map is usually considered the "true" representative of the action, the change of basis is as important as the monomial map from a security viewpoint. Making it transparent would render the entire group action solvable in polynomial time, in fact we have:

**Proposition 3.13.** *Let* $\mathbf{G}$ *and* $\mathbf{G}' = \mathbf{SGQ}$ *be the generator matrices for two linearly equivalent codes. If* $\mathbf{S}$ *is known then it is possible to recover* $\mathbf{Q}$ *in polynomial time by using sorting algorithms.*

*Proof.* Knowing the linear map $\mathbf{S}$, we can have the two codes written with respect to the same basis. In particular we can consider the matrix $\mathbf{S}^{-1}\mathbf{G}' = \mathbf{GQ}$. Note that this matrix and $\mathbf{G}$ have the same columns vectors, only permuted and multiplied by a constant. So we can use Algorithm 5 to recover the monomial map. The algorithm ends in polynomial time since it only requires basic linear algebra operation and the sorting of two lists ($O(n \log(n))$ complexity).

**Algorithm 5** Monomial Map Calculation

**Require:** Two generator matrices $\mathbf{G}, \mathbf{G}'$ satisfying $\mathbf{G}' = \mathbf{GQ}$ with $\mathbf{Q}$ being a monomial map;
**Ensure:** The monomial matrix $\mathbf{Q}$.

Set $S \leftarrow \varnothing$;
**for** each column $\mathbf{c}$ in $\mathbf{G}$ **do**
    Multiply $\mathbf{c}$ by the inverse of the first non-zero entry;
                ▷ We are setting the column in a standard form
    $S \leftarrow S \cup \{\mathbf{c}\}$;
Sort the list $S$ using any order.
Memorize the resulting permutation as $\mathbf{Q}_1$.
Set $S \leftarrow \varnothing$;
**for** each column $\mathbf{c}'$ in $\mathbf{G}'$ **do**
    Multiply $\mathbf{c}'$ by the inverse of the first non-zero entry;
    $S \leftarrow S \cup \{\mathbf{c}'\}$;
Sort the list $S$ using any order.
Memorize the resulting permutation as $\mathbf{Q}_2$.
Get the permutation matrix $\mathbf{P} \leftarrow \mathbf{Q}_1\mathbf{Q}_2^{-1}$.
Find the coefficients by confronting $\mathbf{GP}$ and $\mathbf{G}'$.
**return** $\mathbf{Q}$

□

**Code Equivalence and Public Key Cryptography** Coding theory problems (Problems 17,16) and code equivalence were already used 40 years ago to define one of the oldest public key cryptosystems, ideated by McEliece in [McE78]. The idea is to consider a particular $[n,k]_q$-code $\mathcal{C}$ generated by $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ where, thanks to additional structure, we can solve efficiently the NCP (Problem 17) for a low weight $w$. The structure of the code is masked by taking a random monomial map $\mathbf{Q} \in \mathsf{Mono}_n$ and applying the equivalence:

$$\mathbf{G}' := \mathbf{Q} \star \mathbf{G} = \mathrm{SF}(\mathbf{G} \cdot \mathbf{Q}) = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{Q} \ .$$

This way $n, k, q$ are the public parameters $\mathsf{pp}$, $\mathbf{G}'$ is the public key $\mathsf{pk}$ and $\mathbf{S}, \mathbf{Q}, \mathbf{G}$ the private key $\mathsf{sk}$.

The scheme then work as follows:

- Given a message $\mathsf{m} \in \mathbb{F}_q^k$ to encrypt, we sample a random error vector $\mathbf{e} \in \mathbb{F}_q^n$ of weight $w = w_H(\mathbf{e})$ and we evaluate $\mathbf{y} = \mathsf{m} \cdot \mathbf{G}' + \mathbf{e}$.
- Given a ciphered text $\mathbf{y}$ we apply the decoding algorithm for $\mathbf{G}$ on $\mathbf{y}\mathbf{Q}^{-1}$ and get $\mathbf{z} \in \mathbb{F}_q^k$, then we have $\mathsf{m} = \mathbf{S}^{-1}\mathbf{z}$. In fact:

$$\mathbf{y}\mathbf{Q}^{-1} = (\mathsf{m} \cdot \underbrace{\mathbf{G}'}_{\mathbf{S}\cdot\mathbf{G}\cdot\mathbf{Q}} + \mathbf{e})\mathbf{Q}^{-1} = \mathsf{m} \cdot \mathbf{S} \cdot \mathbf{G} + \underbrace{\mathbf{e}\mathbf{Q}^{-1}}_{w_H(\mathbf{e}\mathbf{Q}^{-1})=w} \ ,$$

  hence from the decoding algorithm we got $\mathbf{z} = \mathsf{m}\cdot\mathbf{S}$, that thanks to $\mathbf{S}$ lead to the message.

### 3.2.2 Rank Metric

Analogously let's consider three linear transformations on a matrix $\mathbf{M} \in \mathbb{F}_q^{n\times m}$ that preserves the rank:

- multiplication on the left side by an invertible matrix in $\mathsf{GL}_m$;
- multiplication on the right side by an invertible matrix in $\mathsf{GL}_n$;
- transposition of the matrix if $n = m$.

These three linear maps can be used to define any possible linear equivalence in the rank metric, since by [Mor14] we have:

**Proposition 3.14.** *If $\phi : \mathbb{F}_q^{n\times m} \to \mathbb{F}_q^{n\times m}$ is a linear isometry for the Rank metric, then there exist a $n \times n$ invertible matrix $\mathbf{A}$ and a $m \times m$ invertible matrix $\mathbf{B}$ such that*

1. *$\phi(\mathbf{M}) = \mathbf{AMB}$ for all $\mathbf{M}$ in $\mathbb{F}_q^{n\times m}$, or*
2. *$\phi(\mathbf{M}) = \mathbf{AM}^{\perp}\mathbf{B}$ for all $\mathbf{M}$ in $\mathbb{F}_q^{n\times m}$;*

*where the latter case can occur only if $n = m$.*

In the cryptographic literature, even if $n = m$ ,only the case 1 is considered, since we only need to check twice the map to consider also the transposition. When the map is induced by the pair of matrices $(\mathbf{A}, \mathbf{B})$ we may write $\phi_{(\mathbf{A},\mathbf{B})}$ or directly $(\mathbf{A}, \mathbf{B})$. Also for a linear matrix code $\mathcal{C}$ it is costum to indicate its image through $\phi_{(\mathbf{A},\mathbf{B})}$ as $\mathbf{A}\mathcal{C}\mathbf{B}$. We may then define the group action as:

$$\begin{aligned} \star : (\mathsf{GL}_n \times \mathsf{GL}_m) \times X &\to X \\ ((\mathbf{A}, \mathbf{B}), \mathcal{C}) &\to (\mathbf{A}, \mathbf{B}) \star \mathcal{C} := \mathbf{A}\mathcal{C}\mathbf{B} \ . \end{aligned}$$

To actually represent the action as linear map we need to *vectorize* the matrices by a map $\mathsf{vec} : \mathbb{F}_q^{n\times m} \to \mathbb{F}_q^{nm}$ that slices the matrices to a vector of dimension $nm$.

$$\mathbf{M} = \begin{pmatrix} m_{1,1} & \dots & m_{1,n} \\ \vdots & \ddots & \vdots \\ m_{m,1} & \dots & m_{m,n} \end{pmatrix} \mapsto \mathsf{vec}(\mathbf{M}) = (m_{1,1}, \dots, m_{1,n}, \dots, m_{m,1}, \dots, m_{m,n}) \ . \tag{3.16}$$

The inverse operation is labeled $\mathsf{mat}$. This way we can use as generator matrix for $\mathcal{C}$ the matrix in $\mathbb{F}_q^{k\times nm}$ that generates the following $[nm, k]$ code:

$$\mathsf{vec}(\mathcal{C}) := \{\mathsf{vec}(\mathbf{M}) \mid \mathbf{M} \in \mathcal{C}\} \ .$$

An important property of the $\mathsf{vec}$ operator is the so called *vec trick*:

$$\mathsf{vec}(\mathbf{AMB}) = \mathsf{vec}(\mathbf{M}) \cdot (\mathbf{A}^\perp \otimes \mathbf{B}) \ ; \tag{3.17}$$

where $\otimes$ is the Kronecker Product:

$$(\mathbf{A}^\perp \otimes \mathbf{B}) := \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{m1}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mm}\mathbf{B} \end{bmatrix} \ . \tag{3.18}$$

We can finally compactly describe the group action induced by the linear rank metric codes equivalence. In this case, the set $X$ is formed by the $k$-dimensional matrix codes of size $m \times n$ over some base field $\mathbb{F}_q$ that are represented via generator matrices $\mathbf{G} \in \mathbb{F}_q^{k\times mn}$. Then, the action of the group $G = \mathsf{GL}_m \times \mathsf{GL}_n$ on this set can be described compactly as follows:

$$\begin{aligned} \star : G \times X &\to X \\ ((\mathbf{A}, \mathbf{B}), \mathbf{G}) &\to \mathrm{SF}(\mathbf{G}(\mathbf{A}^\perp \otimes \mathbf{B})) \end{aligned} \tag{3.19}$$

Note that this is equivalent to applying the matrices $\mathbf{A}$ and $\mathbf{B}$ to each codeword $\mathbf{M}$ in the matrix code as $\mathbf{AMB}$. We can finally define the rank metric version of Problem 19 and its decisional version:

**Problem 25** (Matrix Code Equivalence (MCE))**.** Given two $k$-dimensional matrix codes $\mathcal{C}, \mathcal{C}'$, find, if any, $\mathbf{A} \in \mathsf{GL}_m, \mathbf{B} \in \mathsf{GL}_n$ such that $\mathcal{C}' = \mathbf{A}\mathcal{C}\mathbf{B}$.

**Problem 26** (Decisional Matrix Code Equivalence (dMCE))**.** Given two $k$-dimensional matrix codes $\mathcal{C}, \mathcal{C}'$ say if the are linear equivalent in the rank-metric.

Interestingly for the rank-metric there is no known polynomial time reduction between the decisional and the search version of the problem.

**Relation to Tensor Isomorphism** Tensors are a well known generalization of the concept of matrices in higher dimension. They can also be used to define a group actions: given the $d$-tensor space $T = \bigoplus_{i=1}^{d} \mathbb{F}_q^{n_i}$ and the group $G = \mathsf{GL}_{n_1} \times \cdots \times \mathsf{GL}_{n_d}$ we can define

$$\begin{aligned} \star : G \times T &\to T \\ ((\mathbf{A}_1, ..., \mathbf{A}_d), \mathbf{T}) &\to \sum_{i_1,\ldots,i_d} \mathbf{T}\left(i_1, \ldots, i_d\right) \mathbf{A}_1 \mathbf{e}_{i_1}^{(1)} \otimes \cdots \otimes \mathbf{A}_d \mathbf{e}_{i_d}^{(d)} \ . \end{aligned} \tag{3.20}$$

Note that we have used the canonical basis $\{\mathbf{e}_1^{(i)}, ..., \mathbf{e}_{n_i}^{(i)}\}$ for any vector space $\mathbb{F}_q^{n_i}$ so that the following equality holds:

$$\mathbf{T} = \sum_{i_1,\ldots,i_d} \mathbf{T}\left(i_1, \ldots, i_d\right) \mathbf{e}_{i_1}^{(1)} \otimes \cdots \otimes \mathbf{e}_{i_d}^{(d)} \ .$$

The orbits of the group action (3.20) are exactly the class of isomorphism for tensors, this way we can define the equivalent of GAIP (Problem 7) for tensors:

**Problem 27.** (d-TI) Given two tensors $\mathbf{T}, \mathbf{T}' \in \bigoplus_{i=1}^{d} \mathbb{F}_q^{n_i}$ find, if any, the isomorphism $(\mathbf{A}_1, ..., \mathbf{A}_d) \in \mathsf{GL}_{n_1} \times \cdots \times \mathsf{GL}_{n_d}$ between them.

A pivotal result for the study of this problem is the reduction:

**Theorem 3.15** (Theorem B of [GQ21])**.** *$d$-TI reduces to 3-TI in polynomial time.*

Since in the same paper (Theorem A) it is showed a reduction of 3-TI to several other problems from different areas of algebra it makes sense to define the complexity class TI-complete that contains all the problems with polynomial time reductions to and from $d$-TI.

In particular also MCE is proven to be TI-complete in [DAl22b]. In fact it is straightforward to see a matrix code as a 3-tensors and the equivalence as matrix multiplication on the 3 axis.

### Change of Basis

As for the Hamming-metric case we can ask if the change of basis (here done again setting the generator matrix in systematic form) is necessary for this group action too. The answer of this question can be found in the analysis contained in [RST22] for the problem MCEbase:

**Problem 28** (MCEbase)**.** Given two $k$-dimensional matrix codes $\mathcal{C}, \mathcal{C}'$ with generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times nm}$, find, if any, $\mathbf{A} \in \mathsf{GL}_m, \mathbf{B} \in \mathsf{GL}_n$ such that $\mathbf{G}' = \mathbf{G}(\mathbf{A}^{\perp} \otimes \mathbf{B})$.

In fact in Remark 16 [RST22] they show how via reductions to problems related to the isomorphism of polynomials that Problem 28 can be solved in polynomial time on random instances ($O(k^6)$).

For completeness we should point out that this reduction was already known in the tensor isomorphism community via a reduction to the module isomorphism problem, which admits a deterministic polynomial-time solution with the Brooksbank-Luks algorithm [BL08]. More on that can be seen in Section 3.3.2 of [JQSY19].

### 3.2.3 General Considerations

The two problems share clearly some similarities, in particular they are both non-abelian group actions. This clearly reduce the possible cryptographic primitives based on them, for example we cannot build a Diffie-Hellmann like key exchange. However this 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 (see Section 2.1.1).

Also both these group actions are based entirely on linear algebra on finite (prime) fields, thus the operations required are simple and well understood by the algebraic community from long time. To compute them we need:

- permutation and rescaling of a matrix for the monomial map, that requires only $O(n)$ operations;
- multiplication of matrices, that requires $O(n^\omega)$ field operations. The value $\omega$ depends on the particular algorithm used for multiplication, for example the Strassen's algorithm (used in practice for finite fields) achieve $\omega = \log_2(7) \simeq 2.807$. At the moment the best asymptotic algorithm in 2023 achieves $\omega = 2.371552$;
- systematic form computation, this is the most expensive computation since it requires to compute a right reduced echelon form that requires $O(n^3)$ field operations for the *de facto* inversion of a $k \times k$ matrix, as explained before.

This is an important advantage both for speed and for implementation with respect to other code based schemes that requires also complex decoding algorithms and to other group actions like isogenies that uses complex procedures like Velu's formulas [BDLS20b].

Suppose now that the base field is not prime, this mean that there exists non-trivial automorphism on the field, e.g. the Frobenius one $x \mapsto x^p$, that induces linear isometries when applied over the codes. This means that in this case we should consider also the Galois group $\mathsf{Gal}(\mathbb{F}_{p^m}/\mathbb{F}_p)$ for the group actions. For both the constructions, cryptographers studying them observed that the Galois Group does not improve neither the security or the efficiency of the protocols, so discarding this possibility.

**Trivial Automorphism Group** As seen in Section 2.2.1 an important question for the quantum security of these group actions is the sizes of the group of automorphisms, i.e. the linear equivalences of the code on itself.

It is a common belief that the automorphism group of the codes is trivial with overwhelming probability. To our knowledge it is computationally infeasible to compute automorphism of codes of this size, so we can conjecture that the Stabilizer Computation Problem (Problem 10) is hard.

In Appendix B of [BBPS21] there are some estimates for the probability of finding a permutation or a monomial map that is an automorphism for a random code, while the MEDS team ([Cho+22]) analysed computational data on smaller sized matrix codes seeing that the probability of a random code having trivial automorphism group grows rapidly in the parameters $q, n$ and $m$. Thus both schemes use the following assumption:

**Assumption 29.** For all in use parameter sets, our code equivalence induced group actions, (3.14) and (3.19), are free, i.e. the stabilizers are trivial.

**Reduction Between Them** We have noted before some differences in the two group actions, thus a question should arise about their equivalence. It is well known in literature that LEP reduces to MCE in polynomial time, it has been proved in Lemma 34 of [CDG20], but also you can see Proposition 3.6 of [GQ19] in combination with [DAl22b].

We briefly introduce here the reduction from [CDG20]. Recall thst $\mathbf{a}_i$ is the $i$-th column of the matrix $\mathbf{A}$. Consider now the operator $\mathsf{Row}_i : \mathbb{F}_q^k \to \mathbb{F}_q^{n\times k}$ that maps a vector to an empty matrix with only the $i$-th row non zero:

$$\mathsf{Row}_i(\mathbf{x}) := \begin{pmatrix} 0 & \cdots & 0 \\ & \mathbf{x} & \\ 0 & \cdots & 0 \end{pmatrix} .$$

Given a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ we can build the following $[k + n \times k, k]$-matrix code:

$$\mathcal{C}_{\mathsf{Mat}}(\mathbf{G}) := \left\{ \sum_{i=1}^{n} \lambda_i \begin{pmatrix} \mathbf{g}_i \mathbf{g}_i^{\perp} \\ \mathsf{Row}_i(\mathbf{g}_i) \end{pmatrix} : \lambda_i \in \mathbb{F}_q \right\} .$$

This construction is compatible with the structure of the codes, in fact we have that if $(\mathbf{A}, \mathbf{B})$ is a positive instance of the monomial equivalence problem, i.e. they generate two linear equivalent codes, then $(\mathcal{C}_{\mathsf{Mat}}(\mathbf{A}), \mathcal{C}_{\mathsf{Mat}}(\mathbf{B}))$ is a positive instance of dMCE, i.e. they are equivalent in the rank-metric.

As shown in [CDG20] if they are linear equivalent there exists $\mathbf{S} \in \mathsf{GL}_k, \mathbf{Q} \in \mathsf{Mono}_n$ such that:

$$\mathbf{A} = \mathbf{SBQ} ;$$

with $\mathbf{Q} = \mathbf{DP}$, where $\mathbf{P}$ is a permutation matrix and $\mathbf{D}$ a diagonal matrix. Denote by $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q^*$ the diagonal entries of $\mathbf{D}$. Then, for any $1 \leqslant i \leqslant n$, it exists $1 \leqslant j \leqslant n$ (image of $i$ by the permutation given by $\mathbf{P}$ ) such that,

$$\mathbf{a}_i = \alpha_j \mathbf{S}\mathbf{b}_j \quad \text{and} \quad \mathbf{a}_i\mathbf{a}_i = \alpha_j^2 \mathbf{S}\mathbf{b}_j\mathbf{b}_j^{\perp}\mathbf{S}^{\perp} .$$

This gives:

$$\mathcal{C}_{\mathsf{mat}}(\mathbf{A}) = \begin{pmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{pmatrix} \mathcal{C}_{\mathsf{mat}}(\mathbf{B})\mathbf{S}^{\perp} \ .$$

The matrices for the equivalence are clearly nonsingular, therefore, we have a positive instance of dMCE. Also we have the following:

**Lemma 3.16** (Lemma 34 [CDG20])**.** *Given two generator matrix* $\mathbf{A}, \mathbf{B} \in \mathbb{F}_q^{k\times n}$ *and* $\mathbf{U} \in \mathsf{GL}_{k+n}$ *and* $\mathbf{V} \in \mathsf{GL}_k$ *which verify:*

$$\mathcal{C}_{\mathsf{Mat}}(\mathbf{A}) = \mathbf{U}\mathcal{C}_{\mathsf{Mat}}(\mathbf{B})\mathbf{V} \ .$$

*Then, there exists a permutation* $\sigma \in \mathbb{S}_n$ *and* $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q^*$ *such that:*

$$\begin{pmatrix} \mathbf{a}_i\mathbf{a}_i^{\perp} \\ \mathsf{Row}_i\left(\mathbf{a}_i\right) \end{pmatrix} = \alpha_{\sigma(i)}\mathbf{U} \begin{pmatrix} \mathbf{b}_{\sigma(i)}\mathbf{b}_{\sigma(i)}^{\perp} \\ \mathsf{Row}_{\sigma(i)}\left(\mathbf{b}_{\sigma(i)}\right) \end{pmatrix} \mathbf{V} \text{ for all } i \ .$$

The lemma does not only prove that positive instances of dMCE are sent to positive instances of dLEP (that would be trivial using the diagonal operator), but also that a solution to Problem 25 can be used to find a solution to Problem 22.

Suppose now that the two codes $\mathcal{C}_{\mathsf{Mat}}(\mathbf{A}), \mathcal{C}_{\mathsf{Mat}}(\mathbf{B})$ are equivalent, then the two underlying linear codes are linear equivalent and we can find the monomial map such that solves LEP in polynomial time from two non-singular matrices $\mathbf{U}$ and $\mathbf{V}$ solving MCE, i.e. such that:

$$\mathcal{C}_{\mathsf{Mat}}(\mathbf{A}) = \mathbf{U}\mathcal{C}_{\mathsf{Mat}}(\mathbf{B})\mathbf{V} \ .$$

By Lemma 3.16, for any $1 \leqslant i \leqslant n$, it exists $1 \leqslant \sigma(i) \leqslant n$ and $\alpha_{\sigma(i)} \in \mathbb{F}_q^*$ such that:

$$\begin{aligned} \begin{pmatrix} \mathbf{a}_i\mathbf{a}_i^{\perp} \\ \mathsf{Row}_i\left(\mathbf{a}_i\right) \end{pmatrix} &= \alpha_{\sigma(i)}\mathbf{U} \begin{pmatrix} \mathbf{b}_{\sigma(i)}\mathbf{b}_{\sigma(i)}^{\perp} \\ \mathsf{Row}_{\sigma(i)}\left(\mathbf{b}_{\sigma(i)}\right) \end{pmatrix} \mathbf{V} \\ &= \alpha_{\sigma(i)}\mathbf{U} \begin{pmatrix} \mathbf{V}^{\perp}\mathbf{b}_{\sigma(i)}\mathbf{b}_{\sigma(i)}^{\perp} \\ \mathsf{Row}_{\sigma(i)}\left(\mathbf{V}^{\perp}\mathbf{b}_{\sigma(i)}\right) \end{pmatrix} \ . \end{aligned}$$

These two matrices are both of rank 1 and their row spaces are generated by $\mathbf{a}_i^{\perp}$ and $\mathbf{b}_{\sigma(i)}^{\perp}V$. Therefore since they represent the same spaces because of the equality $\mathbf{a}_i$ and $\mathbf{b}_{\sigma(i)}\mathbf{V}^{\perp}$ are linearly dependant. By combining the linear coefficient between them and the permutation we get the monomial map solving LEP. Also observe that $\mathbf{V}^{\perp}$ is the change of basis matrix.

## 3.3 Hardness

We need now to study the practical hardness of recovering a linear code isomorphism. We go through the most important techniques known in literature, in particular the algebraic modeling and the Leon's like attacks, some of them introduced already 40 years ago.

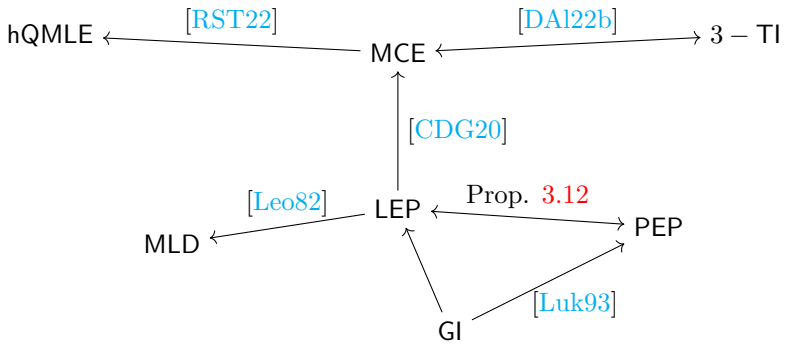



Figure 3.1: Reduction for code equivalence related problems

We give an estimate of the hardness, but also prove some reductions and discriminate a set of hardest instances for the equivalence problems. A summary diagram with reductions between the problems can be seen in Section 3.3. When relevant we also explain how exploiting a quantum computer can improve the final results.

### 3.3.1 Algebraic Attacks

A classical technique to solve a problem in complexity theory is to describe it via the use of polynomial equations, so that the set of solutions to the system corresponds to the set of solutions to the initial problem (a nice example of this reduction can be seen for MLD in [MPS23]).

For example consider a LEP instance $\mathcal{C}, \mathcal{C}'$ that satisfies

$$\mathbf{G}\mathbf{X}\mathbf{H}'^{\perp} = \mathbf{0} \ , \tag{3.21}$$

where $\mathbf{G}$ is the generator matrix of $\mathcal{C}$, $\mathbf{H}'$ the parity check matrix of $\mathcal{C}'$ and $\mathbf{X}$ a monomial map.[1] To consider a system of equations that finds LEP solutions we can assign as variables the coefficients of $\mathbf{X}$ as $\mathbf{X} = [x_{i,j}]_{i,j=1}^{n}$, then request them to satisfy:

- Equation (3.21);
- on each column there is only one non-zero elements, i.e. $x_{i,j}x_{i',j} = 0$ for all $i, i', j \in [1, n]$ with $i \neq i'$;
- the sum of any row is equal to a element of $\mathbb{F}_q^*$, i.e. that $(\sum_{l=1} x_{il})^{q-1} = 1$ for all $i \in [1, n]$[2].

Thus a matrix $\mathbf{X}$ is a monomial isometry for $\mathcal{C}, \mathcal{C}'$ if and only if it is a solution of the following polynomial system:

[1] we use $\mathbf{X}$ instead of $\mathbf{Q}$ to avoid confusion with the field cardinality $q$.
[2] By basic algebra since we are in a prime field the set of solutions of $z^{q-1} = 1$ is $\mathbb{F}_q^*$

$$\mathcal{Q}_{\mathcal{C},\mathcal{C}'} := \begin{cases} \mathbf{G}\mathbf{Z}\mathbf{H}'^{\perp} = \mathbf{0} \\ \sum_{j'=1}^{n} x_{i,j'} = 1, \quad 1 \leqslant i \leqslant n \\ x_{i,j}x_{i',j} = 0, \quad i \neq i' \end{cases} \tag{3.22}$$

So we can use well known techniques from computational algebra to solve LEP. We can do the same for PEP using the following system:

$$\mathcal{P}_{\mathcal{C},\mathcal{C}'} := \begin{cases} \mathbf{G}\mathbf{P}\mathbf{H}'^{\perp} = \mathbf{0} \\ \sum_{j'=1}^{n} p_{i,j'} = 1, \quad 1 \leqslant i \leqslant n \\ p_{i,j}p_{i',j} = 0, \quad i \neq i' \end{cases} \tag{3.23}$$

This approach have been extensively studied in the work [Sae17] by adjoining the systems with redundant equations and using Faugere's algorithms for Groebner Basis calculations. In details you can see [Sae17, Section 3] for PEP and [Sae17, Section 4] for LEP.

Thanks to this analysis we can state that algebraic attacks does not pose a threat to instances already secure for successive attacks.

Instead when trying to model the MCE problem in Section 6.2 of [Cho+22] they were to obtain competitive attacks.

For MCE start by considering two matrix $[n \times m, k]$-codes $\mathcal{C}$,$\mathcal{C}'$ with basis $(\mathbf{C}^{(1)}, ..., \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, ..., \mathbf{D}^{(k)})$ such that it exists $\mathbf{A} \in \mathbb{F}_q^{m\times m}$, $\mathbf{B} \in \mathbb{F}_q^{n\times n}$ and a change of basis $\mathbf{S} \in \mathbb{F}_q^{k\times k}$, all three invertible with :

$$\sum_{t=1}^{k} s_{rt}\mathbf{D}^{(t)} = \mathbf{A}\mathbf{C}^{(r)}\mathbf{B}, \quad \forall r \in [1, k] \ . \tag{3.24}$$

These three are the same that satisfies $\mathbf{G}' = \mathbf{S}\mathbf{G}(\mathbf{A}^{\perp} \otimes \mathbf{B})$, where $\mathbf{G}$ and $\mathbf{G}'$ have as $i$-th row $\mathsf{vec}(\mathbf{C}^{(i)})$ and $\mathsf{vec}(\mathbf{D}^{(i)})$ respectively. The variables are still the coefficients of these three matrices, this way we have $knm$ bilinear equations for $k^2 + n^2 + m^2$ unknowns. In [Cho+22] they showed how to improve the previous system. The first observation is that by guessing the $\mathbf{A}$ equations they get a linear system that can be solved efficiently in $O((n^2 + k^2)^3)$ time. To refine this idea we should try to guess only $\alpha m$ variables from the first $\alpha$ rows of $\mathbf{A}$ and consider only the first $\alpha kn$ equation. Clearly at this point choosing the smaller $\alpha$ so that we still obtain an indeterminate linear system is the optimal choice, in particular this way with $\alpha = \lceil \frac{n^2+k^2}{kn} \rceil$ the complexity goes to:

$$O\left(q^{m\frac{n^2+k^2}{kn}}(n^2 + k^2)^3\right) \ . \tag{3.25}$$

*Remark* 30. As seen in [DAl22b] and noted before MCE problem is equivalent to 3-TI problem, where none of the three coordinates plays a special role and can be cyclically exchanged by a change of basis. By using the reductions between the problems thus we can also move a MCE instance in tensor representation, change the representation basis and go back to matrix code representation. With this observation we see that none of the three secret matrices plays a special

role and the parameters $k, n, m$ can be exchanged easily in Equation (3.25) (and any other possible attack complexity).

Because of this clever trick the MEDS team fixed $n = m = k$ for any instance used in the final submission.

For the case $n = m = k$ we get $\alpha = 2$ and we can try to use a quantum speed up via the Grover Algorithm [Gro96] for the enumeration of the $2n$ variables for $\mathbf{A}$. Since the speed up is quadratic we go from a complexity of $O(q^{2n}n^6)$ to

$$O(\underbrace{q^n}_{\text{Grover}} \cdot \underbrace{n^6}_{\text{Gauss elim.}}) \ . \tag{3.26}$$

Since for implementation choices the actual instantiations have field size $q \geqslant 2039$ this speed up does not pose a threat to the quantum security of MCE.

In [Cho+22] they improved again on the current algebraic modeling by removing the $\mathbf{S}$ variables. This can be done in different ways, also shown in [Cho+23] and in relation to the tensor modeling. Here we show one of them, that uses the parity check matrix $\mathbf{H}' \in \mathbb{F}_q^{nm\times(nm-k)}$ of the code $\mathcal{C}'$ in the same way as Equation (3.21) to discard the change of basis map.

Consider the matrix:

$$\tilde{\mathbf{G}} := \mathbf{G}(\mathbf{A}^\perp \otimes \mathbf{B}) \ .$$

This matrix generates the same matrix code as $\mathbf{G}'$, only with a different base, thus as done before

$$\tilde{\mathbf{G}} \cdot \mathbf{H}'^\perp = \mathbf{0} \ . \tag{3.27}$$

This system has $k(nm-k)$ bilinear equations for $n^2 + m^2$ variables and can be solved with classical techniques to solve polynomial systems, like bilinear XL. In [Cho+23] a similar approach is then applied with variables $(\mathbf{A}, \mathbf{S})$, $(\mathbf{B}, \mathbf{S})$ and then all together. This approach cannot be improved on quantum computers since no relevant quantum speedup is known for this problem.

### 3.3.2 Leon Like Attacks

In [Leo82] Leon proposed an algorithm to find equivalences between codes (in the Hamming metric) built on the observation that usually the set of codewords with low weight is small, thus when considering two set of codewords of low weight from the two codes they contains enough information to recover the secret isomorphism via a trial and error matching of the codewords in the two lists.

The algorithm to find an isometry $\phi$ such that $\phi(\mathcal{C}) = \mathcal{C}'$ can be summarized as:

1. find $B_w := \{\mathbf{c} \in \mathcal{C} \mid w(\mathbf{c}) = w\}$ and $B'_w := \{\mathbf{c} \in \mathcal{C}' \mid w(\mathbf{c}) = w\}$;
2. find the set

$$\mathsf{Mor}_{\mathsf{Isom}_n}(B_w, B'_w) := \{\ \psi \in \mathsf{Isom}_n \mid \psi(B_w) = B'_w\}\ ; \tag{3.28}$$

where $\mathsf{Isom}_n$ is the set of isometries for the weight $w$;

3. find $\phi \in \mathsf{Mor}_{\mathsf{Isom}_n}(B_w, B'_w)$ with $\phi(\mathcal{C}) = \mathcal{C}'$.

The second step is polynomial in the average cardinality of $B_w$: $N_w := \mathbb{E}_{\mathcal{C}}[B_w]$, thus when choosing the parameter $w$ we need to observe that:

- $w$ is not high, since $N_w$ grow exponentially with respect to it;
- $w$ is not too low, otherwise the two sets $B_w, B'_w$ do not contain enough structure and the set of solution $\mathsf{Mor}_{\mathsf{Isom}_n}(B_w, B'_w)$ grows exponentially too;
- the low weight codewords computation represents the most relevant bottleneck for the algorithm, since it is equivalent to solve the well studied NP-complete problem Maximum Likelihood Decoding (Problem 16).

**Permutation Equivalence** The algorithm described before from [Leo82] has complexity:

$$O(\log(N_w) C_{ISD}(n, k, q, w)) \ ; \tag{3.29}$$

where the cost of computing $\mathsf{Mor}_{\mathbb{S}_n}$ and verifying the map is ignored.

The proof of this can be found in Appendix C [BBPS22], where the authors also explore optimal values of $w$, finding that for random codes Leon's procedure excels when $w$ is slightly larger than the GV distance ($d_{GV} + 1$ or $d_{GV} + 2$).

In [Beu20] Beullens observe that the Leon's idea can be improved using the Birthday Paradox. Instead of evaluating all $B_w, B'_w$ we can:

1. find two lists $L_{\mathcal{C}}, L_{\mathcal{C}'}$ of codewords from $\mathcal{C}$ and $\mathcal{C}'$ with fixed weight $w$;
2. for any pair $(\mathbf{c}, \mathbf{c}') \in L_{\mathcal{C}} \times L_{\mathcal{C}'}$ set it as a collision if

$$\{c_i\}_{i=1}^n = \{c'_i\}_{i=1}^n \ ; \tag{3.30}$$

   i.e. see if they have the same entries in different order;
3. from each collision imply that $\pi(i) \neq j$ if $c_i \neq c'_j$;
4. retrieve $\pi$ when there are enough collisions, approximately in [Beu20] $2n \log(n)$ codewords of weight $w$ are required to recover a map;
5. the algorithm succeeds if $\pi(\mathcal{C}) = \mathcal{C}'$.

Clearly point 2 lead to useful collisions, i.e. such that actually $\pi(\mathbf{c}) = \mathbf{c}'$, with much higher probability for large field sizes $q$. In this cases the complexity goes down to:

$$O\left(\sqrt{\frac{n \log(n)}{N_w}} C_{ISD}(n, k, q, w)\right) \ . \tag{3.31}$$

More details can be found again in Appendix C of [BBPS22].

**Linear Equivalence** When we take into consideration a monomial map $\tau \in \mathsf{Mono}_n$ instead of a permutation the multiset comparison as in (3.30) cannot be used anymore, thus Beullens observed that subcodes of fixed dimension and small support size can be used instead. In particular he proposed the use of:

$$X_w := \{\ \mathcal{B} \subset \mathcal{C} \mid \dim(\mathcal{B}) = 2 \text{ and } \#\ \mathsf{supp}(\mathcal{B}) = w\}$$

and $X'_w$ with the same characteristics, but for $\mathcal{C}'$ subcodes. Both in [Beu20; BBPS22] the list of elements from $X_w$ and $X'_w$ are compared using the first lexicographic base for them, i.e. the smallest generator matrix for the subcodes of the same orbit with respect to the lexicographic order. This invariant can be evaluated for any 2-dimensional subcode $\mathcal{B} \subset \mathcal{C}$ in $O(\max(q, n))$ computations via:

1. finding the minimal weigh codeword $\mathbf{x}$ of $\mathcal{B}$ via listing them all $(q + 1$ operations);
2. complete the basis with another independent codeword $\mathbf{y}$;
3. using a monomial map $\mu$ to have all the zeros at the start and ones in all the other $w(\mathbf{x})$ entries;
4. reorder the last $w(\mathbf{x})$ entries of $\mu(\mathbf{y})$ so that they are in lexicographic order.

From this point [BBPS22] improved by generating these 2-dimensional subcodes with small support size, say $w$, via:

1. using ISD to find codewords of weight $w'$ with $2w'$ slightly larger that $w$;
2. consider all the matrices generated from pairs $(\mathbf{x}, \mathbf{y})$ of these codewords;
3. keep only the one such that $\mathsf{supp}(\mathbf{x}) \cap \mathsf{supp}(\mathbf{y})$ has cardinality $2w' - w$, this way $\langle \mathbf{x}, \mathbf{y} \rangle$ has support size $w$.

In Proposition 8 [BBPS22] they estimated the optimal complexity as:

$$O\left(\frac{C_{ISD}(n, k, q, w')}{\sqrt{N_{w'}}} \left(\frac{n \log(n)}{\zeta_{w',w}}\right)^{1/4}\right)\ ; \tag{3.32}$$

where $\zeta_{w',w}$ is the probability that a subcode generated by two random codewords of weight $w'$ has support size $w$ (Lemma 2 [BBPS22]), that is equal to $\binom{w'}{2w'-w}\binom{n-w'}{w-w'}\binom{n}{w'}^{-1}$.

**Rank-metric version** First of all in the rank-metric the low weight codewords search is equivalent to the MinRank Problem (Problem 6).

The strategy proposed in [Cho+22] consist in finding collisions between pair of codewords $(\mathbf{C}_1, \mathbf{C}_2)$ and $(\mathbf{C}'_1, \mathbf{C}'_2)$. An isometry between them lead to the

following linear equations:

$$\begin{cases} \mathbf{A}\mathbf{C}_1 = \mathbf{C}'_1\mathbf{B}^{-1} \\ \mathbf{A}\mathbf{C}_2 = \mathbf{C}'_2\mathbf{B}^{-1} \\ \mathbf{A}^{-1}\mathbf{C}'_1 = \mathbf{C}_1\mathbf{B} \\ \mathbf{A}^{-1}\mathbf{C}'_2 = \mathbf{C}_2\mathbf{B} \end{cases} \tag{3.33}$$

that lead to $2n^2$ linear equations for $2n^2$ variables when $n = m$. Sadly since the codewords are not of full rank the number of independent equations is only at most $2n^2 - 2(n-r)^2$ leading to a solution space of dimension $2(n-r)^2$. This equations thus have to be combined with the equations from the algebraic model in Section 3.3.1. Now let's focus in the case $n = m = k$, in which the optimal rank to get a solution is $r = n - \sqrt{n}$. With this parameter choice the solution spaces have dimension $2n = 2(n - n + \sqrt{n})^2$ thus when we substitute Equation (3.33) into 3.27 we get a system that is bilinear with $2n$ variables for $\mathbf{A}$ and the same for $\mathbf{B}$ with $n(n^2 - n)$ linear equations. Hence we can linearise the system via assigning to each of the $4n^2$ quadratic monomials a variable and solve it via Gaussian elimination in $(4n^2)^\gamma$ operations with $\gamma \leqslant 3$ (the exact value depends on the particular algorithm used).

Now we need to evaluate the cost of the list enumeration and collision search. Let's set as $C_{MRk}(n, m, k, q)$ the complexity of solving Problem 6 (a recent estimate can be found in [Bar+20]). The average number of $n \times m$ matrices in $\mathbb{F}_q$ of rank $r$ in a matrix code of dimension $k$ can be estimated as $O(q^\sigma)$ with $\sigma = r(n + m - r) - nm + k$. This because the probability for a random matrix to lie in the code is $q^{nm-k}$, while the probability to have rank $r$ is approximated as $q^{r(r-n+m)}$.

By the Birthday Paradox the two lists need to have size $\sqrt{2q^\sigma}$. The list enumeration cost is dominated by the value $C_{MRk}(n, m, k, q)$, that we use as complexity. Eventually it can be improved by observing that since the number of expected solution for MinRank is $q^\sigma$ we can try to assign $\sigma$ variables and then run the MinRank solver bringing the complexity to $O(q^\sigma C_{MRk}(n, m, k - \sigma, q))$. This remark can be used also to exploit the Grover's Algorithm to bring the quantum complexity down to $O(q^{\sigma/2} C_{MRk}(n, m, k - \sigma, q))$. Thus we do the implicit assignment:

$$C_{MRk}(n, m, k, q) = \min(C_{MRk}(n, m, k, q), \underbrace{q^\sigma}_{\text{param. enum.}} C_{MRk}(n, m, k, q)) \tag{3.34}$$

Instead when looking at the collision search complexity the number of pairs from a list of size $\sqrt{2q^\sigma}$ is $\binom{\sqrt{2q^\sigma}}{2}$, thus the total number of elements we need to scan is:

$$\binom{\sqrt{2q^\sigma}}{2}^2 = O(q^{2\sigma}) \ . \tag{3.35}$$

Thus when $n = m = k$ and $r = n - \sqrt{n}$, the classical complexity is:

$$O(\max(\underbrace{C_{MRk}(n, m, k, q)}_{(3.34)}, \underbrace{q^{2\sigma}}_{(3.35)} \underbrace{n^{2\gamma}}_{\text{Gauss elim.}})) \ ; \tag{3.36}$$

that for a quantum computer via groverization can be reduced to:

$$O(\max(q^{\sigma/2} C_{MRk}(n, m, k-\sigma, q), \underbrace{q^{\sigma}}_{\text{Grover}} \underbrace{n^{2\gamma}}_{\text{Gauss elim.}})) \ . \tag{3.37}$$

If we consider just a single a pair of codewords $(\mathbf{C}, \mathbf{C}')$ of the same rank evaluating the rank-metric isometry boils down to solving the system $\mathbf{ACB} = \mathbf{C}'$. From this equation we can find linear equations via:

- taking an element $\mathbf{v}$ in the kernel of $\mathbf{C}'$, this way $\mathbf{ACBv} = \mathbf{0}$ and we have that $\mathbf{Bv}$ lie in the kernel of $\mathbf{C}$, yielding to $rm - r^2$ equations;
- taking an element $\mathbf{s}$ in the left kernel of $\mathbf{C}'$, this way $\mathbf{s}^{\perp}\mathbf{ACB} = \mathbf{0}$ and we have that $\mathbf{s}^{\perp}\mathbf{A}$ lie in the left kernel of $\mathbf{C}$, yielding again to $rn - r^2$ equations.

When these equations are combined with the one from Section 3.3.1 they cannot be solved in polynomial time via linearization. However, after fixing as $C_1(n, r)$ the complexity of this system when $n = m = k$, in [Cho+23] they carried over the analysis leading to a classical complexity of:

$$O(\max(\underbrace{q^{\sigma/2} C_{MRk}(n, m, k-\sigma, q)}_{\text{list enum.}}, \underbrace{q^{\sigma}}_{\text{num of pairs}} \cdot C_1(n, r)) \ ; \tag{3.38}$$

where $\sigma = (2n-r)r + n^2 - n$ and list sizes is $O(\sqrt{2q^{\sigma}}) = O(q^{\sigma/2})$. Consequently the quantum complexity goes down to:

$$O(\max(\underbrace{q^{\sigma/4}}_{\text{Grover}} C_{MRk}(n, m, k-\sigma, q), \underbrace{q^{\sigma/2}}_{\text{Grover}} \cdot C_1(n, r)) \ . \tag{3.39}$$

### 3.3.3 Support Splitting Algorithm

The Support Splitting Algorithm [Sen99] is a way to solve PEP (Problem 23) in polynomial time for codes with small hull dimension.

The idea is to define a signature function $\mathcal{S}(\mathcal{C}, i)$, with $i \in \{1, ..., n\}$, that it is invariant under the action of any permutation $\pi \in \mathbb{S}_n$:

$$\mathcal{S}(\mathcal{C}, i) = \mathcal{S}(\pi(\mathcal{C}), \pi(i)) \ . \tag{3.40}$$

Since the constant function technically is a signature function we need additional properties for $\mathcal{S}$:

- $\mathcal{S}$ is *discriminant* if for any code $\mathcal{C}$ there exists $i, j \in [1, n]$ with $\mathcal{S}(\mathcal{C}, i) \neq \mathcal{S}(\mathcal{C}, j)$;
- $\mathcal{S}$ is *fully discriminant* if for any code $\mathcal{C}$ for all $i \neq j \in [1, n]$ $\mathcal{S}(\mathcal{C}, i) \neq \mathcal{S}(\mathcal{C}, j)$ with overwhelming probability.

Observe that if $\mathcal{S}$ is fully discriminant, then for any instance of PEP $\mathcal{C}, \mathcal{C}' = \pi(\mathcal{C})$ we have that $\mathcal{S}(\mathcal{C}, i) = \mathcal{S}(\mathcal{C}', j)$ if and only if $j = \pi(i)$. So, via a fully discriminant signature, to recover the secret map $\pi$ in $2n$ steps is enough to compute all the values $s_i = \mathcal{S}(\mathcal{C}, i)$ and $s'_j = \mathcal{S}(\mathcal{C}', j)$ for $i \in [1, n]$, then use that $s_i = s'_j$ if and only if $\pi(i) = j$. This is the core process of the so called *Support Splitting Algorithm* (SSA).

The question now is if there *exists* an *efficient* fully discriminant signature. For existence we simply have to consider the any fine grained invariant for permutation equivalence on the punctured code $\mathcal{C}_i$, for example a brute force solution is

$$\mathcal{S}(\mathcal{C}, i) := \{(\pi(\mathcal{C}), \pi(\mathcal{C}_i)) | \pi \in \mathbb{S}_n\}$$

from Proposition 3.8 of [Sen99].

By looking closer to this framework we can see that:

**Proposition 3.17.** *PEP can be reduced to dPEP in polynomial time.*

*Proof.* Suppose we have a probabilistic polynomial-time algorithm $\mathsf{Evl}$ that solves $dPEP$, given $\mathcal{C}, \mathcal{C}' = \pi(\mathcal{C})$ if $\pi(i) = j$ then:

$$\pi(\mathcal{C}_i) = \pi(\mathcal{C})_{\pi(j)} = \mathcal{C}'_j \ .$$

Then we can use again the core idea of SSA, we go through all the pairs $(i, j) \in [1, n]^2$ and verify with $\mathsf{Evl}$ if $(\mathcal{C}_i, \mathcal{C}'_j)$ is a positive dPEP instance. If the codes are permutation equivalent then we find that $\pi(i) = j$. In $n^2$ calls to $\mathsf{Evl}$ thus we recover $\pi$. □

Now we need to focus in finding an invariant $\mathcal{V}$ efficiently computable so that the signature $\mathcal{S}(\mathcal{C}, i) := \mathcal{V}(\mathcal{C}_i)$ is fully discriminant. A possible idea is the use of the weight enumerator function $\mathsf{Wef}$, since the probability for two random codes to have the same weight distribution without being equivalent is negligible. However, this function requires the computation of weights for all the $q^k$ codewords, thus we need to consider a smaller code. The intuition of [Sen99] is to use the hull of the code, in fact for every permutation $\pi$:

$$\pi(\mathcal{H}(\mathcal{C})) = \pi(\mathcal{C} \cap \mathcal{C}^\perp) = \pi(\mathcal{C}) \cap \pi(\mathcal{C}^\perp) = \pi(\mathcal{C}) \cap \pi(\mathcal{C})^\perp = \mathcal{H}(\pi(\mathcal{C})) \ .$$

The we can finally define:

$$\mathcal{S}(\mathcal{C}, i) := \{ \ \mathsf{Wef}(\mathcal{H}(\mathcal{C}_i)), \mathsf{Wef}(\mathcal{H}(\mathcal{C}_i^\perp)) \ .\} \tag{3.41}$$

By the use of the closure $\mathcal{C} \otimes \mathbf{a}$ and the reduction in Proposition 3.12 we can use the SSA on the closure to solve also the LEP. For the particular details on how to transform a permutation in a monomial map you can see Lemma 4 [SS13].

It is clear that the complexity of this algorithm is exponential in the dimension of the hull $h$, in particular in Proposition 7 of [BBPS21], the authors evaluate the complexity as:

$$O(\underbrace{n^3}_{\mathcal{H}} + n^2 \cdot \underbrace{q^h}_{\mathsf{Wef}} \cdot \underbrace{\log(n)}_{\text{repetitions}} \ )$$

Finally, we can note that the hull of a random code in $\mathbb{F}_q^n$ is of small dimension with very high probability, as showed in [Sen97; Sen99]. This implies that PEP can be solved efficiently on a random instance, that rules out the use for cryptographic purposes of a good chunk of codes. However there are still some considerations to be done:

- it is possible to generate codes with arbitrary large hulls, as shown in Section B.0.1 of [Sae17], thus we can still use PEP instances in our algorithm, but we should be careful during the codes generation;
- the closure of a code on $\mathbb{F}_q$ with $q \geqslant 5$ is always weakly self dual, as pointed out in Proposition 6 of [SS13], thus we have a very large family we can use as hard instances.

**The Case for MCE** At our current state of knowledge it is not possible to use similar strategies for MCE, this is strongly related to the fact that there is no reduction from the search version to the decisional one.

A possible strategy for example would be to use the matrix spaces from [Rav16]:

$$\mathrm{Mat}_U(n \times m, \mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times m} \mid \mathrm{colsp}(M) \subset U\} \ ,$$

that can substitute the role of $\mathcal{E}_i$ in the puncturing and shortening definition (Definition 3.7). The problem is that, even using additional assumption like the use of congruent matrices for the equivalence, we can only hope to try to get the image of a dimension 1 vector, for which we have $q^n$ possibilities.

### 3.3.4 Relation to Graph Isomorphism

Before we have seen that when the hull of the code is trivial (i.e. $\mathcal{C} \cap \mathcal{C}^\perp = \{0\}$) the SSA does not work, however we can still solve this instances by studying their relations with graph theory, as showed in [BOS19].

In particular we can reduce PEP on codes with trivial hull to the problem of finding isomorphism in undirected weighted graphs, that is quasipolynomial [Bab16].

An undirected weighted graph is a graph $\Gamma = (V, E)$ with $V$ a finite set of $n$ vertices, $E$ the set of edges (pairs of elements in $V$), associated to a function $\mathsf{weight} : E \to \mathbb{F}$ that associates to each edge a weight in a field $\mathbb{F}$. If we order the set of vertices as $V = \{v_1, ..., v_n\}$ we can represent any weighted graph as an $n \times n$ matrix $\mathbf{A}$, called adjacency matrix, with:

$$a_{ij} := \begin{cases} 0 & \text{if } \{v_i, v_j\} \notin E \ , \\ \mathsf{weight}(\{v_i, v_j\}) & \text{otherwise} \ . \end{cases} \tag{3.42}$$

Given a code $\mathcal{C}$ with generator matrix $\mathbf{G}$ and trivial hull we can associate to a weighted graph $\Gamma_\mathcal{C}$ of $n$ vertex with the following adjacency matrix:

$$\mathbf{A}_\mathcal{C} := \mathbf{G}^\perp \left(\mathbf{G}\mathbf{G}^\perp\right)^{-1} \mathbf{G} \ . \tag{3.43}$$

*Remark* 31. The previous construction does not depends on the particular generator matrix $\mathbf{G}$, in fact if we apply a change of basis $\mathbf{S}$ we get:

$$(\mathbf{SG})^{\perp}\left(\mathbf{SG}(\mathbf{SG})^{\perp}\right)^{-1}\mathbf{SG} = \mathbf{G}^{\perp}\mathbf{S}^{\perp}\left(\mathbf{S}^{\perp}\right)^{-1}\left(\mathbf{GG}^{\perp}\right)^{-1}\mathbf{S}^{-1}\mathbf{SG} = \mathbf{A}_{\mathcal{C}} \ .$$

Now consider the following problem:

**Problem 32** (Weighted Graph Isomorphism Problem (WGI))**.** Given two weighted graphs $\Gamma, \mathsf{weight}$ and $\Gamma', \mathsf{weight}'$ of size $n$ find a graph isomorphism $\pi$, i.e. a permutation such that $\{v_i, v_j\} \in E$ if and only if $\{v_{\pi(i)}, v_{\pi(j)}\} \in E'$, that also satisfies:

$$\mathsf{weight}(\{v_i, v_j\}) = \mathsf{weight}(\{v_{\pi(i)}, v_{\pi(j)}\}) \ .$$

If $\mathbf{P}$ is the matrix representation of $\pi$ this is equivalent to ask:

$$\mathbf{A}_{\Gamma'} = \mathbf{P}^{\perp}\mathbf{A}_{\Gamma}\mathbf{P} \ .$$

At this point we have the following:

**Theorem 3.18.** *Two linear codes $\mathcal{C}$,$\mathcal{C}'$ are permutation equivalent with respect to the permutation $\pi$ if and only if $\Gamma_{\mathcal{C}}$,$\Gamma_{\mathcal{C}'}$ are isomorph with respect to the same permutation $\pi$.*

This way PEP can be decided in time

$$O(\underbrace{n^{\omega}}_{\text{matrix mult.}} \cdot C_{WGI}(n)) \ ;$$

where at today state of knowledge $\omega \simeq 2.373$ is the best time for matrix inversion while $C_{WGI}(n)$ is the complexity of deciding Problem 32.

In [BOS19] the analysis continue also to the non trivial hull case by observing that when considering an information set $I \subseteq [1, n]$ for the hull $\mathcal{H}(\mathcal{C})$ then the shortened $\mathcal{C}_{\setminus I}$ has trivial hull. In literature this attack is also called BOS in relation to the initials of the authors. This way in Theorem 10 [BOS19] they proved that the complexity of deciding PEP with BOS approach for a code with hull dimension $h$ is asymptotically:

$$O(\underbrace{n^{\omega}}_{\text{matrix mult.}} \cdot \underbrace{n^{(h+1)}h}_{\text{search info. set}} \cdot C_{WGI}(n)) \ . \tag{3.44}$$

**Graph Theory Approach for MCE** Also for MCE we can use results from graph theory [BFV13] via a reduction to problems related to isomorphism of polynomials [RST22].

In particular they prove in [RST22, Theorem 15] that, if we assume the symmetric matrix representations of the instances having trivial automorphism group, MCE reduces to the following problem for $N = n + m$:

**Problem 33** (hQMLE)**.** Give two tuples of quadratic polynomials in $\mathbb{F}_q[x_1, ..., x_N]$:

$$\mathcal{F} = (f_1, ..., f_k) \text{ and } \mathcal{F}' = (f'_1, ..., f'_k)$$

find a linear isomorphism between them, i.e. $\mathbf{S} \in \mathsf{GL}_N$ and $\mathbf{T} \in \mathsf{GL}_k$ that satisfy:

$$\mathcal{F}'(\mathbf{x}) = (\mathcal{F}(\mathbf{x}\mathbf{S}))\mathbf{T} \ .$$

At this point the approach resembles the Leon's like ones. It requires the use of a binary predicate $\mathbb{P}$ that is invariant under isomorphism, which is used to generate two lists $L_{\mathcal{F}}$ and $L_{\mathcal{F}'}$ starting from $\mathcal{F}$ and $\mathcal{F}'$ respectively, satisfying $\mathbb{P}$. The universe for the elements of the list depends on the choice of the predicate. At this point they search for collisions in the two lists that result in a simpler equivalence problem via the Birthday Paradox.

In [RST22] they consider as predicate the functions $\mathbb{F}_q^N \ni \mathbf{a} \mapsto \dim\ker D_{\mathbf{a}}\mathcal{F} = \kappa$ and $\mathbb{F}_q^N \ni \mathbf{b} \mapsto \dim\ker D_{\mathbf{b}}\mathcal{F}' = \kappa$, obtaining complexity:

$$O(\max(\underbrace{\sqrt{\frac{q^{m+n}}{d_\kappa}}}_{\text{lists enum.}}, \underbrace{q^{m+n}d_\kappa}_{\text{collisions find}})) \ ; \tag{3.45}$$

where $d_\kappa$ is the probability of $\mathbb{P}$ being true as a function of the integer $\kappa$. In the particular instance $k = n = m$ described in Remark 30 this approach perform poorly with respect to the one in Section 3.3.2 even for optimal choices of $\kappa$.

*Remark* 34. Here the Grover's Algorithm can be used both during lists enumeration and collision finding, leading to a quantum complexity of:

$$O(\max(\underbrace{\left(\frac{q^{m+n}}{d_\kappa}\right)^{\frac{1}{4}}}_{\text{lists enum.}}, \underbrace{\left(q^{m+n}d_\kappa\right)^{\frac{1}{2}}}_{\text{collisions find}})) \ ; \tag{3.46}$$

## 3.4 Uses and Parameters

The two isometry problems derived from coding theory are used in two of the schemes submitted to the NIST additional call for post quantum cryptography [NIS23]. In both these schemes Protocol 2.2.1 is instantiated with the group actions proposed in Section 3.2, rendered via the Fiat-Shamir transform to secure digital signatures. Both these protocols benefit from optimizations proposed in Section 2.3: fixed weight challenges, multiple bits challenges and seed tree generation for the ephemeral elements.

### 3.4.1 LESS

The *Linear Equivalence Signature Scheme* proposed in [BMPS20; BBPS21] is based on the Linear Equivalence Problem (Problem 22 in Section 3.2.1) and uses the group action from Equation (3.14). The identification protocol underlying the signature is Protocol 3.4.1.

Before discussing in details the parameters achieved by LESS we insert here one optimizations specific for LEP based protocols.

| | | |
|---|---|---|
| Public Data : Parameters $n, k, q$, linear $[n,k]$-code with generator matrix $\mathbf{G}$, group action $\star$ from (3.14) and hash function $\mathsf{H}$. | | |
| Private Key : Monomial map $\mathbf{Q} \in \mathsf{Mono}_n$. | | |
| Public Key : $\mathbf{G}' = \mathbf{Q} \star \mathbf{G}$. | | |
| **PROVER** | | **VERIFIER** |
| Get $\tilde{\mathbf{Q}} \xleftarrow{\$} \mathsf{Mono}_n$, send $\mathsf{com} = \mathsf{H}(\tilde{\mathbf{Q}} \star \mathbf{G})$ | $\xrightarrow{\mathsf{com}}$ | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0,1\}$. |
| If $\mathsf{ch} = 0$ then $\mathsf{resp} \leftarrow \tilde{\mathbf{Q}}$. | $\xrightarrow{\mathsf{resp}}$ | Accept if $\mathsf{H}(\mathsf{resp} \star \mathbf{G}) = \mathsf{com}$. |
| If $\mathsf{ch} = 1$ then $\mathsf{resp} \leftarrow \mathbf{Q}^{-1}\tilde{\mathbf{Q}}$. | | Accept if $\mathsf{H}(\mathsf{resp} \star \mathbf{G}') = \mathsf{com}$. |

Protocol 3.4.1: One round of the LESS signature.

**Information Set LEP** In [PS23] the authors showed how to half the communication cost of the group elements with only negligible additional computations. The core idea is to verify the group action only on one information set.

First they define a version of LEP focused on just one information set:

**Problem 35** (Information Set - Linear Equivalence Problem (IS-LEP))**.** Given two linear $[n,k]$-codes $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$ say if they are Information Set (IS) linearly equivalent, i.e. say if there exist monomial transformations $\tilde{\mu} \in \mathsf{Mono}_n, \zeta \in \mathsf{Mono}_{n-k}$ and an information set $J'$ for both $\mathcal{C}'$ and $\widetilde{\mathcal{C}} = \tilde{\mu}(\mathcal{C})$ such that, given generator matrices $\widetilde{\mathbf{G}}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ for $\tilde{\mathcal{C}}$ and $\mathcal{C}'$, it must be

$$\widetilde{\mathbf{G}}_{J'}^{-1} \widetilde{\mathbf{G}}_{\{1,\cdots,n\} \setminus J'} = \zeta \left( \mathbf{G}_{J'}^{-1} \mathbf{G}'_{\{1,\cdots,n\} \setminus J'} \right).$$

In Theorem 1 [PS23] they proved that two linear codes are linearly equivalent if and only if they are IS linearly equivalent, implying immediately a reduction between IS-LEP and LEP. With this result in mind they proceed to define Protocol 3.4.2 for IS-LEP, that is equivalent to Protocol 3.4.1, but on challenge $\mathsf{ch} = 1$ halves the communication cost.

We do not include here the details about the proof for the protocol (that can be read on [PS23]), but we go through the protocol to explain the computations done here.

First of all we have to understand more precisely the meaning of Information Set - Linear Equivalence between the two codes. The idea is to consider a monomial map $\tilde{\mu}$ that generates a new code from $\mathcal{C}$. The natural choice for this monomial map is the linear equivalence between them.

We need a common information set $J' = \{j'_1, ..., j'_k\}$ in these two codes, such that, when we compute the systematic form on it (i.e. we put the columns in $J'$ in equal to the identity), the two non-systematic parts of the matrices are identical, up to a monomial transformation.

To verify this equivalence up to monomial transformation we can consider the orbit of all the matrices obtained via a monomial transformation, then take a representative for this set. This is done in the protocol via the function $\mathsf{MinLex}$,

that evaluates the smaller matrix with respect to the lexicographical order of the columns.

*Remark* 36. For the map $\tilde{\mu}$ we only need to know how it acts on the information set of cardinality $k$, while we are not interested in how the rest of the code is handled, since via MinLex the non-systematic parts are confronted without the need of the monomial map $\zeta$. We compute these informations via the function Trunc, that returns a sequence of integers that represent the preimage of the information set $J'$ and the sequence of coefficients of the monomial map $\tilde{\mu}$ on $J'$. These can then be used with the function apply over $\mathbf{G}'$ to retrieve the columns associated to the information set $J$ in the right order and scale them up accordingly, essentially the matrix we need to invert to put $\mathbf{G}'$ and $\tilde{\mathbf{G}}$ on systematic form.

This observations contains all the tools used to verify IS-Linear Equivalence in Protocol 3.4.2.

- The protocol start by applying a random monomial map on $\mathcal{C}$. To commit it uses $\mathrm{SF}^*$, that find an information set $J^*$ in a deterministic way and evaluate the systematic form on it by doing $\mathbf{A}^* = (\mathbf{G}^*_{J'})^{-1}\mathbf{G}^*$.
- On challenge 0 he just need to send the ephemeral map and the verifier repeats the commitment procedure.
- On challenge 1 the prover consider the monomial map between $\mathcal{C}'$ and $\mathcal{C}\mathbf{Q}$, at this point as pointed out in Remark 36 he only need to send the trunked map on the information set $J^*$ used for the commitment. The verifier now simply follows the procedure explained in Remark 36.

| | | |
|---|---|---|
| Public Data : Parameters $n, k, q$, linear $[n,k]$-code with generator matrix $\mathbf{G}$, and hash function $\mathsf{H}$. | | |
| Private Key : Monomial map $\mathbf{Q} \in \mathsf{Mono}_n$. | | |
| Public Key : $\mathbf{G}' = \mathrm{SF}(\mathbf{GQ})$. | | |
| Get $\tilde{\mathbf{Q}} \xleftarrow{\$} \mathsf{Mono}_n$, set $\mathbf{G}^* \leftarrow \mathbf{G}\tilde{\mathbf{Q}}$; Set $J^*, \mathbf{A}^* \leftarrow \mathrm{SF}^*(\mathbf{G}^*)$ Send $\mathsf{com} = \mathsf{H}(\mathbf{A}^*)$ | $\xrightarrow{\mathsf{com}}$ | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0,1\}$. |
| If $\mathsf{ch} = 0$ then $\mathsf{resp} \leftarrow \tilde{\mathbf{Q}}$. | $\xrightarrow{\mathsf{resp}}$ | Set $J^*, \mathbf{A}^* \leftarrow \mathrm{SF}^*(\tilde{\mathbf{G}} \cdot \mathsf{resp})$; Accept if $\mathsf{H}(\mathbf{A}^*) = \mathsf{com}$. |
| If $\mathsf{ch} = 1$ then $\mathbf{Q}' \leftarrow \mathbf{Q}^{-1}\tilde{\mathbf{Q}}$. Set $\mathsf{resp}_0, \mathsf{resp}_1 \leftarrow \mathsf{Trunc}(\mathbf{Q}', J^*)$ | $\xrightarrow{(\mathsf{resp}_0, \mathsf{resp}_1)}$ | Set $\mathbf{U} \leftarrow \mathsf{apply}((\mathsf{resp}_0, \mathsf{resp}_1), \mathbf{G}')$; Set $\mathbf{A}^* \leftarrow \mathsf{MinLex}(\mathbf{U}^{-1}\mathbf{G}'_{[1,n]\setminus \mathsf{resp}_0})$; Accept if $\mathsf{H}(\mathbf{A}^*) = \mathsf{com}$. |

Protocol 3.4.2: Identification protocol equivalent to Protocol 3.4.1.

**Parameters** We can finally see how to instantiate the LESS signature. We use the IS-LEP identification protocol in combination with Fixed-weight (Section 2.3.2) and Multiple bits (Section 2.3.5) challenges, while to compress the seeds we use a Seed Tree structure (Section 2.3.3).

| Name | $n$ | $k$ | $q$ | $t$ | $w$ | $s$ | pk (KiBytes) | sig (KiBytes) | KeyGen (Mcyc.) | Sign (Mcyc.) | Verify (Mcyc.) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NIST Security Level I | | | | | | | | | | | |
| LESS-1b | 252 | 126 | 127 | 247 | 30 | 2 | 13.7 | 8.4 | 0.9 | 263.6 | 271.4 |
| LESS-1i | | | | 244 | 20 | 4 | 41.1 | 6.1 | 2.3 | 254.3 | 263.4 |
| LESS-1s | | | | 198 | 17 | 8 | 95.9 | 5.2 | 5.1 | 206.6 | 213.4 |
| NIST Security Level III | | | | | | | | | | | |
| LESS-3b | 400 | 200 | 127 | 759 | 33 | 2 | 34.5 | 18.4 | 2.8 | 2446.9 | 2521.4 |
| LESS-3s | | | | 895 | 26 | 3 | 68.9 | 14.1 | 5.2 | 2984.3 | 3075.1 |
| NIST Security Level V | | | | | | | | | | | |
| LESS-5b | 548 | 274 | 127 | 1352 | 40 | 2 | 64.6 | 32.5 | 6.4 | 10212.6 | 10458.8 |
| LESS-5s | | | | 907 | 37 | 3 | 129.0 | 26.1 | 11.7 | 6763.2 | 7016.5 |

Table 3.1: Parameters for LESS-FM contained in the specifications [Bal+23b] submitted to NIST.

By the combination of Corollary 2.12, Proposition 2.14, Proposition 2.16 we have a quantum secure digital signature in the sEUF-CMA model, under the assumption of LEP (Problem 22) hardness and Assumption 29 on the automorphisms of random linear codes.

Given a security parameter $\lambda$ take $t, w, r$ that satisfy:

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

Then using $[n,k]_q$ random linear codes we get a public key of size:

$$\underbrace{\lambda}_{\text{seed}} + (r-1)\underbrace{k(n-k)\lceil \log_2(q)\rceil}_{\mathbf{G}_i \text{ weight}}$$

and a signature of size

$$\underbrace{2\lambda}_{\text{salt}} + \underbrace{\lceil t\log_2(r)\rceil}_{\text{com}} + \underbrace{w\log_2\left(\frac{t}{w}\right)\lambda}_{\text{seed tree paths}} + w\underbrace{k(\lceil \log_2(q)\rceil + \lceil \log_2(n)\rceil)}_{\text{IS-LEP mono. weight}} \ .$$

The secure parameter choices for NIST Security Level I ($\lambda = 128$), III ($\lambda = 192$) and V ($\lambda = 256$) used in the submitted specifications [Bal+23b] are listed in Table 3.1. Observe that for any level both a balanced instantiation, that tries to optimize together public key and signature sizes (marked with $b$ in the name), and a low signature size oriented one are proposed (marked with $s$).

### 3.4.2 MEDS

The *Matrix Equivalence Digital Signature* Scheme proposed in [Cho+22] is based on the Matrix Code Equivalence Problem (Problem 25) and uses the group action from Equation (3.19).

Public Data : Parameters $n, q$, linear $[n \times n, n]$-code with generator matrix $\mathbf{G} \in \mathbb{F}_q^{n^2 \times n}$, group action $\star$ from (3.19) and hash function $\mathsf{H}$.
Private Key : Equivalence given by $\mathbf{A}, \mathbf{B} \in \mathsf{GL}_n$.
Public Key : $\mathbf{G}' = (\mathbf{A}, \mathbf{B}) \star \mathbf{G}$.

| **PROVER** | | **VERIFIER** |
|---|---|---|
| Get $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}}) \xleftarrow{\$} \mathsf{GL}_n$, send $\mathsf{com} = \mathsf{H}((\tilde{\mathbf{A}}, \tilde{\mathbf{B}}) \star \mathbf{G})$ | $\xrightarrow{\mathsf{com}}$ | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0, 1\}$. |
| If $\mathsf{ch} = 0$ then $\mathsf{resp} \leftarrow (\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$. | $\xrightarrow{\mathsf{resp}}$ | Accept if $\mathsf{H}(\mathsf{resp} \star \mathbf{G}) = \mathsf{com}$. |
| If $\mathsf{ch} = 1$ then $\mathsf{resp} \leftarrow (\mathbf{A}^{-1}\tilde{\mathbf{A}}, \tilde{\mathbf{B}}\mathbf{B}^{-1})$. | | Accept if $\mathsf{H}(\mathsf{resp} \star \mathbf{G}') = \mathsf{com}$. |

Protocol 3.4.3: One round of MEDS.

**Public Key Compression** The natural way to generate a public key is to generate from random seeds an origin matrix $\mathbf{G}$ and the secret key, evaluate the group action to get $\mathbf{G}' = (\mathbf{A}, \mathbf{B}) \star \mathbf{G}$ and publish $\mathbf{G}'$ as a full matrix and $\mathbf{G}$ via the seed used in the generation to save space.

However in [Cho+22; Cho+23] they noted that, via generating the secret matrices with more attention, it is possible to save some more bytes when publishing the key. The new key generation proceeds as follows.

1. Generate from a seed $\mathbf{G} \in \mathbb{F}_q^{mn \times k}$ and two codewords $\mathbf{P}'_0, \mathbf{P}'_1 \in \mathbb{F}_q^{m \times n}$.
2. Generate from a secret seed a change of basis matrix $\mathbf{T} \in \mathsf{GL}_k$ and compute $\hat{\mathbf{G}} = \mathbf{TG}$. Label as $\mathbf{P}_0, \mathbf{P}_1 \in \mathbb{F}_q^{m \times n}$ the first two codewords of the generator matrix $\hat{\mathbf{G}}$.
3. Solve the following equations using as variables the entries of $\mathbf{A}$ and $\mathbf{B}^{-1}$:

$$\begin{cases} \mathbf{AP}_1 = \mathbf{P}'_1\mathbf{B}^{-1} \\ \mathbf{AP}_2 = \mathbf{P}'_2\mathbf{B}^{-1} \end{cases} ; \tag{3.47}$$

similarly as it was done for Leon's like attacks in Section 3.3.2. Eventually the process can be restarted if ranks of the codewords do not match, but this happens with negligible probability. The linear the system in (3.47) require relatively intensive overhead computations, thus in Section 5.2 [Cho+23] the authors suggest to use as codewords $\mathbf{P}'_0, \mathbf{P}'_1$ the identity and the identity shifted by one, this way the system became sparse and structured, leading to optimized solutions.

This optimization can be used for each matrix created for the public key when using the Multiple Bits challenges (Section 2.3.5), with the caution of using a different change of base $\mathbf{T}$ for each of the $r-1$ non ephemeral matrices. Thus the final public key generated this way uses $\lambda+(r-1)((k-2)(mn-k)+n)\lceil \log_2(q) \rceil$ bits.

**Group Element Compression** To reduce signature size another group element compression technique similar is proposed in [Cho+23, Section 8]. The

process uses the same idea of retrieving the invertible matrix via two bidimensional subcodes.

The optimized protocol can be seen in 3.4.4. During the commitment the user generate via a public seed a full rank matrix $\mathbf{R} \in \mathbb{F}_q^{2\times mn}$, i.e. two random independent codewords, and take two random codewords in the code generated by $\mathbf{G}$ using $\mathbf{M}$. Then he use this pair of dimension 2 subcodes to solve Equation (3.47), as for the key generation, and get $\tilde{\mathbf{A}}_M, \tilde{\mathbf{B}}_M$, used to evaluate the final code as usual.

This process is repeated in the same way during verification for $\mathsf{ch} = 0$ case, instead for $\mathsf{ch} = 1$ the prover need to send a pair of codewords in the code generated by $\mathbf{G}'$, associated to the pair $\mathbf{R}$. Since we have $\mathbf{TG}(\mathbf{A}^{\perp} \otimes \mathbf{B}) = \mathbf{G}'$ and $\mathbf{MG}(\tilde{\mathbf{A}}^{\perp} \otimes \tilde{\mathbf{B}}) = \mathbf{R}$ we get:

$$\mathbf{R} = \mathbf{MG}(\tilde{\mathbf{A}}^{\perp} \otimes \tilde{\mathbf{B}}) = \mathbf{MT}^{-1}\mathbf{G}'(\mathbf{A}^{\perp} \otimes \mathbf{B})^{-1}(\tilde{\mathbf{A}}^{\perp} \otimes \tilde{\mathbf{B}}) = \\ = \mathbf{MT}^{-1}\mathbf{G}'((\mathbf{A}^{-1}\tilde{\mathbf{A}})^{\perp} \otimes \mathbf{B}^{-1}\tilde{\mathbf{B}}) \ . \quad (3.48)$$

This way from $\mathbf{MT}^{-1}\mathbf{G}'$ and $\mathbf{R}$ the verifier can recover the equivalence $(\mathbf{A}^{-1}\tilde{\mathbf{A}}), \mathbf{B}^{-1}\tilde{\mathbf{B}})$ and verify the commitment. Note that $\mathbf{R}$ is ephemeral and can be obtained from the public seed, while the response from $\mathsf{ch} = 0$ is only composed again by ephemeral matrices, while for $\mathsf{ch} = 1$ only the $2 \times k$ matrix $\mathbf{MT}^{-1}$ is sent, cutting the communication cost to $\lceil 2 \cdot k \cdot \log_2(q) \rceil$ bits.

| | | |
|---|---|---|
| Public Data : Parameters $n, k, m, q$, linear $[m \times n, k]$-code with generator matrix $\mathbf{G}$, pair of codewords $\mathbf{P}' \in \mathbb{F}_q^{2\times nm}$ used for key generation and hash function $\mathsf{H}$. | | |
| Private Key : Equivalence given by $\mathbf{A} \in \mathsf{GL}_m, \mathbf{B} \in \mathsf{GL}_n$ and change of basis $\mathbf{T} \in \mathsf{GL}_k$. | | |
| Public Key : $\mathbf{G}' = (\mathbf{A}, \mathbf{B}) \star \mathbf{G}$. | | |
| Get $\mathbf{R} \xleftarrow{\$} \mathbb{F}_q^{2\times mn}$, $\mathbf{M} \xleftarrow{\$} \mathbb{F}_q^{2\times k}$; | | |
| Solve (3.47) using the pairs $\mathbf{MG}$ and $\mathbf{R}$ and get $\tilde{\mathbf{A}} \in \mathsf{GL}_m, \tilde{\mathbf{B}} \in \mathsf{GL}_n$; | | |
| Send $\mathsf{com} \leftarrow \mathsf{H}((\tilde{\mathbf{A}}, \tilde{\mathbf{B}}) \star \mathbf{G})$ | $\xrightarrow{\mathsf{com}, \mathbf{R}}$ | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0, 1\}$. |
| If $\mathsf{ch} = 0$ then $\mathsf{resp} \leftarrow \mathbf{M}$. | $\xrightarrow{\mathsf{resp}}$ | Solve (3.47) using the pairs $\mathsf{resp} \cdot \mathbf{G}$ and $\mathbf{R}$ and get $\tilde{\mathbf{A}}', \tilde{\mathbf{B}}'$; Accept if $\mathsf{H}((\tilde{\mathbf{A}}', \tilde{\mathbf{B}}') \star \mathbf{G}) = \mathsf{com}$. |
| If $\mathsf{ch} = 1$ then $\mathsf{resp} \leftarrow \mathbf{MT}^{-1}$. | $\xrightarrow{(\mathsf{resp}_0, \mathsf{resp}_1)}$ | Solve (3.47) using the pairs $\mathsf{resp} \cdot \mathbf{G}'$ and $\mathbf{R}$ and get $\tilde{\mathbf{A}}', \tilde{\mathbf{B}}'$; Accept if $\mathsf{H}((\tilde{\mathbf{A}}', \tilde{\mathbf{B}}') \star \mathbf{G}') = \mathsf{com}$. |

Protocol 3.4.4: Identification protocol equivalent to Protocol 3.4.3.

**Parameters** We can finally instantiate also the MEDS signature, to do that we can use again Fixed-weight (Section 2.3.2) and Multiple bits (Section 2.3.5) challenges, while we compress seeds via a Seed Tree structure (Section 2.3.3) and public key using the procedure presented before.

| Parameter Set | $q$ | $n = m = k$ | $r$ | $t$ | $w$ | pk (B) | sig (B) | sig* (B) | KeyGen (Mcyc.) | Sign (Mcyc.) | Verify (Mcyc.) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NIST Security Level I | | | | | | | | | | | |
| MEDS-9923 | 4093 | 14 | 4 | 1152 | 14 | 9'923 | 9'896 | 2'264 | 1.90 | 518.05 | 515.58 |
| MEDS-13220 | | | 5 | 192 | 20 | 13'220 | 12'976 | 2'544 | 2.51 | 88.90 | 87.48 |
| NIST Security Level III | | | | | | | | | | | |
| MEDS-41711 | 4093 | 22 | 4 | 608 | 26 | 41'711 | 41'080 | 5'344 | 9.80 | 1467.00 | 1461.00 |
| MEDS-69497 | | | 5 | 160 | 36 | 55'604 | 54'736 | 5'135 | 12.82 | 387.27 | 380.70 |
| NIST Security Level V | | | | | | | | | | | |
| MEDS-134180 | 2039 | 30 | 5 | 192 | 52 | 134'180 | 132'528 | 9'496 | 44.75 | 1629.84 | 1612.57 |
| MEDS-167717 | | | 6 | 112 | 66 | 167'717 | 165'464 | 9'653 | 55.83 | 961.80 | 938.89 |

Table 3.2: Parameters for MEDS contained in the specifications [Cho+23] submitted to NIST.

By the combination of Corollary 2.13, Proposition 2.14, Proposition 2.16 we have a quantum secure digital signature in the sEUF-CMA model, under the assumption of MEC Problem (Problem 25) hardness and Assumption 29 on the automorphisms of random linear matrix codes.

Given a security parameter $\lambda$ consider $t, w, r$ (as used in Section 2.3) satisfying:

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

When using $[n \times n, n]_q$ random linear codes we get a public key of size:

$$\underbrace{\lambda}_{\text{seed}} + (r-1)\underbrace{n(n^2-3n+3)\lceil \log_2(q) \rceil}_{\mathbf{G}_i \text{ weight compressed}}$$

and a signature (without using group element compression) of size

$$\underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lceil t \log_2(r) \rceil}_{\mathsf{com}} \underbrace{w \log_2\left(\frac{t}{w}\right)\lambda}_{\text{seed tree paths}} + w \underbrace{2n^2\lceil \log_2(q) \rceil}_{(\mathbf{A},\mathbf{B}) \text{ weight}} \ .$$

When using the group element compression the signature size goes down to

$$\underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lceil t \log_2(r) \rceil}_{\mathsf{com}} + \underbrace{w \log_2\left(\frac{t}{w}\right)\lambda}_{\text{seed tree paths}} + w \underbrace{\lceil 2 \cdot n \cdot \lceil \log_2(q) \rceil \rceil}_{(\mathbf{A},\mathbf{B}) \text{ compr. weight}} \tag{3.49}$$

The secure parameter choices for NIST Security Level I, III and V used in the submitted specifications [Cho+23] are listed in Table 3.2. Observe that for any level they have two instantiations, one with smaller key and signature size and another with smaller number of rounds $t$, which decreases significantly the signing and verification time.

Note that in the round 1 specifications the group element compression was not used since it needs further study and ad-hoc parameter choices. Thus we inserted the signature sizes from the specifications in the eighth column, while

in the ninth one ($\mathsf{sig}^*$)we have estimated by our selves the signature size with the group compression technique to give an idea of the effectiveness of the idea.

*Remark* 37. During the write up of this thesis a new optimization for MEDS and LESS signature size has been proposed by Chou, Persichetti, and Santini in [CPS23].

CHAPTER

# FOUR

# THRESHOLD FUNCTIONALITIES

In this chapter we start by briefly recalling some notions for threshold signature schemes, using as reference [GJKR96] and other texts from the literature, but trying to adapt the notation to the one used before. Then in Section 4.1 we design a full threshold scheme for any group action and, after discussing the security concerns associated to the construction, we prove its security. Successively we proceed in leveraging it to a general threshold scheme. Finally, in Section 4.4, we explain how to use the construction to obtain a threshold version of the LESS and MEDS signature, adapting also optimizations from Section 2.3 and 3.4.

**Secret Sharing** Threshold signatures often require to share some secret $s$ in a field $\mathbb{F}$. Here we highlight two of the most common algorithm to do so.

- *Replicated secret sharing* schemes define additive shares for a secret in a ring, then each share is sent to a particular subset of user so that only authorized set of users can have access to all the shares. This technique was introduced in [ISN89], but can be read also in [CS20; BBY20]. The ideas can be used in the same way also for abelian groups and, with some drawbacks, also in the non-abelian ones (the interesting situations for us).
- *Linear secret sharings* schemes are the most efficient way to achieve it. They require that each authorized set of users is able to efficiently compute a linear transformation on the shares which outputs the secret $s$. The main examples are the Shamir's scheme (the most used one, [Sha79]) and the Blakley's scheme ([Bla79]).

  Briefly the idea of the first is to consider a polynomial $f$ of degree $T-1$, then share to $P_i$ the value $f(i)$, while the secret is $s = f(0)$. Through linear Lagrange interpolation any $T$ parties can recover the secret $f(0)$.

**Communication Model** We take into consideration a set of $n$ users (or parties) $P_1, ..., P_n$ that can compute probabilistic polynomial-time algorithms.

We assume that each user has access to a secure, reliable and authenticated private channel with each of the other users. There is also for each user a broadcast channel that may be use to send authenticated messages to all the other parties. Here we are not interested in the specific design and peculiarities of the channel, such as latency and noise during messages exchanges.

We finally give a formal definition of a threshold signature defined with respect to a "centralized" digital signature. The idea behind a threshold signature scheme is to allow any subset of $T$ out of $N$ user that agree on a message $\mathsf{m}$ to sign it so that it results signed by the whole group, while this should be unfeasible in probabilistic polynomial-time by any smaller group of $T-1$ or less users. Note that setup and verification are the same of the centralized one.

**Definition 4.1.** Give a digital signature $\mathsf{DS}$ = ($\mathsf{DS.Setup}$, $\mathsf{DS.KeyGen}$, $\mathsf{DS.Sign}$, $\mathsf{DS.Verify}$) defined according to Definition 1.1 a $(T, N)$-threshold signature for $\mathsf{DS}$ for the users $P_1, ..., P_n$ is a tuple $\mathsf{TDS.DS}$ = ($\mathsf{DS.Setup}$, $\mathsf{TDS.KeyGen}$, $\mathsf{TDS.Sign}$, $\mathsf{DS.Verify}$) where:

- $\mathsf{DS.Setup}$, $\mathsf{DS.Verify}$ are the same for the original signature scheme, also referred as *centralized scheme*.
- $\mathsf{TDS.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk}_1, ..., \mathsf{sk}_N)$, on input the public parameters $\mathsf{pp}$ returns a public key and $N$ secret keys. Each of secret keys $\mathsf{sk}_i$ is known by only by the user $P_i$. There exists also a master secret key $\mathsf{sk}$, not necessarily computed explicitly, associated to $\mathsf{pk}$ that can be recovered from any subset of $T$ secret keys in polynomial time. The Key Generation my be run by a trusted third party (in this case we say that it is a *centralized key generation*) or by the $N$ users collectively (called *decentralized key generation*).
- $\mathsf{TDS.Sign}(\{\mathsf{sk}_{i_j}\}_{j=1}^{\perp}, \mathsf{m}) \rightarrow \mathsf{sig}$, on input the secret keys of a set of $T$ distinct users and a message $\mathsf{m}$ the algorithm is run by the $T$ users collectively. It outputs a signature $\mathsf{sig}$.

Since $\mathsf{KeyGen}$ and $\mathsf{Sign}$ may be run collectively we refer as the *view* of a set of users as the probability distribution on the transcripts of all the data available to them during the execution of the protocol, like the secret keys, the message $\mathsf{m}$, the signature; but also the outputs received by the other parties necessary to complete the protocols executions.

*Remark* 38. In general we may allow also the possibility that a threshold signature is designed from scratch without a centralized scheme, but this case is not of interest to us now.

As for classical signature scheme the main security property for threshold signature schemes is *Existential Unforgeability under Chosen Message Attacks (EUF-CMA)*. We present it below the version of threshold signature schemes.

**Definition 4.2.** A threshold digital signature $\mathsf{TDS.DS}$ is secure in the EUF-CMA if for any probabilistic polynomial-time adversary $\mathsf{Evl}$ that is allowed to:

1. Query a key generation oracle that runs $\mathsf{DS.Setup}$ and $\mathsf{TDS.KeyGen}$ for the public parameters $\mathsf{pp}$ and the public key $\mathsf{pk}$ (but not the private key);
2. Corrupt $T-1$ out of $N$ users and know their private keys $\mathsf{sk}_i$;
3. In the case of a decentralized key generation he get also access to the view of the protocol $\mathsf{TDS.KeyGen}$;
4. Perform a polynomial number of query to a signing oracle that on chosen messages $\mathsf{m}_i$ obtaining the view of $\mathsf{TDS.Sign}$ (also on the non-corrupted users);

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

$$\mathbb{P}\left[\mathsf{DS.Verify}(\mathsf{pk},\mathsf{m}^*,\mathsf{sig}^*)=1 \;\middle|\; \begin{array}{l}\mathsf{m}^*,\mathsf{sig}^* \leftarrow \mathsf{Evl}\ , \\ \mathsf{m}^* \neq \mathsf{m}_i\ \forall i\ .\end{array}\right] \leqslant \mathsf{negl}(\lambda) \tag{4.1}$$

Informally the idea is that $T-1$ views cannot be combined to obtain a valid signature. A stronger security requirement from [GJKR96] would be that the views actually does not contain any information (similarly how we ask for $\Sigma$-protocols with the zero-knowledge property). This can be formalized as follows:

**Definition 4.3** (Definition 3 [GJKR96])**.** A threshold signature scheme $\mathsf{TDS.DS}$ is *simulatable* if:

– When decentralized the protocol $\mathsf{TDS.KeyGen}$ is simulatable. That is, there exists a simulator $\mathsf{Sim}_{\mathsf{KeyGen}}$ for any adversarial set of users that, on input the public key $\mathsf{pk}$ and the outputs generated by an execution of the key generation from the adversary, can simulate the view of the adversary on that execution.

– The protocol $\mathsf{TDS.Sign}$ is simulatable. That is, there exists a simulator $\mathsf{Sim}_{\mathsf{Sign}}$ that, on input the public input of $\mathsf{Sign}$, $T-1$ private keys $\mathsf{sk}_{i_1},\ldots,\mathsf{sk}_{i_{T-1}}$, and a signature $\mathsf{sig}$ of $\mathsf{m}$, can simulate the view of the adversary on an execution of $\mathsf{Sign}$ that generates $\mathsf{sig}$ as an output.

When the centralized signature is unforgeable and the threshold version is simulatable, then also the threshold signature is unforgeable [GJKR96].

## 4.1 The Full Threshold Scheme

We start our analysis in the simpler case, the *full threshold* one, in which $T=N$, i.e. all the users are required to participate to produce a signature.

Given a cryptographic group action $(G,X,\star)$ with origin element $x\in X$ we can consider a secret key given by the product of $N$ random group elements:

$$g = g_N \cdots g_1. \tag{4.2}$$

Thus the public key is $y = g \star x$ and to each user $P_i$ is given the share $g_i$. In Section 4.1.3 we show how to obtain a decentralized key generation for it. Also the users receive the intermediate set elements $x_j = g_j \star x_{j-1}$ used to evaluate the public key $y = x_N$.

First we can consider in Protocol 4.1.1 a natural threshold computation for the classical Identification Protocol 2.2.1.

Public Data Parameters : Group $G$ acting on $X$ via $\star$, element $x \in X$ and hash function $\mathsf{H}$.
Private Key : Group element $g = g_N \cdots g_1$ with $g_i \in G$.
Shares for $P_i$ : Group element $g_i$, set elements $x_j = g_j \cdots g_1 \star x$ for all $j$.
Public Key : $y = g \star x$.

| **PROVERS** | | **VERIFIER** |
|---|---|---|
| Set $\tilde{x}_0 \leftarrow x$ and for $i = 1, ..., N$ do : | | |
| $P_i$ get $\tilde{g}_i \xleftarrow{\$} G$ and set $\tilde{x}_i \leftarrow \tilde{g}_i \star \tilde{x}_{i-1}$ | $\xrightarrow{\mathsf{com}}$ | |
| Set $\mathsf{com} = \mathsf{H}(\tilde{x}_N)$. | | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0, 1\}$. |
| Set $\mathsf{resp} \leftarrow e$. | | |
| for $i = 1, ..., N$ do : | $\xrightarrow{\mathsf{resp}}$ | If $\mathsf{ch} = 0$ accept if $\mathsf{H}(\mathsf{resp} \star x) = \mathsf{com}$. |
| $P_i$ set $\mathsf{resp} \leftarrow \tilde{g}_i \cdot \mathsf{resp} \cdot g_i^{-\mathsf{ch}}$. | | If $\mathsf{ch} = 1$ accept if $\mathsf{H}(\mathsf{resp} \star y) = \mathsf{com}$. |

Protocol 4.1.1: Full threshold identification protocol for the knowledge of the Private Key.

An relevant limitation of the proposed protocol is that each user $P_i$ need to receive the set element $\tilde{x}_{i-1}$ by $P_{i-1}$ before starting its computations. Thus as explained in [DM20] it is necessary to adopt a *sequential round-robin* communication structure that makes impossible to parallelize the algorithm and combine the output of each party later, slowing down the execution. Moreover the users need to agree upon a precise execution order at the beginning of the execution.

*Remark* 39. When the action is induced by an abelian group (like isogenies over supersingular elliptic curves) the response phase can be compressed further since each party can simply disclose $\tilde{g}_i^{\mathsf{ch}} \cdot g_i$ to combine them later as the following product (or actually the sum as in [DM20; CS20])

$$\prod_{i=1}^{N} g_i{}^{-\mathsf{ch}} \cdot \tilde{g}_i = g^{-\mathsf{ch}} \prod_{i=1}^{N} \tilde{g}_i \ .$$

The most natural idea would be to apply the Fiat-Shamir transform on it, by evaluating the challenge as:

$$\mathsf{ch} = \mathsf{H}(x_N^1 \| ... \| x_N^\lambda \| \mathsf{m}) \ ,$$

to obtain a secure threshold signature, which can be read in Algorithm 6.

However, the above construction is secure only in the *honest-but-curious* adversary model, for example in [DM20] they prove that this protocol is simulatable under the assumption of the hardness of Problem 7.

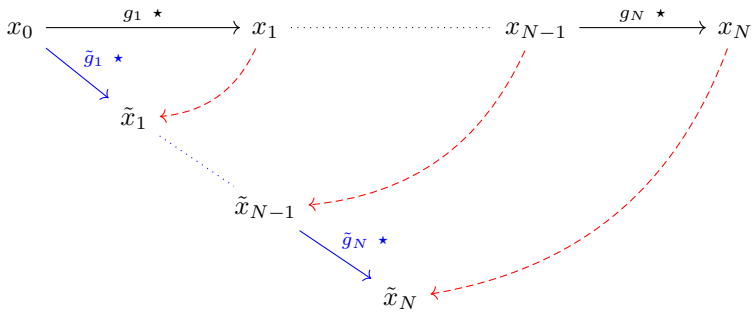



Figure 4.1: Scheme representing the idea behind Protocol 4.1.1. In blue there are the ephemeral group elements revealed on $\mathsf{ch} = 0$, while in red the map reconstructed for $\mathsf{ch} = 1$.

**Algorithm 6** TDS.Sign, no active security

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

Set $x_0^j = x$ for all $j = 1$ to $\lambda$ ▷ 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 $\lambda$
  **for** $j = 1$ to $\lambda$ **do**
    $P_i$ chooses $\tilde{g}_i^j \in G$ and computes $x_i^j = \tilde{g}_i^j \star x_{i-1}^j$
    $P_i$ outputs $x_i^j$;
Set $x^j = x_N^j$ for all $j = 1$ to $\lambda$
Compute $\mathsf{ch} = \mathsf{H}(x^1 \| ... \| x^\lambda || \mathsf{m})$ ▷ Non-iterative challenges evaluation
Set $u_0^j = e$ for all $j = 1$ to $\lambda$ ▷ 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 $j = 1, ..., \lambda$
  **for** $j = 1$ to $\lambda$ **do**
    $P_i$ computes $u_i^j = \tilde{g}_i^j u_{i-1}^j g_i^{-\mathsf{ch}_j}$;
    $P_i$ outputs $u_i^j$;
$\mathsf{resp}_j = u_N^j$ for all $j = 1$ to $\lambda$;
$\mathtt{sig} = \mathtt{ch} || \mathtt{resp}_1 || ... || \mathtt{resp}_\lambda$

In fact the scheme can be attacked by a *malicious* adversary tempering with it by opening several concurrent sessions. Suppose the adversary is in control of the $N$-th user and want to sing the message $\mathsf{m}$ for the public key $y = g \star x$ knowing only $g_N$. He can proceed in the following way:

1. The adversary start $\lambda$ signing sessions for any messages $\mathsf{m}_1$,...,$\mathsf{m}_\lambda$.
2. For any 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. Lets call this element $\hat{x}^s$ for each session.
3. Evaluate the challenge $\mathsf{ch} = \mathsf{H}(\hat{x}^1 \| ... \| \hat{x}^\lambda \| \mathsf{m})$.
4. For each session $s$ now he evaluates $x^2_N, ..., x^{\lambda-1}_N$ legitimately, then he 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 $\mathsf{ch}$. This point can be simply adapted to more general challenge spaces and be evaluated in probabilistic linear time on the challenge space.
5. He then closes all the concurrent sessions obtaining for the session $s$ the response $u^1_{N-1}$ received from $P_{N-1}$ can be used to evaluate $\mathsf{resp}_1$. Lets call this $\hat{\mathsf{resp}}_s$; we can use it to answer $\mathsf{ch}_s$. In this way we have obtained a valid signature $\mathsf{ch} \| \hat{\mathsf{resp}}_1 \| ... \| \hat{\mathsf{resp}}_\lambda$.

By this example we see that the key point for the adversary was the possibility of imposing a condition on the challenges, via the freedom of choice in the evaluation of the commitment, used to evaluate the $\mathsf{H}$. To have a secure scheme we should avoid this possibility.

### 4.1.1 Sashimi Solution

The first solution proposed in [CS20] consists in the use of additional non-interactive Zero-Knowledge proofs. These ZKPs need to prove the existence of a witness $h$ for the following relation

$$\bigwedge_{j=0}^{q} y_j = h \star x_j \ . \tag{4.3}$$

The protocol presented below is a generalization of the one presented in Section 3.1 of [CS20], for a generic group action.

**Proposition 4.4** ([CS20], [BDPV21])**.** *The protocol in Protocol 4.1.2 can be rendered to a non interactive computationally zero-knowledge quantum proof of knowledge for a free weakly pseudorandom group action (Definition 2.6) in the QROM.*

*Proof.* First we prove that the underlying protocol is sound and computationally zero-knowledge, since the completeness is straightforward. We need to prove soundness and zero knowledge.

| | | |
|---|---|---|
| Public Data : $x_a, x_b \in X$ and hash function $\mathsf{H}$. | | |
| Private Key : Group element $g \in G$. | | |
| Public Key : $y_j = h \star x_j$ for $j = 0, ..., q$. | | |
| **PROVER** | | **VERIFIER** |
| Choose $\tilde{g} \xleftarrow{\$} G$ and set: | | |
| $\tilde{x}_j = \tilde{g} \star x_j$ for $j = 0, ..., q$. | $\xrightarrow{\mathsf{com}}$ | |
| Set $\mathsf{com} = \mathsf{H}(\tilde{x}_0 \| ... \| \tilde{x}_q))$. | | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0, 1\}$. |
| If $\mathsf{ch} = 0$ then $\mathsf{resp} = \tilde{g}$. | $\xrightarrow{\mathsf{resp}}$ | Accept if $\mathsf{H}(\mathsf{resp} \star x_0 \| ... \| \mathsf{resp} \star x_q) = \mathsf{com}$. |
| If $\mathsf{ch} = 1$ then $\mathsf{resp} = \tilde{g}h^{-1}$. | | Accept if $\mathsf{H}(\mathsf{resp} \star y_0 \| ... \| \mathsf{resp} \star y_q) = \mathsf{com}$. |

Protocol 4.1.2: Identification protocol to prove relation 4.3.

- **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 $h$ as $u_1^{-1}u_0$ against the one-wayness of the group action (thus also against weak pseudorandom) 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 $(\bar{x}, h \star \bar{a})$ from random ones, against the pseudorandomness.

For the quantum resistance we can observe that since the automorphisms are all trivial the sigma protocol has perfect unique responses (see [Blä+22, Lemma 1], the perfect uniqueness response version of Lemma 2.11) then by [DFMS19, Theorem 25] (perfect uniqueness response version of Theorem 1.15) the protocol is a quantum proof of knowledge. Then the protocol has completeness, high min entropy[1] and HVZK and is zero-knowledge against quantum adversaries thanks to [Unr17]. □

In our case we have $q = 1$ and the relation we want to prove for the commitment phase is:

$$y_i = \tilde{g}_i \star x \wedge x_i = \tilde{g}_i \star x_{i-1} \ ;$$

where $y_i$ are committed at the start of the protocol with a ZKP proving the relation $y_i = \tilde{g}_i \star x$. Then the each user applies the random group element $\tilde{g}_i$ on

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

the received set element and produces a non-interactive zero-knowledge proof for the above relation, which is verified from the other parties. In this way it is not possible for any adversary to deviate from the protocol and the signature can be simulated also for a *malicious* adversary. The similarity between Protocol 4.1.2 and Protocol 2.2.1 is straightforward, thus producing non-interactive zero-knowledge proof is equivalent to a classical signature, with the only difference that the number of group actions is multiplied by $q + 1$.

Since this schemes are designed for a post-quantum scenario we have also verified that the non-interactive ZKP is secure in the quantum random oracle model. For this reason we require the additional assumption that the action is free. Note that it would also be enough to ask that the group action is free to use Corollary 2.13.

Even if this scheme achieves the desired level of the security its main drawback is the number of ZKP necessary for a secure execution, $\lambda$ for each user for every single signature. In isogeny-based group actions this implies, in combination with the sequential communication model, an important slow down due to difficulties in computing group actions. In [CS20] they estimate that when implemented for CSI-FiSh ([BKV19]) each party has a latency of 238 $\mathsf{s}$, almost 5 minutes! Also for code-based group actions we have this problem, adjoined by the fact that ZKPs would require heavy bandwidth use. Using LESS-1b parameters, assuming a signature and verification time of around 50 $\mathsf{ms}$ from [Bal+23b, Table 7], each non-interactive ZKP would require around 100 $\mathsf{ms}$ (since the number of group actions is doubled in Protocol 4.1.2). So the latency due to the generation and verification of the 128 non-interactive ZKPs is around 25 $\mathsf{s}$.

### 4.1.2 A Salty Solution

As said before, when using a seed tree optimization for the centralized signature, an hash salting technique is necessary to preserve the security level. In this section we show how to use a similar idea to avoid excessive use of ZKP to control the adversary's behaviour

The modifications to the protocol works as follow:

- Each party $P_i$ chooses a random $\mathsf{salt}_i \in \{0,1\}^{2\lambda}$ and commits themselves to it.
- The protocol continues normally until the last step before the computation of $\mathtt{ch}$.
- $P_N$ broadcast $x^1, ..., x^\lambda$ to all players. Then each party reveals $\mathsf{salt}_i$. Each party checks that each revealed value matches the corresponding commitment.
- All the party can compute $\mathtt{ch} = H(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m})$ where $\mathsf{salt}$ is defined as $\mathsf{salt} = \sum_i (\mathsf{salt}_i)$.[2]

[2] Other combining functions for the computation of $\mathsf{salt}$ are possible, as long as the adversary

– During the response computations the parties verify the received partial responses.

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 $\tilde{x}_i^j = u_i^j \star x_i$.

**Algorithm 7** TDS.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 $g = g_N \cdots g_1$ and all the intermidiate set elements $x_j = g_j \cdots g_1 \star x$.

**Ensure:** A valid signature for the message $\mathsf{m}$ under the public key $(x, y)$.

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 $\mathsf{salt}_i$ randomly 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: 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: $\mathsf{sig} = \mathsf{ch} \| \mathsf{salt} \| \mathsf{resp}_1 \| ... \| \mathsf{resp}_\lambda$.

Thus the final version of the threshold signature can be read in Algorithm 7. The verification procedure is the same to the standard one and is reported in Algorithm 8 for completeness.

Note that, to verify the responses, the parties not only need their secret share, but also the intermediate set elements $x_i$ and $\tilde{x}_i^j$. In fact, during the

is not able to bias the distribution of it without knowing all the $\mathsf{salt}_i$, including the one of the honest party

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 $\tilde{x}_i^j = u_i^j \star x_i$.

**Algorithm 8** Verify

**Require:** $x \in X$, the security parameter $\lambda$, a hash function $\mathsf{H}$, a 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 $\mathsf{ch}, \mathsf{salt}, \mathsf{resp}_1, ..., \mathsf{resp}_\lambda$ from $\sigma$
2: **for** $j = 1$ to $\lambda$ **do**
3: **if** $\mathsf{ch}_j = 0$ **then**
4: set $\hat{x}^j = \mathsf{resp}_j \star x$
5: **else**
6: set $\hat{x}^j = \mathsf{resp}_j \star y$
7: Accept if $\mathsf{ch} = \mathsf{H}(\hat{x}^1 \| ... \| \hat{x}^\lambda \| \mathsf{salt} \| \mathsf{m})$

We need now to prove its security.

**Theorem 4.5.** *For a free cryptographic group action, if the centralized signature is unforgeable in the quantum random oracle model, then the full-threshold signature scheme (Algorithm 7) is EUF-CMA secure in the quantum random oracle model, as required in Definition 4.2.*

The proof of Theorem 4.5 follows the game-based argument proposed in [GHHM21, 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 9), and we need to reprogram the random oracle, thanks to Theorem 1.16. We discuss later how to insert also a Distributed Key Generation algorithm in the proof.

*Proof Theorem 4.5.* Consider a probabilistic polynomial-time adversary $\mathsf{Evl}$ that make up to $q_s$ sign queries and $q_h$ quantum call to the random oracle $\mathsf{H}$. Consider the games from Algorithm 9. Since the protocol $\mathsf{TDS.Sign}$ is executed in multiparty, if by any reason the protocol is aborted because of $\mathsf{Evl}$ misbehaviour, the game ends and returns 0.

*Game* $G_0$. This game is the same one played for the EUF-CMA security in Definition 4.2, 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 $\mathsf{ch}$ 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 as the one in Theorem 1.16; in particular we can adapt the distinguisher from the proof of [GHHM21, Theorem 3]. In total, we reprogram the

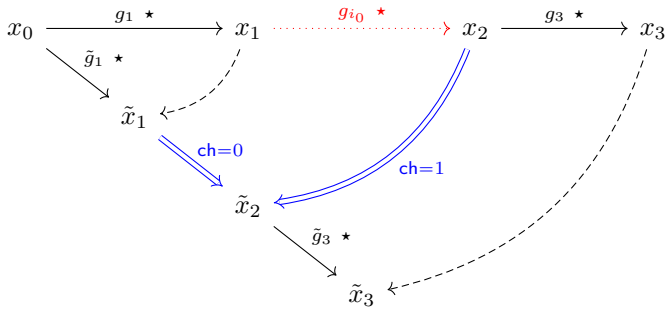



Figure 4.2: Example of simulation for $N = 3$ and $i_0 = 2$, in red the missing link, while in blue the elements used to generate $x_{i_0}$ and to answer the challenge.

oracle $q_s$ times (one for every signature) and $\mathsf{Evl}$ performs $q_h$ quantum calls. Moreover, note that $x^1, ..., x^\lambda, \mathsf{m}$ are (at least partially) controlled by the adversary, while $\mathsf{salt}$ is randomly sampled thanks to the initial commitments and the secure aggregation, so the random set is $X_1 = \{0,1\}^{2\lambda}$. Thus, by Theorem 1.16 we have:

$$|\mathbb{P}[G_0^{\mathsf{Evl}} \to 1] - \mathbb{P}[G_1^{\mathsf{Evl}} \to 1]| \leqslant \frac{3g_s}{2^{1+\lambda}}\sqrt{q_h} \tag{4.4}$$

*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 $\mathsf{ch}_j$ (Figure 4.2 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 is not rejected during the response computation. When, instead, the adversary sends wrong data before $P_{i_0}$, the simulator follows the protocol normally. Even if it would not be able to answer to the challenge, that step is never reached, since the error allows for an early abort. Thus the simulation is perfect.

We have shown that $G_2$ simulates the multiparty signature protocol $\mathsf{TDS.Sign}$, 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 $\mathsf{salt}_{i_0}$, $x^j_{i_0}$, $\tilde{g}^j_{i_0}$ for all $j = 1, ..., \lambda$. The salt and the group elements are uniformly distributed 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 TDS.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:

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

□

*Remark* 40. Note that the rewinding procedure (lines 16-21, Algorithm 9) is not necessary when the action is regular, in fact in this case surely $\tilde{x}^j_{i_0-1} \in \mathcal{O}(x_{i_0-1}) = \mathcal{O}(x)$, thus they are again indistinguishable. Still the checks in line 19 for Algorithm 7 can not be omitted, since otherwise the simulation would lead to a valid signature even for invalid responses sent by the corrupted users $P_{i_0-1}$.

### 4.1.3 Distributed Key Generation

We now proceed in defining a secure distributed key generation mechanism, introduced in [CS20]. The idea is fairly simple: the users sequentially apply a previously committed random group element to the origin $x$ and add the non-interactive Zero-Knowledge proof in Protocol 4.1.2 to show the freshness of the group element. You can see the protocol in Algorithm 10. 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 distributed key generation *de facto* is a multiparty computation protocol used to jointly generate from an origin $x \in X$ a set element in its orbit that cannot be biased by any strict subset of the users. This construction strongly resembles in fact the distributed trusted-setup protocol for the generation of a supersingular elliptic curve with unknown endomorphism ring defined in [Bas+23].

**Algorithm 9** Threshold Signature Simulation

**procedure** GAMES $G_0 - G_1 - G_2$
  Evl chose at least a non corrupted user $P_{i_0}$;
  Execute KeyGen with Evl;
  $\mathsf{m}^*, \sigma^* \leftarrow \mathsf{Evl}^{\mathsf{Sign},|\mathsf{H}\rangle}$;
  **return** $\mathsf{DS.Verify}((x,y), \sigma^*, \mathsf{m}^*) \wedge \mathsf{m}^* \notin S_M$.

**procedure** Sign(m)
  $S_M \leftarrow S_M \cup \{\mathsf{m}\}$;
  Run $\mathsf{TDS.Sign}(\mathsf{m}, g_{i_0})$ up to line 11; ▷ $G_0 - G_1$
  $\mathsf{ch} \leftarrow \mathsf{H}(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m})$; ▷ $G_0$
  Get $\mathsf{ch} \xleftarrow{\$} \{0,1\}^\lambda$; ▷ $G_1$
  $\mathsf{H} \leftarrow \mathsf{H}^{(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m}) \mapsto \mathsf{ch}}$; ▷ $G_1$
  Run $\mathsf{TDS.Sign}(\mathsf{m}, g_{i_0})$ to the end; ▷ $G_0 - G_1$
  Run $\mathsf{Sim.TDS.Sign}(\mathsf{m})$; ▷ $G_2$
  **return** $\mathsf{salt}_{i_0}, \tilde{x}^j_{i_0}, u^j_{i_0}$ for all $j$.

**procedure** SIM.TDS.Sign($\mathsf{m}, g_i$ for $i \neq i_0$)
  Run $\mathsf{TDS.Sign}(\mathsf{m}, g_{i_0})$ until line 15.
  Check all the $u^j_{i_0-1}$ received.
  **if** At least one $u^j_{i_0-1}$ is not correct **then**:
    **return** 0 ▷ Abortion in TDS.Sign
  **else**
    Rewind Evl to line 4 after having received $x^j_{i_0-1}$
    **for** $j = 1, ..., \lambda$ **do**
      Get $\tilde{g}^j_{i_0} \leftarrow G$;
      Set $\tilde{g}^j_{i_0} \leftarrow \tilde{g}^j_{i_0} \cdot (g_N \cdots g_{i_0+1})^{-\mathsf{ch}_j}$;
      **if** $\mathsf{ch}_j = 0$ **then**
        output $x^j_{i_0} = \tilde{g}^j_{i_0} \star x^j_{i_0-1}$;
      **else**
        output $x^j_{i_0} = \tilde{g}^j_{i_0} \star y$;
    **After** receiving $x^j_N$, open $\mathsf{salt}_{i_0}$;
    **if** $\mathsf{salt}_i$ are correct **then**
      compute $\mathsf{salt} = \sum_i \mathsf{salt}_i$;
    **else return** 0 ▷ Abortion in TDS.Sign
    $\mathsf{H} \leftarrow \mathsf{H}^{(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m}) \mapsto \mathsf{ch}}$;
    Output $u^j_{i_0} = \tilde{g}^j_{i_0}$ for all $j$;

**Algorithm 10** TDS.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 $x'_i = g_i \star x$.
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 Protocol 4.1.2 proving $x'_i = \tilde{g}_i \star x \wedge x_i = \tilde{g}_i \star x_{i-1}$.
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 $g_i$.

Thanks to the use of Zero-Knowledge proofs we have the following security result

**Lemma 4.6.** *For a weakly pseudorandom free group action (Definition 2.6), the protocol* TDS.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 (contained in [CS20]) is to the ZKPs to recover their secret shares and simulate a view of the protocol. Differently by [CS20] here we only have one ZKP for any user, thus we rely in rewinding the tape to change the sent set element sent in state 6. Moreover because of this we need an additional assumption since GAIP alone is not enough. This proof works in the quantum random oracle model since the protocol in Protocol 4.1.2 is a non-interactive zero-knowledge quantum proof of knowledge in the quantum random oracle for a free group action (Proposition 4.4).

As a consequence we can combine the result with Theorem 4.5 to obtain:

**Corollary 4.7.** *For a weakly pseudorandom free group action, if the centralized signature is unforgeable in the quantum random oracle model, then the full-threshold signature scheme composed by* TDS.DS.KeyGen, TDS.Sign *(Algorithms 10 and 7) and the verification* DS.Verify *is EUF-CMA secure in the quantum random oracle model.*

*Proof of Lemma 4.6.* Algorithm 11 shows the simulation strategy for a probabilistic polynomial-time adversary 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 (Lemma 1.12) 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 $x_{i_0} = (g_{i_0+1}^{-1} ... g_N^{-1}) \star y$, 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 weakly pseudorandom property (Definition 2.6). 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 Protocol 4.1.2 is a non-interactive zero-knowledge quantum proof of knowledge (see Proposition 4.4). □

**Algorithm 11** Simulation of TDS.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 $x_{i_0} = (g_{i_0+1}^{-1} ... g_N^{-1}) \star y$;
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.

*Remarks* 41 (On the utility of the ZKP). The ZKP inserted in the KeyGen algorithm is not only necessary to prove the security using the game based style, but also without it opens the possibilities for attacks to the Key Generation of the scheme. In fact without it a malicious user could send a malformed element $\hat{x} \in \mathcal{X}$ instead of the correct set element that can possibly be used to extract information on the secret shares of the honest parties. For example in the code equivalence setting he could send a code with a codeword of weight 1 that can be used to get the evaluation of monomial maps on it.

Also the requirement to previously commit to a secret group element is necessary, otherwise a malicious user could chose a group element in a way that the output of the group action has some particular characteristics (a naive example could be that he forces the last bit of the set element to be 0).

Instead as shown by the proof is not necessary to incorporate an additional ZKP for the committed values $x'_i$, as done in [CS20], since by rewinding we can extract the secret share directly from the ZKP in state 5 of Algorithm 10.

## 4.2 Construction for Non-Abelian Group Actions

In this section, we explain how to modify the construction for the full threshold scheme, to obtain a $T$-out-of-$N$ scheme. The main problem is that for a non-abelian group action we have no hope of using linear secret sharing, but instead we should rely on replicated secret sharing. Curiously this approach was firstly proposed in [CS20], that takes into consideration group actions in the cyclic setting. We start by showing the elementary way to leverage the full threshold signature to obtain a threshold one. Then we show how this technique can be seen as a particular case of replicated secret sharing, so that we can obtain a distributed signature for any access structure.

**Subset solution** 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 $g_{I_M} \cdots g_{I_1}$, 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 4.8.** *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 prove that it is possible to recover $g_I$ for any $I$ of cardinality $N - T + 1$. By the inclusion-exclusion principle, $|J \cap I| = |J| + |I| - |I \cup J| \geqslant T + N - T + 1 - N = 1$, 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 $A^C = \{1, ..., N\} \backslash A$, 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. Observe that as long as the parties agree on a particular order for the shares the non-commutativity of the group action is not a problem. 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 4.1.

In particular, the structure of the protocol is 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 $\tau$ that, on input the set of participants $J$ and the current round, allows to consistently choose which user carries on the operations during the current turn. A possible example of $\tau$ 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 $\min J \cap I_i \in I_i$).

We can see in Protocol 4.2.1 an example on how to carry over the identification protocol at the basis of the signature. This protocol is then rendered to a

Public Data : Group $G$ acting on $X$ via $\star$, element $x \in X$ and hash function $\mathsf{H}$.
Private Key : Group element $g = g_{I_M} \cdots g_{I_1}$ 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 $\tilde{x} \leftarrow \tilde{g}_i \star \tilde{x}$ | $\xrightarrow{\mathsf{resp}}$ | |
| Set $\mathsf{com} = \mathsf{H}(\tilde{x})$. | | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0, 1\}$. |
| Set $\mathsf{resp} \leftarrow e$. | | |
| for $i = 1, ..., M$ do : | $\xrightarrow{\mathsf{resp}}$ | If $\mathsf{ch} = 0$ accept if $\mathrm{H}(\mathsf{resp} \star x) = \mathsf{com}$. |
| $P_{\tau(J,i)}$ set $\mathsf{resp} \leftarrow \tilde{g}_i \cdot \mathsf{resp} \cdot g_{I_i}^{-\mathsf{ch}}$. | | If $\mathsf{ch} = 1$ accept if $\mathrm{H}(\mathsf{resp} \star y) = \mathsf{com}$. |

Protocol 4.2.1: Threshold identification protocol for the shared knowledge of the Private Key using the subset technique, executed by a set $J$ of at leats $T$ honest users

threshold signature via one of the techniques explained in Section 4.1.1 or 4.1.2. In Algorithm 12 you can see the final protocol augmented with a secure salt.

**Replicated Secret Sharing** To understand how this technique based on subsets can be seen as a particular case of Replicated Secret sharing we firstly start by defining a monotone access structure:

**Definition 4.9.** An *access structure* $\mathcal{A}$ for the parties $\mathcal{P} := \{P_1, .., P_N\}$ is the family of subsets $S \subset \mathcal{P}$ that are authorized (to sign a message). An access structure is said *monotone* if given any $S \in \mathcal{A}$ and $S' \supset S$ then $S' \in \mathcal{A}$.

For each access structure we can associate a family of unqualified sets $\mathcal{U}$ satisfying that, for all $S \in \mathcal{A}$, $U \in \mathcal{U}$, then $S \cap U = \varnothing$. For all the section we 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.

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

Then for each $I_i \in \mathcal{I}$ we define a share $s_I$ so that:

$$s = \prod_{I \in \mathcal{I}} s_I \ ;$$

then each party $P_i$ is given access to $s_I$ if and only if $I \ni i$.

Observe now that we have for any monotone access structure:

**Proposition 4.10.** *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.*

**Algorithm 12** $\mathsf{TDS.Sign}_{T,N}$

**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 $g = g_{I_M} \cdots g_{I_1}$ so that $I_j \ni i$.

**Ensure:** A valid signature for the message $\mathsf{m}$ under the public key $(x, y)$.

**for** $t \in J$ **do**
  $P_t$ pick $\mathsf{salt}_t$ randomly and sends $\mathsf{COM}(\mathsf{salt}_t)$
Set $x_0^j = x$ for all $j = 1$ to $\lambda$ ▷ Shared commitment generation phase
**for** $i = 1$ to $M$ **do**
  If $i > 1$ $P_{\tau(J,i)}$ receives $x_{i-1}^j$ from $P_{\tau(J,i-1)}$ for all $j = 1$ to $\lambda$
  **for** $j = 1$ to $\lambda$ **do**
    $P_{\tau(J,i)}$ chooses $\tilde{g}_i^j \in G$ and computes $x_i^j = \tilde{g}_i^j \star x_{i-1}^j$
    $P_i$ outputs $x_i^j$
Set $x^j = x_N^j$ for all $j = 1$ to $\lambda$. Party $\tau(J, N)$ broadcast all $x^j$ to all players.
Each party publish $\mathsf{salt}_t$ and checks the consistency of the received data with the initial commitment.
$\mathsf{salt} = \sum_t \mathsf{salt}_t$
Compute $\mathsf{ch} = \mathsf{H}(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m})$ ▷ Non-interactive challenges evaluation
Set $u_0^j = e$ for all $j = 1$ to $\lambda$ ▷ Shared response generation phase
**for** $i = 1$ to $M$ **do**
  If $i > 1$ $P_{\tau(J,i)}$ receives $u_{i-1}^j$ from $P_{\tau(J,i-1)}$ for all $j = 1, ..., \lambda$
  **for** $j = 1$ to $\lambda$ **do**
    $P_{\tau(J,i)}$ computes $u_i^j = \tilde{g}_i^j u_{i-1}^j g_i^{-\mathsf{ch}_j}$
    $P_{\tau(J,i)}$ outputs $u_i^j$
    All users verify $u_i^j$ is valid;
$\mathsf{resp}_j = u_N^j$ for all $j = 1$ to $\lambda$
$\mathsf{sig} = \mathsf{ch} \| \mathsf{salt} \| \mathsf{resp}_1 \| ... \| \mathsf{resp}_\lambda$

*Proof.* We prove that they can 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 mean that $I \notin P_i$ 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 then we have $I^c \in \mathcal{A}$, but $I^c$ lie also in $\mathcal{U}^+$ because of $\mathcal{I}$ definition, so $I^c \in \mathcal{A} \cap \mathcal{U}$, that is absurd for Definition 4.9.

Also 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 $B^c \cap A = \varnothing$. Also 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. □

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

When we are working in abelian group (we use additive notation for simplicity) this implies that for any authorized set $J$ the secret $s$ can be seen as a sum of $|J|$ shares:

$$s = \sum_{P_i \in J} \left[ \sum_{\psi_J(I) = P_i} s_I \right] . \tag{4.5}$$

This is used in [CS20] to reduce the number of rounds in their threshold signature scheme, but it cannot be used for non-abelian groups.

It should be clear at this point how the subset solution is just a particular instance of replicated secret sharing in which the authorized sets are the one of cardinality at least $T$, in fact in this way $\mathcal{U}^+$ are the subsets of cardinality $T-1$ and $\mathcal{I}$ the ones of cardinality $N-T+1$. Then the turn function $\tau$ can be derived as $\tau(J, i) := \psi_J(I_i)$.

**Usability of Replicated Secret Sharing** The main drawback of Replicated Secret sharing is that the number of shares goes as the cardinality of $\mathcal{U}^+$, that usually is exponential in the number of parties. For example in the threshold case there are in total $\binom{N}{T-1}$ shares, also each party has the knowledge of $\binom{N}{T}$ shares (this can be proved with elementary combinatorics techniques).

Also if the group is non-abelian the number of rounds cannot be compressed and is equal to the total number of shares, thus exponential. In the threshold case it is equal to $\binom{N}{T}$, as said before.

So 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 $N$. As said the main issue here is the non-commutativity of the group, which precludes the usage of traditional techniques for secret sharing and the compression of the share.

We would like to point out however that for the most used combinations of $(T, N)$, like $(2, 3)$ or $(3, 5)$, the increase in the number of shares and rounds is so contained that it does not impact the efficiency of the protocol.

**Distributed Key Generation** The distributed key generation protocol in Algorithm 10 can be leveraged also to the threshold case using replicated secret sharing again, in the same way as for the signing algorithm. There is only some considerations to be made. 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 similiar to the key generation;
2. delegate one of the users that should know a share the due of executing the turn and then he shares it to the others.

The first option is, for example, implemented efficiently with the protocol $\Pi_{\mathsf{Rand}}$ in Section 2.4 of [CS20]. The efficient implementation can only be implemented in an abelian group since it is based on the same idea of Equation (4.5).

Instead the second option has a lower latency for the non-abelian case since we don't have intermediate rounds. Moreover:

– the key generation protocol is secure according to Lemma 4.6, so the party executing it cannot bias it;
– every authorized party can directly check that the shares corresponds to the one applied.

This said, the key generation is performed as before, where each party sends messages according to the function $\tau$. The signature algorithm is also performed in the same way as the full threshold scheme, using the function $\tau$ to determine which party sends which messages at each round.

Said this 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 4.11.** *For a 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* $\mathsf{TDS.KeyGen}_{T,N}$, $\mathsf{TDS.Sign}_{T,N}$ *and the verification* $\mathsf{DS.Verify}$ *is EUF-CMA secure in the quantum random oracle model.*

*Sketch.* The proof is very similar to that of the full threshold case (Theorem 4.5). 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 4.8. Thus we just use the strategies from Algorithm 11 and Algorithm 9 using as non corrupted user $P_{\tau(J,\mathsf{ho})}$. □

**Abelian Group Actions** Thanks to the reduced overhead computations it would be interesting to use the salt solution also for abelian group actions. However observe that for the security reduction a round robin structure is necessary also in the response for challenges $\mathsf{ch} \neq 0$, in fact in the simulation from Algorithm 9 the adversary in line 21 is rewinded after the receival of $u^j_{i_0-1}$. Thus we cannot use the observation from Remark 39.

Instead the classical recombination strategy from Equation (4.5) can be leveraged to cut the number of rounds from $M = \binom{N}{T-1}$ to $T$, even if each response $\sum_{\psi_J(I)=P_i} s_I$ needs to be verified independently. However for verifying the responses $u_i^j$ firstly they need to recompute the intermediate values via a multiparty computation protocol equal to the one in Algorithm 10.

### 4.2.1 A LESS Tailored Approach

The scheme proposed in Algorithm 12 becomes rapidly impractical for large values of $T, N$, since the number of round and shares necessaries is $\binom{N}{T-1}$. Solving this using only non-abelian group actions as black boxes is not an option, thus it is desirable to find some tailored solution that exploits the group action peculiarities.

For the Hamming metric Code Equivalence monomial maps are too structured for a threshold like sharing, but maybe other approaches are possible. In Proposition 3.13 we have shown how the change of basis matrix $\mathbf{S} \in \mathsf{GL}_k$ contains all the secret information and may eventually be used to recover efficiently the monomial map.

Since invertible matrices are simpler object to deal with, there may be a way to define a multiplicative threshold secret sharing for matrices, i.e. a scheme in which for any set $P = \{i_1, ..., i_T\}$ of $T$ parties each stakeholder $P_{i_j}$ can create a matrix $\hat{\mathbf{S}}_{i_j} \in GL_k(\mathbb{F}_q)$ such that a secret matrix $\mathbf{S}$ can be recovered as $\mathbf{S} = \hat{\mathbf{S}}_{i_1} \cdots \hat{\mathbf{S}}_{i_T}$.

With a scheme like that, if, during the commitment phase, each party memorizes not only the ephemeral monomial map $\mathbf{Q}_j$, but also the linear map $\tilde{\mathbf{S}}_{i_j}$, they can retrieve $\mathbf{S}\tilde{\mathbf{S}}$ (and so $\mathbf{Q}\tilde{\mathbf{Q}}$) by recursively combining $\mathsf{resp} \leftarrow \tilde{\mathbf{S}}_{i_j} \cdot \mathsf{resp} \cdot \hat{\mathbf{S}}_{i_j}$, as shown in Protocol 4.2.2.

There are surely several ways to obtain a threshold secret sharing for matrix multiplication, but, as we can seen from the protocol, we need additional properties:

1. the secret matrix $\mathbf{S}$ (or the shares) should not have a structure that leaks information on the monomial map, i.e. it should still be hard to find $\mathbf{Q}$ given $\mathbf{G}$ and $\mathbf{SGQ}$;

2. during the recombination phase it should be infeasible to use the publicly exchanged information to retrieve the share $\hat{\mathbf{S}}_{i_j}$ or the ephemeral map $\tilde{\mathbf{Q}}_{i_j}$.

Satisfying these properties is not easy and, at the moment, we do not know any practical and secure way to instantiate this sharing and this protocol. As pedagogical example we go through a way to instantiate the protocol, even if flawed.

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:

Public Data Parameters : Parameters $n, k, q$, linear $[n, k]$-code with generator matrix $\mathbf{G}$ and hash function $\mathsf{H}$.
Private Key : Invertible matrix $\mathbf{S} \in \mathsf{GL}_k$ and monomial matrix $\mathbf{Q}$.
Shares for $P_j$ : Share of $\mathbf{S}$ for on the set of parties, wlog, $J = [1, T]$.
Public Key : $\mathbf{G}' = \mathbf{SGQ}$.

| **PROVERS** | | **VERIFIER** |
|---|---|---|
| Set $\tilde{\mathbf{G}} \leftarrow \mathbf{G}'$ and for $i = 1, ..., T$ : | | |
| get $\tilde{\mathbf{S}}_i \xleftarrow{\$} \mathsf{GL}_k(\mathbb{F}_q)$ and $\tilde{\mathbf{Q}}_i \xleftarrow{\$} \mathsf{Mono}_n$ | $\xrightarrow{\mathsf{com}}$ | |
| set $\tilde{\mathbf{G}} \leftarrow \tilde{\mathbf{S}}_i \cdot \tilde{\mathbf{G}} \cdot \tilde{\mathbf{Q}}_i$; | | |
| Set $\mathsf{com} = \mathsf{H}(\mathrm{SF}(\tilde{\mathbf{G}}))$; | | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0, 1\}$. |
| If $\mathsf{ch} = 0$ then $\mathsf{resp} \leftarrow \tilde{\mathbf{Q}}$ | | |
| (retrieved opening all $\tilde{\mathbf{Q}}_i$) | | Accept if $\mathsf{H}(\mathrm{SF}(\mathbf{G}' \cdot \mathsf{resp})) = \mathsf{com}$. |
| If $\mathsf{ch} = 1$ then $\nu \leftarrow \mathbf{I}$; | $\xrightarrow{\mathsf{resp}}$ | |
| for $i = 1, ..., T$: | | |
| $\nu \leftarrow \tilde{\mathbf{S}}_i \cdot \nu \cdot \hat{\mathbf{S}}_i$; | | |
| Use $\nu$ and algo. 5 to get $\mathsf{resp} \leftarrow \mathbf{Q}\tilde{\mathbf{Q}}$ | | Accept if $\mathsf{H}(\mathrm{SF}(\mathbf{G} \cdot \mathsf{resp})) = \mathsf{com}$; |

Protocol 4.2.2: Threshold identification protocol using only $T$ shares.

$$U = \left\{ \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix} \middle| \ \boldsymbol{A} \in \mathbb{F}_q^{\frac{k}{2} \times \frac{k}{2}} \right\} . \tag{4.6}$$

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

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{I} & \boldsymbol{B} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} + \boldsymbol{B} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix} \tag{4.7}$$

We can use this property to define a threshold secret sharing scheme for matrix multiplication by simply using any Linear Secret Sharing scheme, like Shamir's one [ÇBCS22, Section 1.5] for a secret $\frac{k}{2} \times \frac{k}{2}$ matrix $\mathbf{S}$ (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. $\mathbf{S} = \delta_1 \mathbf{S}_1 + ... + \delta_T \mathbf{S}_T$. At this point we have that:

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

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. In fact we have:

$$\underbrace{\begin{bmatrix} \mathbf{G}'_1 \\ \mathbf{G}'_2 \end{bmatrix}}_{\mathbf{G}'} = \begin{bmatrix} \boldsymbol{I} & \mathbf{S} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix} \cdot \underbrace{\begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \end{bmatrix}}_{\mathbf{G}} \cdot \mathbf{Q} = \begin{bmatrix} (\mathbf{G}_1 + \mathbf{S}\mathbf{G}_2) \cdot \mathbf{Q} \\ \mathbf{G}_2 \cdot \mathbf{Q} \end{bmatrix} ; \tag{4.8}$$

hence using Algorithm 5 on the pair $\mathbf{G}_2, \mathbf{G}'_2 = \mathbf{G}_2 \cdot \mathbf{Q}$ we can recover, at least partially, the secret.

To solve this we we can consider matrices in the product group $U \times U'$, where $U'$ is the transpose of $U$:

$$U' = \left\{ \begin{bmatrix} \boldsymbol{I} & \mathbf{0} \\ \mathbf{A} & \boldsymbol{I} \end{bmatrix} \mid \boldsymbol{A} \in \mathbb{F}_q^{\frac{k}{2} \times \frac{k}{2}} \right\} . \tag{4.9}$$

This way the secret matrices have the form:

$$\underbrace{\begin{bmatrix} \boldsymbol{I} & \mathbf{S}_1 \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix}}_{\hat{\mathbf{S}}_1} \cdot \underbrace{\begin{bmatrix} \boldsymbol{I} & \mathbf{0} \\ \mathbf{S}_2 & \boldsymbol{I} \end{bmatrix}}_{\hat{\mathbf{S}}'_2} = \begin{bmatrix} \mathbf{S}_1\mathbf{S}_2 + \boldsymbol{I} & \mathbf{S}_1 \\ \mathbf{S}_2 & \boldsymbol{I} \end{bmatrix} . \tag{4.10}$$

On the assumption that a change of basis of the form (4.10) hides the monomial map, we may hope to get a secure scheme (Protocol 4.2.3) by

1. sharing differently the two matrices $\mathbf{S}_1, \mathbf{S}_2$ using a linear secret sharing LSS;
2. repeating the for loop two times during the commitment and the response phases.

Sadly Protocol 4.2.3 has a flaw, that allows to use the vulnerable structure from (4.8). In fact by looking at the first user an honest-but-curious adversary can store $\tilde{\mathbf{G}}^* = \tilde{\mathbf{S}}_{1,1}\mathbf{G}'\tilde{\mathbf{Q}}_{1,1}$ during the commitment and, on challenge $\mathsf{ch} = 1$, during the response phase stores $\mathbf{R} = \tilde{\mathbf{S}}_{1,1} \cdot \hat{\mathbf{S}}_{\mathrm{LSS}(1),1}$. This way he compute:

$$\begin{aligned} \mathbf{R}^{-1}\tilde{\mathbf{G}}^* &= \hat{\mathbf{S}}^{-1}_{\mathrm{LSS}(1),1} \cdot \tilde{\mathbf{S}}^{-1}_{1,1} \cdot \tilde{\mathbf{S}}_{1,1}\mathbf{G}'\tilde{\mathbf{Q}}_{1,1} = \hat{\mathbf{S}}^{-1}_{\mathrm{LSS}(1),1} \cdot \mathbf{G}'\tilde{\mathbf{Q}}_{1,1} = \\ &= \begin{bmatrix} \boldsymbol{I} & -\mathbf{S}_{\mathrm{LSS}(1),1} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix} \cdot \underbrace{\begin{bmatrix} \mathbf{G}'_1 \\ \mathbf{G}'_2 \end{bmatrix}}_{\mathbf{G}'} \cdot \tilde{\mathbf{Q}}_{1,1} = \begin{bmatrix} (\mathbf{G}'_1 - \mathbf{S}_{\mathrm{LSS}(1),1}\mathbf{G}'_2) \cdot \tilde{\mathbf{Q}}_{1,1} \\ \mathbf{G}'_2 \cdot \tilde{\mathbf{Q}}_{1,1} \end{bmatrix} . \end{aligned} \tag{4.11}$$

From $\mathbf{G}'_2 \cdot \tilde{\mathbf{Q}}_{1,1}$ and $\mathbf{G}'_2$ via Algorithm 5 $\tilde{\mathbf{Q}}_{1,1}$, so the share $\mathbf{S}_{\mathrm{LSS}(1),1}$, can be retrieved, breaking the protocol security.

**Possible Future Direction** Even if the double triangular matrix solution is not secure, the question arises about the existence of other possible ways and research direction to get a multiplicative matrix secret sharing. Note that results in this directions would be interesting also for MEDS, thanks to the discussion in Section 3.2.2 on Problem 28. Surely, a possible direction would be to implement classical MPC technique, but this requires describing and computing the group action as an explicit arithmetic circuit, that could in principle be feasible, but surely the practicality of the scheme would be drastically reduced.

With regards to other possible $q$-subgroups (i.e. subgroup of order a power of $q$, like $U \simeq \mathbb{F}_q^{\frac{k^2}{4}}$), as the triangular matrix subgroup $U$, we should note that,

| | | |
|---|---|---|
| Public Data Parameters : Parameters $n, k, q$, linear $[n,k]$-code with generator matrix $\mathbf{G}$ and hash function $\mathsf{H}$. | | |
| Private Key : Invertible matrix $\hat{\mathbf{S}}_1\hat{\mathbf{S}}'_2 \in U \times U'$ and monomial matrix $\mathbf{Q}$. | | |
| Shares for $P_j$ : Additive shares of $\mathbf{S}_1$ and $\mathbf{S}_2$ for LSS | | |
| Public Key : $\mathbf{G}' = \hat{\mathbf{S}}_1\hat{\mathbf{S}}'_2\mathbf{G}\mathbf{Q}$. | | |
| **PROVERS** | | **VERIFIER** |
| Set $\tilde{\mathbf{G}} \leftarrow \mathbf{G}'$ and for $i = T, ..., 1$, $r = 1, 2$ do : | | |
| get $\tilde{\mathbf{S}}_{i,r} \xleftarrow{\$} \mathsf{GL}_k$ and $\tilde{\mathbf{Q}}_{i,r} \xleftarrow{\$} \mathsf{Mono}_n$ | $\xrightarrow{\mathsf{com}}$ | |
| set $\tilde{\mathbf{G}} \leftarrow \tilde{\mathbf{S}}_{i,r}\tilde{\mathbf{G}}\tilde{\mathbf{Q}}_{i,r}$. | | |
| Set $\mathsf{com} \leftarrow \mathsf{H}(\mathrm{SF}(\tilde{\mathbf{G}}))$. | | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0,1\}$. |
| If $\mathsf{ch} = 0$ then $\mathsf{resp} \leftarrow \tilde{\mathbf{Q}}$ | | |
| (retrieved opening all $\tilde{\mathbf{Q}}_{i,r}$) | | Accept if $\mathsf{H}(\mathrm{SF}(\mathbf{G}' \cdot \mathsf{resp})) = \mathsf{com}$. |
| If $\mathsf{ch} = 1$ then $\nu \leftarrow \mathbf{I}$. | $\xrightarrow{\mathsf{resp}}$ | |
| for $i = T, ..., 1$, $r = 1, 2$ do : | | |
| $\nu \leftarrow \tilde{\mathbf{S}}_{i,r} \cdot \nu \cdot \hat{\mathbf{S}}_{\mathrm{LSS}(i),r}$. | | |
| Use $\nu$ and algo. 5 to get $\mathsf{resp} \leftarrow \mathbf{Q}\tilde{\mathbf{Q}}$ | | Accept if $\mathsf{H}(\mathrm{SF}(\mathbf{G} \cdot \mathsf{resp})) = \mathsf{com}$. |

Protocol 4.2.3: Identification protocol for the threshold version

since

$$\#\mathsf{GL}_k = \prod_{i=0}^{k-1}(q^k - q^i) = q^{\sum_{i=1}^{k-1} i}\prod_{i=0}^{k-1}(q^{k-i} - 1) = q^{\frac{k(k-1)}{2}}\prod_{i=0}^{k-1}(q^{k-i} - 1)\ ,$$

the Sylow $q$-groups of $\mathsf{GL}_k$ have cardinality $\frac{k(k-1)}{2}$.[3] Note that all Sylow $q$-groups are conjugate and one of them is the group of upper triangular matrices with the identity on the diagonal $\mathsf{T} \subset \mathsf{GL}_k$. Hence they are all conjugate with the triangular matrices, so, since Sylow $q$-subgroup are maximal, for any $q$-subgroup $V \subset \mathsf{GL}_k$ there exists $\mathbf{R} \in \mathsf{GL}_k$ such that $\mathbf{R}^{-1} \cdot V \cdot \mathbf{R} \subset \mathsf{T}$.

This means that, for any $\mathbf{S} \in V$, $\mathbf{R}^{-1} \cdot \mathbf{S} \cdot \mathbf{R} = \mathbf{S}^*$ is upper triangular. Hence from $\mathbf{G}$ and $\mathbf{G}' = \mathbf{S}\mathbf{G}\mathbf{Q}$ we can get $\mathbf{G}^* = \mathbf{R}^{-1}\mathbf{G}$ and

$$\mathbf{R}^{-1}\mathbf{G}' = \mathbf{R}^{-1}\mathbf{S}\mathbf{G}\mathbf{Q} = \mathbf{S}^*\mathbf{R}^{-1}\mathbf{G}\mathbf{Q} = \mathbf{S}^*\mathbf{G}^*\mathbf{Q}\ .$$

Thanks to the triangularity of $\mathbf{S}^*$ leakages like (4.8) may be exploited. This means that any secure sharing structure, should not rely exclusively on some particular $q$-subgroup.

Another possible direction to solve this would be to drop the use of Shamir Secret Sharing and instead rely on something like the Blakley's Secret Sharing scheme [Bla79], that recover the secret by intersecting $T$ hyperplanes, for example by applying the change of basis on some masked hyperplanes.

[3] Theory regarding Sylow groups and theorems can be read in [Lan12, Chapter 1.6]

## 4.3 Constructions for Cyclic Group Actions

In this section we see how the previous construction can be improved for the case in which the group $G$ is cyclic with order $q$ know and we have access to a generator $g$; as introduced in the cutting-edge paper [DM20]. We show a scheme that, at the moment, even if it uses also a salt, is only proven secure in the honest-but-curious model. The scheme can be adjoined by ZKPs to obtain an active security, as shown in [CM22; BDPV21], even if much more sophisticated than Protocol 4.1.2. Instead the proof technique used for Theorem 4.5 before cannot be adapted here since we do not have a simple way to access the intermediate elements $x_i$ used for the verification.

However, we rely here in a different proof technique based on simulatability [GJKR96], that follows the path of [DM20], without using an ad-hoc assumption.

Under our assumptions we may see the group action as an action defined on the integers modulo $q$:

$$\begin{aligned} [\cdot] : \mathbb{Z}/q\mathbb{Z} \times X &\to X \\ (a, E) &\to [a]x := g^a \star x \ . \end{aligned} \tag{4.12}$$

The only known instance of cryptographic group action (HHS) with this characteristic is the one used in CSI-FiSh ([BKV19]). For example in CSI-FiSh:

$$\begin{aligned} q = 3 \cdot 37 \cdot 1407181 \cdot 51593604295295867744293584889 \cdot \\ \cdot 3159941450468199585300827874558783220490 9 \simeq 2^{257.136} \ . \end{aligned} \tag{4.13}$$

The idea at this point is simple: given a secret $s$, so that $y = [s]x$, we share it through a linear secret sharing technique like Shamir Secret Sharing, in this way for any subset $J$ of $T$ users they are able to find $s_i$ for $i \in J$ so that $s = \sum_{i \in J} s_i$, thus:

$$[s_{i_T}] \cdots [s_{i_1}]x = [\sum_{i \in J} s_i]x = [s]x = y \ . \tag{4.14}$$

This scheme requires to adapt the linear secret sharing also for the case in which $q$ is not prime ([DM20]).

**Secret Sharing on Modular Integers** The classical $(T, N)$-Shamir Secret Sharing for finite field $\mathbb{Z}/q\mathbb{Z}$ with $q > N$ goes as following:

- the dealer generate a secret uniformly random polynomial $f \in \mathbb{Z}/q\mathbb{Z}[t]$ of degree $T - 1$;
- the secret is fixed to $s = f(0)$;
- the party $P_i$ get access to the share $s_i = f(i)$;
- when a set $J$ of $T$ parties decides to recover the secret they all share $s_j \delta_{J,j}(0)$, where $\delta_{J,j}(t)$ is the unique polynomial of degree $k - 1$ equal to

1 on $j$ and null for the other entries in $J$, also said Lagrange polynomial, that is evaluated as:

$$\delta_{J,j}(t) = \prod_{i \in J, i \neq j} \frac{i-t}{i-j} \mod q \ ; \tag{4.15}$$

– by summing the published values they obtain:

$$\sum_{j \in J} s_j \delta_{J,j}(0) \stackrel{*}{=} f(0) = s \ . \tag{4.16}$$

For $*$ observe that $f$ is the unique polynomial of degree $T-1$ passing through all the $T$ points $(j, s_j)$.

As said, the order of the group may not be prime. It is possible to overcome this difficulty maintaining the security of the sharing employing some classical techniques, used for example in [Sho00], as explained in [DM20].

The only critical point is the step 4.3, where to evaluate the Lagrange polynomial in (4.15) we need to invert the term $\prod_{i \in J, i \neq j} i - j$. By classical results from modular arithmetic, we can do that if and only if $i - j$ is not coprime to $q$ for all $i \in J$ different from $j$. Suppose now that $q_1$ is the smallest prime factor for $q$, if $N < q_1$ we have that

$$|i - j| \leqslant \max i, j \leqslant N < q_1 \ ;$$

since also $i - j \neq 0$ we have that $i - j$ and $q$ are coprimes. Under this additional assumption the scheme remains secure:

**Proposition 4.12** (Proposition 1 in [DM20])**.** *The classical $(T, N)$-Shamir Secret Sharing for a ring $\mathbb{Z}/q\mathbb{Z}$ with $p > N$ for all positive primes dividing $q$ is perfectly secure, where by perfectly secure we mean that the random variable associated to the secret is independent to any set of $T-1$ random variables associated to the shares.*

Perfect secrecy is the best level of security since it is equivalent to ask that any unbounded adversary cannot retrieve the secret from $T-1$ shares with probability greater than $q^{-1}$.

*Remark* 42. One Time Pad, the classical example of cryptosystem achieving perfect secrecy, can be seen as a particular case of a $(2, 2)$-Shamir Secret sharing on the ring of integers modulo $2^n$.

To remove the small factor $q_0$ in the order of a group we simply have to consider the subgroup generated by $g^{q_0}$, this way we get a group of order $\frac{q}{q_0}$. Clearly we must check that the subgroup satisfies the level of security required. For example for CSI-FiSh to have a SSS with up to 1407180 users we can consider the group generated by $\mathfrak{l}^{3 \cdot 37}$, that has order

$$q' = 1407181 \cdot 515936042952958677442935848 89 \cdot \\ \cdot 3159941450468199585300827874558783220490 9 \simeq 2^{250.342} \ . \tag{4.17}$$

**Algorithm 13** TDS.Sign

**Require:** $x \in X$, a security parameter $\lambda$, a hash function $\mathsf{H}$, a public key $(x, y = [s]x)$, a secure commitment scheme $\mathsf{COM}$, a set $J$ of $T$ parties. Observe that there exists $f(t)$ so that $s = f(0)$ and the party $P_i$ knows $f(i)$.

**Ensure:** A valid signature for the message $\mathsf{m}$ under the public key $(x, y)$.

1: **for** $i \in J$ **do**
2: $P_i$ pick $\mathsf{salt}_i$ randomly and sends $\mathsf{COM}(\mathsf{salt}_i)$
3: Set $x_0^j = x$ for all $j = 1$ to $\lambda$ ▷ Shared commitment generation phase
4: **for** $i \in J$ **do**
5: **for** $j = 1$ to $\lambda$ **do**
6: $P_i$ chooses $b_i^j \in G$ and computes $x_i^j = [b_i^j]x_{i-1}^j$
7: $P_i$ outputs $x_i^j$
8: Set $x^j = x_T^j$ for all $j = 1$ to $\lambda$.
9: Each party publishes $\mathsf{salt}_i$ and checks the consistency of the received data with the initial commitment.
10: $\mathsf{salt} = \sum_i \mathsf{salt}_i$
11: Compute $\mathsf{ch} = \mathsf{H}(x^1 \| ... \| x^\lambda \| \mathsf{salt} \| \mathsf{m})$ ▷ Non-iterative challenges evaluation
12: **for** $i \in J$ **do** ▷ Shared response generation phase
13: **for** $j = 1$ to $\lambda$ **do**
14: $P_i$ outputs $u_i^j = b_i^j - \mathsf{ch}_j \cdot s_i \cdot \delta_{J,i}(0)$;
15: $\mathsf{resp}_j = \sum_i u_i^j$ **for** $j = 1$ to $\lambda$;
16: $\mathsf{sig} = \mathsf{ch} \| \mathsf{salt} \| \mathsf{resp}_1 \| ... \| \mathsf{resp}_\lambda$

You can see the instantiation for this scheme in Algorithm 13. Let's point out some differences from Algorithm 7 and 12.

- The number of rounds is limited to the number of users $T$.
- The combination phase, in state 14, can be computed at the same time, halving the latency in the protocol.
- The intermediate set elements $x_1, ..., x_{T-1}$ with

$$x = x_0 \xrightarrow{s_1\delta_{J,1}(0)} x_1 \xrightarrow{s_2\delta_{J,2}(0)} x_2 \text{ --------} x_{T-1} \xrightarrow{s_T\delta_{J,T}(0)} x_T = y \quad ,$$

  vary for any set $J$, so they are unknown.

**Security of LSSS** In [DM20] they prove simulatability (Definition 4.3) under a new assumption:

**Problem 43** ($a$-Power-DDHA problem)**.** Let $(G, X, \star)$ be a cryptographic group action. Given $x \in X$, $1 < a < |G|$ and $g$ be a uniformly random element in $G$ distinguish between tuples of the form $(a, x, g \star x, g^a \star x)$ and $(a, x, g \star x, y)$, where $y$ is sampled from the uniform distribution.

We can now repeat their proof, without this assumption, using instead Proposition 4.12 to simulate the intermediate elements.

**Theorem 4.13.** *For a ring $\mathbb{Z}/q\mathbb{Z}$ with $p > N$ for all positive primes dividing $q$, the signature scheme of Algorithm 13 is simulatable.*

Since the centralized scheme is unforgeable in the random oracle model, then we get the following corollary:

**Corollary 4.14.** *The signature scheme in Algorithm 13, with the classical centralized verification scheme, is unforgeable for honest-but-curious users under the assumption of the hardness of GAIP (Problem 7), in the random oracle model.*

*Proof of Theorem 4.13.* We proceed as in the proof of [DM20, Theorem 1]. Let $A$ be a set of indices of corrupted shares $s_i$, with cardinality strictly smaller than $T$. Given a valid signature $(\mathsf{ch}\|\mathsf{salt}\|\mathsf{resp}_1\|...\|\mathsf{resp}_\lambda)$ for a message chosen by the adversary and a set of user $J$ we want to simulate the view for the users in $A$ using the corrupted shares. Wlog we can assume $J = [1, T]$ and not specify $A$ instead.

The salts $\mathsf{salt}_1, ..., \mathsf{salt}_{T-1}$ can be simulated straightforwardly at random, then the last one is obtained by fixing $\mathsf{salt}_T = \mathsf{salt} - \sum_{i=1}^{T-1} \mathsf{salt}_i$.

For the responses (line 14) for $j = 1, ..., \lambda$ we generate uniformly random integers $u_i^j \in \mathbb{Z}/q\mathbb{Z}$ for $i = 1, ..., T-1$ and fix $u_T^j = \mathsf{resp}_j - \sum_{i=1}^{T-1} u_i^j$. Since the responses are uniformly sampled also the view $u_i^j$ are uniformly random for any user. Hence they are indistinguishable.

The commitments $x_i^j$ (line 7) instead need a different take for the two possible challenges. For $j$ with $\mathsf{ch}_j = 0$, since $u_i^j = b_i^j$, we can simply set, as in line 6,

$$x_i^j = \left[\sum_{k=1}^{i} b_k^j\right] x = \left[\sum_{k=1}^{i} u_k^j\right] x \ .$$

For $j$ with $\mathsf{ch}_j = 1$ we need instead to build a chain replacing:

$$x = x_0 \xrightarrow{s_1\delta_{J,1}(0)} x_1 \xrightarrow{s_2\delta_{J,2}(0)} x_2 \text{ --------- } x_{T-1} \xrightarrow{s_T\delta_{J,T}(0)} x_T = y \quad .$$

The corrupted shares can be used to define a direct chain from $x$ applying recursively $s_i\delta_{J,1}(0)$ and a backwards one from $y$ applying instead $-\delta_{J,1}(0)$. If there is only one non corrupted user $P_{i_0}$ we can define all the intermediate elements up to $x_{i_0-1}$ with a direct chain, while the other using a backwards one. Then we know that $x_{i_0} = [s - \sum_{i\neq i_0} s_i\delta_{J,i}(0)]x_{i_0-1} = [s_{i_0}\delta_{J,i_0}(0)]x_{i_0-1}$, as in an honest execution, since $s = \sum_i s_i\delta_{J,i}(0)$. An example for $T = 4$ and $i_0 = 3$ is showed:

$$x \xrightarrow{s_1\delta_{J,1}(0)} x_1 \xrightarrow{s_2\delta_{J,2}(0)} x_2 \text{ --------- } x_3 \xleftarrow[-s_4\delta_{J,4}(0)]{} y \quad .$$

The main problem is that, when there are less than $T-1$ corrupted shares the chains are not enough to define all the intermediate elements. We can solve this by sampling the missing shares uniformly for all users, but one, say $i_0$ the relative index. This way (under the assumption that $p > N$ for all positive primes dividing $q$) using Proposition 4.12 there exists a uniformly distributed polynomial $\hat{f} \in \mathbb{Z}/q\mathbb{Z}[t]$ with $\hat{f}(0) = s$ and $\hat{f}(i) = s_i$ for $i = 1, ..., T$ and $i \neq i_0$. This way we get again $x_{i_0} = [\hat{f}(i_0)\delta_{J,i_0}(0)]x_{i_0-1}$. Since the original polynomial $f$ and $\hat{f}$ are both sampled uniformly they are indistinguishable. We can see here an example for $T = 6$, $A = \{1, 3, 6\}$ and $i_0 = 6$:

$$x \xrightarrow{s_1\delta_{J,1}(0)} x_1 \dashrightarrow^{s_2^*\delta_{J,2}(0)} x_2 \xrightarrow{s_3\delta_{J,3}(0)} x_3 \text{ --------- } x_4 \xleftarrow[-s_4^*\delta_{J,4}(0)]{} x_5 \xleftarrow[-s_6\delta_{J,6}(0)]{} y \quad ;$$

where sampled shares are labeled $s_i^*$. We can also see directly that the intermediate link $r \in \mathbb{Z}/q\mathbb{Z}$ with $x_{i_0} = [r\delta_{J,i_0}(0)]x_{i_0-1}$ can be found also solving for $r$:

$$\sum_{i\in A, i\neq i_0} s_i^*\delta_{J,i}(0) + \sum_{i\in J\setminus A} s_i\delta_{J,i}(0) + r\delta_{J,i_0}(0) = s \ ;$$

that has always a solution since $\delta_{J,i_0}(0)$ is invertible for the requirement on $q$ prime divisors. Note that we only need the existance of $r$, not its knowledge. Using these intermediate elements we can finally simulate the commitments as:

$$x_i^j = [u_i^j + \cdots u_1^j]x_i \text{ for } i \in [1, T], j \in [1, \lambda] \ .$$

These are the same as in Algorithm 13 since $b_i^j = u_i^j + \hat{f}(i)\delta_{J,i}(0)$. □

## 4.4 Concrete Instantiations

In this section, we present concrete instantiations of our protocols, using the code equivalence group actions behind the LESS and MEDS signature schemes [BBPS21; Cho+22]; 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 discuss these again using the generic group action notation, since they can work in general.

### 4.4.1 Multi-bit Challenges

It is straightforward to use the multibit optimization from Section 2.3.5 for the threshold schemes, as already done for [CS20]. As consequence, we consider $r$ public keys $y_1, ..., y_r$ generated from the initial element $x$ by $r$ shared keys $g^{(1)}, ..., g^{(r)}$. To do this, we repeat the $\mathsf{TDS.KeyGen}$ algorithm $r$ times in order to generate $r$ shared secret keys $g^{(1)}, ..., g^{(r)}$ and public keys $y_1, ..., y_r$. To ease the reading we also fix $y_0 := x$.

At this point we can modify Protocol 4.2.1 to produce challenges in a larger space, as shown in Protocol 4.4.1.

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

| **PROVERS** | | **VERIFIER** |
|---|---|---|
| Set $\tilde{x} \leftarrow x$ and for $i = 1, ..., M$ do : | | |
| $P_{\tau(J,i)}$ get $\tilde{g}_i \xleftarrow{\$} X$, set $\tilde{x} \leftarrow \tilde{g}_i \star \tilde{x}$ | $\xrightarrow{\mathsf{resp}}$ | |
| Set $\mathsf{com} \leftarrow \mathsf{H}(\tilde{x})$. | | |
| | $\xleftarrow{\mathsf{ch}}$ | $\mathsf{ch} \xleftarrow{\$} \{0, 1, ..., r\}$. |
| Set $\mathsf{resp} \leftarrow e$. | | |
| for $i = 1, ..., M$ do : | $\xrightarrow{\mathsf{resp}}$ | |
| $P_{\tau(J,i)}$ set $\mathsf{resp} \leftarrow \tilde{g}_i \cdot \mathsf{resp} \cdot \left(g_{I_i}^{(\mathsf{ch})}\right)^{-1}$. | | Accept if $\mathsf{H}(\mathsf{resp} \star y_{\mathsf{ch}}) = \mathsf{resp}$. |

Protocol 4.4.1: Threshold identification protocol with soundness error $\frac{1}{r+1}$.

With this protocol the soundness error is reduced to $\frac{1}{r+1}$, thus in the signing algorithm we only need to execute $\lceil \frac{\lambda}{\log_2(r+1)} \rceil$ rounds, reducing both the signature size and the computational cost, but increasing the public key size. It is straightforward to obtain a security reduction to from the multibit version of the threshold scheme to the centralized version of it, in the same way as for the proof of Theorem 4.5 and Lemma 4.6.

| Version | #rounds | \|pk\| | \|sig\| |
|---|---|---|---|
| Classic | $\lambda$ | $l_X$ | $\lambda l_G + 3\lambda$ |
| Multibit | $\left\lceil \frac{\lambda}{l} \right\rceil$ | $(r-1)l_X$ | $\left\lceil \frac{\lambda}{\log_2(r)} \right\rceil (l_G + l) + 2\lambda$ |
| Fixed | $t$ s.t. $\binom{t}{\omega} \geqslant 2^\lambda$ | $l_X$ | $(N_{\mathsf{seeds}} M + 2)\lambda + \omega l_G + t$ |
| Both | $t$ s.t. $\binom{t}{\omega}(r-1)^\omega \geqslant 2^\lambda$ | $(r-1)l_X$ | $(N_{\mathsf{seeds}} M + 2)\lambda + \omega l_G + t$ |

Table 4.1: Overview of the sizes for different variants.

### 4.4.2 Fixed-Weight Challenges

Another possible variant for group action-based signature schemes is the use of fixed-weight challenge strings, as shown in Section 2.3.2.

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.

Also it is clear that here the seeds can be further compressed by using a seed tree structure described in Section 2.3.3. As said before the combination with a fresh salt is necessary to avoid the attack from [Cha22], since already $\mathsf{salt}_i$ is necessary to achieve the security of the threshold construction the parties could use it also for the PRNG call.

### 4.4.3 Scheme Parameters

In Table 4.1, 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 use $l_X$ to denote the weight in bits of an element of the set $X$, and $l_G$ to denote that of a group element of $G$.

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:

- $\#\mathsf{rounds} \cdot l_X$ bits for the commitment phase,
- $\#\mathsf{rounds} \cdot l_G + 2\lambda$ bits in general and $(N_{\mathsf{seeds}} M + 2)\lambda + \omega l_G$ when using fixed-weight challenges.

At this point, we can see specific choices for LESS and MEDS. We select the public parameters that satisfy the requirement of 128 bits of classical security and at least 64 bits of quantum security, and evaluate $l_X$ and $l_G$ accordingly. We include here the data for the original signature schemes, as well as parameters that we found in order to optimize the sum $|\mathsf{pk}| + |\mathsf{sig}|$ 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 $M\lambda \geqslant l_G$).

**Instantiations with LESS.**

From [Bal+23b] we have taken the secure balanced LESS parameters for the NIST Security Level I: $n = 252, k = 126$ (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 $l_X = 13.7\mathsf{kB}$. Instead to send a monomial map we can use the IS-LEP optimization [PS23]. This new optimization requires the use of a new canonical representation of the generator matrices via information sets, this way the equality can then be verified using only the monomial map truncated on the preimage of the information set, 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 [CPS23]) can be used also for the threshold protocol since:

- for the commitment phase the last user can simply commit using the modified canonical form, they then store the additional info received (the information set used);
- for the response phase when the monomial map $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$ is recovered can then be truncated again by the last user 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 require to increase the code length up to $n = 440$ for $q = 127$ leading to a public key size of 17.1kB and truncated monomial map size of 100B. Numbers are reported in Table 4.2, where we report, in the last column, also the total amount of exchanged data.

**Instantiations with MEDS.**

From [Cho+23] 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 $l_X = 3.84\mathsf{kB}$. Observe that in the distributed key generation case we cannot use the public key compression mechanism from [Cho+22, 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

| Case | Variant | $t$ | $\omega$ | $\lvert\mathsf{pk}\rvert$ (KiB) | $\lvert\mathsf{sig}\rvert$ (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 4.2: Parameters for the threshold version of LESS

| Case | Variant | $t$ | $\omega$ | $r$ | $\lvert\mathsf{pk}\rvert$ (KiB) | $\lvert\mathsf{sig}\rvert$ (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 |
| [Cho+23, Section 8] | M | - | - | 3 | 7.50 | 3.37 | $\binom{N}{T-1}$0.342 |

Table 4.3: Parameters for the threshold version of MEDS

$l_G = 588\mathsf{B}$. Numbers are reported in Table 4.3; as above, in the last column we report the total amount of exchanged data.

The group element compression technique from [Cho+23, Section 8] (see it in Section 3.4.2) cannot be used in the same way since the change of basis matrix $\mathbf{T}$ cannot be shared, however here we propose a slightly less efficient version suitable for the multiparty calculations in which the last user modify its executions.

- **Commitment**: the last user generate via a public seed a full rank matrix $\mathbf{R} \in \mathbb{F}_q^{2\times mn}$, i.e. random independent codewords; and take two random codewords in the code received by the previous user. Solve Equation (3.47) to get $\tilde{\mathbf{A}}_M, \tilde{\mathbf{B}}_M$ and evaluate the final code as usual.
- **Response**: At the end of the classical 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 this 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 Protocol 3.4.4 thus the communication price per round is cut down to $2k\lceil \log_2(q) \rceil$ bits.

*Remark* 44. Differently from the original optimization 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.

# BIBLIOGRAPHY


[AABN02] 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: *Advances in Cryptology — EUROCRYPT 2002*. Ed. by L. R. Knudsen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 418–433. ISBN: 978-3-540-46035-0.

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[Adj+23] G. Adj, L. Rivera-Zamarripa, J. Verbel, E. Bellini, S. Barbero, A. Esser, C. Sanna, and F. Zweydinger. *MiRitH (MinRank in the Head)*. https://pqc-mirith.org/assets/downloads/mirith_specifications_v1.0.0.pdf. Accessed: 2023-09-13. 2023.

[ADMP20] N. Alamati, L. De Feo, H. Montgomery, and S. Patranabis. "Cryptographic group actions and applications". In: *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*. Springer. 2020, pp. 411–439.

[AFLT12] M. Abdalla, P.-A. Fouque, V. Lyubashevsky, and M. Tibouchi. "Tightly-secure signatures from lossy identification schemes". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2012, pp. 572–590.

[AFMP20] N. Alamati, L. D. Feo, H. Montgomery, and S. Patranabis. *Cryptographic Group Actions and Applications*. Cryptology ePrint Archive, Paper 2020/1188. https://eprint.iacr.org/2020/1188. 2020. URL: https://eprint.iacr.org/2020/1188.

[Agu+20] 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.

[Agu+23] C. Aguilar-Melchor, N. Gama, J. Howe, A. Hü lsing, D. Joseph, and D. Yue. "The return of the SDitH". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 564–596.

[Alb+20] M. R. Albrecht et al. *Classic McEliece*. NIST PQC Submission. 2020.

[Ara+20] N. Aragon et al. *BIKE*. NIST PQC Submission. 2020.

[Bab16] L. Babai. "Graph isomorphism in quasipolynomial time". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 2016, pp. 684–697.

[Bal+23a] M. Baldi, S. Bitzer, A. Pavoni, P. Santini, A. Wachter-Zeh, and V. Weger. "Zero Knowledge Protocols and Signatures from the Restricted Syndrome Decoding Problem". In: *Cryptology ePrint Archive* (2023).

[Bal+23b] M. Baldi et al. *Matrix Equivalence Digital Signature*. https://www.less-project.com/LESS-2023-08-18.pdf. Accessed: 2023-09-15. 2023.

[Bar+20] M. Bardet, M. Bros, D. Cabarcas, P. Gaborit, R. Perlner, D. Smith-Tone, J.-P. Tillich, and J. Verbel. "Improvements of algebraic attacks for solving the rank decoding and MinRank problems". In: *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part I 26*. Springer. 2020, pp. 507–536.

[Bas+23] 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*. Springer. 2023.

[BBBGT23] M. Bardet, P. Briaud, M. Bros, P. Gaborit, and J.-P. Tillich. "Revisiting algebraic attacks on MinRank and on the rank decoding problem". In: *Designs, Codes and Cryptography* (2023), pp. 1–37.

[BBMP23] M. Battagliola, G. Borin, A. Meneghetti, and E. Persichetti. *Cutting the GRASS: Threshold GRoup Action Signature Schemes*. Cryptology ePrint Archive, Paper 2023/859. https://eprint.iacr.org/2023/859. 2023. URL: https://eprint.iacr.org/2023/859.

[BBPS21] 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*. Springer. 2021, pp. 23–43.

[BBPS22] A. Barenghi, J.-F. Biasse, E. Persichetti, and P. Santini. *On the Computational Hardness of the Code Equivalence Problem in Cryptography*. Cryptology ePrint Archive, Paper 2022/967. https://eprint.iacr.org/2022/967. 2022. URL: https://eprint.iacr.org/2022/967.

[BBY20] A. Baccarini, M. Blanton, and C. Yuan. "Multi-party replicated secret sharing over a ring with applications to privacy-preserving machine learning". In: *Cryptology ePrint Archive* (2020).

[BD21] J. Burdges and L. De Feo. "Delay encryption". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2021, pp. 302–326.

[BDLS20a] D. J. Bernstein, L. De Feo, A. Leroux, and B. Smith. "Faster computation of isogenies of large prime degree". In: *Open Book Series* 4.1 (2020), pp. 39–55. DOI: 10.2140/obs.2020.4.39.

[BDLS20b] D. J. Bernstein, L. De Feo, A. Leroux, and B. Smith. "Faster computation of isogenies of large prime degree". In: *Open Book Series* 4.1 (2020), pp. 39–55.

[BDPV21] W. Beullens, L. Disson, R. Pedersen, and F. Vercauteren. "CSI-RAShi: distributed key generation for CSIDH". In: *International Conference on Post-Quantum Cryptography*. Springer. 2021.

[BDV] L. T. A. N. Brandão, M. Davidson, and A. Vassilev. *NIST Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives*. Accessed: 2020-08-27. URL: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8214A.pdf.

[Bea92] D. Beaver. "Efficient multiparty protocols using circuit randomization". In: *Advances in Cryptology—CRYPTO'91: Proceedings 11*. Springer. 1992, pp. 420–432.

[Ber10] D. J. Bernstein. "Grover vs. mceliece". In: *Post-Quantum Cryptography: Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings 3*. Springer. 2010, pp. 73–80.

[Beu20] W. Beullens. "Not enough LESS: An improved algorithm for solving code equivalence problems over F q". In: *International Conference on Selected Areas in Cryptography*. Springer. 2020, pp. 387–403.

[BFS99] J. F. Buss, G. S. Frandsen, and J. O. Shallit. "The computational complexity of some problems of linear algebra". In: *Journal of Computer and System Sciences* 58.3 (1999), pp. 572–596.

[BFV13] C. Bouillaguet, P.-A. Fouque, and A. Véber. "Graph-theoretic algorithms for the "isomorphism of polynomials" problem". In: *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*. Springer. 2013, pp. 211–227.

[BGZ23] D. Boneh, J. Guan, and M. Zhandry. "A Lower Bound on the Length of Signatures Based on Group Actions and Generic Isogenies". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 507–531.

[BKP20] W. Beullens, S. Katsumata, and F. Pintore. *Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices*. Cryptology ePrint Archive, Paper 2020/646. https://eprint.iacr.org/2020/646. 2020. URL: https://eprint.iacr.org/2020/646.

[BKV19] W. Beullens, T. Kleinjung, and F. Vercauteren. "CSI-FiSh: efficient isogeny based signatures through class group computations". In: *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I*. Springer. 2019, pp. 227–247.

[BL08] P. A. Brooksbank and E. M. Luks. "Testing isomorphism of modules". In: *Journal of Algebra* 320.11 (2008), pp. 4020–4029.

[Blä+22] 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". In: *Cryptology ePrint Archive* (2022).

[Bla+23] M. Blaser, D. H. Duong, A. K. Narayanan, T. Plantard, Y. Qiao, A. Sipasseuth, and G. Tang. *ALTEQ Signature Specification*. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/ALTEQ-Spec-web.pdf. Accessed: 2023-10-11. 2023.

[Bla79] G. R. Blakley. "Safeguarding cryptographic keys". In: *Managing Requirements Knowledge, International Workshop on*. Los Alamitos, CA, USA: IEEE Computer Society, 1979, p. 313. DOI: 10.1109/AFIPS.1979.98. URL: %5Curl%7Bhttps://doi.ieeecomputersociety.org/10.1109/AFIPS.1979.98%7D.

[BMPS20] J.-F. Biasse, G. Micheli, E. Persichetti, and P. Santini. "LESS is More: Code-Based Signatures Without Syndromes". In: *Progress in Cryptology - AFRICACRYPT 2020*. Ed. by A. Nitaj and A. Youssef. Cham: Springer International Publishing, 2020, pp. 45–65. ISBN: 978-3-030-51938-4.

[BN06] 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. Association for Computing Machinery, 2006, pp. 390–399. ISBN: 1595935185. DOI: 10.1145/1180405.1180453. URL: https://doi.org/10.1145/1180405.1180453.

[Bon+11] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. "Random oracles in a quantum world". In: *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*. Springer. 2011, pp. 41–69.

[BOS19] M. Bardet, A. Otmani, and M. Saeed-Taha. "Permutation Code Equivalence is Not Harder Than Graph Isomorphism When Hulls Are Trivial". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. 2019, pp. 2464–2468. DOI: 10.1109/ISIT.2019.8849855.

[BPS23] G. Borin, E. Persichetti, and P. Santini. *Zero-Knowledge Proofs from the Action Subgraph*. Cryptology ePrint Archive, Paper 2023/718. https://eprint.iacr.org/2023/718. 2023. URL: https://eprint.iacr.org/2023/718.

[BR06] M. Bellare and P. Rogaway. "The security of triple encryption and a framework for code-based game-playing proofs". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2006, pp. 409–426.

[ÇBCS22] S. Çalkavur, A. Bonnecaze, R. dela Cruz, and P. Solé. *Code Based Secret Sharing Schemes: Applied Combinatorial Coding Theory*. World Scientific, 2022.

[CD23] W. Castryck and T. Decru. "An efficient key recovery attack on SIDH". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 423–447.

[CDG20] A. Couvreur, T. Debris-Alazard, and P. Gaborit. *On the hardness of code equivalence problems in rank metric*. 2020. DOI: 10.48550/ARXIV.2011.04611. URL: https://arxiv.org/abs/2011.04611.

[Cha+23] J. Chavez-Saab et al. *SQISign Specification*. https://sqisign.org/spec/sqisign-20230601.pdf. Accessed: 2023-10-04. 2023.

[Cha22] A. Chailloux. "On the (In) security of optimized Stern-like signature schemes". In: WCC. 2022.

[Cho+22] 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*. Cryptology ePrint Archive, Paper 2022/1559. https://eprint.iacr.org/2022/1559. 2022. URL: https://eprint.iacr.org/2022/1559.

[Cho+23] 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. Accessed: 2023-09-12. 2023.

[CJS14] A. Childs, D. Jao, and V. Soukharev. "Constructing elliptic curve isogenies in quantum subexponential time". In: *Journal of Mathematical Cryptology* 8.1 (2014), pp. 1–29.

[CLMPR18] W. Castryck, T. Lange, C. Martindale, L. Panny, and J. 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*. Springer. 2018, pp. 395–427.

[CM22] F. Campos and P. Muth. "On actively secure fine-grained access structures from isogeny assumptions". In: *International Conference on Post-Quantum Cryptography*. Springer. 2022, pp. 375–398.

[Cou06] J.-M. Couveignes. *Hard Homogeneous Spaces*. Cryptology ePrint Archive, Paper 2006/291. https://eprint.iacr.org/2006/291. 2006. URL: https://eprint.iacr.org/2006/291.

[CPS23] T. Chou, E. Persichetti, and P. Santini. *On Linear Equivalence, Canonical Forms, and Digital Signatures*. Cryptology ePrint Archive, Paper 2023/1533. https://eprint.iacr.org/2023/1533. 2023. URL: %5Curl%7Bhttps://eprint.iacr.org/2023/1533%7D.

[CS19] D. Cozzo and N. P. Smart. "Sharing the LUOV: threshold post-quantum signatures". In: *IMA International Conference on Cryptography and Coding*. Springer. 2019, pp. 128–153.

[CS20] D. Cozzo and N. P. Smart. "Sashimi: Cutting up CSI-FiSh Secret Keys to Produce an Actively Secure Distributed Signing Protocol". In: *Post-Quantum Cryptography*. Ed. by J. Ding and J.-P. Tillich. Cham: Springer International Publishing, 2020, pp. 169–186. ISBN: 978-3-030-44223-1.

[CVE11] P.-L. Cayrel, P. Véron, and S. M. El Yousfi Alaoui. "A Zero-Knowledge Identification Scheme Based on the q-ary Syndrome Decoding Problem". In: *Selected Areas in Cryptography*. Ed. by A. Biryukov, G. Gong, and D. R. Stinson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 171–186. ISBN: 978-3-642-19574-7.

[DAl22a] G. D'Alconzo. "A Note on the Hardness of Problems from Cryptographic Group Actions". In: *arXiv preprint arXiv:2202.13810* (2022).

[DAl22b] G. D'Alconzo. *Code Equivalence in the Sum-Rank Metric: Hardness and Completeness*. Cryptology ePrint Archive, Paper 2022/968. https://eprint.iacr.org/2022/968. 2022. URL: https://eprint.iacr.org/2022/968.

[De 17] L. De Feo. "Mathematics of isogeny based cryptography". In: *arXiv preprint arXiv:1711.04062* (2017).

[Del78] P. Delsarte. "Bilinear forms over a finite field, with applications to coding theory". In: *Journal of combinatorial theory, Series A* 25.3 (1978), pp. 226–241.

[DFMS19] J. Don, S. Fehr, C. Majenz, and C. Schaffner. "Security of the Fiat-Shamir transformation in the quantum random-oracle model". In: *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019 , Proceedings, Part II 39*. Springer. 2019, pp. 356–383.

[DG19] L. De Feo and S. 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*. Springer. 2019, pp. 759–789.

[DM20] L. De Feo and M. Meyer. "Threshold schemes from isogeny assumptions". In: *Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part II 23*. Springer. 2020, pp. 187–212.

[DS23] A. J. Di Scala and C. Sanna. "Smaller public keys for MinRank-based schemes". In: *arXiv preprint arXiv:2302.12447* (2023).

[Duc+18] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehl é. "Crystals-dilithium: A lattice-based digital signature scheme". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 238–268.

[DW22] L. Ducas and W. van Woerden. "On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2022, pp. 643–673.

[EZ17] Y. Erlich and D. Zielinski. "DNA Fountain enables a robust and efficient storage architecture". In: *science* 355.6328 (2017), pp. 950–954.

[Fen22] T. Feneuil. "Building MPCitH-based signatures from MQ, MinRank, Rank SD and PKP". In: *Cryptology ePrint Archive* (2022).

[FF93] J. Feigenbaum and L. Fortnow. "Random-self-reducibility of complete sets". In: *SIAM Journal on Computing* 22.5 (1993), pp. 994–1005.

[FGHLS12] E. Farhi, D. Gosset, A. Hassidim, A. Lutomirski, and P. Shor. "Quantum money from knots". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 2012, pp. 276–289.

[FJR22] T. Feneuil, A. Joux, and M. Rivain. "Syndrome decoding in the head: shorter signatures from zero-knowledge proofs". In: *Annual International Cryptology Conference*. Springer. 2022.

[FLP08] J.-C. Faugere, F. Levy-dit-Vehel, and L. Perret. "Cryptanalysis of minrank". In: *Annual International Cryptology Conference*. Springer. 2008, pp. 280–296.

[FS87] A. Fiat and A. Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology — CRYPTO' 86*. Ed. by A. M. Odlyzko. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194.

[Gab12] E. Gabidulin. "A brief survey of metrics in coding theory". In: *Mathematics of Distances and Applications* 66 (2012), pp. 66–84.

[Gab85] E. M. Gabidulin. "Theory of codes with maximum rank distance". In: *Problemy peredachi informatsii* 21.1 (1985), pp. 3–16.

[GGN16] 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*. Springer. 2016, pp. 156–174.

[GHHM21] A. B. Grilo, K. Hövelmanns, A. Hülsing, and C. Majenz. "Tight adaptive reprogramming in the QROM". In: *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings , Part I 27*. Springer. 2021, pp. 637–667.

[GHS02] S. D. Galbraith, F. Hess, and N. P. Smart. "Extending the GHS Weil descent attack". In: *Advances in Cryptology—EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, The Netherlands, April 28–May 2, 2002 Proceedings 21*. Springer. 2002, pp. 29–44.

[GJKR96] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. "Robust threshold DSS signatures". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1996, pp. 354–371.

[GMR19] S. Goldwasser, S. Micali, and C. Rackoff. "The knowledge complexity of interactive proof-systems". In: *Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali*. 2019, pp. 203–225.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. "A digital signature scheme secure against adaptive chosen-message attacks". In: *SIAM Journal on computing* 17.2 (1988), pp. 281–308.

[GPS22] S. Gueron, E. Persichetti, and P. Santini. "Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup". In: *Cryptography* 6.1 (2022), p. 5.

[GQ19] J. A. Grochow and Y. Qiao. "Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions". In: *arXiv preprint arXiv:1907.00309* (2019).

[GQ21] J. A. Grochow and Y. Qiao. "On the Complexity of Isomorphism Problems for Tensors, Groups, and Polynomials I: Tensor Isomorphism-Completeness". In: *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*. Ed. by J. R. Lee. Vol. 185. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021, 31:1–31:19. ISBN: 978-3-95977-177-1. DOI: 10.4230/LIPIcs.ITCS.2021.31. URL: https://drops.dagstuhl.de/opus/volltexte/2021/13570.

[Gro96] L. K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866. URL: https://doi.org/10.1145/237814.237866.

[GRS15] P. Gaborit, O. Ruatta, and J. Schrek. "On the complexity of the rank syndrome decoding problem". In: *IEEE Transactions on Information Theory* 62.2 (2015), pp. 1006–1019.

[GS86] S. Goldwasser and M. Sipser. "Private coins versus public coins in interactive proof systems". In: *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. 1986, pp. 59–68.

[ISN89] M. Ito, A. Saito, and T. Nishizeki. "Secret sharing scheme realizing general access structure". In: *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)* 72.9 (1989), pp. 56–64.

[Jao+20] D. Jao et al. *SIKE*. 2020.

[Jou23] A. Joux. *MPC in the head for isomorphisms and group actions*. Cryptology ePrint Archive, Paper 2023/664. https://eprint.iacr.org/2023/664. 2023. URL: https://eprint.iacr.org/2023/664.

[JQSY19] Z. Ji, Y. Qiao, F. Song, and A. Yun. "General linear group action on tensors: A candidate for post-quantum cryptography". In: *Theory of Cryptography Conference*. Springer. 2019, pp. 251–281.

[Kis12] V. V. Kisil. "Erlangen program at large: an overview". In: *Advances in applied analysis* (2012), pp. 1–94.

[KLS18] E. Kiltz, V. Lyubashevsky, and C. Schaffner. "A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model". In: *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part III 37*. Springer. 2018, pp. 552–586.

[KT17] G. Kachigar and J.-P. Tillich. "Quantum information set decoding algorithms". In: *International Workshop on Post-Quantum Cryptography*. Springer. 2017, pp. 69–89.

[KTT23] N. Kimura, A. Takayasu, and T. Takagi. "Memory-Efficient Quantum Information Set Decoding Algorithm". In: *Australasian Conference on Information Security and Privacy*. Springer. 2023, pp. 452–468.

[Kup05] G. Kuperberg. "A subexponential-time quantum algorithm for the dihedral hidden subgroup problem". In: *SIAM Journal on Computing* 35.1 (2005), pp. 170–188.

[Lan12] S. Lang. *Algebra*. Vol. 211. Springer Science & Business Media, 2012.

[Leo82] J. Leon. "Computing automorphism groups of error-correcting codes". In: *IEEE Transactions on Information Theory* 28.3 (1982), pp. 496–511. DOI: 10.1109/TIT.1982.1056498.

[LN94] R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications*. Cambridge university press, 1994.

[Luk93] E. Luks. "Permutation groups and polynomial-time computation". In: *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* (1993), pp. 139–175.

[Lyu09] V. Lyubashevsky. "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2009, pp. 598–616.

[LZ19] Q. Liu and M. Zhandry. "Revisiting post-quantum fiat-shamir". In: *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019 , Proceedings, Part II 39*. Springer. 2019, pp. 326–355.

[McE78] R. J. McEliece. "A public-key cryptosystem based on algebraic". In: *Coding Thv* 4244 (1978), pp. 114–116.

[MMPPW23] L. Maino, C. Martindale, L. Panny, G. Pope, and B. Wesolowski. "A direct key recovery attack on SIDH". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 448–471.

[Mor14] K. Morrison. "Equivalence for rank-metric and matrix codes and automorphism groups of Gabidulin codes". In: *IEEE Transactions on Information Theory* 60.11 (2014), pp. 7035–7046.

[MPS23] A. Meneghetti, A. Pellegrini, and M. Sala. "On the equivalence of two post-quantum cryptographic families". In: *Annali di Matematica Pura ed Applicata (1923-)* 202.2 (2023), pp. 967–991.

[NIS17] NIST. *Post-Quantum Cryptography Standardization*. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography. 2017.

[NIS22a] NIST. *Post-Quantum Cryptography, Round 4 Submissions*. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions. 2022.

[NIS22b] NIST. *Post-Quantum Cryptography, Selected Algorithms 2022*. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. 2022.

[NIS23] NIST. *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. URL: https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals. 2023.

[Pet10] C. Peters. "Information-set decoding for linear codes over F q". In: *Post-Quantum Cryptography: Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings 3*. Springer. 2010, pp. 81–94.

[Poh69] I. Pohl. *Bi-directional and heuristic search in path problems.* Tech. rep. Stanford Linear Accelerator Center, Calif., 1969.

[Pra62] E. Prange. "The use of information sets in decoding cyclic codes". In: *IRE Transactions on Information Theory* 8.5 (1962), pp. 5–9.

[PS00] D. Pointcheval and J. Stern. "Security arguments for digital signatures and blind signatures". In: *Journal of cryptology* 13 (2000), pp. 361–396.

[PS23] E. Persichetti and P. Santini. "A New Formulation of the Linear Equivalence Problem and Shorter LESS Signatures". In: *Cryptology ePrint Archive* (2023).

[PWBJ17] R. Pellikaan, X.-W. Wu, S. Bulygin, and R. Jurrius. *Codes, Cryptology and Curves with Computer Algebra*. Cambridge University Press, 2017. DOI: 10.1017/9780511982170.

[Rav16] A. Ravagnani. "Rank-metric codes and their duality theory". In: *Designs, Codes and Cryptography* 80.1 (2016), pp. 197–216. DOI: 10.1007/s10623-015-0077-3. URL: https://doi.org/10.1007/s10623-015-0077-3.

[Reg04a] O. Regev. "A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space". In: *arXiv preprint quant-ph/0406151* (2004).

[Reg04b] O. Regev. "Quantum computation and lattice problems". In: *SIAM Journal on Computing* 33.3 (2004), pp. 738–760.

[Rob23] D. Robert. "Breaking SIDH in polynomial time". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 472–503.

[RST22] K. Reijnders, S. Samardjiska, and M. Trimoska. *Hardness estimates of the Code Equivalence Problem in the Rank Metric*. Cryptology ePrint Archive, Paper 2022/276. https://eprint.iacr.org/2022/276. 2022. URL: https://eprint.iacr.org/2022/276.

[Sae17] M. A. Saeed. "Algebraic Approach for Code Equivalence". In: 2017.

[Sch90] C.-P. Schnorr. "Efficient identification and signatures for smart cards". In: *Advances in Cryptology—CRYPTO'89 Proceedings 9*. Springer. 1990, pp. 239–252.

[Sch91] C.-P. Schnorr. "Efficient signature generation by smart cards". In: *Journal of cryptology* 4 (1991), pp. 161–174.

[Sen97] N. Sendrier. "On the dimension of the hull". In: *SIAM Journal on Discrete Mathematics* 10.2 (1997), pp. 282–293.

[Sen99] N. Sendrier. "The Support Splitting Algorithm". In: 1999.

[Sha79] A. Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://doi.org/10.1145/359168.359176.

[Sha92] A. Shamir. "Ip= pspace". In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 869–877.

[Sho00] V. Shoup. "Practical threshold signatures". In: *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*. Springer. 2000, pp. 207–220.

[Sho94] P. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[Sil09] J. H. Silverman. *The arithmetic of elliptic curves*. Vol. 106. Springer, 2009.

[SK11] D. Silva and F. R. Kschischang. "Universal secure network coding via rank-metric codes". In: *IEEE Transactions on Information Theory* 57.2 (2011), pp. 1124–1135.

[SS13] N. Sendrier and D. E. Simos. "How easy is code equivalence over Fq?" In: *International Workshop on Coding and Cryptography-WCC 2013*. 2013.

[Ste94] J. Stern. "Designing identification schemes with keys of short size". In: *Annual International Cryptology Conference*. Springer. 1994, pp. 164–173.

[Sto12] A. Stolbunov. "Cryptographic schemes based on isogenies". In: (2012).

[Unr12] D. Unruh. "Quantum proofs of knowledge". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2012, pp. 135–152.

[Unr16] D. Unruh. "Computationally binding quantum commitments". In: *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*. Springer. 2016, pp. 497–527.

[Unr17] D. Unruh. "Post-quantum security of Fiat-Shamir". In: *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer. 2017, pp. 65–95.

[Vél71] J. Vélu. "Isogénies entre courbes elliptiques". In: *Comptes Rendus de l'Académie des Sciences de Paris*. A et B, Sciences mathématiques et Sciences physiques 273 (July 1971), pp. 238–241. URL: https://gallica.bnf.fr/ark:/12148/bpt6k56191248/f52.item.

[Vér97] P. Véron. "Improved identification schemes based on error-correcting codes". In: *Applicable Algebra in Engineering, Communication and Computing* 8 (1997), pp. 57–69.