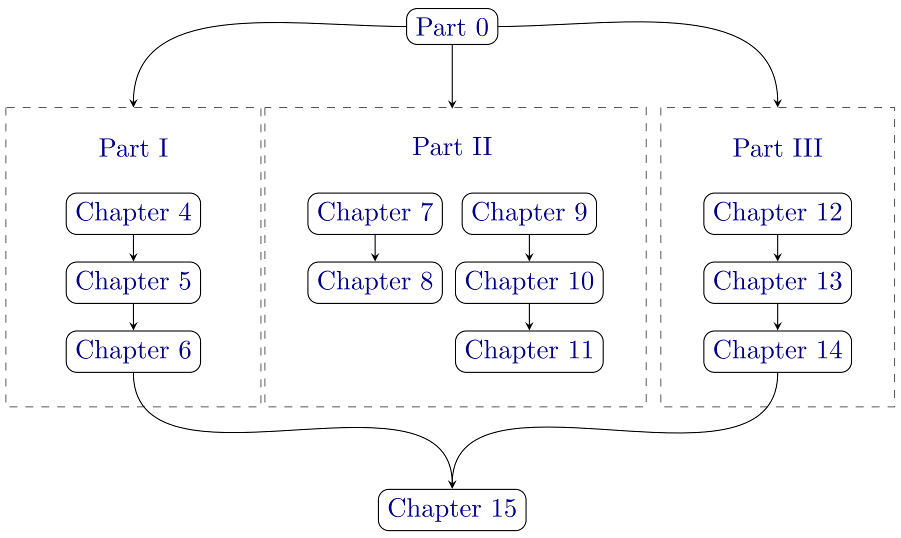
isogenies & isometries

Krijn Reijnders

Contents

Introduction

This thesis will be super cool.



Organizational chart of the components of the thesis.

Part 0

Background

Background

This thesis concerns itself with cryptography. In general, modern cryptography relies on the hardness of mathematical problems to ensure its security and fields within cryptography are often named after the underlying mathematical structures on which such problems are based. *Post-quantum cryptography*, which takes into account an adversary with access to a quantum computer, usually relies on one of five mathematical or cryptographical structures: lattices, codes, isogenies, hash functions and multivariate systems. This thesis concerns itself with code-based and isogeny-based cryptography, and, in particular, has a focus on code-based cryptography using *isometries*. There is some overlap in theory between *isometries* and *isogenies*: both are maps that preserve the essential structure between their domains and codomains, and both such maps should be hard to derive given their domains and codomains. This makes both types of functions suitable for cryptography. Beyond these similarities, there is little overlap in the concrete theory and cryptographical application and we must be careful not to get carried away in their semblance. Nevertheless, we will see throughout this thesis that ideas in one field may inspire the other, and vice versa.

We start with an outline of general constructions in public key cryptography in Chapter 1. We continue with the theoretical background on codes and their isometries (Chapter 2) and curves and their isogenies (Chapter 3).

Chapter 1

General Cryptography

1.1 Public Key Cryptography

We start by outlining the general concepts of public key cryptography that will be used throughout the work such as cryptographic primitives and security notions. A more thorough introduction is the textbook by Galbraith [87], which inspired this section. Although cryptography has many spectacular and 'fancy' applications, we restrict ourselves to three basic constructions in public key cryptography:

1. *non-interactive key exchanges* (NIKEs) which exchange a key between two parties without any interaction between these parties,
2. *key-encapsulation mechanisms* (KEMs), which simply allow two parties to agree on a shared value (a key), but require interaction, and
3. *digital signatures*, which allow anyone with access to some public key to verify that a message or piece of data was signed by the owner of the associated private key.

These three core primitives are crucial building blocks for an extensive list of advanced primitives, protocols and applications, and are used daily in any digital devices connected to other devices. Hence, improving the cryptanalysis or performance of particular NIKEs, KEMs or signatures is essential to ensure safety, security and speed in our day-to-day lives. As we will see later, code-based and isogeny-based KEMs and signatures both face very different challenges in terms of practicality.

1.1.1 Non-interactive key exchange

Informally, a non-interactive key exchange allows two parties A and B to agree on some shared value K, with A only needing to know some public key pk_B of B and vice versa. Any public key pk, known to all, is associated with a secret key

sk, known to only one party. Such a pair $(\mathsf{sk}, \mathsf{pk})$ is generated by an algorithm known as Keygen which takes as input the security parameter λ. Deriving the shared value K requires an algorithm Derive, which returns the same value for both parties. Altogether, we get the following definition.

Definition 1.1. A *non-interactive key exchange (NIKE)* is a collection of two algorithms Keygen and Derive, such that

- Keygen is a probabilistic algorithm that on input 1^λ, with λ the security parameter, outputs a keypair $(\mathsf{sk}, \mathsf{pk})$, and
- Derive is a deterministic algorithm that on input a public key pk and a secret key sk outputs a key K.

A NIKE is *correct* if for every two pairs $(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A}) \leftarrow \mathsf{Keygen}(1^\lambda)$ and $(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B}) \leftarrow \mathsf{Keygen}(1^\lambda)$ it holds that $\mathsf{Derive}(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}) = \mathsf{Derive}(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{A})$.

Note that, given the public information pk of some party x, any party y with private key sk' can derive $\mathsf{K} = \mathsf{Derive}(\mathsf{sk}', \mathsf{pk})$ without any interaction with x.

Pre-quantum key exchange. The quintessential example of a NIKE is the *Diffie-Hellman-Merkle key exchange* [63, 115], usually given for finite fields or elliptic curves, which is in general applicable to any cyclic group G. As a scheme parameter, it requires a generator g of order q for G. It relies on the core equality

$$(g^a)^b = g^{ab} = (g^b)^a$$

for positive integers a and b, which allows two parties A and B to compute g^{ab}, where A only needs to know her (secret) value $\mathsf{sk}_\mathsf{A} := a$ and the public value $\mathsf{pk}_\mathsf{B} := g^b$, and similarly, B can compute g^{ab} given $\mathsf{pk}_\mathsf{A} := g^a$ and $\mathsf{sk}_\mathsf{B} := b$. In this example, Keygen simply samples a random positive integer $a \in \mathbb{Z}_q$ and returns the pair (a, g^a), whereas Derive computes g^{ab} given g^b and a.

It is easy to see that this cryptographic scheme would be completely broken if one could derive either a from g and g^a or g^{ab} from g, g^a and g^b, as this would allow an adversary without any of the required secret knowledge to derive the secret value K that should only be accessible to A and B. The security of this scheme therefore relies on two *hardness assumptions*, or equivalently, the difficulty of two mathematical problems, known as the *discrete logarithm problem* and the *computational Diffie-Hellman problem.*

Problem 1.1 (Discrete logarithm problem)**.** Let G be a group. The *discrete logarithm problem* is:

$$\text{Given } g, h \in G, \quad \text{find } a \in \mathbb{N} \text{ such that } h = g^a.$$

For Diffie-Hellman-Merkle key exchange, we usually require a (sub)group G of prime order q, and have to choose this group G and the generator g specifically so that Problem 1.1 is cryptographically hard, i.e. there exist no polynomial time algorithms to find a given g and h.

Problem 1.2 (Computational Diffie-Hellman problem)**.** Let G be a group. The *computational Diffie-Hellman problem* is:

$$\text{Given } g, g^a, g^b \in G, \quad \text{compute } g^{ab}.$$

Sometimes, the decisional variant is required: in this case, one is given g, g^a, g^b and some z, which is either a random value g^c or the value g^{ab}. The goal is then to decide if z is random or $z = g^{ab}$.

If the group G is selected carefully, for example as the group of points on a specific elliptic curve E over a finite field $\mathbb{F}_q$, this problem remains secure for classical adversaries. Unfortunately, Shor [149] shows that an adversary with access to a quantum computer can break both problems in polynomial time.

Post-quantum key exchange. In post-quantum cryptography, we require our problems to remain cryptographically hard, even with access to a quantum computer. As of the moment of writing, there are two classes of post-quantum NIKEs available:

1. CSIDH [46], and generalizations [77], which is the subject of Part II of thesis, and

2. Swoosh [84], a lattice-based key exchange originally considered "folklore", with a first proper attempt given by Kock [104].

Both NIKEs are, at the moment of writing, practically computable, yet far from practical. The performance and security of CSIDH will be analyzed in Part II, Swoosh suffers from large public keys.

1.1.2 Key-encapsulation mechanisms

A key-encapsulation mechanism achieves a similar goal to a NIKE, sharing some key between two parties, but requires interaction. More precisely, in key-encapsulation we again generate a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Keygen}(1^\lambda)$ for each party, but the public key pk is now used to encapsulate some key K in a ciphertext ct using an algorithm Encaps. The associated sk is required to compute K from ct. Thus, any party that knows pk can get a K from Encaps (and keeps it secret) and knows that only the owner of sk can compute the same K too, using Decaps. Altogether, we get the following definition.

Definition 1.2. A *key-encapsulation mechanism (KEM)* is a collection of three algorithms Keygen, Encaps, and Decaps, such that

- Keygen is a probabilistic algorithm that on input 1^λ, with λ the security parameter, returns a keypair $(\mathsf{sk}, \mathsf{pk})$, and

- Encaps is a probabilistic algorithm that on input a public key pk returns returns a ciphertext ct and a key K, and

- Decaps is a deterministic algorithm that on input a secret key sk and a ciphertext ct returns a key K.

A KEM is *correct* if for every pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Keygen}(1^\lambda)$ and $(\mathsf{ct}, \mathsf{K}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$, it holds that $\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}) = \mathsf{K}$.

In 2017, to find replacements for pre-quantum key exchange mechanisms, the National Institute of Standards and Technology (NIST) launched a standardization effort for post-quantum KEMs, which resulted in the selection of Kyber [33] in 2022.

Difference with NIKE. Although both NIKEs and KEMs are used to exchange a key K, we can be more flexible with a NIKE as it is non-interactive. Pre-quantum cryptography therefore uses the Diffie-Hellman-Merkle NIKE as a building block for an incredibly wide range of applications, from XXX to XXX. In a post-quantum setting, the limitations of current NIKEs prevent them from being used in most settings. In general, the usage of NIKEs and KEMs in protocols falls into three categories:

1. Some protocols require an interaction by nature, and therefore use a KEM. The usage of a NIKE in such a situation is possible but does not offer any additional benefit. A traditional example is TLS, where the current usage of Diffie-Hellman-Merkle can migrate to post-quantum KEMs [34, 146].

2. Some protocols require a NIKE by nature, which cannot be replaced by a KEM. An example is Signal's X3DH protocol [112], which still needs to work when participants are offline and can therefore not interact.

3. Some protocols can use KEMs but at some additional cost, usually an extra round of communication. In such instances, the comparison between post-quantum NIKEs or KEMs is especially interesting. Chapter 10 discusses this topic in more detail.

1.1.3 Digital signatures

A digital signature allows some party A to compute a signature σ for some piece of data, usually a message msg, using its secret key sk with an algorithm called Sign. Anyone with the associated public key pk can verify this signature, which assures that A computed σ for msg, with an algorithm called Verif. A successful verification is usually interpreted as A authenticating msg and provides integrity of the content of msg. Altogether, we get the following definition.

Definition 1.3. A *signature scheme* is a collection of three algorithms Keygen, Sign, and Verif, such that

- Keygen is a probabilistic algorithm that on input 1^λ, with λ the security parameter, returns a keypair $(\mathsf{sk}, \mathsf{pk})$, and

- Sign is a probabilistic algorithm that on input a message msg and a secret key sk returns a signature σ, and
- Verif is a deterministic algorithm that on input a public key pk, a message msg and a signature σ returns either "valid" or "invalid".

A signature scheme is *correct* if for every pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Keygen}(1^\lambda)$ and signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, \mathsf{sk})$, it holds that $\mathsf{Verif}(\mathsf{msg}, \sigma, \mathsf{pk})$ returns "valid".

Pre-quantum signatures. The most elegant yet simple and natural example of pre-quantum signatures is the *Schnorr signature scheme* [145], which is based on the same discrete logarithm problem (Problem 1.1) as Diffie-Hellman-Merkle key exchange. This starts from the following *interactive* protocol: Assuming a finite cyclic group G with generator g of order q in which the discrete logarithm problem is hard, set the secret key of A to a random $\mathsf{sk_A} \leftarrow a \in \mathbb{Z}_q^*$ and the public key as $\mathsf{pk_A} \leftarrow g^a$. To convince B that she knows $\mathsf{sk_A}$, A takes another random $r \leftarrow \mathbb{Z}_q^*$ and sends $R \leftarrow g^r$ to B (the *commitment*). As a *challenge*, B sends back a value $c \in \mathbb{Z}_q$. As a *response*, A computes $s \leftarrow r - c \cdot \mathsf{sk_A} \mod q$ and sends this back to B. The idea is that s can only be computed correctly if one knows $\mathsf{sk_A}$, and so, by confirming that the commitment R equals $g^s \cdot \mathsf{pk_A}^c$, party B is convinced that A really does know $\mathsf{sk_A}$.

To turn the above identification protocol into a non-interactive signature scheme, all users agree on a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$, and we replace the challenge $c \leftarrow \mathbb{Z}_q$ by the hash of R (as a bit string) concatenated with msg to get $c \leftarrow H(R || \mathsf{msg})$. The resulting $s \leftarrow r - c \cdot \mathsf{sk_A} \mod q$ together with c becomes the signature $\sigma \leftarrow (s, c)$. To verify, we recompute $R \leftarrow g^s \cdot \mathsf{pk_A}^c$ and ensure that $H(R || \mathsf{msg})$ equals c.

Sigma protocols and the Fiat-Shamir heuristic. The method of the previous example to turn an identification protocol into a signature scheme is a well-known method named the *Fiat-Shamir heuristic* [79] and applies to a broad range of identification protocols. In this thesis, we restrict ourselves to Sigma protocols [54] which are identification protocols based on the commitment-challenge-response structure we saw in the Schnorr identification protocol. Such Sigma protocols assume two parties: a "prover" A and a "verifier" B, with A proving knowledge of some public statement x tied to some secret knowledge w, often called the *witness*. For example, in Scnorr's identification protocol, A convinces B that she knows the witness $w = \mathsf{sk_A}$ such that $x = \mathsf{pk_A} = g^{\mathsf{sk_A}}$. We use the following definition for Sigma protocols.

Definition 1.4. A *Sigma protocol* is a collection of two prover algorithms, the commitment algorithm P_{cmt} and the response algorithm P_{resp}, and a verification algorithm V_{verif}, with the following flow. To prove knowledge of a witness w for the statement x,

1. A computes a commitment $\mathsf{cmt} \leftarrow P_{\mathsf{cmt}}$ and sends it to B,

2. B samples $\mathsf{chal} \leftarrow S$ uniformly random from a predefined set S, and sends chal to A,

3. A computes $\mathsf{resp} \leftarrow P_{\mathsf{resp}}(x, w, \mathsf{cmt}, \mathsf{chal})$ and sends resp to B,

4. B computes $V_{\mathsf{verif}}(x, \mathsf{cmt}, \mathsf{chal}, \mathsf{resp})$ and outputs **pass** or **fail**.

If the *transcript* $(\mathsf{cmt}, \mathsf{chal}, \mathsf{resp})$ verifies x, i.e. $V_{\mathsf{verif}}(x, \mathsf{cmt}, \mathsf{chal}, \mathsf{resp})$ passes, we call the transcript *valid* for x.

A Sigma protocol will only be useful if a prover A can actually convince B, that is, if only A who knows a witness w for x can generate a valid transcript for x, and furthermore that such a transcript can always be verified by B. These are distinct properties of a Sigma protocol, and we list the three most important here. We provide an intuitive explanation, formal definitions are abundant in the literature [**xxx**]

- A Sigma protocol is **complete** if a verifier that actually knows the secret w for x will be able to generate a valid transcript. More precisely, any honestly generated transcript $(\mathsf{cmt}, \mathsf{chal}, \mathsf{resp})$ for x is valid: $V_{\mathsf{verif}}(x, \mathsf{cmt}, \mathsf{chal}, \mathsf{resp})$ passes.

- A Sigma protocol is **honest-verifier zero-knowledge** if the verifier B gains no knowledge on the secret w from the transcript $(\mathsf{cmt}, \mathsf{chal}, \mathsf{resp})$. Zero-knowledge comes in different gradations:

  - the zero-knowledge is *perfect* if it is mathematically impossible to get information from the transcript,
  - the zero-knowledge is *statistical* if the distribution of valid transcripts is indistinguishable from a random transcript, and
  - the zero-knowledge is *computational* if differentiating between a valid transcript and a random transcript is a computationally hard problem.

- A Sigma protocol is **special sound** if, given two transcripts for the same cmt but different chal, it is possible to derive w in polynomial time. This implies that someone without knowledge of w can only convince the verifier with probability $1/|S|$, essentially by guessing the right response resp.

TODO: finish up this section bla bla

1.2 Cryptographic group actions

Cryptographic group actions are a useful framework to unify several different approaches to designing signature schemes[1]. After having defined a single group action, we can design advanced protocols in terms of this group action, whose

[1]This section is a summary of the results by Borin, Persichetti, Santini Pintore and me [32].

security reduces to the hardness of recovering the group element that acted. This has led to an abundance of schemes, one of which is the main topic of Part I.

We first introduce the general terminology around group actions, before we summarize several techniques used to amplify the performance of digital signatures based on group actions. We furthermore present a table summarizing possible combinations of these techniques together with their impact on performance.

1.2.1 Properties of cryptographic group actions

Let G be a group and X a set. Denote by $\star$ a group action of G on X, that is, a function

$$\star : G \times X \to X, \quad (g, x) \mapsto g \star x,$$

which behaves well with regard to the group structure: $e \star x = x$ for all $x \in X$, with e the neutral element of G, and $g_1 \star (g_2 \star x) = (g_1 \cdot g_2) \star x$ for all $x \in X$ and all $g_1, g_2 \in G$.

Several properties of group actions are particularly relevant when used in a cryptographic context.

Definition 1.5. Let G act on X. The group action is said to

- *transitive*, if for every $x, y \in X$ there is a $g \in G$ such that $y = g \star x$,
- *faithful*, if no $g \in G$, except the neutral element, acts trivially on X, i.e. $g \star x = x$ for all $x \in X$,
- *free*, if no $g \in G$, except the neutral element, acts trivially on any element of X, i.e. $g \star x = x$ for one $x \in X$ implies $g = e$,
- *regular*, if the group action is both transitive and free, which implies there is a unique $g \in G$ for any two $x, y \in X$, such that $y = g \star x$.

Beyond these mathematical properties, we require several cryptographic properties for a group action to be useful in cryptography.

Definition 1.6. Let G act on X. The group action is *cryptographic* if

- *vectorisation* is hard, that is, given $y = g \star x$ and x, find g
- *parallellisation* is hard, that is, given x, $g_1 \star x$ and $g_2 \star x$, find $(g_1 \cdot g_2) \star x$.

Vectorisation is sometimes called the *Group Action Inverse Problem*, or GAIP, and parallellisation is sometimes called the *computational Group Action Diffie-Hellman Problem*, as it asks to find the bottom right corner of a Diffie-Hellman protocol. A *cryptographic group action* is secure to use for protocol design, but may still not be very useful. This depends on yet a third set of properties, defined in [3], adapted here to our context.

Definition 1.7. Let G act on X, and let both be finite. The group action is *efficient* if the following procedures have efficient (probabilistic polynomial time) algorithms. For the group G,

- *membership testing*, that is, to decide if some element is a valid element of G,
- *equality testing*, that is, to decide if two elements represent the same element of G,
- *sampling*, that is, to sample (statistically close) to uniformly random from G,
- *multiplication*, that is, to compute $g_1 \cdot g_2$ given any $g_1, g_2 \in G$,
- *inversion*, that is, to compute g^{-1} given any $g \in G$.

For the set X,

- *membership testing*, that is, to decide if some element is a valid element of X,
- *uniqueness of representation*, that is, to give a unique representation x for any arbitrary $x \in X$.

Furthermore, most crucially, *evaluation* must be efficient, that is, to compute $g \star x$ given any $g \in G$ and any $x \in X$.

Beyond these properties, *commutativity* of the group action is especially nice to have. As we will see in Parts I and II, whereas general cryptographic group actions allow constructions of digital signatures, commutative cryptographic group actions furthermore allow the construction of a NIKE. However, as of the moment of writing, CSIDH is the only known post-quantum commutative cryptographic group action.

1.2.2 Sigma protocols from cryptographic group actions

Given a cryptographic group action, it is easy to define a Sigma protocol and thus a digital signature scheme, after applying the Fiat-Shamir heuristic. The central idea is to prove knowledge of a secret element $g \in G$, by giving as a public key a pair $(x, y) \in X \times X$ with $y = g \star x$. Using a random element $\tilde{g} \in G$, party A can commit to $\tilde{x} = \tilde{g} \star x$ and only A can compute both paths $x \xrightarrow{\tilde{g}} \tilde{x}$ or $y \xrightarrow{\tilde{g} \cdot g^{-1}} \tilde{x}$. Thus, repeating such a *round*, as visualised in Figure 1.1, $t = \lambda$ times, where party B may each time ask for either of these paths, should convince B that A knows g.

For a λ-bit security level, assume we repeat this diagram t times. We denote by $\tilde{g}_i$ the uniformly random group element in round i, and by $\tilde{x}_i$ the commitment in round i. The challenge space of this Sigma protocol then becomes strings

$$\begin{tikzcd} x \arrow[rr, "g"] \arrow[dr, dashed, "\tilde{g}"'] & & y \arrow[dl, dashed, "\tilde{g} \cdot g^{-1}"] \\ & \tilde{x} & \end{tikzcd}$$

(a) Commitment generation.

$$\begin{tikzcd} x \arrow[rr, "g"] \arrow[dr, "\tilde{g}"'] & & y \arrow[dl, dashed] \\ & \tilde{x} & \end{tikzcd}$$

(b) $\mathsf{ch} = 0$, we reveal $\tilde{g}$.

$$\begin{tikzcd} x \arrow[rr, "g"] \arrow[dr, dashed] & & y \arrow[dl, "\tilde{g} \cdot g^{-1}"] \\ & \tilde{x} & \end{tikzcd}$$

(c) $\mathsf{ch} = 1$, we reveal $\tilde{g} \cdot g^{-1}$.

Figure 1.1: Simple one-round Sigma protocol based on group actions.

$\mathsf{ch} \in \{0,1\}^t$, where each ch_i asks to reveal either $x \to \tilde{x}_i$ or $y \to \tilde{x}_i$. The response then consists of each $\tilde{g}_i$, resp. $\tilde{g}_i \cdot g^{-1}$ for $1 \leq i \leq t$.

One can easily show that this very basic Sigma protocol is complete, honest-verifier zero-knowledge, and 2-special sound, hence, the security reduces to the vectorisation problem with x and $y = g \star x$.

TODO: the remainder of this section becomes the separate SOK paper in the last part of the thesis

Chapter 2

Codes & Isometries

Part I of this thesis concerns itself with code-based cryptography: our fundamental hardness assumption is a presumably difficult problem in coding theory. Contrary to classical code-based cryptography, the underlying hardness assumption on which much of the work in Part I is based is not syndrome decoding, but code equivalence. We first describe codes, and then their isometries, to give an overview of the ingredients required for Part I.

2.1 Codes

This thesis only concerns itself with *linear codes*, e.g. subspaces $\mathcal{C}$ of a vector space V over a finite field $\mathbb{F}_q$, equipped with some metric d. Let n denote the dimension of V, and k the dimension of $\mathcal{C}$, then such codes are called $[n, k]$-codes. Most well-known are *Hamming metric* codes, where $V = \mathbb{F}_q^n$ and $d(v, v')$ simply counts the number of different coefficients $v_i \neq v'_i$.

This thesis, however, focuses on a different class of codes, namely matrix codes: subspaces $\mathcal{C}$ of the vector space $V = \mathbb{F}_q^{n \times m}$ of matrices over a finite field. A more thorough introduction on matrix codes is given by Gorla [90]. The usual choice for measuring distance between matrices over a finite field is the so-called *rank metric*, defined as follows.

Definition 2.1. Let $\mathrm{Rank}(\mathbf{M})$ denote the rank of a matrix $\mathbf{M} \in \mathbb{F}_q^{n \times m}$. The *rank distance* between two $m \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$ over $\mathbb{F}_q$ is defined as

$$d(\mathbf{A}, \mathbf{B}) = \mathrm{Rank}(\mathbf{A} - \mathbf{B}).$$

By symmetry, without loss of generality, in the rest of the works on matrix codes, we assume $n \geqslant m$.

Definition 2.2. A *matrix code* is a subspace $\mathcal{C}$ of $m \times n$ matrices over $\mathbb{F}_q$ endowed with the rank metric. We let k denote the dimension of $\mathcal{C}$ as a subspace of $\mathbb{F}_q^{m \times n}$ and we denote the basis by $\langle \mathbf{C}_1, \ldots, \mathbf{C}_k \rangle$, with $\mathbf{C}_i \in \mathbb{F}_q^{m \times n}$ linearly independent.

TODO: write a bit on vector rank codes versus matrix codes?
TODO: write a bit on the vectorization construction

2.2 Isometries

Definition 2.3. Let V be a linear space equipped with a metric d. An *isometry* is a map $\mu : V \to V$ that preserves the rank,

$$\mu(d(v)) = d(\mu(v)), \quad \text{for all } v \in V.$$

An isometry of a code $\mathcal{C} \subset V$ preserves the rank of the codewords

$$\mu(d(c)) = d(\mu(c)), \quad \text{for all } c \in \mathcal{C}.$$

It is then natural to ask if isometries of codes $\mu : \mathcal{C} \to \mathcal{C}$ extend to the space V they are contained in. For the Hamming metric, the result is satisfactory: The MacWilliams Extension Theorem [110] shows that any isometry μ of a linear code $\mathcal{C}$ can be extended to an isometry $\mu' : V \to V$ such that $\mu'_{|\mathcal{C}} = \mu$.

For the rank metric, this no longer holds. It is not difficult to construct counterexamples of isometries that do not extend, and recent research has explored the question to what extend such a theorem may hold for the rank metric [91]. In light of such obstructions, we have to admit defeat and only consider isometries of the full matrix space $\mathbb{F}_q^{n\times m}$ in this thesis. Luckily, such isometries are easy to categorize.

Lemma 2.4. Let $\mu : \mathbb{F}_q^{n\times m} \to \mathbb{F}_q^{n\times m}$ be an isometry with respect to the rank metric. If $n \neq m$, there exist matrices $\mathbf{A} \in \mathrm{GL}_n(q)$, $\mathbf{B} \in GLmq$ such that

$$\mu(\mathbf{M}) = \mathbf{AMB},$$

for all $\mathbf{M} \in \mathbb{F}_q^{n\times m}$. When $n = m$, we may additionally transpose $\mathbf{M}$, i.e.

$$\mu(\mathbf{M}) = \mathbf{AM}^T\mathbf{B}.$$

Definition 2.5. Two codes $\mathcal{C}$, $\mathcal{D}$ are *equivalent* if there exists an isometry $\mu : \mathcal{C} \to \mathcal{D}$.

In light of Lemma 2.4, we therefore say that two rank-metric codes $\mathcal{C}, \mathcal{D} \subseteq \mathbb{F}_q^{n\times m}$ are equivalent, if there exist two matrices $\mathbf{A} \in \mathrm{GL}_n(q)$, $\mathbf{B} \in \mathrm{GL}_m(q)$ such that

$$\mathbf{C} \mapsto \mathbf{ACB}$$

is an isometry $\mathcal{C} \to \mathcal{D}$. Note that this disregards transpositioning codewords; this has no impact for the cryptographic applications in this thesis. In short, when we write about isometries in the rank metric, we assume isometries in $\mathcal{I}_{n,m}(q) := \mathrm{GL}_n(q) \rtimes \mathrm{GL}_m(q)$. For any scalars $\lambda, \nu \in \mathbb{F}_q^*$, the isometry $\mu = (\mathbf{A}, \mathbf{B})$ and $\mu' = (\lambda\mathbf{A}, \nu\mathbf{B})$ are essentially equal, for all purposes. We can thus think of isometries as elements $(\mathbf{A}, \mathbf{B}) \in \mathrm{PGL}_n(q) \rtimes \mathrm{PGL}_m(q)$. This is conceptually the better interpretation, but often cumbersome. Thus, we stay with the notation $\mu = (\mathbf{A}, \mathbf{B})$ with $\mathbf{A} \in \mathrm{GL}_n(q)$, $\mathbf{B} \in \mathrm{GL}_m(q)$ and refer to the projective interpretation where needed. The set of isometries between two codes $\mathcal{C}$ and $\mathcal{D}$ is denoted $\mathcal{I}(\mathcal{C}, \mathcal{D}) \subseteq \mathcal{I}_{n,m}(q)$.

Automorphisms of matrix codes.

Definition 2.6. An *automorphism* of a code $\mathcal{C}$ is an isometry $\mu : \mathcal{C} \to \mathcal{C}$. The full group of automorphisms is denoted $\mathrm{Aut}^{\bullet}(\mathcal{C})$. We denote the subgroup of automorphisms for matrix codes derived from $\mu : \mathbb{F}_q^{n\times m} \to \mathbb{F}_q^{n\times m}$ by

$$\mathrm{Aut}(\mathcal{C}) = \{(\mathbf{A}, \mathbf{B}) \in \mathcal{I}_{n,m}(q) \mid \mathbf{A}\mathcal{C}\mathbf{B} = \mathcal{C}\}.$$

We write $\mathrm{Aut}_L(\mathcal{C})$, resp. $\mathrm{Aut}_R(\mathcal{C})$ to denote the subgroups of $\mathrm{Aut}(\mathcal{C})$ where $\mathbf{B} = \lambda\mathbf{I}_m$, resp. $\mathbf{A} = \lambda\mathbf{I}_n$.

Necessarily, $(\lambda\mathbf{I}_n, \nu\mathbf{I}_m) \in \mathrm{Aut}(\mathcal{C})$ for any code $\mathcal{C}$ and any scalars $\lambda, \nu \in \mathbb{F}_q^*$. If the automorphism groups contains no other elements than these, we say the automorphism group is *trivial*[1].

Both $\mathrm{Aut}_L(\mathcal{C})$ and $\mathrm{Aut}_R(\mathcal{C})$ are normal subgroups of $\mathrm{Aut}(\mathcal{C})$. We can thus define the group $\mathrm{Aut}_{LR}(\mathcal{C}) := \mathrm{Aut}(\mathcal{C}) / \mathrm{Aut}_L(\mathcal{C}) \times \mathrm{Aut}_R(\mathcal{C})$, which measures the number of distinct non-trivial two-sided automorphisms a code has.

Lemma 2.7. Let $\mathcal{C}$ and $\mathcal{D}$ be equivalent codes. Then, for any $\mu : \mathcal{C} \to \mathcal{D}$,

$$\mu \cdot \mathrm{Aut}(\mathcal{C}) = \mathcal{I}(\mathcal{C}, \mathcal{D}) = \mathrm{Aut}(\mathcal{D}) \cdot \mu.$$

Proof. As $\mathcal{C}$ and $\mathcal{D}$ are equivalent, there is some $\mu \in \mathcal{I}(\mathcal{C}, \mathcal{D})$. For any $\mu' \in \mathrm{Aut}(\mathcal{C})$, it is clear that $\mu \cdot \mu' \in \mathcal{I}(\mathcal{C}, \mathcal{D})$. For any other $\mu' \in \mathcal{I}(\mathcal{C}, \mathcal{D})$, we get $\mu' = \mu \cdot \mu^{-1} \cdot \mu'$ with $\mu^{-1} \cdot \mu' \in \mathrm{Aut}(\mathcal{C})$. $\square$

As a consquence, for any two codes $\mathcal{C}$ and $\mathcal{D}$, if $\mathcal{I}(\mathcal{C}, \mathcal{D})$ is non-empty, this set is as large as $\mathrm{Aut}(\mathcal{C})$, which is therefore as large as $\mathrm{Aut}(\mathcal{D})$.

Finally, we discuss a central hardness problem for Part I of this thesis, which studies cryptography based on isometries.

Problem 2.1 (Matrix Code Equivalence)**.** Given two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subseteq \mathbb{F}_q^{n\times m}$, find, if any, a map $(\mathbf{A}, \mathbf{B}) \in \mathcal{I}_{n,m}(q)$ such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mathbf{ACB} \in \mathcal{D}$.

We often refer to Matrix Code Equivalence by MCE. Its hardness has been studied previously [21, 53], and we extend this research in Chapter 4.

Group action from isometries.

The group of isometries $\mathcal{I}_{n,m}(q)$ acts on the set X of k-dimensional matrix codes by simply mapping any code $\mathcal{C} \in X$ to $\mathbf{A}\mathcal{C}\mathbf{B} \in X$ for an isometry $(\mathbf{A}, Bb)$. By abuse of notation, we refer to this group action also by MCE, which is justified as vectorisation for this group action of $\mathcal{I}_{n,m}(q)$ on X is precisely MCE.

With respect to the properties given in Section 1.2.1, we find that MCE is neither commutative, nor transitive, nor free. However, MCE is *efficient*, and so it can be used to construct sigma protocols, as is studied in Chapter 5.

[1]From the projective perspective, this is the trivial subgroup of $\mathrm{PGL}_n(q) \rtimes \mathrm{PGL}_m(q)$.

The orbits of X under this group action are the classes of equivalent codes, and the stabilizer group of a code $\mathcal{C}$ is the automorphism group $\operatorname{Aut}(\mathcal{C})$. By the orbit-stabilizer theorem, we find the orbit of $\mathcal{C}$ by the action of $\mathcal{I}_{n,m}(q)/\operatorname{Aut}(\mathcal{C})$. For a cryptographic group action, we want the largest orbit and thus a trivial automorphism group. By restricting X to such an orbit, the restriced group action becomes transitive and free. Finding the automorphism group of a code is a difficult problem, and it seems infeasible to determine $\operatorname{Aut}(\mathcal{C})$ in polynomial time, given $\mathcal{C}$. It is however folklore that random codes have trivial automorphism groups, which we study in Chapter 6.

Chapter 3

Curves & Isogenies

Parts II and III of this thesis concern themselves with two subareas of isogeny-based cryptography, i.e. cryptography based on isogenies between abelian varieties. Luckily, most of this thesis concerns itself with isogenies between elliptic curves, which are well-known objects to any cryptographer, and a small part on (isogenies between) hyperelliptic curves of genus 2. This chapter introduces the main players: curves, their isogenies and pairings on curves. We assume basic familiarity with elliptic curves, such as an understanding of Silverman [151] or Galbraith [87], and a basic understanding of quaternion algebras, for which Voight [159] is our main reference.

3.1 Curves

All curves in this thesis are smooth, projective varieties of dimension 1.

Definition 3.1. An *elliptic curve* $(E, \mathcal{O})$ is a smooth projective curve E of genus 1 together with a rational point $\mathcal{O} \in E$. An elliptic curve is defined over k, whenever E is defined over k as a curve, and $\mathcal{O} \in E(k)$.

Elliptic curves are pleasant to work with: we do not have to consider the more abstract algebraic geometry, discussing polarizations, duals and Jacobians, as each elliptic curve comes with a canonical principal polarization, and $E(k)$ is furthermore isomorphic to its Jacobian, ensuring a group structure on this set of points. Even more pleasantly, the fields we are working over never have characteristic 2 or 3, which ensures that every elliptic curve has an isomorphic curve E in *short Weierstrass form* defined by the affine equation

$$E : y^2 = x^3 + ax + b, \qquad a, b \in k,$$

with the set of points $E(k)$ precisely those points $P = (x, y)$ with $x, y \in k$ such that the equation holds. Such a curve is smooth, i.e. non-singular whenever $\Delta(E) = -16(4a^3 + 27b^2)$ is non-zero. Although E is given in affine form, we assume the reader understands that, properly speaking, we imply the projective closure with $\mathcal{O} = (0 : 1 : 0)$ the *point at infinity* not corresponding to any affine point.

In practice, isogeny-based cryptography uses different curve models too, as the cost of crucial operations highly depends on the choice of model. A popular model was given by Montgomery [118].

Definition 3.2. A *Montgomery curve* E over k of characteristic $\operatorname{char} k > 2$ is an elliptic curve with affine equation

$$E : By^2 = x^3 + Ax^2 + x, \qquad A, B \in k$$

with $\Delta(E) = B(A^2 - 4)$ non-zero.

Whenever we work with Montgomery curves in this thesis, we can choose $B = 1$, and we refer to $A \in k$ as the *Montgomery coefficient* of the curve. This reduces smoothness to $A \neq \pm 2$. Furthermore, in such cases, or whenever it is clear, E_A refers to the Montgomery curve with Montgomery coefficient A. In particular, E_0 refers to the Montgomery curve with the affine equation

$$E_0 : y^2 = x^3 + x.$$

This curve has many 'nice' properties, which make E_0 appear frequently in isogeny-based cryptography as a system parameter.

Curve arithmetic using only the x-coordinate.

Even the very first articles introducing elliptic curves in cryptography [102, 117], Miller and Koblitz independently noted that one could perform most operations required for elliptic curve cryptography using only the x-coordinate of points $P \in E(k)$. In particular for curves in Montgomery form, such x-only approaches are significantly faster. This holds not only for classical ECC, but also for current-day isogeny-based cryptography.

As the x-coordinate of a point is equal for both P and $-P$, mathematically working with x-only arithmetic means an identification of these two. That is, we work on E with equivalence between P and $-P$. This turns out to be an algebraic variety $\mathcal{K}_E = E/\langle \pm 1 \rangle$ isomorphic to the projective line $\mathbb{P}^1$, named the *Kummer line* of E. The Kummer line $\mathcal{K}_E$ does not inherit the group structure of E due to the identification of P and $-P$. Nevertheless, $\mathcal{K}_E$ is a *pseudo-group*, which means we can still perform scalar multiplication and differential addition. This turns out to be enough for most cryptographic applications, such as a Diffie-Hellman key exchange based on the group $E(k)$, as it only requires the computation of $P \mapsto [n]P$ for $n \in \mathbb{Z}$.

3.1.1 Hyperelliptic curves

Beyond elliptic curves, we require an understanding of hyperelliptic curves. Such objects are well-studied in cryptography for their use in hyperelliptic curve cryptography (HECC) since Koblitz [103]. Hyperelliptic curves are curves of genus $g > 1$ with a ramified double cover of the projective line. As we assume we are working over fields with $\operatorname{char} k \neq 2$, such hyperelliptic curves are isomorphic to curves with a nice affine model.

Lemma 3.3. Any hyperelliptic curve over k, with $\operatorname{char} k \neq 2$, is isomorphic over k to a curve $\mathcal{C}$ given by the affine model

$$\mathcal{C} : y^2 = f(x), \qquad f(x) \in k[x]$$

with $\deg f = 2g + 1$ or $\deg f = 2g + 2$. The curve is smooth if $\gcd(f, f') = 0$, e.g. f has no repeated roots.

The polynomial $f(x)$ may have roots $w_i \in k$, which implies points $(w_i, 0) \in \mathcal{C}(k)$. These points are the *Weierstrass points* of $\mathcal{C}$. In cryptographic applications, we again often require a special curve model suitable for our purposes. In the case of genus 2 hyperelliptic curves, a particularly interesting curve model is given by Rosenhain [143].

Definition 3.4. A hyperelliptic curve $\mathcal{C}$ over k of genus 2 is in *Rosenhain form* if we have

$$\mathcal{C} : y^2 = x(x-1)(x-\lambda)(x-\mu)(x-\nu), \qquad \lambda, \mu, \nu \in k.$$

The coefficients λ, μ, ν are called the *Rosenhain invariants.*

Hyperelliptic curves are isomorphic over k to a curve in Rosenhain form when $f(x)$ splits in $k[x]$, or equivalently, all Weierstrass points are rational. In this situation, by a Möbius transform κ, one can choose three Weierstrass points and map these to $\mathcal{O}$, $(0, 0)$ and $(1, 0)$. This determines an isomorphism κ, which determines $(\lambda, 0)$, $(\mu, 0)$ and $(\nu, 0)$ as the images of the other three Weierstrass points. We get a total of 720 possible choices for such a map κ, and therefore 720 possible Rosenhain forms for any hyperelliptic curve.

Unfortunately, the set of points $\mathcal{C}(k)$ no longer have a group structure, as they did in the case of elliptic curves. Fortunately, to every hyperelliptic curve we can associate an abelian variety, the *Jacobian* $\mathcal{J}_\mathcal{C}$ of $\mathcal{C}$, which does carry the structure of the group[1]. Formally, a description of the curve provides us with a canonical principal polarization, and so we can understand $\mathcal{J}_\mathcal{C}$ as the *Picard variety of degree 0* $\operatorname{Pic}^0_k(\mathcal{C})$, that is, the group of degree zero divisors quotiented out by principal divisors. For the purpose of this thesis, these formalities are of lesser importance, and we refer the reader to Cassels and Flynn [42] for an elaborate introduction. We mainly need to understand how elements of $\mathcal{J}_\mathcal{C}$ arise from points on $\mathcal{C}$, how we represent such elements, and how these form a group. We keep our introduction slightly informal, for brevity and clarity.

There is a map $\mathcal{C}^{(g)} \to \mathcal{J}_\mathcal{C}$, which maps tuples of points $P = (P_1, \dots, P_g) \in \mathcal{C}$ to divisors $D_P = (P_1) + (P_2) + \ldots + (P_g)$, up to symmetry. It is known that any element $D_P \in \mathcal{J}_\mathcal{C}$ can be associated with such a divisor with $0 \leq n \leq g$ points, which is then called *reduced.* A practical representation of such divisors is given by Mumford [120].

[1]The set of points of an elliptic curve E form a group precisely because E is isomorphic to its Jacobian $\mathcal{J}_E$.

Definition 3.5. The *Mumford representation* of an element $D_P = \sum_{i=1}^{n}(P_i)$ with $P_i = (x_i, y_i) \in \mathcal{C}(\overline{k})$ is given by a pair of polynomials $a(x), b(x) \in \overline{k}[x]$, uniquely defined by the properties

$$a(x) = \prod x - x_i, \qquad b(x_i) = y_i$$

with $\deg a = n$ and $\deg b < \deg a$. The representation is denoted as $\langle a(x), b(x)\rangle$.

It can be shown that each reduced divisor $D_P \in \mathcal{J}_{\mathcal{C}}$ has a unique Mumford representation $\langle a(x), b(x)\rangle$. Furthermore, a divisor is defined over k whenever both $a(x)$ $b(x)$ are defined over k. Note that this does not imply that each $P_i \in \mathcal{C}(k)$! For example, D_P can be rational whenever this divisor contains each Galois conjugate of $P_i \in \mathcal{C}(K)$ for some finite field extension K/k.

An efficient algorithm to add elements D_P and D_Q is given by Cantor [40], which takes the Mumford representations of D_P and D_Q and returns the Mumford representation of $D_P + D_Q$. This algorithm is a higher-dimensional generalization of the chord-tangent construction for the group operation on elliptic curves.

Kummer surfaces.

Similar to the elliptic curve situation, arithmetic on the Jacobian $\mathcal{J}_{\mathcal{C}}$ using Cantor's algorithm is rather slow and not suitable for cryptography. Where for an elliptic curve E we could simply use the x-coordinate[2] which keeps enough structure to do cryptography, for hyperelliptic curves we can similarly work on a Kummer surface $\mathcal{K}_{\mathcal{C}} = \mathcal{J}_{\mathcal{C}}/\langle\pm\rangle$, for which models exists as algebraic varieties in $\mathbb{P}^3$. Several different models of Kummer surfaces are in use in cryptography. A more detailed introduction is given in Chapter 14. One of the most fun challenges in current isogeny-based cryptography is the generalization of techniques we have for Kummer lines to Kummer surfaces, which often drastically increases the complexity. Fortunately, an improved understanding of Kummer surfaces also aids our understanding of Kummer lines.

3.2 Isogenies of Elliptic Curves

As the name suggests, isogenies are the main object of interest in isogeny-based cryptography. Shor's algorithm [149] breaks most cryptography based on the hardness of the discrete logarithm problem for (hyper)elliptic curves. However, so far, no polynomial-time quantum algorithms have been found for hard problems in isogeny-based cryptography. As cryptographic primitives in this domain provide key exchanges and signatures with key or signature sizes far smaller than other areas of post-quantum cryptography, isogeny-based cryptography has seen an explosion of interest in recent years. We formally define an isogeny only between elliptic curves, although they can be defined more generally for abelian varieties and even algebraic groups.

[2]More abstractly, work on a representation of the Kummer line $\mathcal{K}_E = E/\langle\pm 1\rangle$

Definition 3.6. An isogeny between elliptic curves E and E' over k is a non-zero group homomorphism

$$\varphi : E \to E'.$$

Any isogeny is a rational map, and we say that φ is defined over k whenever φ can be written as a map $(x, y) \mapsto (f_1(x, y)/f_2(x, y), g_1(x, y)/g_2(x, y))$, where f_i, g_i are polynomials in $k[x]$. Whenever an isogeny exists between two curves E and E', they are called *isogenous*. Surprisingly, it is easy to decide if two curves over a finite field are isogenous, due to Tate's theorem [154]: with $k = \mathbb{F}_q$, the curves E and E' are isogenous over $\mathbb{F}_q$ if and only if $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$. Tate's theorem holds in more generality, and in particular also for Jacobians of hyperelliptic curves. On the other hand, providing such an isogeny $E \to E'$ given only E and E' is presumable hard.

Problem 3.1. Given two isogenous elliptic curves E and E' over $k = \mathbb{F}_q$, find an isogeny $\varphi : E \to E'$.

Any isogeny φ furthermore induces a map $\varphi^* : k(E') \to k(E)$ between the function fields of E and E' by pre-composition, and the degree of φ is defined as the degree of the induced field extension $[k(E) : \varphi^* k(E')]$. The degree is multiplicative: $\deg(\varphi \circ \psi) = \deg \varphi \cdot \deg \psi$.

An isogeny is said to be *separable* if the field extension $\varphi^* k(E')/k(E)$ is separable. In particular, whenever the degree of an isogeny is coprime to char k, the isogeny is separable.

Separable isogenies are pleasant to work with. It can be shown that the degree of a separable isogeny over a finite field is precisely the cardinality of its kernel.

Example 3.1. Let $k = \mathbb{F}_{11}$ and let $E : y^2 = x^3 + x$ and $E' : y^2 = x^3 + 5$. then

$$\varphi : (x, y) \mapsto \left(\frac{x^3 + x^2 + x + 2}{(x - 5)^2}, y \cdot \frac{x^3 - 4x^2 + 2}{(x - 5)^3} \right)$$

is a separable isogeny $E \to E'$ of degree 3. The kernel of φ are the three points $\mathcal{O}$, $(5, 3)$ and $(5, -3)$.

Example 3.2. For any curve E, the map $[n] : P \mapsto [n]P$ is a seperable isogeny. As $E[n] := \ker[n] \cong \mathbb{Z}/\mathbb{Z}_n \times \mathbb{Z}/\mathbb{Z}_n$, the isogeny $[n]$ is of degree n^2.

Definition 3.7. The isogeny $\pi : (x, y) \mapsto (x^p, y^p)$ is the *Frobenius isogeny*. The Frobenius isogeny is a (purely) inseparable isogeny.

All inseparable isogenies essentially sprout from this single source, as it can be shown that any isogeny φ can be uniquely decomposed as

$$\varphi = \varphi_{\text{sep}} \circ \pi^k,$$

where φ_{sep} is a separable isogeny, up to composition with isomorphisms.

Any isogeny $\varphi : E \to E'$ naturally has a counterpart $\hat{\varphi} : E' \to E$ of the same degree called the *dual* of φ, with the property that $\hat{\varphi} \circ \varphi$ acts as multiplication by $\deg \varphi$ on E, and vice-versa that $\varphi \circ \hat{\varphi}$ acts as multiplication by $\deg \varphi$ on E'. This implies that φ is the dual of $\hat{\varphi}$, and furthermore that the isogeny $[n]$ is self-dual.

The set of isogenies between two elliptic curves E and E', together with the zero map $E \to E', P \mapsto O$, is denoted $\mathrm{Hom}(E, E')$, with subsets $\mathrm{Hom}_k(E, E')$ for those isogenies defined over k. By pointwise addition $(\varphi + \psi)(P) := \varphi(P) + \psi(P)$ for $\varphi, \psi \in \mathrm{Hom}(E, E')$, we find an abelian group structure.

Computing isogenies.

The mathematical description of isogenies tells us nothing about how to find or compute such isogenies. It turns out that for separable isogenies there is a one-to-one correspondence between finite subgroups $G \leq E$ and separable isogenies $\varphi : E \to E'$, up to post-composition with an isomorphism. We use the notation E/G to denote the codomain of the isogeny φ_G associated to $G \leq E$. Velu's formulas [158] give an explicit method to compute φ given G in $\mathcal{O}(\#G)$. This means we can both compute the codomain E/G, as well as compute $\varphi(Q) \in E/G$ for any $Q \in E$.

Any separable isogeny of degree $n = \prod \ell_i^{e_i}$ with ℓ_i prime can be decomposed into a sequence of isogenies of degrees ℓ_i, which drastically improves the efficiency from $\mathcal{O}(n)$ to $\mathcal{O}(\ell)$ with $\ell = \max_i \ell_i$. Furthermore, the improvements by Bernstein, De Feo, Leroux, and Smith [23] provide a square-root speedup over Velu's formulas to compute ℓ-isogenies in $\tilde{\mathcal{O}}(\sqrt{\ell})$.

3.2.1 Endomorphisms and the Endomorphism Ring

Isogenies from E to itself are called *endomorphisms* and they turn out to be more crucial and interesting than one would at first glance surmise. We have seen two examples of endomorphisms already: the multiplication-by-n map $[n] : E \to E$ and the Frobenius $\pi : (x, y) \mapsto (x^p, y^p)$.

Definition 3.8. For an elliptic curve E over k, the set of endomorphisms of E is denoted $\mathrm{End}(E) := \mathrm{Hom}(E, E')$. With addition by $(\varphi+\psi)(P) = \varphi(P)+\psi(P)$ and multiplication $(\varphi \cdot \psi)(P)$ by composition $\varphi \circ \psi(P)$, we obtain the structure of a ring on $\mathrm{End}(E)$, called the *endomorphism ring* of E. The subring of k-rational endomorphisms is denoted $\mathrm{End}_k(E)$, or $\mathrm{End}_q(E)$ when $k = \mathbb{F}_q$.

The maps $[n] \in \mathrm{End}(E)$ for $n \in \mathbb{Z}$ define an embedding of $\mathbb{Z}$ in $\mathrm{End}(E)$, and similarly $\pi \in \mathrm{End}(E)$ then implies a rank-2 subring $\mathbb{Z}[\pi] \subseteq \mathrm{End}(E)$ for any curve E over a field of non-zero characteristic. A more natural way to study the endomorphism ring is using the *endomorphism algebra* $\mathrm{End}^0(E) := \mathrm{End}(E) \otimes_{\mathbb{Z}} \mathbb{Q}$, which is a quaternion algebra. In turn, $\mathrm{End}(E)$ is isomorphic to an order of this quaternion algebra. This neatly characterizes elliptic curves over finite fields.

Theorem 3.9. Let E be an elliptic curve over $k = \mathbb{F}_q$. Then the rank of $\mathrm{End}(E)$ as a $\mathbb{Z}$-module is either

- two, in which case we call E *ordinary*, and $\mathrm{End}(E)$ is isomorphic to a quadratic order in $\mathrm{End}^0(E)$,
- or four, in which case we call E *supersingular*, and $\mathrm{End}(E)$ is isomorphic to a maximal order in $\mathrm{End}^0(E)$.

Supersingular curves lie at the core of isogeny-based cryptography: we only consider supersingular curves in Parts II and III. The main reason is that we have a fine control on the number of points on a supersingular curve. For example, any supersingular curve E over $\mathbb{F}_p$ has $p+1$ points, hence, as we can choose p freely, we can ensure E has points of order $\ell \mid p+1$ with ℓ prime. This turns out to be useful in many isogeny-based protocols, due to the correspondence between subgroups and isogenies.

3.2.2 The Action of the Class Group

Assume now a curve E, either ordinary or supersingular, with an endomorphism $\alpha : E \to E$ such that α^2 acts as $[-D] : P \mapsto [-D]P$ for some $D \in \mathbb{N}$. Thus, we may identify a subring of $\mathrm{End}(E)$ with $\mathcal{O} = \mathbb{Z} + \sqrt{-D}\mathbb{Z}$, by $\iota : \sqrt{-D} \mapsto \alpha$. Let K denote a quadratic imaginairy field such that $\mathcal{O}$ is a quadratic order in K, then ι is an *$\mathcal{O}$-orientation* if it can be extended to an embedding $K \to \mathrm{End}^0(E)$ such that $\iota(\mathcal{O}) = \mathrm{End}^0(E) \cap \iota(K)$. In this section, we slightly abuse notation by not explicitly writing the embedding ι.

We quickly get an action of any ideal $\mathfrak{a} \subseteq O$ on such curves: We define a kernel on E associated to $\mathfrak{a}$ by

$$E[\mathfrak{a}] = \bigcap_{\alpha \in \mathfrak{a}} \ker \alpha,$$

and use Velu's formulas to translate the kernel $E[\mathfrak{a}]$ into an isogeny $\varphi_{\mathfrak{a}} : E \to E/\mathfrak{a}$. As before, this is only efficient if the degree of $\varphi_{\mathfrak{a}}$ is smooth enough and $E[\mathfrak{a}]$ is either rational or defined over only a small field extension. Whenever $\mathfrak{a}$ is a principal ideal (ω), the kernel $E[\mathfrak{a}]$ is simply $\ker \omega$, and so $\omega = \varphi_{\mathfrak{a}} : E \to E$. However, one can show that any non-principal (fractional) ideal $\mathfrak{a}$ is never an endomorphism, and two fractional ideals $\mathfrak{a}$ and $\mathfrak{b}$ have isomorphic codomains only when $\mathfrak{a} = (\omega)\mathfrak{b}$ for some principal ideal (ω). In other words, the class group $\mathcal{C}\ell(\mathcal{O})$ of fractional ideals quotiented out by principal ideals, acts (commutatively) on the set of such curves E over k with $\mathcal{O} \subseteq \mathrm{End}(E)$, which we denote $\mathcal{E}\ell\ell_k(O)$, and Onuki [129] shows that this action $\mathcal{C}\ell(\mathcal{O}) \times \mathcal{E}\ell\ell_k(\mathcal{O}) \to \mathcal{E}\ell\ell_k(\mathcal{O})$ is free and transitive.

In theory, assuming vectorization and parallelization is hard for such a group action, we quickly get a Diffie-Hellman-like non-interactive key exchange: given a starting curve E, Alice and Bob both sample their private keys as elements $\mathfrak{a}, \mathfrak{b} \in \mathcal{C}\ell(\mathcal{O})$, their public keys are respectively $E/\mathfrak{a}$ and $E/\mathfrak{b}$, and they compute shared secrets by the action of $\mathfrak{a}$ on $E/\mathfrak{b}$ resp. $\mathfrak{b}$ on $E/\mathfrak{a}$ so that both derive $E/\mathfrak{a}\mathfrak{b}$, summarized in the following diagram.

$$\begin{tikzcd}
E \arrow[r, "\mathfrak{a}"] \arrow[d, "\mathfrak{b}"] & E/\mathfrak{a} \arrow[d, "\mathfrak{b}"] \\
E/\mathfrak{b} \arrow[r, "\mathfrak{a}"] & E/\mathfrak{ab}
\end{tikzcd}$$

However, the difficulty of using such a group action cryptographically lies in the efficiency of the group action: not only is it difficult to sample random elements of $\mathcal{C}\ell(\mathcal{O})$, computing the action of such elements on random curves in $\mathcal{E}\ell\ell_k(\mathcal{O})$ can be terribly or impossibly slow. An approach is to find an order $\mathcal{O}$ in a quadratic imaginairy field K such that many small odd primes ℓ split in $\mathcal{O}$.

CSIDH [46] finds a beautiful solution: by choosing primes of the form

$$p = 4 \cdot \ell_1 \cdot \ldots \cdot \ell_n - 1,$$

where the ℓ_i are small odd primes, then in the order $\mathcal{O} = \mathbb{Z} + \pi\mathbb{Z}$, with $\pi = \sqrt{-p}$ in the quadratic imaginairy field $K = \mathbb{Q}(\sqrt{-p})$, each of these ℓ_i splits as

$$(\ell_i) = \mathfrak{l} \cdot \bar{\mathfrak{l}}, \qquad \mathfrak{l} = (\ell, \pi - 1), \quad \bar{\mathfrak{l}} = (\ell, \pi + 1).$$

Supersingular curves now have $\#E(\mathbb{F}_p) = p + 1$, and the Frobenius endomorphism π on E, defined over $\mathbb{F}_p$, has precisely the property that $\pi^2 = [-p]$. The set $\mathcal{E}\ell\ell_p(\mathcal{O})$ then consists precisely of those curves E with $\mathrm{End}_p(E) \xrightarrow{\sim} \mathbb{Z} + \pi\mathbb{Z}$. Furthermore, for any such curve E, we can easily compute $E[\mathfrak{l}]$ or $E[\bar{\mathfrak{l}}]$, because by definition

$$E[\mathfrak{l}] = \ker(\ell) \cap \ker(\pi - 1),$$

thus we need both $\pi P = P$ and $\ell P = \mathcal{O}_E$, which means $E[\mathfrak{l}]$ is precisely the set of $\mathbb{F}_p$-rational points of order ℓ on E. As $\ell \mid p+1$ and supersingular curves have $\#E(\mathbb{F}_p) = p + 1$, we can easily find and compute with rational points of order ℓ, and thus compute the action of $\mathfrak{l}$, resp. $\bar{\mathfrak{l}}$ on $E \in \mathcal{E}\ell\ell_p(E)$.

We still face the difficulty of sampling random elements of $\mathcal{C}\ell(\mathcal{O})$. Thus, CSIDH requires the heuristic assumption that $\mathcal{C}\ell(\mathcal{O})$ is generated by these ideals $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$, so that we can sample essentially random elements by

$$\mathfrak{a} = \prod \mathfrak{l}_i^{e_i}, \qquad e_i \in \mathbb{Z},$$

where $\mathfrak{l}_i^{-1}$ denotes $\bar{\mathfrak{l}}_i$, for large enough integers e_i. The action of $\mathfrak{a}$ can then be computed per $\mathfrak{l}_i$, that is, we decompose $\varphi_{\mathfrak{a}} : E \to E/\mathfrak{a}$ into its prime-degree components given by $\mathfrak{l}_i$.

The last problem that needs to be solved to get the efficient NIKE described above, is the efficiency of the membership test: upon receiving a curve E, how are we ensured that $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ before we act with a secret key? Precisely in the CSIDH situation, this simply requires us to verify that E is supersingular, which is again efficient. The practicality of CSIDH is the main topic of Part II.

3.2.3 The Deuring correspondence

In the previous section, we have seen that $\mathrm{End}(E)$ for supersingular curves is isomorphic to some maximal order in a quaternion algebra $\mathcal{B}_{p,\infty}$. In more generality, the world of supersingular elliptic curves and their isogenies have many properties in common with the world of quaternion algebras and their ideals. The *Deuring correspondence*, after Max Deuring [62], provides us with the general bridge between these two worlds, mainly based on the functorial equivalence given by

$$\begin{array}{rcl}\{\text{ss. curves over } \mathbb{F}_{p^2}\}/\sim & \to & \{\text{max. orders in } \mathcal{B}_{p,\infty}\}/\sim \\ E & \mapsto & \mathrm{End}(E)\end{array}$$

where $\sim$ on both sides implies equivalence up to isomorphism. This thesis does not require an in-depth understanding of quaternion algebras, or the equivalence of categories. However, we sketch the situation informally so that we can understand SQIsign, an signature scheme whose complex signing procedure relies mainly on this correspondence. Part III of this thesis will focus only on the verification part of SQIsign, which can be understood in terms of elliptic curves and isogenies only. A reader can therefore safely skip the remainder of this section if one is only interested in SQIsign verification.

Similar to the previous section, the Deuring correspondence allows us to see isogenies $\varphi : E \to E'$ as ideals I_φ, but this time of a maximal order instead of a quadratic order. Again, the association relies on the kernel: all endomorphisms ω that vanish on $\ker \varphi$ form a left ideal I_φ of $\mathrm{End}(E)$, and similarly any left ideal I of $\mathrm{End}(E)$ give a subgroup $E[I] \subseteq E$ by the intersection of the kernel of all generators of I, which gives a unique isogeny $\varphi_I : E \to E/I$, up to post-composition with an isomorphism. The *norm* of such an ideal I, the greatest common divisor of the norms of its elements, is then equal to the degree of φ_I.

The codomain associated to such a left ideal I of $\mathcal{O}$ should then be some 'natural' maximal order with I as a right ideal. This natural object is the *right order*[3]

$$\mathcal{O}_r(I) := \{\beta \in \mathcal{B}_{p,\infty} \mid I\beta \subseteq I\},$$

which is isomorphic to the endomorphism ring of the elliptic curve E/I. In such a situation, we say that I is a *connecting ideal* of two maximal orders $\mathcal{O}_1$ and $\mathcal{O}_2$, that is, when I is a left ideal of $\mathcal{O}_1$ whose right order is isomorphic to $\mathcal{O}_2$, or vice versa.

The KLPT algorithm

The main algorithmic building block in SQIsign is a polynomial time algorithm by Kohel, Lauter, Petit, and Tignol [105], the KLPT algorithm. The idea is simple: we may often encounter isogenies $\varphi : E \to E'$ of some generic large degree d. Most likely, d is neither smooth, nor is the d-torsion rational or defined over a small extension field. If we could somehow, given φ construct

[3]No pun intended.

another isogeny $\varphi' : E \to E'$ whose degree d' is very smooth, such as $d' = \ell^k$ for some small prime ℓ, we could easily compute φ' instead which could alleviate our burdens.

In the world of supersingular elliptic curves, this problem is presumably hard: finding such a φ' implies we find a non-trivial endomorphism $\hat{\varphi'} \circ \varphi$, which is the hard problem all isogeny-based cryptography is based on. However, if we know $\mathrm{End}(E)$, and can therefore use the Deuring correspondence to move this problem to the world of quaternion algebras by $\varphi \mapsto I_\varphi$, our problems are magically solved by the KLPT algorithm, as long as the domain of φ is of a special form. We leave out the details of this special form, and simply refer to *special* maximal orders $\mathcal{O}$. Informally, we can state the KLPT algorithm as follows.

Theorem 3.10 (The KLPT algorithm)**.** Let I be a left ideal of a special maximal order $\mathcal{O}$ in $\mathcal{B}_{p,\infty}$. There exists a heuristic algorithm that returns an ideal J, equivalent to I, of norm ℓ^k for some $k \in \mathbb{N}$.

The result is wonderful: instead of working with the isogeny/ideal of ghastly degree/norm d, we can apply KLPT to work with the ideal J of smooth norm ℓ^k. By translating such an ideal back to an isogeny[4], the isogeny $\varphi_J : E \to E'$ can be efficiently computed in polynomial time.

So far, we have ignored two main problems: first, are we ensured the maximal orders we will encounter 'in the wild' are of the special form, and second, how large is k in the output of KLPT? As an answer to the first question, it turns out that the special curve $E_0 : y^2 = x^3+x$ has an endomorphism ring of precisely the special form we need, and this is all we require for the applications in SQIsign. For the second question, the norm of the output ideal can under some heuristic assumptions be estimated as $p^{7/2}$. In practice, this means that φ_J is a rather long isogeny, but not unpractically long.

The SQIsign signature scheme

SQIsign [47, 59, 60] is a signature scheme that cleverly uses the Deuring correspondence and the KLPT algorithm to construct an identification scheme using a Sigma protocol. Although there are some significant details between subsequent versions of the scheme, the main idea is as follows.

We assume the starting elliptic curve E_0, whose endomorphism ring $\mathcal{O}_0 = \mathrm{End}(E_0)$, a special maximal order, is publicly known. Alice, the prover, samples a secret isogeny $\varphi_A : E_0 \to E_A$, and provides E_A as her public key. By knowing φ_A, she ensures she, and only she, knows $\mathrm{End}(E_A)$. Her goal is to prove this knowledge.

The Sigma protocol then starts with Alice committing to some curve E_1 which is the codomain of another random isogeny $\varphi_{\mathrm{com}} : E_0 \to E_1$. She sends E_1 to Bob, the verifier. Again, only Alice can compute $\mathrm{End}(E_1)$ as she keeps φ_{com} hidden. Bob samples a random isogeny $\varphi_{\mathrm{chall}} : E_1 \to E_2$ and communicates φ_{chall} to Alice, which allows her to compute $\mathrm{End}(E_2)$ using her knowledge

[4]A highly non-trivial and slow task, which we will not discuss in detail in this thesis.

$$\begin{tikzcd}
E_0 \arrow[r, "\varphi_{\text{com}}"'] \arrow[d, dashed, "\varphi_A"] & E_1 \arrow[d, "\varphi_{\text{chall}}"'] \\
E_A \arrow[r, "\varphi_{\text{resp}}"] & E_2
\end{tikzcd}$$

Figure 3.1: The SQIsign protocol as a diagram.

of $\text{End}(E_1)$. Alice thus knows $\text{End}(E_A)$ and $\text{End}(E_2)$, and furthermore can compute the ideals I_{secret}, I_{com}, I_{chall}. Thus, she can compute an ideal I_{resp} as the product $I_{\text{chall}} \cdot I_{\text{com}} \cdot \bar{I}_{\text{secret}}$ to which she can apply KLPT to obtain a smooth equivalent ideal J. She then computes the associated isogeny $\varphi_J : E_A \to E_2$, which is her response $\varphi_{\text{resp}} := \varphi_J$. Bob verifies the response by recomputing $\varphi_{\text{resp}} : E_A \to E_2$ and confirms that the codomain is indeed (isomorphic to) the challenged E_2. The protocol is summarized by the Figure 3.1.

The SQIsign works ingeniously solve many obstacles that are not visible in the above description to allow Alice to sign, e.g. compute φ_{resp}, in practice, and to ensure the security of the scheme. However, with our main focus in Part III being SQIsign verification, we do not go in-depth on the signing procedure. Instead, we focus on the verification procedure.

The ideal J associated with φ_{resp} is an output of the KLPT algorithm and therefore, as noted before, of rather large norm ℓ^k. In practice, we find $\deg \varphi_{\text{resp}} \approx p^{15/4}$. For NIST Level I security, where $\log p \approx 2\lambda = 256$ bits, the current parameters give $\deg \varphi_{\text{resp}} = 2^e$ with $e \approx 1000$. The maximal available rational $2^\bullet$-torsion on supersingular curves is determined by the largest f such that $2^f \mid p+1$. For the given prime for NIST Level I security, this gives $f = 75$, and thus, in verification, we recompute φ_{resp} as a composition of $n := \lceil \frac{e}{f} \rceil$ isogenies $\varphi_i : E^{(i)} \to E^{(i+1)}$ of degree 2^f, such that $\varphi_{\text{resp}} = \varphi_n \circ \dots \varphi_1$. In Part III, in particular in Chapter 12 and Chapter 13, we analyze the performance of SQIsign verification in detail.

3.3 Moving to higher-dimensions

During the years that spanned the work in this thesis, SIDH/SIKE [57, 99] was spectacularly broken by higher-dimensional isogenies [43, 111, 140] using a (generalization of) Kani's lemma [100] combined with Zahrin's trick. From the ashes of SIDH arose a phoenix: higher-dimensional isogenies allow for an incredible constructive tool that increased the general flexibility for cryptodesign. This has sprouted numerous new schemes and will continue to provide significant improvements in isogeny-based cryptography as this paradigm shift to higher-dimensional isogenies is explored.

For this thesis, we sketch a quick understanding of higher-dimensional isogenies and their applications in variants of SQIsign. Although a detailed understanding is not required, a high-over understanding is useful to grasp the current landscape of isogeny-based cryptography and the precise role Chapters 12 to 14 play in our exploration of this landscape.

3.3.1 Higher-dimensional Isogenies

By higher-dimensional isogenies, we refer to isogenies between Abelian varieties of dimension larger than 1. In practice, we usually encounter 2-dimensional Abelian varieties over some field k, and these are easily classified. Let A be a 2-dimensional Abelian variety, then a) A is (isomorphic to) the Jacobian of some hyperelliptic curve $\mathcal{C}$ of genus 2 over k, or b) A is (isomorphic to) the product of two elliptic curves E and E' over k, or c) A is (isomorphic to) the Weil restriction of an elliptic curve E over K, where K/k is a degree-2 field extension.

We specifically focus on isogenies $f : A \to B$ whose kernel as a subgroup is isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$, and furthermore maximally isotropic[5]. Such isogenies as maps between hyperelliptic curves are precisely the *Richelot isogenies*, which have appeared in hyperelliptic curve cryptography literature. In such cases, we say that f is a $(2,2)$-isogeny. A concatenation of n such $(2,2)$-isogenies, modulo some requirements on cyclicity, is then a $(2^n, 2^n)$-isogeny.

Kani's Lemma

Kani's lemma [100] now allows for a very powerful tool: given the right setup, two 1-dimensional isogenies φ and ψ between elliptic curves can be associated with a specific 2-dimensional isogeny Φ. Computing Φ, as a 2-dimensional isogeny[6], will allow us to derive information on φ and ψ that was practically unavailable using only 1-dimensional isogenies. The right setup in this situation is an *isogeny diamond*.

Definition 3.11. Let $\psi : E_1 \to E_2$ and $\varphi : E_1 \to E_3$ be two isogenies of coprime degrees. An *isogeny diamond* is a square of isogenies

$$\begin{tikzcd}
E_1 \arrow[r, "\varphi"] \arrow[d, "\psi"] & E_3 \arrow[d, "\psi'"] \\
E_2 \arrow[r, "\varphi'"] & E_4
\end{tikzcd}$$

such that $\varphi \circ \psi' = \psi \circ \varphi'$, with $\deg \varphi = \deg \varphi'$ and $\deg \psi = \deg \psi'$.

Kani's lemma provides the crucial 2-dimensional isogeny that arises from an isogeny diamond.

[5]A technical requirement that we will not explore in more detail.

[6]The reason we only consider $(2^n, 2^n)$-isogeny is precisely that we can only efficiently compute 2-dimensional isogenies of this degree, although recent progress also explores $(3,3)$-isogenies and beyond [49].

Lemma 3.12 (Kani's lemma)**.** Consider an isogeny diamond given by φ, φ', ψ and ψ', and let $N = \deg\varphi + \deg\psi$. then the map

$$\Phi : E_2 \times E_3 \to E_1 \times E_4,$$

given in matrix form as

$$\begin{pmatrix} \hat{\psi} & \hat{\varphi} \\ -\varphi' & \psi \end{pmatrix},$$

is an (N, N)-isogeny of abelian surfaces with kernel

$$\ker\Phi = \{(\varphi(P), \psi(P)) \mid P \in E_1[N]\}.$$

The main idea is now simple: Although φ and ψ individually might be difficult to compute, with the right set-up Φ might be efficient to compute, and can provide the same information.

Embedding isogenies

A practical example of the above idea is given by Robert [141] to *embed an isogeny* φ of dimension 1 into a higher-dimensional isogeny Φ, so that evaluation of Φ allows us to evaluate φ.

Consider as a first example an isogeny $\varphi : E \to E'$, such that $2^n - \deg\varphi = x^2$ for some $n \in \mathbb{N}$ and $x \in \mathbb{Z}$. In such a case, we construct the isogeny diamond

$$\begin{tikzcd} E \arrow[r, "\varphi"] \arrow[d, "{[x]}"] & E' \arrow[d, "{[x]}"] \\ E \arrow[r, "\varphi"] & E' \end{tikzcd}$$

where $[x]$ denotes the simple-to-compute multiplication-by-x map $P \mapsto [x]P$. Kani's lemma now implies that $\Phi : E \times E' \to E \times E'$ is an (N, N)-isogeny with $N = \deg\varphi + \deg[x] = 2^n$, and so

$$\ker\Phi = \{(\varphi(P), [x]P) \mid P \in E[2^n]\}.$$

Thus, if we know $\varphi(P)$ for $P \in E[2^n]$ and we simply compute $[x]P$, we can compute $\varphi(Q)$ for any other $Q \in E$ using Φ, which we can efficiently compute as a $(2^n, 2^n)$-isogeny. Thus, we can 'embed' the 1-dimensional isogeny φ into the 2-dimensional isogeny Φ, if $\deg\varphi$ differs from a power of 2 by precisely a square.

This situation is highly unlikely to happen for any random isogeny φ, as only very few integers differ from a power of 2 by precisely a square. However, by Legendre's four-square theorem, any natural number can be represented as the sum of four squares. This inspired Robert [141] to generalize the result by Kani to a result between 8-dimensional abelian varieties $\Phi : E^4 \times E'^4 \to E^4 \times E'^4$. Then, instead of hoping $\deg\varphi$ precisely fits $2^n - x^2$ for some $x \in \mathbb{Z}$, we find four integers $x_1, x_2, x_3, x_4 \in \mathbb{Z}$ such that $2^n - \deg \varphi = x_1^2 + x_2^2 + x_3^2 + x_4^2$. A similar setup as before then allows us to embed the 1-dimensional isogeny φ into an 8-dimensional isogeny Φ. Using several technical tricks, one can often already achieve a similar result with a 4-dimensional embedding.

3.3.2 Higher-dimensional SQIsign

The technique to embed 1-dimensional isogenies into higher dimensional isogenies is powerful and versatile. For example, whenever the 1-dimensional isogeny φ is of large non-smooth degree but embeddable in a smooth higher-dimensional isogeny Φ, we can simply evaluate φ on points using Φ, which is efficient to compute.

This has led to a revolution in many areas of isogeny-based cryptography, and most notably in SQIsign: instead of having to use KLPT to obtain a smooth response isogeny (of fairly large degree), we may embed the response isogeny into a smooth higher-dimensional isogeny Φ. Verification is then reduced to the efficient recomputation of Φ. To achieve this requires solving several technical difficulties which was done by Dartois, Leroux, Robert, and Wesolowski [55]. The resulting signature scheme, using 4- or 8-dimensional isogenies is named SQIsignHD and allows significantly faster signing than the original SQIsign. A drawback is slow verification, as computing 4- or 8-dimensional isogenies is much more complex than 1-dimensional isogenies.

Using an algorithmic building block by Nakagawa and Onuki [121], three works independently from each other were able to improve SQIsignHD to achieve verification using 2-dimensional isogenies only. These works, SQIsign2D-West [18], SQIsign2D-East [122] and SQIPrime [69], achieve both fast signing and verification, ushering in an era of practical higher-dimensional isogeny-based cryptography.

In summary, SQIsign2D is currently the most practical approach to isogeny-based signatures, with relatively fast signing and verification times on par with 1-dimensional SQIsign. However, 1-dimensional SQIsign is still an interesting topic for research, as many questions remain unanswered when comparing the real-world practicality of these two approaches. Chapter 13 deals with some of these questions.

3.4 Pairings

Like dogs to humans, pairings are great companions for all of isogeny-based cryptography. Pairings, bilinear maps between Abelian groups, appear naturally in many versatile use cases due to their interesting properties, in particular in relation to isogenies of abelian varieties. This thesis contains many of these applications of pairings, mostly using the Tate pairing[7] although the Weil

[7]More properly named the Tate-Lichtenbaum pairing, or also the Frey-Rück pairing, most authors stick to *Tate pairing*, perhaps simply because it is easiest to write.

pairing is more commonly used in general. Both are usually used in very specific and concrete situations, yet we will introduce these in their general forms, following Bruin [39], Garefalakis [88], and Silverman [150]. This allows us to provide more intuition to the reader in what these pairings are fundamentally doing. More properties, details and computational aspects are widely described in the literature, see [48, 85, 86]. Inspiration is also drawn from [44, 142].

3.4.1 The Weil Pairing

Let $\varphi : E \to E'$ be an isogeny with $\operatorname{char} k \nmid \deg \varphi$. To φ, we can associate the generalized φ-Weil pairing

$$e_\varphi : \ker \varphi \times \ker \hat{\varphi} \to \overline{k}^*,$$

which is a bilinear, non-degenerate pairing invariant under the action of $\operatorname{Gal}(\overline{k}/k)$, taking as values m-th roots in $\overline{k}^*$, for m such that $\ker \varphi \subset E[m]$. We denote the cyclic group of m-th roots by μ_m. Several other desirable properties are described in [44, 86].

Taking $\varphi = [m] : E \to E$, we recover the more traditional Weil pairing

$$e_m : E[m] \times E[m] \to \mu_m.$$

Computing $e_m(P, Q)$ for two points $P, Q \in E[m]$ can be done using two Miller functions $f_{m,P}, f_{m,Q} \in \overline{k}(E)$, defined by the property that

$$\operatorname{div} f_{m,P} = m(P) - m(\mathcal{O}), \quad \operatorname{div} f_{m,Q} = m(Q) - m(\mathcal{O}).$$

We want to evaluate $f_{m,P}$ on the divisor $(Q) - (\mathcal{O})$, and vice versa. However, as $\mathcal{O}$ is a root of $f_{m,P}$, we must first find a divisor D_Q equivalent to $(Q) - (\mathcal{O})$, with support disjoint from P, OO, and similar for D_P. Then

$$e_m(P, Q) = f_{m,P}(D_Q)/f_{m,Q}(D_P).$$

Computing the Miller functions $f_{m,P}$ and $f_{m,Q}$ can be done using Miller's algorithm [116].

One exemplary use case of the Weil pairing is computing a change of basis matrix for two bases P, Q and P', Q' of $E[m]$. We must have

$$\begin{pmatrix} P' \\ Q' \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \cdot \begin{pmatrix} P \\ Q \end{pmatrix}$$

for values $a_i \in \mathbb{Z}/m\mathbb{Z}$. An easy way to compute these values is by

$$a_1 = e_m(P, P'),\ a_2 = e_m(Q, P'),\ a_3 = e_m(P, Q'),\ a_4 = e_m(Q, Q').$$

3.4.2 The Tate Pairing

We focus on the case $k = \mathbb{F}_q$. Let $\varphi : E \to E'$ be an isogeny with $\ker \varphi \subset E[m] \subset E[q-1]$. To φ, we can associate the generalized φ-Tate pairing

$$T_\varphi : \ker \varphi(\mathbb{F}_q) \times \operatorname{coker} \hat{\varphi}(\mathbb{F}_q) \to \mathbb{F}_q^*/\mathbb{F}_q^{*m},$$

which is a bilinear, non-degenerate pairing invariant under the action of $\mathrm{Gal}(\overline{k}/k)$. The values are equivalence classes modulo m-th powers. Sometimes this is enough, but in practice, we often raise the value to the power $(q-1)/m$ to obtain specific m-th roots. We then call the Tate pairing *reduced*, and denote by t_φ the composition of T_φ with the exponentiation $z \mapsto z^{(q-1)/m}$, which is an isomorphism $\mathbb{F}_q^*/\mathbb{F}_q^{*m} \to \mu_m(\mathbb{F}_q)$. Several other desirable properties are described in [44, 86].

Computing $t_\varphi(P, Q)$ for $P \in \ker \varphi$, $Q \in E'$ can be done using the Miller function $f_{m,P}$ evaluated on a divisor D_Q equivalent to $(Q) - (\mathcal{O})$, with disjoint support, where again we can compute $f_{m,P}$ using Miller's algorithm. Taking $\varphi = [m]$ recovers the more traditional Tate pairing

$$t_m : E[m] \times E(\mathbb{F}_q)/[m]E(\mathbb{F}_q) \to \mu_m.$$

The case $\varphi = [m]$ also shows an immediate use case of the Tate pairing: as it is non-degenerate, we can identify the points $Q \in [m]E(\mathbb{F}_q)$ precisely as those points where $t_m(P, Q) = 1$ for all $P \in E[m]$. Conversely, this implies that $t_m(P, Q) \neq 1$ for some $P \in E[m]$ and $Q \in E(\mathbb{F}_q)$ immediately implies that $Q \notin [m]E(\mathbb{F}_q)$. Even more specific for $[m] = [2]$, for any Montgomery curve E_A, which has $(0, 0) \in E[2]$, this implies that if $t_2((0, 0), Q) \neq 1$, then $Q \notin [2]E$. The value $t_2((0, 0), Q)$ can be computed to be precisely the quadratic reciprocity of x_Q. For supersingular curves E_A with $2^f \mid p+1$, this thus implies that the order of Q is divisible by 2^f, whenever x_Q is non-square. Thus, a classical result on the image of E under doubling [98, Thm. 4.1] can concisely be described using the Tate pairing. More generally speaking, we can use the φ-Tate pairing together with $\ker \varphi$ to determine the coset in $\operatorname{coker} \hat{\varphi}(\mathbb{F}_q)$ in which a point $Q \in E$ lies.

Part I

Isometries

Intro

Some introduction on isometries testing cref Chapter 6 in Part I.

Chapter 4

Hardness analysis

4.1 Placeholder

The paper will go here.

4.2 Introduction

Given two mathematical objects of the same type, an equivalence problem asks the question whether there exists an equivalence map between these objects – and how to find it – that preserves some important property of the objects. These kind of problems come in different flavors depending on the objects – groups, graphs, curves, codes, quadratic forms, etc. – and quite often the interesting maps are isomorphisms or isometries. Interestingly, equivalence problems are one of the core hard problems underlying the security of many public-key cryptosystems, especially post-quantum ones. Many multivariate and code-based systems employ an equivalence transformation as a hiding technique, and thus intrinsically rely on the assumption that a particular equivalence problem is intractable, for example [29, 41, 61, 65, 113, 126, 130] . In addition, quite remarkably, a hard equivalence problem gives rise to a Sigma protocol and, through the Fiat-Shamir transform, a provably secure digital signature scheme [80]. This idea has been revisited many times, being the basis of several signature schemes [16, 30, 56, 58, 89, 130]. Two such schemes actually appeared during the writing of this manuscript [68, 152] as a result of NIST's announcement for an additional fourth round on signatures in the post quantum standardization process [123]. Understanding the hardness of these equivalence problems is an essential task in choosing appropriate parameters that attain a certain security level of these cryptographic schemes. Comment Simona: LESS case....

One of these problems is the Code Equivalence problem, which given two codes (with the Hamming metric), asks for an isometry (equivalence transformation that preserves the metric) that maps one code to the other. It was first studied by Leon [108] who proposed an algorithm that takes advantage of the Hamming weight being invariant under monomial permutations. It was improved very recently by Beullens [25] using collision-based techniques. Sendrier [147] proposed another type of algorithm, the Support Splitting Algorithm (SSA), that is exponential in the dimension of the hull (the intersection of a code and its dual). Interestingly, in low characteristic, random codes have very small hull, rendering the problem easy.

In this work, we focus on the code equivalence problem, but for matrix codes (an $\mathbb{F}_q$-linear subspace of the space of $m \times n$ matrices over $\mathbb{F}_q$) endowed with the rank metric - *Matrix Code Equivalence* (MCE). Evaluating the hardness of this problem is only natural – rank-based cryptography has become serious competition for its Hamming-based counterpart, showing superiority in key sizes for the same security level [8, 9, 20, 114]. MCE, and variations of it, has been introduced by Berger in [22], but it was only recently that the first concrete statements about its hardness were shown in two concurrent independent works publicly available as preprints [53, 93][1]. Couvreur et al. [53] showed that MCE is at least as hard as the (Monomial) Code Equivalence problem in the Hamming metric, while for only right equivalence, or when the codes are $\mathbb{F}_{q^m}$-linear, the problem becomes easy. Grochow and Qiao [93] show the same reduction from (Monomial) Code Equivalence to MCE but using a completely different technique of linear algebra coloring gadgets which makes the reduction looser than the one in [53].

4.2.1 Our contributions

In this paper, we investigate the theoretical and practical hardness of the Matrix Code Equivalence (MCE) problem. Our contributions can be summarized as follows:

First, we link in a straightforward manner the MCE problem to hard problems on systems of polynomials by showing that MCE is polynomial-time equivalent to the Bilinear Maps Linear Equivalence (BMLE) problem. We then extend this result by proving that MCE is polynomial-time equivalent to the Quadratic Maps Linear Equivalence (QMLE) problem, under a mild assumption of trivial automorphism groups of the codes in question. While our technique fails to give a proof without this assumption, we consider it to be reasonable for randomly generated codes and for cryptographic purposes. As the QMLE problem is considered to be the hardest equivalence problem for systems of multivariate polynomials, it is essential to understand under which conditions MCE and QMLE reduce to one another. Note that previous work[2] requires much stronger assumptions for related results [19, 83, 93], such as algebraically closed fields or existence of square or third roots. Our reduction to QMLE is tight and gives a tight upper bound on the hardness of MCE. Furthermore, it is very simple,

[1]The two works use different techniques and terminology, and seem to be mutually unaware of the line of work preceding the other. In [93] the MCE problem is referred to as Matrix Space Equivalence problem and 3-Tensor Isomorphism problem.

[2]We were made aware of this line of work by one of the authors after our results were first presented at WCC 2022.

thus establishing connection between code equivalence problems and polynomial equivalence problems that is usable in practice. This is the basis of our contributions on the practical hardness of MCE.

Second, using similar techniques, and under the same assumptions, we show that MCE is polynomial-time equivalent to other code equivalence problems, such as Matrix Sum-Rank Code Equivalence Problem, and at least as hard as the Vector Sum-Rank Code Equivalence Problem. All these connections and our results are visualized in Figure 4.1.

On the practical side, we provide the first two non-trivial algorithms for solving MCE using the connection to QMLE. The first algorithm is a generalization of a known birthday-based algorithm for QMLE [37, 38] for systems of polynomials with the same number of variables as equations. We show that this algorithm extends to different invariance properties and code dimensions, which helps us prove complexity of $q^{\frac{2}{3}(n+m)}$ up to a polynomial factor for MCE for $m \times n$ matrix codes. The algorithm is probabilistic with success probability that can be made arbitrarily close to 1, and can be used for code dimensions up to $2(m+n)$. For larger dimensions, the complexity becomes $q^{(n+m)}$ up to a polynomial factor, but the algorithm is deterministic. The birthday-based algorithm for QMLE [38] assumed existence of a polynomial-time solver for the inhomogeneous variant of QMLE to achieve these complexities. Interestingly, due to the specific instances of the inhomogeneous QMLE arising from the collision search, the problem seems to be much harder than for random instances – a fact previously overlooked in [38]. In contrast, [37] uses a non-polynomial estimate for this solver. We analyse the most recent results regarding such solvers, and show that for parameter sets of cryptographical interest the above complexities hold, even if such solvers do not achieve polynomial time.

Our second algorithm uses the bilinear structure of the polynomials arising from MCE. Because matrix codes show symmetry between the parameters, as given in Lemma 4.14, the complexity of solving MCE using this result and Algorithm 2 becomes $q^{\min\{m,n,k\}}$ up to a polynomial factor. The algorithm is deterministic and does not require a polynomial-time solver for the inhomogeneous QMLE instance, but the weaker assumption that the solver has a complexity of $\mathcal{O}(q^{\min\{m,n,k\}})$ at most. This general result, valid for any m,n, and k, is summarized in our main result Theorem 4.24.

Lastly, to verify the results and performance of these algorithms in practice, we have implemented both and solved randomly generated instances of MCE for different parameter sets. The results of these experiments show that our assumptions are reasonable and the above complexities hold. Our implementations are open source and available at:

https://github.com/mtrimoska/matrix-code-equivalence

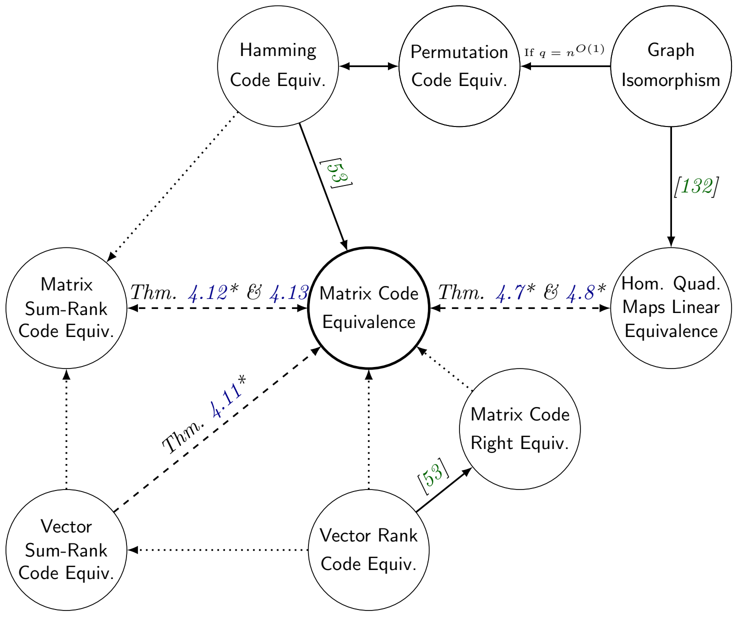


Figure 4.1: Reductions around Matrix Code Equivalence. Dashed arrows are contributions from this work, dotted arrows are trivial reductions. "$\mathsf{A} \longrightarrow \mathsf{B}$" means that "Problem A reduces to Problem B in polynomial time". Results with * assume trivial automorphism groups.

4.3 Preliminaries

Let $\mathbb{F}_q$ be the finite field of q elements. $\mathrm{GL}_n(q)$ and $\mathrm{AGL}_n(q)$ denote respectively the general linear group and the general affine group of degree n over $\mathbb{F}_q$.

We use bold letters to denote vectors $\mathbf{a}, \mathbf{c}, \mathbf{x}, \dots$, and matrices $\mathbf{A}, \mathbf{B}, \dots$. The entries of a vector $\mathbf{a}$ are denoted by a_i, and we write $\mathbf{a} = (a_1, \dots, a_n)$ for a (row) vector of dimension n over some field and $\mathbf{a}^\top = (a_1, \dots, a_n)^\top$ for the respective column vector. Similarly, the entries of a matrix $\mathbf{A}$ are denoted by A_{ij}. A matrix $\mathbf{A}$ is called symmetric if $\mathbf{A}^\top = \mathbf{A}$ and skew-symmetric if $\mathbf{A}^\top = -\mathbf{A}$. The space of matrices over $\mathbb{F}_q$ of size $m \times n$ is denoted $\mathcal{M}_{m,n}(q)$. The set of k-subsets of $\mathcal{M}_{m,n}(q)$ is denoted by $\mathcal{M}_{m,n}^{[k]}(q)$.

Random sampling from a set S is denoted by $a \xleftarrow{\$} S$. We use the notation $\tilde{\mathcal{O}}(f(n))$ to denote $\mathcal{O}(f(n)\log(f(n)))$ whenever we want to omit polynomial factors from the complexity expression. We use the notation $f = \Theta(g)$ whenever f is bounded from below and above by g asymptotically.

For a computational problem P, if we want to emphasize a list of parameters p defining the size of the inputs and the input set S, we will use the notation $\mathsf{P}(p, S)$. If these are not relevant, clear from context, or the set S is the entire universe U, we will use only $\mathsf{P}(p)$ or P.

Our results in Section 4.4 use the following standard notion of Turing reduction.

Definition 4.1. Given two computational problems $\mathsf{P}(\mathsf{p}, S)$ and $\mathsf{P}'(\mathsf{p}', S')$, with inputs coming from sets S and S' respectively, we say that $\mathsf{P}(\mathsf{p}, S)$ reduces to $\mathsf{P}'(\mathsf{p}', S')$ if there exists a probabilistic polynomial-time oracle machine $\mathcal{B}$ such that for every oracle $\mathcal{A}$ that solves $\mathsf{P}'(\mathsf{p}', S')$ on all inputs from S', $\mathcal{B}^{\mathcal{A}}$ ($\mathcal{B}$ given access to $\mathcal{A}$) solves $\mathsf{P}(\mathsf{p}, S)$ on all inputs from S.

Note that our reductions are meaningful only as worst-case to worst-case, and therefore in the definition we include the statement that the oracles solve the problems on all inputs. On the other hand, we do not always require the oracle $\mathcal{A}$ to be able to solve P' on the entire universe U' of inputs in order for $\mathcal{B}^{\mathcal{A}}$ to be able to solve P on the entire universe U of inputs. When this is the case, it will be emphasized through the definition of the input sets S and S'. These restrictions, however, can not be used to show a stronger statement such as worst-case to average-case reduction.

4.3.1 The Matrix Code Equivalence problem.

This section introduces basic notions on matrix codes and their equivalences. A more thorough introduction on matrix codes can be found in [**gorla**]. The usual choice for measuring distance between matrices over a finite field is the so-called *rank metric*, defined as follows.

Definition 4.2. Let Rank($\mathbf{M}$) denote the rank of a matrix $\mathbf{M} \in \mathcal{M}_{m,n}(q)$. The *rank distance* between two $m \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$ over $\mathbb{F}_q$ is defined as

$$d(\mathbf{A}, \mathbf{B}) = \mathrm{Rank}(\mathbf{A} - \mathbf{B}).$$

An *isometry* is a map $\mu : \mathcal{M}_{m,n}(q) \to \mathcal{M}_{m,n}(q)$ that preserves the rank, i.e. $\mathrm{Rank}(\mu(\mathbf{M})) = \mathrm{Rank}(\mathbf{M})$ for all $\mathbf{M} \in \mathcal{M}_{m,n}(q)$.

By symmetry, without loss of generality, in the rest of the text we assume $n \geqslant m$.

Definition 4.3. A *matrix code* is a subspace $\mathcal{C}$ of $m \times n$ matrices over $\mathbb{F}_q$ endowed with the rank metric. Let k denote the dimension of $\mathcal{C}$ as a subspace of $\mathbb{F}_q^{m\times n}$ and its basis by $\langle \mathbf{C_1}, \dots, \mathbf{C_k} \rangle$, with $\mathbf{C_i} \in \mathbb{F}_q^{m\times n}$ linearly independent. Two matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$ are said to be *equivalent* if there exists an isometry μ with $\mu(\mathcal{C}) = \mathcal{D}$.

An isometry from $\mathcal{C}$ to $\mathcal{D}$ is always of the form $\mathbf{M} \mapsto \mathbf{AMB}$, $\mathbf{M} \mapsto \mathbf{M}^\top$ or a composition of these two, where $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$ [96, 160]. We restrict our attention to the isometries of the first form and we will say that two matrix codes are equivalent if there exists a map $\mathbf{C} \mapsto \mathbf{ACB}$ from $\mathcal{C}$ to $\mathcal{D}$ where $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$. We will denote this map as a pair $(\mathbf{A}, \mathbf{B})$. When $n = m$, If there exists a map $(\mathbf{A}, \mathbf{A}^\top) : \mathbf{C} \mapsto \mathbf{ACA}^\top$ from $\mathcal{C}$ to $\mathcal{D}$, where $\mathbf{A} \in \mathrm{GL}_m(q)$, we will say that the codes $\mathcal{C}$ and $\mathcal{D}$ are *congruent*. This is a direct generalization of the notion of congruent matrices. An *automorphism* of a code is a map $(\mathbf{A}, \mathbf{B}) : \mathcal{C} \to \mathcal{C}$, i.e. for each $\mathbf{C} \in \mathcal{C}$, we get $\mathbf{ACB} \in \mathcal{C}$. The *automorphism group* of $\mathcal{C}$ contains all the automorphisms of $\mathcal{C}$. If the automorphism group contains only the maps $(\lambda\mathbf{I}, \nu\mathbf{I})$ for scalars $\lambda, \nu \in \mathbb{F}_q^*$, we say the automorphism group is trivial.

The main focus of this article will be the *Matrix Code Equivalence*(MCE) problem which is formally defined as follows:

Problem 4.1. MCE $(n, m, k, \mathcal{M}_{m,n}^{[k]}(q))$:
Input: Two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$
Question: Find – if any – a map $(\mathbf{A}, \mathbf{B})$, where $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mathbf{ACB} \in \mathcal{D}$.

This is the computational version of MCE which, similarly to its counterpart in the Hamming metric [14, 16, 30], seems to be more interesting for cryptographic applications than its decisional variant. We will thus be interested in evaluating the practical hardness only of MCE, and present algorithms only for MCE and not its decisional variant. It is also interesting to consider the following variant of MCE:

Problem 4.2. MCE base$(n, m, k, \mathcal{M}_{m,n}^{[k]}(q))$:
Input: The bases $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ of two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$
Question: Find – if any – a map $(\mathbf{A}, \mathbf{B})$, where $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{C}^{(i)}$, it holds that $\mathbf{AC^{(i)}B} = \mathbf{D}^{(i)}$.

Intuitively, MCE base seems easier than MCE, and as a matter of fact, we will show later that most random instances are solvable in polynomial time. Another variant of the MCE problem is the *Matrix Codes Right Equivalence* problem (MCRE) (left equivalence could be defined similarly):

Problem 4.3. MCRE $(n, m, k, \mathcal{M}_{m,n}^{[k]}(q))$:
Input: Two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$
Question: Find – if any – $\mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mathbf{CB} \in \mathcal{D}$.

It has been shown in [53] that MCE is at least as hard as code equivalence in the Hamming metric, *Hamming Code Equivalence* (HCE), also known as Linear or Monomial Equivalence. Interestingly, the same paper shows that MCRE is actually easy and can always be solved in probabilistic-polynomial time.

For *vector rank codes* $\mathcal{C} \subset \mathbb{F}_{q^m}^n$, isometries are similar to the case of matrix codes. We get the *Vector Rank Code Equivalence* (VRCE) problem.

Problem 4.4. VRCE $(n, m, k, \mathcal{M}_{m,n}^{[k]}(q))$:
Input: Two k-dimensional vector rank codes $\mathcal{C}, \mathcal{D} \subset \mathbb{F}_{q^m}^n$
Question: Find – if any – a matrix $\mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{c} \in \mathcal{C}$, it holds that $\mathbf{cB} \in \mathcal{D}$.

Given a vector rank code $\mathcal{C} \subset \mathbb{F}_{q^m}^n$ and a basis Γ for $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, each vector $\mathbf{c} \in \mathcal{C}$ can be expanded to a matrix $\Gamma(\mathbf{c}) \in \mathcal{M}_{m,n}(q)$, giving rise to a matrix code $\Gamma(\mathcal{C})$. For any two bases Γ and Γ', an equivalence between two vector rank codes $\mathcal{C}$ and $\mathcal{D}$ implies an equivalence between the matrix codes $\Gamma(\mathcal{C})$ and $\Gamma'(\mathcal{D})$ [**gorla**], so VRCE is trivially a subproblem of MCE. However, using the $\mathbb{F}_{q^m}$-linearity of vector rank codes, VRCE reduces *non-trivially* to MCRE [53].

4.3.2 Systems of quadratic polynomials.

Let $\mathcal{P} = (p_1, p_2, \dots, p_k) : \mathbb{F}_q^N \to \mathbb{F}_q^k$ be a vectorial function of k quadratic polynomials in N variables $x_1, \dots, x_N$, where

$$p_s(x_1, \dots, x_N) = \sum_{1 \leqslant i \leqslant j \leqslant N} \gamma_{ij}^{(s)} x_i x_j + \sum_{i=1}^{N} \beta_i^{(s)} x_i + \alpha^{(s)},$$

with $\gamma_{ij}^{(s)}, \beta_i^{(s)}, \alpha^{(s)} \in \mathbb{F}_q$ for $1 \leqslant s \leqslant k$.

It is common to represent the quadratic homogeneous part of the components of $\mathcal{P}$ using symmetric matrices, but unfortunately, a natural correspondence only exists for finite fields of odd characteristic. For the case of even characteristic, we will adopt a technical representation that is a common workaround in the literature of multivariate cryptography and will still be good for our purposes.

Let $p(x_1, \dots, x_N) = \sum\limits_{1 \leqslant i \leqslant j \leqslant N} \gamma_{ij} x_i x_j$ be a quadratic form over $\mathbb{F}_q$. Then, for fields of odd characteristic, we can associate to p a symmetric matrix $\mathbf{P} = \overline{\mathbf{P}} + \overline{\mathbf{P}}^\top$, where $\overline{\mathbf{P}}$ is an upper triangular matrix with coefficients $\overline{\mathbf{P}}_{ij} = \gamma_{ij}/2$ for $i \leqslant j$. Clearly, there is a one-to-one correspondence between quadratic forms and symmetric matrices, since for $\mathbf{x} = (x_1, \dots, x_N)$ it holds that

$$p(x_1, \dots, x_N) = \mathbf{x}\mathbf{P}\mathbf{x}^\top. \tag{4.1}$$

Now, all operations on quadratic forms naturally transform into operations on matrices since the one-to-one correspondence between quadratic forms and symmetric matrices is in fact an isomorphism. Note that, in matrix form, a change of variables (basis) works as:

$$p(\mathbf{x}\mathbf{S}) = \mathbf{x}\mathbf{S}\mathbf{P}\mathbf{S}^\top\mathbf{x}^\top. \tag{4.2}$$

In what follows, we will interchangeably work with both the quadratic form p and its matrix representation $\mathbf{P}$.

Over fields $\mathbb{F}_q$ of even characteristic, the relation (4.1) does not hold, since for a symmetric matrix $\mathbf{P}$ we have $(\mathbf{P}_{ij} + \mathbf{P}_{ji})x_i x_j = 2\mathbf{P}_{ij} x_i x_j = 0$. The nice correspondence between quadratic forms and symmetric matrices is broken, but we would still like to be able to use some sort of matrix representation for quadratic forms. Thus, in even characteristic we associate to p a symmetric matrix $\mathbf{P} = \overline{\mathbf{P}} + \overline{\mathbf{P}}^\top$, where $\overline{\mathbf{P}}$ is an upper triangular matrix with coefficients $\overline{\mathbf{P}}_{ij} = \gamma_{ij}$ for $i \leqslant j$.

This representation can also be used in odd characteristic when it comes to linear operations and changes of basis, as the correspondence $p \mapsto \mathbf{P}$ is a homomorphism. However, it is not a bijection, since all the quadratic forms in the set $\{\sum_{1\leqslant i<j\leqslant N} \gamma_{ij}x_ix_j + \sum_{1\leqslant i\leqslant N} \gamma_{ii}x_i^2 \mid \gamma_{ii} \in \mathbb{F}_q\}$ map to the same symmetric matrix (note that it has zeros on the diagonal). In practical, cryptographic applications, this typically does not pose a problem, and can be overcome. The same holds for our purpose of solving equivalence problems for systems of quadratic polynomials.

Differential of quadratic functions.

Given a non-zero $\mathbf{a} \in \mathbb{F}_q^N$, an object directly related to the symmetric matrix representation of quadratic forms is the *differential* of $\mathcal{P}$ at $\mathbf{a}$ (see [66, 81]):

$$D_{\mathbf{a}}\mathcal{P} : \mathbb{F}_q^N \to \mathbb{F}_q^k, \quad \mathbf{x} \mapsto \mathcal{P}(\mathbf{x}+\mathbf{a}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{a}).$$

Note that the differential of a quadratic function is closely related to the bilinear form $\beta(\mathbf{x},\mathbf{y}) = q(\mathbf{x}+\mathbf{y}) - q(\mathbf{x}) - q(\mathbf{y})$ associated to a quadratic form q. In this work we are especially interested in the kernel of $D_{\mathbf{a}}\mathcal{P}$, as $D_{\mathbf{a}}\mathcal{P}(\mathbf{x}) = 0$ implies $\mathcal{P}(\mathbf{x}+\mathbf{a}) = \mathcal{P}(\mathbf{x}) + \mathcal{P}(\mathbf{a})$, that is, $\mathcal{P}$ acts linearly on the kernel of $D_{\mathbf{a}}\mathcal{P}$.

4.3.3 Isomorphism of polynomials.

The Isomorphism of Polynomials (IP) problem (or Polynomial Equivalence (PE) [74]) was first defined by Patarin in [130] for the purpose of designing a "graph isomorphism"-like identification scheme and a digital signature scheme using the Fiat-Shamir transform [80]. It is defined as follows.

Problem 4.5. IP $(N, k, \mathbb{F}_q[x_1,\ldots,x_N]^k \times \mathbb{F}_q[x_1,\ldots,x_N]^k)$:
Input: Two k-tuples of multivariate polynomials $\mathcal{F} = (f_1, f_2, \ldots, f_k)$, $\mathcal{P} = (p_1, p_2, \ldots, p_k) \in \mathbb{F}_q[x_1,\ldots,x_N]^k$.
Question: Find – if any – $(\mathbf{S},\mathbf{s}) \in \mathrm{AGL}_N(q), (\mathbf{T},\mathbf{t}) \in \mathrm{AGL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}) = \mathcal{F}(\mathbf{xS}+\mathbf{s})\mathbf{T} + \mathbf{t}. \tag{4.3}$$

The variant of the problem where $(\mathbf{T},\mathbf{t})$ is trivial is known as the Isomorphism of Polynomials with one secret (IP $1\mathcal{S}$), whereas if $\mathcal{P}$ and $\mathcal{F}$ are quadratic and both $\mathbf{s}$ and $\mathbf{t}$ are the null vector, the problem is known as Quadratic Maps Linear Equivalence (QMLE) problem.

The decisional version of IP is not $\mathcal{NP}$-complete [132], but it is known that even IP $1\mathcal{S}$ is at least as difficult as the Graph Isomorphism problem [132]. The IP problem has been investigated by several authors, initially for the security of the C^* scheme [132]. In [134] it was shown that the IP $1\mathcal{S}$ is polynomially solvable for most of the instances with $k \geq N$, and Bouillaguet et al. [36] gave an algorithm with running time of $\mathcal{O}(N^6)$ for random instances of the IP $1\mathcal{S}$ problem, thus fully breaking Patarin's identification scheme [130]. The authors of [132] gave an algorithm for solving the general IP, called To-and-Fro, that runs in time $\mathcal{O}(q^{2N})$ for $q > 2$ and $\mathcal{O}(q^{3N})$ for $q = 2$. It was noted in [38] that the algorithm is only suited for bijective mappings $\mathcal{F}$ and $\mathcal{P}$. Getting rid of the bijectivity constraint has been explored in [37] with the conclusion that the proposed workarounds either have a non-negligible probability of failure or it is unclear how greatly they affect the complexity of the algorithm.

Regarding QMLE, the linear variant of IP, an empirical argument was given in [74] that random inhomogeneous instances are solvable in $\mathcal{O}(N^9)$ time, but a rigorous proof for this case still remains an open problem. Under this assumption, the same paper provides an algorithm of complexity $\mathcal{O}(N^9 q^N)$ for the homogeneous case which is considered the hardest, that was subsequently improved to $\mathcal{O}(N^9 q^{2N/3})$ in [38]. Both works reduce a homogenous instance to an inhomogenous instance and assume the obtained inhomogeneous instance behaves as a random instance. This, however, is a wrong assumption which questions the claimed complexity of the algorithm.

In this work, we will be interested in the homogeneous variant of QMLE, that we denote hQMLE, as the hardest and most interesting instance of QMLE.

Formally, the hQMLE problem is defined as follows.

Problem 4.6. hQMLE $(N, k, \mathbb{F}_q[x_1, \dots, x_N]^k \times \mathbb{F}_q[x_1, \dots, x_N]^k)$:
Input: Two k-tuples of homogeneous multivariate polynomials of degree 2

$$\mathcal{F} = (f_1, f_2, \dots, f_k),\ \mathcal{P} = (p_1, p_2, \dots, p_k) \in \mathbb{F}_q[x_1, \dots, x_N]^k.$$

Question: Find – if any – a map $(\mathbf{S}, \mathbf{T})$ where $\mathbf{S} \in \mathrm{GL}_N(q), \mathbf{T} \in \mathrm{GL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}) = (\mathcal{F}(\mathbf{xS}))\mathbf{T}. \tag{4.4}$$

Interestingly, the case of $k = 1$, which we will call Quadratic Form Equivalence (QFE) has been completely solved for more than 80 years already in the works of Witt [155] and Arf [156]. It is known that every quadratic form is equivalent to a unique canonical diagonal (for odd characteristic) or block diagonal (for even characteristic) form which can be obtained in time $\mathcal{O}(N^3)$. Thus, QFE can also be solved in time $\mathcal{O}(N^3)$ by first calculating the transformations to the canonical forms of the two quadratic forms. If the canonical forms are the same, by composition, one can find the equivalence. If the canonical forms are not the same, the two quadratic forms are not equivalent.

In this work, we also consider a variant of QMLE where $\mathcal{F}$ and $\mathcal{P}$ are bilinear forms. We call this problem Bilinear Maps Linear Equivalence (BMLE). In this variant, $\mathcal{F}$ and $\mathcal{P}$ are k-tuples of homogeneous polynomials of degree 2 in two sets of variables $[x_1, \dots, x_n]$ and $[y_1, \dots, y_m]$, where each monomial is of the form $x_i y_j$. Formally, the BMLE problem is defined as follows.

Problem 4.7. BMLE $(n, m, k, \mathbb{F}_q[x_1, \dots, x_n, y_1, \dots, y_m]^k \times \mathbb{F}_q[x_1, \dots, x_n, y_1, \dots, y_m]^k)$:
Input: Two k-tuples of bilinear forms

$$\mathcal{F} = (f_1, f_2, \dots, f_k),\ \mathcal{P} = (p_1, p_2, \dots, p_k) \in \mathbb{F}_q[x_1, \dots, x_n, y_1, \dots, y_m]^k$$

Question: Find – if any – a triplet $(\mathbf{S}_1, \mathbf{S}_2, \mathbf{T})$ where $\mathbf{S}_1 \in \mathrm{GL}_n(q), \mathbf{S}_2 \in \mathrm{GL}_m(q)$, $\mathbf{T} \in \mathrm{GL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}, \mathbf{y}) = (\mathcal{F}(\mathbf{x}\mathbf{S}_1, \mathbf{y}\mathbf{S}_2))\mathbf{T}. \tag{4.5}$$

The inhomogenous versions of QMLE and BMLE will be referred to as $\mathsf{inhQMLE}$ and $\mathsf{inhBMLE}$ respectively. We write $\mathsf{inh(Q/B)MLE}$ these two problems. when it does not matter if we are referring to the quadratic or the bilinear version.

4.4 How hard is MCE?

In this section we investigate the relation of the MCE problem to other known problems that we notably split in two groups – equivalence problems for systems of multivariate quadratic polynomials and equivalence problems for codes.

4.4.1 Relations to equivalence problems for qaudratic polynomials

We start with establishing a straightforward link between MCE and polynomial equivalence problems by proving that the MCE and BMLE problems are equivalent.

Theorem 4.4. The MCE problem is at least as hard as the BMLE problem.

Proof. In order to prove our claim, we need to show that an oracle $\mathcal{A}$ solving any instance of the MCE problem can be transformed in polynomial time to an oracle $\mathcal{B}$ solving any instance of the BMLE problem.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{BMLE}}(\mathcal{F}, \mathcal{P})$ of BMLE $(n, m, k, \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k \times \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k)$, where $\mathcal{F} = (f_1, f_2, \ldots, f_k)$, $\mathcal{P} = (p_1, p_2, \ldots, p_k) \in \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k$ are k-tuples of bilinear forms. Without loss of generality, we assume $f_1, f_2, \ldots, f_k$ (respectively $p_1, p_2, \ldots, p_k$) to be linearly independent. $\mathcal{B}$ can efficiently construct an instance of the MCE problem as follows.

$\mathcal{B}$ represents the components f_s and p_s, $s \in \{1, \ldots, k\}$ of the mappings $\mathcal{F}$ and $\mathcal{P}$ as $m \times n$ matrices $\mathbf{F}^{(s)}$ and $\mathbf{P}^{(s)}$, where $\mathbf{F}^{(s)}_{i,j}$ equals the coefficient of $x_i y_j$ in f_s and $\mathbf{P}^{(s)}_{i,j}$ equals the coefficient of $x_i y_j$ in p_s. Taking $(\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(k)})$ to be a basis of a matrix code $\mathcal{C}$ and $(\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(k)})$ a basis of a matrix code $\mathcal{D}$, $\mathcal{B}$ obtains an instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ of MCE $(n, m, k, \mathcal{M}^{[k]}_{m,n}(q))$.

$\mathcal{B}$ gives the instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ as an input to $\mathcal{A}$. $\mathcal{A}$ outputs either a solution $(\mathbf{A}, \mathbf{B})$ to the MCE instance (in the case it was a positive instance) or outputs that there is no solution (in the case it was a negative instance). In the latter case, $\mathcal{B}$ immediately outputs: no solution. In the former case, $\mathcal{B}$ constructs the matrices $\mathbf{R}^{(s)} = \mathbf{A}\mathbf{F}^{(s)}\mathbf{B} \in \mathcal{D}$ and solves the following system of equations in the variables $t_{i,j}$:

$$\sum_{j=1}^{k} t_{j,i} \cdot \mathbf{R}^{(j)} = \mathbf{P}^{(i)}, \forall i \in \{1, \ldots, k\} \tag{4.6}$$

The system has always a solution, since $(\mathbf{R}^{(1)}, \dots, \mathbf{R}^{(k)})$ is a basis of the code $\mathcal{D}$.

$\mathcal{B}$ sets $\mathbf{T} = (t_{i,j})$, and outputs $(\mathbf{A}, \mathbf{B}^\top, \mathbf{T})$ as the solution to $\mathcal{I}_{\mathsf{BMLE}}(\mathcal{F}, \mathcal{P})$. $\mathcal{B}$ succeeds whenever $\mathcal{A}$ succeeds and the reduction runs in time $\mathcal{O}(k^6)$.

□

Theorem 4.5. BMLE is at least as hard as MCE.

Proof. We proceed similarly as in the other direction – Given an oracle $\mathcal{A}$ solving any instance of BMLE, we can construct in polynomial time an oracle $\mathcal{B}$ with access to $\mathcal{A}$ that can solve any instance of MCE.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ of MCE $(n, m, k, \mathcal{M}^{[k]}_{m,n}(q))$. $\mathcal{B}$ takes arbitrary bases $(\mathbf{C}^{(1)}, \dots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \dots, \mathbf{D}^{(k)})$ of the codes $\mathcal{C}$ and $\mathcal{D}$ respectively. For each of the matrices $\mathbf{C}^{(s)}$, $\mathcal{B}$ constructs the bilinear forms $c_s(\mathbf{x}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n} \mathbf{C}^{(s)}_{ij} x_i y_j$ and for the matrices $\mathbf{D}^{(s)}$ the bilinear forms $d_s(\mathbf{x}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n} \mathbf{D}^{(s)}_{ij} x_i y_j, \forall s, 1 \leqslant s \leqslant k$. Taking $\mathcal{F} = (c_1, c_2, \dots, c_k)$ and $\mathcal{P} = (d_1, d_2, \dots, d_k)$ we obtain an instance $\mathcal{I}_{\mathsf{BMLE}}(\mathcal{F}, \mathcal{P})$ of BMLE $(n, m, k, \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k \times \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k)$.

$\mathcal{B}$ queries $\mathcal{A}$ with the instance $\mathcal{I}_{\mathsf{BMLE}}(\mathcal{F}, \mathcal{P})$ and $\mathcal{A}$ outputs a solution $(\mathbf{S}_1, \mathbf{S}_2, \mathbf{T})$ to the BMLE instance, or no solution if there isn't any. In the first case, this immediately gives a solution $(\mathbf{S}_1, \mathbf{S}_2^\top)$ to the MCE instance. In the second case, there is no solution to the MCE instance. □

In order to prove the connection of MCE to the more general problem hQMLE we first need to establish some properties of matrix codes.

Lemma 4.6. Let $\mathcal{C}$ and $\mathcal{D}$ be matrix codes generated by the bases $= (\mathbf{C}_1, \dots, \mathbf{C}_k)$ and $(\mathbf{D}_1, \dots, \mathbf{D}_k)$ of (skew-)symmetric matrices, and assume that $\mathcal{C}$ and $\mathcal{D}$ have trivial automorphism groups. Then $\mathcal{C}$ is equivalent to $\mathcal{D}$ if and only if $\mathcal{C}$ is congruent to $\mathcal{D}$.

Proof. Clearly, by definition if $\mathcal{C}$ is congruent to $\mathcal{D}$, then $\mathcal{C}$ is equivalent to $\mathcal{D}$.

For the opposite direction, let $\mathcal{C}$ be equivalent to $\mathcal{D}$. Then there exist non-singular matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{T}$ such that

$$\sum_{i=1}^{k} t_{j,i} \mathbf{D}_i = \mathbf{A} \mathbf{C}_j \mathbf{B}$$

Since $\mathbf{C}_i$ and $\mathbf{D}_i$ are (skew-)symmetric the last rewrites as

$$\sum_{i=1}^{k} t_{j,i} \mathbf{D}_i = \mathbf{B}^\top \mathbf{C}_j \mathbf{A}^\top$$

Combining the two, and since $\mathbf{A}$ and $\mathbf{B}$ are non-singular, we obtain

$$\mathbf{C}_j = \mathbf{A}^{-1} \mathbf{B}^\top \mathbf{C}_j \mathbf{A}^\top \mathbf{B}^{-1}$$

The automorphism group being trivial implies $\mathbf{A} = \lambda \mathbf{B}^\top$ for some $\lambda \in \mathbb{F}_q$ which in turn implies that $\mathcal{C}$ is congruent to $\mathcal{D}$. □

Remark 4.1. The result of Lemma 4.6 has already been known for algebraically closed fields of non-even characteristic [19, 148]. Since finite fields are not algebraically closed, this result is not useful in our context. On the other hand, requiring a trivial automorphism group for the codes is not a huge restriction, and we typically expect the automorphism group to be trivial for randomly chosen matrix codes. Specifically for cryptographic purposes with regards to MCE, one wants the orbit of $\mathcal{C}$ to be maximal under the action of suitable isometries, which happens when the automorphism group of $\mathcal{C}$ is trivial. Similar requirements for trivial or small automorphism groups occur in the Hamming metric, where it is known that without this requirement there might exist weak keys [72, 73].

Theorem 4.7. Let $\mathcal{T}$ denote the subset of $\mathcal{M}^{[k]}_{m,n}(q)$ of k-dimensional matrix codes of symmetric matrices with trivial automorphism groups. Further, let $\mathcal{T}'$ denote the subset of $\mathbb{F}_q[x_1, \ldots, x_N]^k$ of k-tuples of polynomials with trivial automorphism groups.

The MCE $(\mathcal{T})$ problem is at least as hard as the hQMLE $(\mathcal{T}')$ problem

Proof. We perform the reduction in a similar manner as previously.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ of hQMLE $(N, k, \mathcal{T}')$, where $\mathcal{F} = (f_1, f_2, \ldots, f_k)$, $\mathcal{P} = (p_1, p_2, \ldots, p_k) \in [x_1, \ldots, x_N]$ are k-tuples of linearly independent quadratic forms from $\mathcal{T}'$. $\mathcal{B}$ can efficiently construct an instance of the MCE $(N, N, k, \mathcal{T})$ problem as follows.

$\mathcal{B}$ forms the $N \times N$ symmetric matrices $\mathbf{F}^{(s)}$ and $\mathbf{P}^{(s)}$ associated to the components f_s and p_s, $s \in \{1, \ldots, k\}$ of the mappings $\mathcal{F}$ and $\mathcal{P}$. Taking $(\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(k)})$ to be a basis of a matrix code $\mathcal{D}$ and $(\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(k)})$ a basis of a matrix code $\mathcal{C}$, $\mathcal{B}$ obtains an instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ of MCE. Per assumption, the matrix codes $\mathcal{C}$ and $\mathcal{D}$ have trivial automorphism groups, hence the instance is from MCE $(N, N, k, \mathcal{T})$.

$\mathcal{B}$ queries $\mathcal{A}$ with the instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$. , $\mathcal{A}$ answers with a solution $(\mathbf{A}, \mathbf{B})$ to the MCE instance if it is positive, and no solution otherwise. In the former case, from Lemma 4.6, since the matrices are symmetric, $\mathbf{A} = \mathbf{B}^\top$. Now, $\mathcal{B}$ applies the change of variables $\mathbf{xA}$ to $\mathcal{F}$ and obtains $\mathcal{R}(\mathbf{x}) = \mathcal{F}(\mathbf{xA})$. It then solves the system

$$\sum_{j=1}^{k} t_{j,s} \cdot r_j = p_s, \forall s \in \{1, \ldots, k\} \tag{4.7}$$

The system has a solution if $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ is a positive instance. This is always the case in odd characteristic, because there is a one-to-one correspondence between polynomials and their symmetric matrix representation. Over characteristic 2, it may happen that the $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ is not a positive instance while its symmetric matrix representation $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ is. In this case, the system (4.7) does not have a solution and $\mathcal{B}$ outputs no solution.

If the system has a solution, $\mathcal{B}$ sets $\mathbf{T} = (t_{i,j})$, and outputs $(\mathbf{A}, \mathbf{T})$ as the solution to $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$. $\mathcal{B}$ succeeds whenever $\mathcal{A}$ succeeds and the reduction takes time $\mathcal{O}(k^6)$.

□

For the following theorem, we define the symmetric matrix representation of a matrix code $\mathcal{C}$ as the code $\{ \begin{bmatrix} \mathbf{0} & \mathbf{C}^\top \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \mid \mathbf{C} \in \mathcal{C}\}$.

Theorem 4.8. Let $\mathcal{T}_s$ denote the subset of $\mathcal{M}^{[k]}_{m,n}(q)$ of k-dimensional matrix codes whose symmetric matrix representation has a trivial automorphism group. Similarly, let $\mathcal{T}'_s$ denote the subset of $\mathbb{F}_q[x_1, \dots, x_N]^k$ of k-tuples of polynomials with trivial automorphism groups.

The hQMLE $(\mathcal{T}'_s)$ problem is at least as hard as the MCE $(\mathcal{T}_s)$ problem.

Proof. We show that given any oracle $\mathcal{A}$ that solves the hQMLE $(\mathcal{T}'_s)$ problem there exists an oracle $\mathcal{B}$ running in polynomial time that solves the MCE $(\mathcal{T}_s)$.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ of MCE $(n, m, k, \mathcal{T}_s)$. $\mathcal{B}$ can efficiently construct an instance of the hQMLE $(n+m, k, \mathcal{T}'_s)$ problem as follows.

$\mathcal{B}$ fixes bases $(\mathbf{D}^{(1)}, \dots, \mathbf{D}^{(k)})$ of the code $\mathcal{D}$ and $(\mathbf{C}^{(1)}, \dots, \mathbf{C}^{(k)})$ of the code $\mathcal{C}$. For each of the matrices $\mathbf{C}^{(s)}$, $\mathcal{B}$ constructs the quadratic forms $c_s(\mathbf{x}) = \sum\limits_{1 \leqslant i \leqslant m, m+1 \leqslant j \leqslant m+n} \mathbf{C}^{(s)}_{ij} x_i x_j$ and for the matrices $\mathbf{D}^{(s)}$ the quadratic forms $d_s(\mathbf{x}) = \sum\limits_{1 \leqslant i \leqslant m, m+1 \leqslant j \leqslant m+n} \mathbf{D}^{(s)}_{ij} x_i x_j, \forall s, 1 \leqslant s \leqslant k$, where $\mathbf{x} = (x_1, \dots, x_{m+n})$. Taking $\mathcal{F} = (c_1, c_2, \dots, c_k)$ and $\mathcal{P} = (d_1, d_2, \dots, d_k)$ $\mathcal{B}$ obtains an instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ of hQMLE $(n+m, k, \mathcal{T}'_s)$.

$\mathcal{B}$ queries $\mathcal{A}$ with the instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ which outputs a solution $(\mathbf{S}, \mathbf{T})$ to the hQMLE instance.

We argue that this solution can be transformed to a solution to the MCE instance, if it is a positive instance. The symmetric matrix representation of the codes $\mathcal{C}$ and $\mathcal{D}$ is given by

$$\begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(i)})^\top \\ \mathbf{D}^{(i)} & \mathbf{0} \end{bmatrix} \text{ and } \begin{bmatrix} \mathbf{0} & (\mathbf{C}^{(i)})^\top \\ \mathbf{C}^{(i)} & \mathbf{0} \end{bmatrix}, i \in \{1, \dots, k\}. \tag{4.8}$$

The solution $(\mathbf{S}, \mathbf{T})$ means

$$\sum \tilde{t}_{i,j} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(j)})^\top \\ \mathbf{D}^{(j)} & \mathbf{0} \end{bmatrix} = \mathbf{S} \begin{bmatrix} \mathbf{0} & (\mathbf{C}^{(i)})^\top \\ \mathbf{C}^{(i)} & \mathbf{0} \end{bmatrix} \mathbf{S}^\top, i \in \{1, \dots, k\}. \tag{4.9}$$

If the given MCE instance is positive, then there exist matrices $\mathbf{A}, \mathbf{B}, \mathbf{L}$ such that $\mathbf{A}\mathbf{C}_i\mathbf{B} = \sum_j l_{i,j} \mathbf{D}_j$. This implies

$$\sum l_{i,j} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(j)})^\top \\ \mathbf{D}^{(j)} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{0} & (\mathbf{C}^{(i)})^\top \\ \mathbf{C}^{(i)} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^\top \end{bmatrix}, i \in \{1, \dots, k\}. \tag{4.10}$$

The last two imply

$$\sum \lambda_{i,j} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(j)})^\top \\ \mathbf{D}^{(j)} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \mathbf{S}^{-1} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(i)})^\top \\ \mathbf{D}^{(i)} & \mathbf{0} \end{bmatrix} \mathbf{S}^{-\top} \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^\top \end{bmatrix}, i \in \{1, \dots, k\}. \tag{4.11}$$

By assumption, the automorphism group of the $\begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(i)})^\top \\ \mathbf{D}^{(i)} & \mathbf{0} \end{bmatrix}$ matrices is trivial, which means $\mathbf{S}$ necessarily equals $\begin{bmatrix} \mathbf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix}$ up to scalar multiplication. For such an $\mathbf{S}$, the MCE solution can immediately be extracted. $\mathcal{B}$ then outputs the extracted solution.

If on the other hand, $\mathbf{S}$ is not of such block-diagonal form, $\mathcal{B}$ outputs no solution, as this implies the instance is not positive.

□

Remark 4.2. Using the above reduction between MCE and hQMLE, we can reduce the MCE base problem to and from a special case of IP known as IP $1\mathcal{S}$. Interestingly, Perret [134] shows IP $1\mathcal{S}$ is polynomially solvable for most instances $k \geq N$, and later work [36] gives an algorithm with running time of $\mathcal{O}(N^6)$ for most random instances, although no rigorous proof that bounds the complexity of the problem to polynomial was given. This nevertheless implies that the MCE base problem can practically be solved in polynomial time for most cryptographically interesting parameters.

4.4.2 Relations to equivalence problems for linear codes

In this section, we show that MCE is at the heart of various code equivalence problems. Equivalence problems for different metrics, such as the Hamming metric or the sum-rank metric, reduce to MCE, making the hardness analysis of MCE the more exciting.

Hamming code equivalence.

Codes $\mathcal{C} \subset \mathbb{F}_q^n$ equipped with the *Hamming metric* have isometries of the form

$$\tau : (c_1, \dots, c_n) \mapsto (\alpha_1 c_{\pi^{-1}(1)}, \dots, \alpha_n c_{\pi^{-1}(n)}), \quad \alpha_i \in \mathbb{F}_q^*, \ \pi \in S_n. \tag{4.12}$$

From this, we define *Hamming code equivalence* (HCE) as the problem of finding an isometry between two Hamming codes $\mathcal{C}$ and $\mathcal{D}$.

Problem 4.8. HCE(k, n):
Input: Two k-dimensional Hamming codes $\mathcal{C}, \mathcal{D} \subset \mathbb{F}_q^n$
Question: Find – if any – $\alpha \in \mathbb{F}_q^{*n}, \pi \in S_n$ such that $\alpha\pi(\mathbf{c}) \in \mathcal{D}$ holds for all $\mathbf{c} \in \mathcal{C}$.

The subproblem where α is trivial is called the *monomial equivalence problem.*

It is easy to turn an HCE instance into a MCE instance [53], given the description of isometries in Equation (4.12). First, define $\Phi : \mathbb{F}_q^n \to \mathcal{M}_n(\mathbb{F}_q)$ by

$$\mathbf{x} = (x_1, \ldots, x_n) \mapsto \begin{pmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{pmatrix}.$$

The map Φ is an isometry from the Hamming metric to the rank metric: codewords with weigh t are mapped to matrices of rank t. From this, we quickly get the reduction: Writing π as a matrix $\mathbf{P} \in \mathrm{GL}_n(q)$, Φ translates a Hamming isometry τ to a rank-metric isometry by

$$\Phi(\tau) : \Phi(\mathbf{x}) \mapsto \mathbf{P}^{-1}\Phi(\mathbf{x})\mathbf{A}\mathbf{P}, \quad \text{where } \mathbf{A} = \begin{pmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_n \end{pmatrix} \in \mathrm{GL}_n(q).$$

A second reduction from HCE to MCE is given later in [53], which concerns the search variant of the problem, and is more explicit. Both reductions, however, do not help with solving HCE in practice: both the permutational ($\mathbf{A}$ is trivial) and the linear variant of code equivalence in the Hamming metric have algorithms [16, 135] that perform much better for an HCE instance τ than the algorithms we propose for solving $\Phi(\tau)$ as an MCE instance.

Sum-rank code equivalence.

The *sum-rank metric* [128] is a metric that is gaining in popularity in coding theory. It is commonly given as a generalization of the vector-rank metric, but one can also define a variant that generalizes matrix-rank metric. We will reduce both vector and matrix sum-rank equivalence problems to MCE. The idea is the same as for HCE, we find the right isometry from sum-rank metric to rank metric to get the reduction.

Definition 4.9. Let n be partitioned as $n = n_1 + \ldots + n_\ell$. Let $\mathbf{v}^{(i)} = (v_1^{(i)}, \ldots, v_{n_i}^{(i)}) \in \mathbb{F}_{q^m}^{n_i}$ and $\mathbf{v} = (\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(\ell)}) \in \mathbb{F}_{q^m}^n$. Let Γ be a basis for $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Then the *vector sum-rank* of $\mathbf{v}$ is defined as

$$\mathrm{SumRank}(\mathbf{v}) := \sum_{i=1}^{\ell} \mathrm{Rank}\, \Gamma(\mathbf{v}^{(i)}).$$

Let m be partitioned as $m = m_1 + \ldots + m_\ell$. Let $\mathbf{V}^{(i)} \in \mathcal{M}_{m_i \times n_i}(\mathbb{F}_q)$ and $\mathbf{V} = (\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(\ell)})$. Then the *matrix sum-rank* of $\mathbf{V}$ is defined as

$$\mathrm{SumRank}(\mathbf{V}) = \sum_{i=1}^{\ell} \mathrm{Rank}\, \mathbf{V}^{(i)}.$$

The sum-rank generalizes both the Hamming metric and the rank metric: taking $\ell = n$ gives the Hamming metric, whereas $\ell = 1$ gives the rank metric. We define isometries again as maps that preserve the sum-rank. Sum-rank isometries are simple generalisations of rank isometries (see Problem 4.4).

Proposition 4.10 ([6, Thm. 3.7])**.** Isometries with respect to the vector sum-rank metric are given by vector rank isometries $\mu^{(i)} : \mathbf{x}^{(i)} \mapsto \alpha^{(i)}\mathbf{x}^{(i)}\mathbf{B}^{(i)}$ per 'block' with $\alpha^{(i)} \in \mathbb{F}_{q^m}^*$ and $\mathbf{B}^{(i)} \in \mathrm{GL}_{n_i}(q)$, and suitable permutations π of such blocks if $n_i = n_j$ for $i \neq j$, so

$$\mu : (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell)}) \mapsto (\alpha^{(1)}\mathbf{x}^{\pi^{-1}(1)}\mathbf{B}^{(1)}, \ldots, \alpha^{(\ell)}\mathbf{x}^{\pi^{-1}(\ell)}\mathbf{B}^{(\ell)})$$

is a general description of a vector sum-rank isometry.

Generalizing to matrix sum-rank codes is achieved by simply replacing $\alpha^{(i)} \in \mathbb{F}_{q^m}^*$ with $\mathbf{A}^{(i)} \in \mathrm{GL}_{m_i}(q)$ [124, Prop. 4.25]. This gives us the *Vector Sum-Rank Code Equivalence* (VSRCE) and *Matrix Sum-Rank Code Equivalence* (MSRCE) problems.

Problem 4.9. VSRCE (n, m, k):
Input: Two k-dimensional vector sum-rank codes $\mathcal{C}, \mathcal{D} \subset \mathbb{F}_{q^m}^n$
Question: Find – if any – $\alpha^{(i)} \in \mathbb{F}_{q^m}^*, \mathbf{B}^{(i)} \in \mathrm{GL}_{n_i}(q)$ and a permuation π such that for all $\mathbf{c} \in \mathcal{C}$, it holds that $\mu(\mathbf{c}) \in \mathcal{D}$.

Problem 4.10. MSRCE(n, m, k):
Input: Two k-dimensional matrix sum-rank codes $\mathcal{C}, \mathcal{D} \subset (\mathcal{M}_{m_i \times n_i}(\mathbb{F}_q))_i$
Question: Find – if any – $\mathbf{A}^{(i)} \in \mathrm{GL}_{m_i}(q), \mathbf{B}^{(i)} \in \mathrm{GL}_{n_i}(q)$ and a permuation π such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mu(\mathbf{C}) \in \mathcal{D}$.

In order to give a reduction to MCE, we use the same idea as for HCE. First, we define a 'nice' map $\Psi : \mathbb{F}_q^n \to \mathcal{M}_{\ell \cdot m \times n}(\mathbb{F}_q)$ by

$$\mathbf{x} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell)}) \mapsto \begin{pmatrix} \mathrm{Mat}(\mathbf{x}^{(1)}) & & \\ & \ddots & \\ & & \mathrm{Mat}(\mathbf{x}^{(\ell)}) \end{pmatrix}.$$

It is clear that Ψ is an isometry from the vector sum-rank metric to the rank metric, as it preserves the weight. We get the following reduction.

Theorem 4.11. Let $\mathcal{T}$ denote the subset of $\mathcal{M}_{m,n}^{[k]}(q)$ of k-dimensional matrix codes with trivial automorphism groups. Let $\mathcal{T}'$ denote the subset of k-dimensional vector sum-rank codes that are in the preimage $\Psi^{-1}(\mathcal{T})$. Then MCE $(\mathcal{T})$ is at least as hard as VSRCE $(\mathcal{T}')$.

Proof. Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{VSRCE}}(\mathcal{C}, \mathcal{D})$ of VSRCE $(n, m, k, \mathcal{T}')$, where $\mathcal{C}$ and $\mathcal{D}$ are k-dimensional vector sum-rank codes. $\mathcal{B}$ can efficiently construct an instance of the MCE $(\mathcal{T})$ problem as follows. By writing the permutation π of the 'blocks' by a matrix representation $\mathbf{P}$, $\mathcal{B}$ can translate a vector sum-rank isometry μ into a matrix code isometry $\Psi(\mu)$ by

$$\Psi(\mu) : \Psi(\mathbf{x}) \mapsto \mathbf{P}^{-1}\mathbf{A}\Psi(\mathbf{x})\mathbf{B}\mathbf{P} \quad \text{where } \mathbf{A} = \begin{pmatrix} \alpha^{(1)} & & \\ & \ddots & \\ & & \alpha^{(\ell)} \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \mathbf{B}^{(1)} & & \\ & \ddots & \\ & & \mathbf{B}^{(\ell)} \end{pmatrix}$$

with $\mathbf{A} \in \mathrm{GL}_\ell(q^m)$, $\mathbf{B} \in \mathrm{GL}_n(q)$. Hence, $\Psi(\mu)$ is an instance of MCE $(n, m, k, \mathcal{T})$, with which $\mathcal{B}$ queries $\mathcal{A}$. $\mathcal{A}$ outputs a solution $(\mathbf{A}', \mathbf{B}')$ to this MCE $(\mathcal{T})$ instance. As the automorphism group is trivial, $\mathcal{B}$ computes $\lambda\mathbf{A}' = \mathbf{P}^{-1}\mathbf{A}$ and $\lambda\mathbf{B}' = \mathbf{B}\mathbf{P}$ for $\lambda \in \mathbb{F}_q$, and therefore solves the $\mathcal{I}_{\mathsf{VSRCE}}$ instance. □

From vector sum-rank code equivalence to matrix sum-rank code equivalence is only a small step. Given a partition $m = m_1 + \ldots + m_\ell$, the map we need is only slightly different from Ψ, namely $\tilde{\Psi} : \left(\mathcal{M}_{m_i \times n_i}(\mathbb{F}_q)\right)_i \to \mathcal{M}_{m\times n}(\mathbb{F}_q)$ by

$$\mathbf{X} = (\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(\ell)}) \mapsto \begin{pmatrix} \mathbf{X}^{(1)} & & \\ & \ddots & \\ & & \mathbf{X}^{(\ell)} \end{pmatrix}.$$

Theorem 4.12. Let $\mathcal{T}$ denote the subset of $\mathcal{M}^{[k]}_{m,n}(q)$ of k-dimensional matrix codes with trivial automorphism groups. Let $\mathcal{T}'$ denote the subset of k-dimensional matrix sum-rank codes that are in the preimage $\tilde{\Psi}^{-1}(\mathcal{T})$. Then MCE $(\mathcal{T})$ is at least as hard as MSRCE $(\mathcal{T}')$.

Proof. This is a simple generalization of Theorem 4.11: Replace $\alpha^{(i)}$ by $\mathbf{A}^{(i)} \in \mathrm{GL}_{m_i}(q)$ so that $\mathbf{A} \in \mathrm{GL}_m(q)$. Then again, for a matrix sum-rank μ we get $\tilde{\Psi}(\mu)$ by $\Psi(\mathbf{x}) \mapsto \mathbf{P}^{-1}\mathbf{A}\Psi(\mathbf{x})\mathbf{B}\mathbf{P}$ as an MCE $(\mathcal{T})$ instance. □

The link between such MCE instances $\Psi(\mu)$ coming from vector sum-rank and $\tilde{\Psi}(\mu)$ coming from matrix sum-rank is given by a representation $\rho : \mathbb{F}^*_{q^m} \to \mathrm{GL}_m(q)$. We map a vector sum-rank instance to a matrix sum-rank instance by $\mathbf{A}^{(i)} = \rho(\alpha^{(i)})$, so that $\mathbf{A} \in \mathrm{GL}_{\ell\cdot m}(q)$.

To show the equivalences between the rank and sum-rank instances, we need to show that an MCE instance is also an MSRCE instance. But this is trivial: the sum-rank metric generalizes the rank metric, thus an MCE instance is an MSRCE instance with $\ell = 1$. Hence, we get the following theorem for free.

Theorem 4.13. MSRCE is at least as hard as MCE.

4.5 Solving Matrix Code Equivalence

In this section, we analyze the complexity of solving an instance of MCE (n, m, k). We start by establishing a useful lemma.

Lemma 4.14. An MCE (n, m, k) instance can in polynomial time be turned into an MCE $(\sigma(n), \sigma(m), \sigma(k))$ instance for any permutation σ on the set $\{n, m, k\}$. Furthermore, they are either both positive, or both negative instances.

Proof. Let $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ be a given MCE (n, m, k) instance. Let $(\mathbf{C}^{(1)}, \dots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \dots, \mathbf{D}^{(k)})$ be bases of the codes $\mathcal{C}$ and $\mathcal{D}$ respectively. Without loss of generality, we will turn this instance into an MCE (m, k, n) instance (the rest can be done analogously). We set $\bar{\mathbf{C}}^{(i)}_{j,t} = \mathbf{C}^{(t)}_{i,j}$, $\bar{\mathbf{D}}^{(i)}_{j,t} = \mathbf{D}^{(t)}_{i,j}$ and we take $(\bar{\mathbf{C}}^{(1)}, \dots, \bar{\mathbf{C}}^{(n)})$ and $(\bar{\mathbf{D}}^{(1)}, \dots, \bar{\mathbf{D}}^{(n)})$ to be the bases of the codes $\bar{\mathcal{C}}$ and $\bar{\mathcal{D}}$ respectively. Clearly, $\bar{\mathcal{C}}$ and $\bar{\mathcal{D}}$ are equivalent if and only if $\mathcal{C}$ and $\mathcal{D}$ are equivalent. □

Without loss of generality, and with Lemma 4.14 in mind, we assume $m = \min\{m, n, k\}$.

As a baseline we have a straightforward algorithm that uses a result from [53] that MCRE can be solved in polynomial time. By enumerating either $\mathbf{A}$ or $\mathbf{B}$, we obtain an instance of MCRE. This means the dominating complexity is the enumeration resulting in an overall complexity of $\tilde{\mathcal{O}}(q^{m^2})$ for MCE.

The approach we outline in the section makes use of the reduction of MCE to hQMLE (see Theorem 4.8). This means that we use techniques already applied for solving hQMLE, but generalize and improve them by making use of the specific structure that MCE instances show when viewed as hQMLE instances.

4.5.1 Solving MCE as QMLE

At Eurocrypt 2013, Bouillaguet et al. [38] proposed an algorithm for solving hQMLE using techniques from graph theory. Their main idea was to reduce the homogeneous case to the inhomogeneous case, which they assume is efficiently solvable (e.g. using the heuristic algebraic approach of [74]). Starting from an instance of hQMLE, they build two exponentially-large graphs that correspond to the given maps $\mathcal{F}$ and $\mathcal{P}$ such that, finding an isomorphism between the two graphs is equivalent to finding an isomorphism between the two quadratic maps. Since the graphs are exponentially large, a technique is provided to *walk* through the graphs without constructing them. Walking through the graphs consists of finding adjacent vertices and computing the degree of a vertex, both in polynomial time. The algorithm consists in finding pairs of vertices from the first and the second graph that have the same degree and making queries to an inhomogenous QMLE solver. If the solver finds an isomorphism by which two vertices are related, then the isomorphism between the two graphs, and thus the isomorphism between the two quadratic maps, is found.

4.5.2 First algorithm for solving MCE

The algorithm for solving hQMLE from [38] considers a graph arising from the differential of a given polynomial map – a vertex $\mathbf{a}$ is connected to all the vertices that vanish at the differential at $\mathbf{a}$. It is, however, not entirely clear how the property we choose to construct such graphs impacts the complexity of the algorithm. We revisit the algorithm, and show how it can be generalized, i.e. abstracted from the property used in [38], under certain conditions. In this section we present this generalization – a birthday-based algorithm for finding an isomorphism between two objects when a specific solver exists. In this form, it can be applied to a broader type of equivalence problems, using more general invariants, here implemented as a predicate $\mathbb{P}$.

Let S_1 and S_2 be subsets of a universe U of equal size N. Algorithm 1 finds an equivalence function $\varphi : S_1 \to S_2$. We assume there exists a predicate $\mathbb{P} : U \to \{\top, \bot\}$ that can be computed in polynomial time, and we denote the cost $C_{\mathbb{P}}$. We assume $\mathbb{P}$ is invariant under the equivalence φ, i.e. $\mathbb{P}(x) = \top \leftrightarrow \mathbb{P}(\varphi(x)) = \top$. Let $U_\top = \{x \in U \mid \mathbb{P}(x) = \top\}$, and $d = |U_\top|/|U|$. We will call d the *density* of the predicate $\mathbb{P}$ and we assume the density on S_1 and S_2 is approximately equal to d. We further assume the existence of an algorithm FindFunction, that given $x \in S_1, y \in S_2$ returns φ if $y = \varphi(x)$ and $\bot$ otherwise. We denote the cost of a query to FindFunction by C_{FF}.

Algorithm 1 General Birthday-based Equivalence Finder

```
1: procedure SampleSet($S, \mathbb{P}, \ell$)
2:     $L \leftarrow \emptyset$
3:     repeat
4:         $a \xleftarrow{\$} S$
5:         if $\mathbb{P}(a)$ then $L \leftarrow L \cup \{a\}$
6:     until $|L| = \ell$
7:     return $L$

8: procedure CollisionFind($S_1, S_2$)
9:     $L_1 \leftarrow$ SampleSet($S_1, \mathbb{P}, \ell$)
10:    $L_2 \leftarrow$ SampleSet($S_2, \mathbb{P}, \ell$)
11:    for all $(a, b) \in L_1 \times L_2$ do
12:        $\varphi \leftarrow$ FindFunction($a, b$)
13:        if $\varphi \neq \bot$ then
14:            return solution $\varphi$
15:    return $\bot$
```

Lemma 4.15. For a fixed success probability of $1 - 1/e$, Algorithm 1 performs on average $\mathcal{O}(\sqrt{N/d})$ operations in SampleSet, queries FindFunction at most $d \cdot N$ times.

The optimal value for d, up to a polynomial factor, is $d = N^{-1/3} \cdot C_{\text{FF}}^{-2/3}$, for which the total time complexity of the algorithm is $\mathcal{O}(N^{\frac{2}{3}} \cdot C_{\text{FF}}^{\frac{1}{3}})$ and the memory complexity is $\mathcal{O}(N^{\frac{1}{3}} C_{\text{FF}}^{-\frac{1}{3}})$. If FindFunction runs in polynomial time, this reduces to time complexity of $\tilde{\mathcal{O}}(N^{\frac{2}{3}})$ and memory complexity of $\mathcal{O}(N^{\frac{1}{3}})$.

Proof. First note that the expected number of elements in S_1 and S_2 such that $\mathbb{P}(x)$ holds is equal to dN by the definition of the density d. By the birthday paradox, it is enough to take lists of size $\ell = \sqrt{d \cdot N}$, to be sure that with probability of $1 - \frac{1}{e}$ FindFunction returns a solution [157]. With this length of the lists, the number of queries to FindFunction is dN. On the other hand, the number of samples needed to build the list L_1 (resp. L_2) of elements $a \in S_1$ (resp. $b \in S_2$) such that $\mathbb{P}(a)$ (resp. $\mathbb{P}(b)$) holds is ℓ/d, which gives a complexity of $\mathcal{O}(\sqrt{N/d})$ to build these lists L_i.

The total running time is optimal when these two quantities $\sqrt{N/d}$ and $d \cdot N \cdot C_{\text{FF}}$ are equal, which holds when $d = N^{-1/3} \cdot C_{\text{FF}}^{-2/3}$. Such a density gives complexity of $\mathcal{O}(N^{\frac{2}{3}} \cdot C_{\text{FF}}^{\frac{1}{3}})$ for SampleSet and at most $N^{\frac{2}{3}}$ queries to FindFunction. If C_{FF} is polynomial, this gives a total time complexity of $\tilde{\mathcal{O}}(N^{\frac{2}{3}})$. The memory requirements of the algorithm correspond to the size of the lists L_i. This results in a memory complexity of $\mathcal{O}(N^{\frac{1}{3}} C_{\text{FF}}^{-\frac{1}{3}})$, or $\mathcal{O}(N^{\frac{1}{3}})$ if C_{FF} is polynomial. □

Remark 4.3. The success probability in Lemma 4.15 is chosen rather arbitrarily, mostly for practical verification of the algorithm's correctness. It can be increased to any value $1 - 1/c$ for a positive constant c by appropriately building lists that are larger only by a constant factor compared to the case treated in Lemma 4.15. The overall complexity only differs by a constant factor, i.e., does not change asymptotically.

As said earlier, the algorithm presented in [38] is a special case of Algorithm 1. Their algorithm can be seen as an instantiation of Algorithm 1 by defining $G_{\mathcal{F}}$ (resp. $G_{\mathcal{P}}$) to be the linearity graph of $\mathcal{F}$ (resp. $\mathcal{P}$), where a vertex $\mathbf{a}$ is connected to all vertices $\mathbf{x}$ such that $D_{\mathbf{a}}\mathcal{F}(\mathbf{x}) = 0$ (resp. $D_{\mathbf{a}}\mathcal{P}(\mathbf{x}) = 0$), taking the predicate $\mathbb{P}_{\kappa}(\mathbf{a}) : \dim\ker D_{\mathbf{a}}\mathcal{F} = \kappa$ on the universe $\mathcal{M}_{k,N}(q)$, and taking for FindFunction the assumed polynomial-time solver from [74] for inhQMLE. Finding a collision (α, β) such that $\beta = \alpha S$ makes the instance $\mathcal{P}(\mathbf{x}+\alpha) = \mathcal{F}(\mathbf{x}S+\beta)\mathbf{T}$ an inhomogeneous instance by defining $\mathcal{P}'(\mathbf{x}) = \mathcal{P}(\mathbf{x}+\alpha)$ and $\mathcal{F}'(\mathbf{x}) = \mathcal{F}(\mathbf{x}+\beta)$. Running FindFunction on $\mathcal{P}'$ and $\mathcal{F}'$ then returns $\mathbf{S}$ and $\mathbf{T}$. In this case, Lemma 4.15 gives the precise result from [38, Thm. 1], which we present as a corollary to our Lemma 4.15, for completeness.

Corollary 4.1. Assuming a polynomial-time solver for the inhomogenous case of QMLE, an hQMLE (N, N) instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$ can be solved with complexity and number of queries equal to $\tilde{\mathcal{O}}(q^{\frac{2}{3}N})$ with a success probability of $1 - 1/c$ for any $c > 0$ and a memory complexity of $\mathcal{O}(q^{\frac{1}{3}N})$.

Proof. Let $G_{\mathcal{F}}$ (i.e. $G_{\mathcal{P}}$) be the linearity graph of $\mathcal{F}$ (i.e. $\mathcal{P}$), where a vertex $\mathbf{a}$ is connected to all $\mathbf{x}$ such that $D_{\mathbf{a}}\mathcal{F}(\mathbf{x}) = 0$ (i.e. $D_{\mathbf{x}}\mathcal{P}(\mathbf{a}) = 0$). We use the predicate $\mathbb{P}_{\kappa}(\mathbf{a}) : \dim\ker D_{\mathbf{a}}\mathcal{F} = \kappa$ we have that $\deg(\mathbf{a}) = q^{\kappa}$. The density of the predicate d_{κ} in the universe of $N \times N$ matrices is independent of $\mathcal{F}$ and $\mathcal{P}$, and is therefore the same as the density of linear maps with kernel of dimension κ. Thus, d_{κ} is approximately a monotonic decreasing function in κ, going from 1 to 0. Hence, by Lemma 4.15, there exists some optimal κ for which we get that $d_{\kappa} \approx |G_{\mathcal{P}}|^{-1/3} = q^{-N/3}$, which gives a total time complexity of $q^{\frac{2}{3}N}$ and a memory complexity of $q^{\frac{1}{3}N}$. □

Remark 4.4. The assumption on a polynomial-time solver in [38] turns out to be too strong: such a solver exists for random instances, however, for inhQMLE instances as obtained in Corollary 4.1 the running time is probably not polynomial [37]. Nevertheless, the algorithm and result are valid, but require a different rebalancing depending on C_{FF}. Section 4.6 analyzes C_{FF} in detail for different instances.

To apply this approach to MCE instances, we need to generalize to the case of N not necessarily equal to k. For an MCE (n, m, k) instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$, we get an hQMLE $(n+m, k)$ instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ by Theorem 4.8. We take again the predicate $\mathbb{P}_\kappa(\mathbf{a}) : \dim\ker D_{\mathbf{a}}\mathcal{F} = \kappa$, but this time on the universe $\mathcal{M}_{k,n+m}(q)$, where $D_{\mathbf{a}}\mathcal{F}$ lives. To get a similar result to Corollary 4.1, we need to show two things. **a)**, that this predicate satisfies the assumptions required for Algorithm 1. **b)**, that there is a κ such that the density d_κ of $\mathbb{P}_\kappa$ is optimal as described in Lemma 4.15. If both are satisfied, we get a complexity of $\mathcal{O}(q^{\frac{2}{3}(n+m)} C_{\mathrm{FF}}^{\frac{1}{3}})$, hence $\tilde{\mathcal{O}}(q^{\frac{2}{3}(n+m)})$ when the solver is polynomial, with a success probability of $1 - 1/c$ for any $c > 0$ for an MCE (n, m, k) instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$. We start with **a)**.

Lemma 4.16. The predicate $\mathbb{P}_\kappa(D_{\mathbf{a}}\mathcal{F}) : \dim\ker D_{\mathbf{a}}\mathcal{F} = \kappa$ is a suitable predicate for Algorithm 1, as **i)** $\mathbb{P}_\kappa$ can be computed in polynomial time, **ii)** is invariant under equivalence, **iii)** and d_κ does not depend on $\mathcal{F}$.

Proof.

1. The cost $C_{\mathbb{P}_\kappa}$ is the cost of computing $\dim\ker D_{\mathbf{a}}\mathcal{F}$, i.e. computing the kernel of a $k \times (n+m)$ matrix over $\mathbb{F}_q$. This can be done in polynomial time.

2. Let $\mathcal{P}(\mathbf{x}) = \mathcal{F}(\mathbf{x}\mathbf{S})\mathbf{T}$ be the equivalence. If $\mathbf{x} \in \ker D_{\mathbf{a}}\mathcal{P}$ then $\mathbf{x}\mathbf{S} \in \ker \mathcal{F}_{\mathbf{a}\mathbf{S}}$ and vice versa, as $\mathbf{T}$ does not affect the kernel. As $\mathbf{S}$ is invertible, we get a one-to-one correspondence $\mathbf{x} \mapsto \mathbf{x}\mathbf{S}$ between the kernels, so $\mathbb{P}_\kappa(D_{\mathbf{a}\mathbf{S}}\mathcal{F}) = \mathbb{P}_\kappa(D_{\mathbf{a}}\mathcal{P})$.

3. For $\mathcal{F}$ coming from an MCE instance, we always have $-\mathbf{a} \in \ker D_{\mathbf{a}}\mathcal{F}$. We want to show that the distribution of the rank of $D_{\mathbf{a}}\mathcal{F}$ follows the ranks of linear maps vanishing at $-\mathbf{a}$. This is given by [66, Thm. 2] for even characteristic and easily adapted to odd characteristic, which shows d_κ is independent of $\mathcal{F}$.

$\square$

We now continue with **b)**: we show that there is a κ such that d_κ is optimal. For now, existence of κ is enough to derive a complexity on MCE. We will explicitely compute κ later, in Section 4.6, when we have a detailed view of C_{FF} for specific parameter sets (k, n, m).

The general density d_κ for the predicate $\mathbb{P}_\kappa$ is given by the following lemma, taking $a = k$ and $b = n + m$ to avoid confusion with regards to n, m and $n + m$.

Lemma 4.17. Define the predicate $\mathbb{P}_\kappa : \dim\ker \mathbf{M} = \kappa$ for $\mathbf{M} \in U = \mathcal{M}_{a,b}(q)$ with $a \geqslant b$. Then the density of the predicate $\mathbb{P}_\kappa$ is $d_\kappa = 1/\Theta(q^{(\kappa^2 + \kappa\cdot(a-b))})$.

Proof. There are $|U| = q^{ab}$ matrices in $\mathcal{M}_{a,b}(q)$, out of which

$$\prod_{i=0}^{r-1} \frac{(q^a - q^i)(q^b - q^i)}{q^r - q^i} = \Theta\left(q^{(a+b-r)r}\right)$$

have rank r [107]. We have $\kappa = b - r$ and so $d_\kappa^{-1} = \frac{|U|}{|U_\top|} = \Theta(\frac{q^{ab}}{q^{-(a+b-r)r}}) = \Theta(q^{\kappa^2+\kappa(a-b)})$. Specifically when the matrix is square, $d_\kappa^{-1} = \Theta(q^{\kappa^2})$. □

From Lemma 4.17 we can conclude that for some κ, the density d_κ is optimal. This means we satisfy both **a)** and **b)** and we can apply Lemma 4.15.

In conclusion, we get our first result on the hardness of MCE, which significantly improves straightforward enumeration. This requires that such a κ exists, which happens when $k \leqslant 2(n+m)$, by Lemma 4.17. Note that, in contrast to [38, Thm. 1], we do not assume a polynomial-time solver for the inhomogeneous case of QMLE. Instead, we write this cost as C_{FF} and explore the precise cost in Section 4.6.

Theorem 4.18. An MCE (n, m, k) instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ over $\mathbb{F}_q$ with $k \leqslant 2(n+m)$ can be solved using Algorithm 1 with time complexity equal to $\mathcal{O}(q^{\frac{2}{3}(n+m)} \cdot C_{\mathrm{FF}}^{\frac{1}{3}} \cdot (C_{\mathbb{P}_\kappa} + 1))$, memory complexity equal to $\mathcal{O}(q^{\frac{1}{3}(m+n)} C_{\mathrm{FF}}^{-\frac{1}{3}})$ and with success probability of $1 - 1/c$ for any $c > 0$, where C_{FF} denotes the cost of a single query to FindFunction.

We will show in Section 4.6 that, even though C_{FF} is *not* polynomial-time, the complexity of Algorithm 1 is still $\tilde{\mathcal{O}}(q^{\frac{2}{3}(n+m)})$ for some optimal κ.

When $k > 2(n+m)$, we can no longer assume elements with $\dim \ker D_{\mathbf{a}}\mathcal{F} > 1$ exist, as practically all differentials $D_{\mathbf{a}}\mathcal{F}$ will have only the trivial kernel spanned by $-\mathbf{a}$. In such a scenario, we have two alternatives:

- Take a single element $\mathbf{a}$ and run FindFunction on $(\mathbf{a}, \mathbf{b})$ for all $\mathbf{b} \in \mathbb{F}_q^{n+m}$ until we find the isometry. This deterministic process has a time complexity of $\mathcal{O}(q^{(n+m)} \cdot C_{\mathrm{FF}})$. The memory requirements of this algorithm are negligible, since we do not build lists of elements;

- Alternatively, note that in this case $n \leqslant 2(k+m)$. Thus, we can also use the result of Lemma 4.14, and instead of an MCE (n, m, k) instance, we can solve an MCE (k, m, n) instance using Algorithm 1. In this case we end up with a complexity of $\tilde{\mathcal{O}}(q^{\frac{2}{3}(k+m)})$. However, for the given regime of parameters, this is always larger than $\tilde{\mathcal{O}}(q^{(n+m)})$, so the first deterministic approach is better.

4.5.3 Second algorithm

The algorithm that we presented in the previous section does not take advantage of the bilinear structure of an instance of MCE when viewed as hQMLE. In such a case, the differential $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}$ of a k-dimensional bilinear form admits a special structure.

Lemma 4.19. Let $\mathcal{F}(\mathbf{x},\mathbf{y})$ be a k-dimensional bilinear form with $\mathbf{x} \in \mathbb{F}_q^m$ and $\mathbf{y} \in \mathbb{F}_q^n$. Let $\mathbf{F_a}$ denote the $k \times n$ matrix of the linear map $\mathcal{F}(\mathbf{a},-) : \mathbb{F}_q^n \to \mathbb{F}_q^k$ for a fixed $\mathbf{a} \in \mathbb{F}_q^m$. Similarly let $\mathbf{F_b}$ denote the $k \times m$ matrix of the linear map $\mathcal{F}(-,\mathbf{b}) : \mathbb{F}_q^m \to \mathbb{F}_q^k$ for a fixed $\mathbf{b} \in \mathbb{F}_q^n$. Then

$$D_{(\mathbf{a},\mathbf{b})}\mathcal{F}(\mathbf{x},\mathbf{y}) = (\ \mathbf{F_b}\ \mathbf{F_a}\ ) \begin{pmatrix} \mathbf{x}^\top \\ \mathbf{y}^\top \end{pmatrix}.$$

Proof. By bilinearity, $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}(\mathbf{x},\mathbf{y}) := \mathcal{F}(\mathbf{x}+\mathbf{a},\mathbf{y}+\mathbf{b}) - \mathcal{F}(\mathbf{x},\mathbf{y}) - \mathcal{F}(\mathbf{a},\mathbf{b})$ equals $\mathcal{F}(\mathbf{a},\mathbf{y}) + \mathcal{F}(\mathbf{x},\mathbf{b}) = \mathbf{F_a}\mathbf{y}^\top + \mathbf{F_b}\mathbf{x}^\top$. $\square$

Similarly for $\mathcal{P}$, we use the notation $\mathbf{P_a}$ and $\mathbf{P_b}$. The equivalence in such a case becomes $\mathcal{P}(\mathbf{x},\mathbf{y}) = \mathcal{F}(\mathbf{x}\mathbf{A}, \mathbf{y}\mathbf{B}^\top)\mathbf{T}$, with $\mathbf{A},\mathbf{B}$ precisely the matrices from the MCE instance. Then, as $\mathcal{F}$ and $\mathcal{P}$ are bilinear, one can see SampleSet in Algorithm 1 as sampling both $\mathbf{a} \in \mathbb{F}_q^n$ and $\mathbf{b} \in \mathbb{F}_q^m$ at the same time as one $(\mathbf{a},\mathbf{b}) \in \mathbb{F}_q^{n+m}$, until $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}$ has a kernel of dimension κ. However in the bilinear case, $\mathbf{a}$ influences only the matrix $\mathbf{F_a}$, and $\mathbf{b}$ influences only $\mathbf{F_b}$. Hence, we can sample $\mathbf{a} \in \mathbb{F}_q^m$ and $\mathbf{b} \in \mathbb{F}_q^n$ separately. This hints that we can apply ideas from Algorithm 1 to the smaller universes $U_\mathbf{a} = \mathcal{M}_{k,n}(q)$ and $U_\mathbf{b} = \mathcal{M}_{k,m}(q)$, where $\mathbf{F_a}$ and $\mathbf{F_b}$ live. By finding well-chosen predicates in these smaller universes, we hope to find collisions faster.

We first analyse the properties of $\mathbf{F_a}$ and $\mathbf{F_b}$ a bit more. Let $\mathfrak{F}_a$ be the set of elements $\mathbf{a}$ for which $\dim\ker \mathbf{F_a}$ is non-trivial, and $\mathfrak{F}_b$ similarly, i.e.

$$\mathfrak{F}_a = \{\mathbf{a} \in \mathbb{F}_q^m \mid \dim\ker \mathcal{F}(\mathbf{a},-) > 0\}, \quad \mathfrak{F}_b = \{\mathbf{b} \in \mathbb{F}_q^n \mid \dim\ker \mathcal{F}(-,\mathbf{b}) > 0\}.$$

For $\mathcal{P}$, we define $\mathfrak{P}_a$ and $\mathfrak{P}_b$ similarly. For isomorphic bilinear forms $\mathcal{F}$ and $\mathcal{P}$, these sets have special properties.

Lemma 4.20. Let $(\mathbf{A},\mathbf{B},\mathbf{T}) : \mathcal{F} \to \mathcal{P}$ be an isomorphism between two k-tuples of bilinear homogenous quadratic polynomials $\mathcal{F}$ and $\mathcal{P}$, such that $\mathcal{P}(\mathbf{x},\mathbf{y}) = \mathcal{F}(\mathbf{x}\mathbf{A},\mathbf{y}\mathbf{B}^\top)\mathbf{T}$. We have the following properties:

1. Given $\mathbf{a} \in \mathfrak{F}_a$ and any $\mathbf{b} \in \ker \mathbf{F_a}$, we get $\mathcal{F}(\mathbf{a},\mathbf{b}) = 0$.

2. $\mathfrak{F}_b$ is completely determined by $\mathfrak{F}_a$, as $\mathfrak{F}_b = \bigcup_{\mathbf{a}\in\mathfrak{F}_a} \ker \mathbf{F_a}$.

3. For $\mathbf{a} \in \mathbb{F}_q^n$ and $\mathbf{y} \in \mathbb{F}_q^m$, we have $\mathbf{P_a}(\mathbf{y}) = \mathbf{F_{aA}}(\mathbf{y}\mathbf{B}^\top)\mathbf{T}$.

4. For $\mathbf{a} \in \mathbb{F}_q^n$, we get $\ker \mathbf{P_a} = \ker \mathcal{F}_{\mathbf{aA}} \cdot \mathbf{B}^\top$.

5. The isomorphism $(\mathbf{A},\mathbf{B},\mathbf{T})$ induces the bijections

$$\mathfrak{P}_a \to \mathfrak{F}_a : \mathbf{a} \mapsto \mathbf{aA}, \quad \mathfrak{P}_b \to \mathfrak{F}_b : \mathbf{b} \mapsto \mathbf{b}\mathbf{B}^\top.$$

Proof. 1. $\mathbf{b} \in \ker \mathbf{F_a}$ is equivalent by definition to $\mathbf{F_a}\mathbf{b}^\top = \mathcal{F}(\mathbf{a},\mathbf{b}) = \mathbf{0}$.

2. This follows directly from 1.: $\mathbf{b} \in \mathfrak{F}_b$ only if there exists an $\mathbf{a} \in \mathfrak{F}_a$ such that $\mathcal{F}(\mathbf{a}, \mathbf{b}) = \mathbf{0}$. But then $\mathbf{b} \in \ker \mathbf{F_a}$ for this specific $\mathbf{a}$.
3. Per definition $\mathbf{P_a}(\mathbf{y}) = \mathcal{P}(\mathbf{a}, \mathbf{y}) = \mathcal{F}(\mathbf{aA}, \mathbf{yB}^\top)\mathbf{T} = \mathbf{F_{aA}}(\mathbf{yB}^\top)\mathbf{T}$.
4. This follows directly from 3.: as $\mathbf{T}$ is invertible, it does not affect the kernels, so $\mathbf{y} \in \ker \mathbf{P_a}$ if and only if $\mathbf{yB}^\top \in \ker \mathbf{F_{aA}}$
5. This follows directly from 4.: Given $\mathbf{a} \in \mathfrak{P}_a$ we get $\mathbf{aA} \in \mathfrak{F}_a$ and vice versa as $\mathbf{A} \in \mathrm{GL}_m(q)$. A similar argument gives $\mathfrak{F}_b \to \mathfrak{P}_b$.

□

Lemma 4.20 shows that $\mathbf{a} \in \mathfrak{F}_a$ and $\mathbf{b} \in \mathfrak{F}_b$ describe all non-trivial roots $(\mathbf{a}, \mathbf{b})$ of a given $\mathcal{F}$. For an instance $(\mathbf{A}, \mathbf{B}, \mathbf{T}) : \mathcal{F} \to \mathcal{P}$, Item 5 shows that non-trivial roots are mapped bijectively by $(\mathbf{A}, \mathbf{B}, \mathbf{T})$. Such non-trivial roots can be used to find collisions more easily between $\mathcal{F}$ and $\mathcal{P}$. However, this requires that instances $\mathcal{F} \to \mathcal{P}$ *have* non-trivial roots. We can get an estimate on the sizes of $\mathfrak{F}_a$, $\mathfrak{F}_b$, $\mathfrak{P}_a$, and $\mathfrak{P}_b$ for given parameters n, m, and k, in the following way.

Lemma 4.21. When $k \geqslant n$, $|\mathfrak{F}_a| = |\mathfrak{P}_a| \approx q^{2n-k-1}$ and $|\mathfrak{F}_b| = |\mathfrak{P}_b| \approx q^{2m-k-1}$.

Proof. By Lemma 4.20, we get $|\mathfrak{F}_a| = |\mathfrak{P}_a|$. Then, using Lemma 4.17, we see that the size of these sets is dominated by elements $\mathbf{a}$ with $\kappa = \dim \ker \mathbf{F_a} = 1$ (a one-dimensional kernel). From the same lemma, the density of $\kappa = \dim \ker \mathbf{F_a} = 1$ elements is $d_1 = q^{-(1+1\cdot(k-n))}$. Hence we expect $d_1 \cdot q^n = \Theta(q^{2n-k-1})$ such elements. A similar argument gives $|\mathfrak{F}_b| = |\mathfrak{P}_b|$ as $\Theta(q^{2m-k-1})$. □

Summarizing, this implies

Corollary 4.2. Assuming $n = m$ as the hardest case, a random MCE (n, m, k) instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$ has an expected value $\mathcal{E}_{n,m,k,q}$ of non-trivial roots

- when $k < 2n$, with $\mathcal{E}_{n,m,k,q} = \Theta(q^{2n-k-1})$,
- when $k = 2n$, with $\mathcal{E}_{n,m,k,q} = \Theta(\frac{1}{q})$,
- when $k > 2n$, with $\mathcal{E}_{n,m,k,q} = \Theta(\frac{1}{q^{k-2n+1}})$.

From these results, we can expect non-trivial roots for an MCE (n, m, k) instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$ with $k \leqslant n + m$. These non-trivial roots can be seen as a suitable predicate on the smaller universes $U_\mathbf{a}$ and $U_\mathbf{b}$: we search for collisions $(\mathbf{a}, \mathbf{b}) \times (\mathbf{aA}, \mathbf{bB}^\top)$, where $(\mathbf{a}, \mathbf{b})$ is a non-trivial root of $\mathcal{P}$, and $(\mathbf{aA}, \mathbf{bB}^\top)$ of $\mathcal{F}$. Given such a collision, we proceed as in Section 4.5.2.

The following result shows that we always find such a collision if $\mathcal{F}$ and $\mathcal{P}$ have non-zero roots.

Lemma 4.22. Let $m \leqslant n$ and $k \leqslant n + m$. Let $\mathfrak{F}_a$, $\mathfrak{F}_b$ and $\mathfrak{P}_a$, $\mathfrak{P}_b$ describe the non-trivial roots of an MCE (n, m, k) instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$. Let $\mathbf{x} = (\mathbf{a}, \mathbf{b}) \in \mathfrak{F}_a \times \mathfrak{F}_b$, then looping over $\mathbf{y} \in \mathfrak{P}_a \times \mathfrak{P}_b$ gives a collision $(\mathbf{x}, \mathbf{y})$ with certainty.

Proof. This follows quickly from Lemma 4.20. We have $\mathbf{x} = (\mathbf{a}, \mathbf{b})$ and two bijections $\mathfrak{F}_a \to \mathfrak{P}_a$ and $\mathfrak{F}_b \to \mathfrak{P}_b$, so $\mathbf{x}$ is mapped to some $\mathbf{y} \in \mathfrak{P}_a \times \mathfrak{P}_b$. As this set is finite, we can loop over it in a finite number of steps until we find the collision. □

Therefore, as soon as we have non-trivial roots, we can use a single one of them to find a collision. This leads to the following pseudo-algorithm:

1. compute $\mathfrak{F}_b$ by computing $\ker \mathbf{F_b}$ for all $\mathbf{b} \in \mathbb{F}_q^m$,
2. if $\mathfrak{F}_b$ is non-empty, compute $\mathfrak{F}_a$ using Lemma 4.20-2. Same for $\mathfrak{P}_a$ and $\mathfrak{P}_a$.
3. sample a single $\mathbf{x} \in \mathfrak{F}_a \times \mathfrak{F}_b$
4. loop over $\mathbf{y} \in \mathfrak{P}_a \times \mathfrak{P}_b$ with FindFunction$(\mathbf{x}, \mathbf{y})$ until the solver finds μ.

Corollary 4.3. Let $m \leqslant n$ and $k \leqslant n + m$. The above algorithm terminates successfully and has a total complexity of $\mathcal{O}(q^m \cdot C_{\mathbb{P}_\kappa} + q^{2(n+m-k-1)} \cdot C_{\mathrm{FF}})$, where $C_{\mathbb{P}}$ denotes the cost of computing $\ker \mathbf{F_b}$ and C_{FF} denotes the cost of a single query to FindFunction.

Proof. Building $\mathfrak{F}_b$ and $\mathfrak{P}_b$ has a complexity of $\mathcal{O}(q^m \cdot C_{\mathbb{P}_\kappa})$, and these give us $\mathfrak{F}_a$ and $\mathfrak{P}_a$ by Lemma 4.20. Then for every step in the loop we get a query to FindFunction. By Lemma 4.21, the size of $\mathfrak{P}_a \times \mathfrak{P}_b$ is at most $\mathcal{O}(q^{2(n+m-k-1)})$. □

We will see later in Section 4.6 that the dominating complexity is $q^m \cdot C_{\mathbb{P}_\kappa}$ as for specific parameters (k, n, m) the number of queries z can be reduced so that $z \cdot C_{\mathrm{FF}} < q^m$. As $C_{\mathbb{P}_\kappa}$ is polynomial, we get a complexity of $\tilde{\mathcal{O}}(q^m)$ for such instances.

For efficiency, one can decrease further the number of queries to FindFunction by applying other, secondary predicates. For example, the sets $\mathfrak{F}_a \times \mathfrak{F}_b$ and $\mathfrak{P}_a \times \mathfrak{P}_b$ can be split into zeros $\mathfrak{F}^{\mathbf{0}} = \{\mathbf{x} \in \mathbb{F}_q^{n+m} | \mathcal{F}(\mathbf{x}) = \mathbf{0}\}$ and non-zeros $\mathfrak{F} = \mathfrak{F}_a \times \mathfrak{F}_b \setminus \mathfrak{F}^{\mathbf{0}}$, which reduces the collision search to each of these sets. Another secondary predicate is to only use elements $\mathbf{a}$ with $\dim \ker \mathbf{F_a} = \kappa$ for some specific value $\kappa > 0$.

We summarize the MCE solver for instances with roots in Algorithm 2.

Practically, since the algorithm is deterministic, we do not need to build and store the list $\mathfrak{F}$. We only need to find one element from it. However, for iterating through the list $\mathfrak{P}$, S_a and S_b need to be stored. The estimated size of these lists is $q^{n+m-k-1}$.

The next theorem summarises the conditions and cost of Algorithm 2 for solving MCE.

Theorem 4.23. Assuming a solver for the inhomogenous case of QMLE with cost C_{FF}, an MCE (n, m, k) instance over $\mathbb{F}_q$ with $m \leqslant n$ and $k \leqslant n + m$ (in which case roots exist for $\mathcal{F}$ and $\mathcal{P}$ with overwhelming probability) can be solved using Algorithm 2 with $\mathcal{O}\left(q^m \cdot C_{\mathbb{P}_\kappa}\right)$ operations in SampleZeros and z queries

Algorithm 2 Bilinear MCE-Solver, assuming $n \geqslant m$.

```
1: procedure SampleZeros($\mathcal{F}$)
2:     $S, S_a, S_b \leftarrow \emptyset$
3:     for all $\mathbf{b} \in \mathbb{F}_q^m$ do
4:         if $\dim \ker \mathbf{F_b} > 0$ then
5:             $S_b \leftarrow S_b \cup \{\mathbf{b}\}$
6:             $S_a \leftarrow S_a \cup \ker \mathbf{F_b} \setminus \{0\}$
7:     $S \leftarrow S_a \times S_b$
8:     return $S$

9: procedure CollisionFind($\mathcal{F}, \mathcal{P}$)
10:    $\mathfrak{F} \leftarrow$ SampleZeros($\mathcal{F}$)
11:    $\mathfrak{P} \leftarrow$ SampleZeros($\mathcal{P}$)
12:    $\mathbf{x} \xleftarrow{\$} \mathfrak{F}$
13:    for all $\mathbf{y} \in \mathfrak{P}$ do
14:        $\mu \leftarrow$ FindFunction($\mathbf{x}, \mathbf{y}$)
15:        if $\mu \neq \perp$ then
16:            return solution $\mu$
17:    return $\perp$
```

to the solver. This amounts to a total time complexity of $\mathcal{O}\left(q^m \cdot C_{\mathbb{P}_\kappa} + z \cdot C_{\mathrm{FF}}\right)$. The memory complexity of the algorithm is $\mathcal{O}(q^{n+m-k-1})$.

We will show in Section 4.6 that, even though C_{FF} is *not* polynomial-time, the dominating factor in this complexity is still $q^m \cdot C_{\mathbb{P}_\kappa}$, where $C_{\mathbb{P}_\kappa}$ is the cost to compute the kernel of an $m \times k$ matrix.

The regime of operation of Theorem 4.23 seems to imply that we can use it only if $k \leqslant n + m$. However, note that if $k > n + m$ then $n \leqslant k + m$. Hence, by Lemma 4.14, we can turn the given MCE (n, m, k) instance into an MCE (k, m, n) instance and solve this instance using Algorithm 2. This results in a complexity of $\tilde{\mathcal{O}}(q^m)$. Recall that we assume $m = \min\{m, n, k\}$, thus, we obtain the following general theorem which is our main result about the practical complexity of solving MCE.

Theorem 4.24. An MCE (n, m, k) instance over $\mathbb{F}_q$ can be solved using Algorithm 2 in time $\tilde{\mathcal{O}}\left(q^{\min\{m,n,k\}}\right)$.

4.6 Filling the gaps in the complexity analysis

The cost $C_{\mathbb{P}}$ is polynomial in all of the cases because it either requires computing the rank of a linear map or sampling a random element from a set. The FindFunction in Algorithms 1 and 2 checks whether a given pair of vectors is a collision, and if so, it returns the solution to the MCE instance. This is done by solving an instance of the inhBMLE that has the same solutions as the input MCE instance. Thus, to estimate the value of C_{FF}, we analyse the complexity of inhBMLE on these instances, by relying on algorithms that have been developed for the inhQMLE case with $N = k$.

4.6.1 Algorithms for inhQMLE

The two algorithms described in this section have been used for tackling the inhQMLE problem within the birthday-based algorithm for hQMLE [37, 38].

Their analysis is thus important to estimate C_{FF}. In Section 4.6.2 we adapt this analysis for the inhBMLE case with arbitrary k and N and we see how this affects Algorithms 1 and 2 for different parameter sets.

The Gröbner bases attack

The algebraic attack on the inhQMLE problem starts by reducing $\mathcal{P}(\mathbf{x})\mathbf{T}^{-1} = \mathcal{F}(\mathbf{x}\mathbf{S})$, with $\mathbf{S}$ and $\mathbf{T}$ unknown, to a system of polynomial equations. By rewriting the problem in matrix form we obtain the following constraints

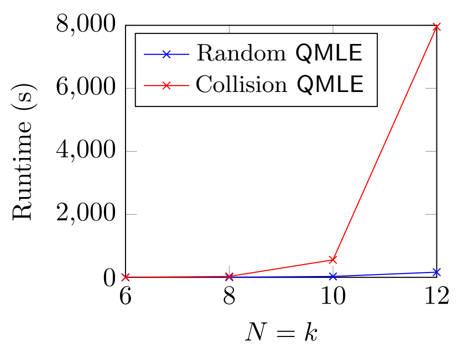
$$\begin{aligned} \sum_{1\leqslant r\leqslant k} \widetilde{T}_{rs}\mathbf{P}^{(r)} &= \mathbf{S}\mathbf{F}^{(s)}\mathbf{S}^{\top}, \ \ \forall s, 1 \leqslant s \leqslant k, \\ \mathbf{P}^{[1]}\mathbf{T}^{-1} &= \mathbf{S}\mathbf{F}^{[1]}, \\ \mathbf{P}^{[0]}\mathbf{T}^{-1} &= \mathbf{F}^{[0]}, \end{aligned} \tag{4.13}$$

where $\mathbf{F}^{[1]} \in \mathbb{F}_q^{N\times k}$ and $\mathbf{P}^{[1]} \in \mathbb{F}_q^{N\times k}$ describe the degree-1 homogeneous part of an inh(Q/B)MLE instance and $\mathbf{F}^{[0]} \in \mathbb{F}_q^{k}$ and $\mathbf{P}^{[0]} \in \mathbb{F}_q^{k}$ describe the degree-0 part. We will denote the subsystem of equations derived from the degree-d homogeneous part as $\mathcal{S}_d$. The resulting system can be solved using Gröbner basis algorithms and this is referred to as the Gröbner attack [74]. The observation that $\mathbf{S}$ and $\mathbf{T}$ are common solutions to homogeneous parts of separate degrees of an inhQMLE instance (also proven in [36, Lemma 1]) and the idea that moving $\mathbf{T}$ to the other side of the equality results in a lower degree system where we solve for $\mathbf{T}^{-1}$ originate from this work.

The complexity of Gröbner basis algorithms depends foremost on the *degree of regularity*, which is usually hard to estimate, but it can sometimes be observed through experimental work. Such experiments applied to inhQMLE instances imply that the system is solved at degree three. A degree-three linearized system in n variables is represented by a matrix of size roughly n^3 and thus, Gaussian Elimination on such a system is performed in $\mathcal{O}(n^{3\omega})$ operations, where ω is the linear algebra constant. This reasoning leads to the assumption that there exists a polynomial-time solver for the inhomogeneous case of QMLE. Another empirical observation made in [74] is that the time to construct the system exceeds the time of the Gröbner basis computation. Since the generation of the system is known to be polynomial, this suggests that the Gröbner basis computation is performed in polynomial time as well. However, these experiments are performed on random inhomogeneous instances of the QMLE problem.

In the birthday-based approach for solving QMLE, $\mathbf{F}^{[1]}$, $\mathbf{P}^{[1]}$, $\mathbf{F}^{[0]}$ and $\mathbf{P}^{[0]}$ are obtained from a collision [38]. Specifically, if we have a collision on $\mathbf{x} \in \mathbb{F}_q^N$ and $\mathbf{y} \in \mathbb{F}_q^N$ such that $\mathbf{y} = \mathbf{x}\mathbf{S}$, they are obtained as

$$\begin{aligned} \mathbf{F}^{[1]} &= D_{\mathbf{y}}\mathcal{F}, & \mathbf{P}^{[1]} &= D_{\mathbf{x}}\mathcal{P}, \\ \mathbf{F}^{[0]} &= \mathcal{F}(\mathbf{y}), & \mathbf{P}^{[0]} &= \mathcal{P}(\mathbf{x}). \end{aligned}$$



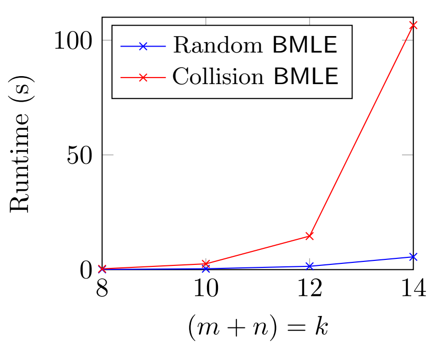


Figure 4.2: Comparison of runtime for solving random and collision-derived inh(Q/B)MLE instances using the Gröbner attack. Results are averaged over 50 runs.

Instances of inhQMLE derived from a collision are, on average, harder to solve than random inhQMLE instances. Recall that in Algorithm 1 the instances of inhQMLE are chosen such that $\dim \ker D_{\mathbf{y}}\mathcal{F} = \dim \ker D_{\mathbf{x}}\mathcal{P} = \kappa$. Hence, the number of linearly independent equations in $\mathcal{S}_1$ is exactly $k(N - \kappa)$, instead of the expected kN on average. The size of $\mathcal{S}_0$ can also depend on the predicate that we choose for the birthday-based algorithm. For instance, when we use the predicate of searching for a collision between the non-trivial roots of $\mathcal{P}$ and $\mathcal{F}$, we obtain no equations in $\mathcal{S}_0$. Additionally, since $\mathbf{F}^{[1]}$ (i.e. $\mathbf{P}^{[1]}$) and $\mathbf{F}^{[0]}$ (i.e. $\mathbf{P}^{[0]}$) are obtained respectively from computing the differential of and evaluating $\mathcal{F}$ (i.e $\mathcal{P}$) at a given point, $\mathcal{S}_1$ and $\mathcal{S}_0$ are not as independent from $\mathcal{S}_2$ as they would be in the random case. It is difficult to estimate the complexity of solving these instances compared to solving random instances with the same structure. Figure 4.2 shows experiments confirming our intuition that the complexity of collision-derived instances is worse than that of random ones. This implies that we can not rely on the experimental observations in [74] to estimate the complexity of these specific instances. We conclude that, in contrast with the literature, we can not assume that C_{FF} is polynomial when the Gröbner attack is used.

The matrix-pencil attack

The matrix-pencil attack was proposed in Bouillaguet's thesis [37] and used for the implementation of the birthday-based attack [38]. This algorithm has a complexity of $\mathcal{O}(N^6)$ with non-negligible probability for random inhQMLE instances where $N = k$. Its complexity for inhQMLE instances derived from a collision attack depends strongly on the parameter κ. We give a general description of the approach. For details on how it relates to the matrix pencil equivalence problem, we refer to [37, Ch. 14].

The first step is to retrieve a basis of the solution space V of the subsystem of linear equations $\mathcal{S}_1$. Let $\ell = \dim V$ and let $(\mathbf{S}^{[1]}, \mathbf{T}^{[1]}), \ldots, (\mathbf{S}^{[\ell]}, \mathbf{T}^{[\ell]})$ be a basis of V. Once the solution space of $\mathcal{S}_1$ is known, in order to find the solution space of the overall system one rewrites $\mathcal{S}_2$ as a system in ℓ variables. Concretely, this is done by replacing $\mathbf{S}$ and $\mathbf{T}$ by $\sum_{i=1}^{\ell} x_i\mathbf{S}^{[i]}$ and $\sum_{i=1}^{\ell} x_i\mathbf{T}^{[i]}$ in Equation (4.13) and then looking for solutions in variables $x_1, \ldots, x_\ell$. This standard approach is also described in [36]. A key idea in the matrix-pencil attack is to use the knowledge of $\mathbf{F}^{[1]}/\mathbf{P}^{[1]}$ and $\mathbf{F}^{[0]}/\mathbf{P}^{[0]}$ to find a (second) collision and double the number of linear equations in $\mathcal{S}_1$. Supposing that there exists $\mathbf{x}'$ such that $\mathbf{x}'\mathbf{P}^{[1]} = \mathbf{P}^{[0]}$, we infer that there also exists $\mathbf{y}'$ such that $\mathbf{y}'\mathbf{F}^{[1]} = \mathbf{F}^{[0]}$ and that $\mathbf{y}' = \mathbf{x}'\mathbf{S}$. We can thus append the equations obtained from $(D_{\mathbf{x}'}\mathcal{P})\mathbf{T}^{-1} = \mathbf{S}(D_{\mathbf{y}'}\mathcal{F})$ to $\mathcal{S}_1$. After applying this technique, the resulting system is usually highly overdetermined and can be solved through direct linearization. The most favorable case is when $\mathbf{x}'$ and $\mathbf{y}'$ are uniquely identified. However, if $\dim\ker\mathbf{F}^{[1]} = \kappa > 1$, then $\mathbf{x}'$ is chosen arbitrarily and we loop through the q^κ possible values for $\mathbf{y}'$. The complexity of the algorithm is $\mathcal{O}(q^\kappa\ell^2N^4)$, under the condition that $\ell(\ell+1)/2 \le |\mathcal{S}_2|$. Another condition for the success of this approach is that $\mathcal{P}(\mathbf{x}) \neq 0$ and there is an $\mathbf{x}$ such that $\mathbf{x}D_{\mathbf{x}}\mathcal{P} = \mathcal{P}(\mathbf{x})$, because this assumption is used to find the second collision. As per the analysis in [37], the probability that the condition for success is met is $1 - 1/q + 1/q^3 + \mathcal{O}(1/q^6)$.

4.6.2 The complexity of inhBMLE

In the following analysis, we use the matrix-pencil algorithm as the inhBMLE solver, as it seems to outperform the Gröbner attack and we have a better understanding of its complexity for these specific instances.

The case $k \le n+m$

Based on the analysis in Section 4.5.3 for the purpose of usage in Algorithm 2 we can assume without loss of generality that $k \le n+m$ and $m = \min\{m, n, k\}$.

The complexity of Algorithm 2 is dominated by the SampleZeros function, as long as the complexity of the inhBMLE solver does not surpass $\mathcal{O}(q^m)$. In the matrix-pencil algorithm, we can not use the zero subsets $\mathfrak{F}^{\mathbf{0}}$ and $\mathfrak{P}^{\mathbf{0}}$, as this contradicts its condition for success $\mathcal{P}(\mathbf{x}) \neq 0$. The non-zeros subsets $\mathfrak{F}$ and $\mathfrak{P}$ can be used with a small adjustment to the algorithm: after finding a basis of the solution space of $\mathcal{S}_1$, we rewrite and solve through linearization the system comprised of both $\mathcal{S}_2$ and $\mathcal{S}_0$. Note that $\mathfrak{F}$ and $\mathfrak{P}$ are non-empty only when the instance has at least two roots. Since in Algorithm 2 we do not restrict the value of κ, we will approximate to the one that has the highest probability, which for the case of $k \le n+m$ is $\kappa = (m+n) - k$. Hence, C_{FF} is approximated to

$$\mathcal{O}(q^{m+n-k} \cdot (m+n)^6).$$

When $k \ge m$, this is always smaller than $\mathcal{O}(q^m)$.

The case $n+m<k<2(n+m)$

This case is not relevant for Algorithm 2, but it is for Algorithm 1. Since the complexity of the inhBMLE solver contains a non-negligible factor of q^κ, the choice of κ needs to be adapted, so that the running times of SampleSet and CollisionFind are equal. Let $N=n+m$ and let $r=N-k$. The optimal κ is chosen such that

$$q^{\frac{N-(\kappa^2+\kappa r)}{2}} \cdot q^{\kappa^2+\kappa r} \approx q^{N-(\kappa^2+\kappa r)} \cdot q^{\kappa}.$$

This gives us $\kappa = \frac{k-(n+m+\sqrt{\delta})}{2} + \frac{1}{3}$, with $\delta = (k-(n+m))^2 + \frac{4}{3}(k+\frac{1}{3})$. The complexity of the overall algorithm with this optimal choice for κ is then

$$q^{\frac{n+m}{2}+\frac{k-\sqrt{\delta}}{6}+\frac{1}{9}}.$$

We get that $\sqrt{\delta} \geqslant |k-(n+m)|$ and so for all values of k between $n+m$ and $2(n+m)$, the term $k-\sqrt{\delta}$ is bounded by $n+m$, and hence this gives a bound on the complexity by $\mathcal{O}(q^{\frac{2}{3}(n+m)+\frac{1}{9}})$. The term $\frac{1}{9}$ adds a few bits at most to this complexity, but is negligible for most cryptographic purposes.

The case $k \geq 2(n+m)$

When $k \geq 2(n+m)$, as per Lemma 4.17, the probability that there exist elements with $\dim\ker D_{(\mathbf{a},\mathbf{b})}\mathcal{F} > 1$ is extremely small, which is why we can not define a distinguishing predicate for Algorithm 1 and $\kappa = 1$ with overwhelming probability. In this case, the complexity of the matrix-pencil algorithm is

$$\mathcal{O}(q\cdot(m+n)^6),$$

as with random inhBMLE instances.

4.7 Experimental results

To confirm our theoretical findings, we solved randomly generated positive instances of the MCE problem, using the two approaches presented in this paper. First, we implement the birthday-based Algorithm 1 in three steps. (1) We randomly generate a positive instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C},\mathcal{D})$ of MCE (n,m,k) and reduce it to an instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F},\mathcal{P})$ of hQMLE $(m+n,k)$. (2) We build the two sample sets for a predefined predicate $\mathbb{P}$ and we combine them to create pairs of potential collisions. (3) For each pair we create an inhQMLE instance and we query an inhQMLE solver until it outputs a solution for the maps $\mathbf{S}$ and $\mathbf{T}$. Our implementation is built on top of the open source birthday-based hQMLE solver from [37], which is implemented in MAGMA [35].

Table 4.1 shows running times for solving the MCE problem using Algorithm 1. The goal of this first experiments was to confirm that there is a parameter choice where the probability of success of the algorithm surpasses

$1 - 1/e$ and that our running times are comparable to the ones given in [38]. These experiments are done with the parameter $q = 2$ and all results are an average of 50 runs.

Table 4.1: Experimental results on solving the MCE problem using Algorithm 1.

$m = n$	k	κ	Sample set size	Runtime (s) SampleSet	Runtime (s) inhQMLE solver	Success probability
10	20	5	2	21	3154	0.70
11	22	5	3	31	2004	0.63
12	24	5	6	76	13873	0.73

The second approach, described in Section 4.5.3 uses the bilinear structure of hQMLE instances derived from MCE instances to have an improved algorithm for building the sample sets and a more precise predicate that results in fewer queries to the inhQMLE solver. The consequence of these two improvements to the runtime can be observed in Table 4.2 where we show experimental results of Algorithm 2 using the non-zeros subsets. Recall that, this approach can be used only when there exist at least two roots of $\mathcal{F}$ and $\mathcal{P}$. Otherwise, the sampled sets contain only the trivial root and the instance is solved using Algorithm 1. Table 4.2 shows results of the case when the sets are non-trivial and the probability of this case for the given parameters is shown in the last column. For efficiency, we take the minimal subset with a common dimension of the kernel of $\mathbf{F_b}$, and when looking for collisions, we are careful to skip pairs $(\mathbf{ab}, \mathbf{a'b'})$ where $\dim\ker \mathbf{F_b} = \dim\ker \mathbf{P_{b'}}$ but $\dim\ker D_{(\mathbf{a},\mathbf{b})}\mathcal{F} \neq \dim\ker D_{(\mathbf{a'},\mathbf{b'})}\mathcal{P}$. In these experiments, $q = 3$ and all results are an average of 50 runs.

Table 4.2: Experimental results on solving the MCE problem using the non-zeros-subsets variant of Algorithm 2.

$m = n$	k	Sample set size	Runtime (s) SampleZeros	Runtime (s) inhQMLE solver	% instances with two roots
8	15	10.4	0.56	175.34	24
	14	35.56	0.60	236.12	68
9	17	12.00	1.74	396.04	22
	16	37.97	1.72	1020.25	70
10	19	25.6	5.13	2822.32	14
	18	36.72	5.05	1809.09	82

Our experiments confirm that Algorithm 2 outperforms Algorithm 1 for solving MCE instances with non-trivial roots.

Acknowledgements. The authors thank Charles Bouillaguet for providing the implementation resulting from [37].

Chapter 5

MEDS

5.1 Placeholder

The paper will go here.

5.2 Introduction

Post-Quantum Cryptography (PQC) comprises all the primitives that are believed to be resistant against attackers equipped with a considerable quantum computing power. Several such schemes have been around for a long time [131, 138], some being in fact almost as old as RSA [113]; however, the area itself was not formalized as a whole until the early 2000s, for instance with the first edition of the PQCrypto conference [125]. The area has seen a dramatic increase in importance and volume of research over the past few years, partially thanks to NIST's interest and the launch of the PQC Standardization process in 2017 [127]. After 4 years and 3 rounds of evaluation, the process has crystallized certain mathematical tools as standard building blocks (e.g. lattices, linear codes, multivariate equations, isogenies etc.). Some algorithms [67, 82, 97] have now been selected for standardization, with an additional one or two to be selected among a restricted set of alternates [1, 5, 7] after another round of evaluation. While having a range of candidates ready for standardization may seem satisfactory, research is still active in designing PQC primitives. In particular, NIST has expressed the desire for a greater diversity among the hardness assumptions behind signature schemes, and announced a partial reopening of the standardization process for precisely the purpose of collecting non-lattice-based protocols.

Cryptographic group actions are a popular and powerful instrument for constructing secure and efficient cryptographic protocols. The most well-known is, without a doubt, the action of finite groups on the integers modulo a prime, or the set of points on an elliptic curve, which give rise to the *Discrete Logarithm Problem (DLP)*, i.e. the backbone of public-key cryptography. Recently, proposals for post-quantum cryptographic group actions started to emerge, based on the tools identified above: for instance, isogenies [45], linear codes [30], trilinear forms [153] and even lattices [95]. All of these group actions provide very promising solutions for cryptographic schemes, for example signatures [16, 56, 153], ring signatures [15, 27] and many others; at the same time, they are very different in nature, with unique positive and negative aspects.

Our Contribution. In this work, we formalize a new cryptographic group action based on the notion of *Matrix Code Equivalence*. This is similar in nature to the *code equivalence* notion at the basis of LESS [16, 30], and in fact belongs to a larger class of isomorphism problems that include, for example, the lattice isomorphism problem, and the well-known isomorphism of polynomials [131]. The hardness of the MCE problem was studied in [53, 139], from which it is possible to conclude that this is a suitable problem for post-quantum cryptography. Indeed, we show that it is possible to use MCE to build a zero-knowledge protocol, and hence a signature scheme, which we name *Matrix Equivalence Digital Signature*, or simply MEDS. For our security analysis, we first study in detail the collision attacks from [139] and then we develop two new attacks. The first attack that we propose uses a nontrivial algebraic modeling inspired from the minors modellings of MinRank in [13, 71]. The second one is an adaptation of Leon's algorithm [109] for matrix codes. Based on this analysis, we provide an initial parameter choice, together with several computational optimizations, resulting in a scheme with great flexibility and very competitive data sizes.

A reference implementation for the full scheme is available at

meds-pqc.org

Furthermore, we show that this group action allows for the construction of (linkable) ring signatures, with performance results that improve on the existing state of the art [15].

5.3 Preliminaries

Let $\mathcal{K}$ be the finite field of q elements. $\mathrm{GL}_n(q)$ and $\mathrm{AGL}_n(q)$ denote respectively the general linear group and the general affine group of degree n over $\mathcal{K}$. We use bold letters to denote vectors $\mathbf{a}, \mathbf{c}, \mathbf{x}, \ldots$, and matrices $\mathbf{A}, \mathbf{B}, \ldots$. The entries of a vector $\mathbf{a}$ are denoted by a_i, and we write $\mathbf{a} = (a_1, \ldots, a_n)$ for a (row) vector of dimension n over some field. Similarly, the entries of a matrix $\mathbf{A}$ are denoted by a_{ij}. Random sampling from a set S is denoted by $a \xleftarrow{\$} S$. For two matrices $\mathbf{A}$ and $\mathbf{B}$, we denote the Kronecker product by $\mathbf{A} \otimes \mathbf{B}$. Finally, we denote the set of all $m \times n$ matrices over $\mathcal{K}$ by $\mathcal{M}_{m,n}(\mathcal{K})$.

5.3.1 Cryptographic group actions

Definition 5.1. Let X be a set and $(G, \cdot)$ be a group. A group action is a mapping

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

such that the following conditions hold for all $x \in X$:

- $e \star x = x$, where e is the identity element of G.
- $g_2 \star (g_1 \star x) = (g_2 \cdot g_1) \star x$, for all $g_1, g_2 \in G$.

A group action can have a number of mathematically desirable properties. For example, we say that a group action is:

- *Commutative*: for any $g_1, g_2 \in G$, we have $g_2 \star (g_1 \star x) = g_1 \star (g_2 \star x)$.
- *Transitive*: given $x_1, x_2 \in X$, there is some $g \in G$ such that $g \star x_1 = x_2$.
- *Free*: if $g \star x = x$, then g is the identity.

In particular, a *cryptographic* group action is a group action with some additional properties that are useful for cryptographic applications. To begin with, there are some desirable properties of computational nature. Namely, the following procedures should be efficient:

- *Evaluation*: given x and g, compute $g \star x$.
- *Sampling*: sample uniformly at random from G.
- *Membership testing*: verify that $x \in X$.

Finally, cryptographic group actions should come with security guarantees; for instance, the *vectorization problem* should be hard:

Problem 5.1 (Group Action Vectorization)**. Given:** The pair $x_1, x_2 \in X$. **Goal:** Find, if any, $g \in G$ such that $g \star x_1 = x_2$.

Early constructions using this paradigm are based on the action of finite groups of prime order, for which the vectorization problem is the discrete logarithm problem. Lately, multiple isogeny-based constructions have appeared: see, for instance, the work of Couveignes in [52] and later by Rostovtsev and Stolbunov [144]. A general framework based on group actions was explored in more detail by [4], allowing for the design of several primitives. The holy grail are those cryptographic group actions that possess both the mathematical and cryptographic properties listed above. Currently, CSIDH [45] is the only post-quantum commutative cryptographic group action, although there is an ongoing debate about the efficiency and quantum hardness of its vectorization problem [31]. In Section 5.4, we introduce the group action that is relevant to our work.

5.3.2 Protocols

We give here an explicit characterization of the protocols we will build. The corresponding security definitions are presented only in an informal manner; formal definitions will are included in the full version of this work.[1]

Definition 5.2. A *Sigma protocol* is a three-pass interactive protocol between two parties: a prover $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$ and a verifier $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$. The protocol is composed of the following procedures:

I. Keygen: on input some public data (including system parameters), output a public key pk (the instance) and the corresponding secret key sk (the witness). Give sk to the prover; pk is distributed publicly and is available to all parties. For simplicity, we assume that the public data is available as input in all the remaining procedures.

II. Commit: on input the public key pk, P_1 outputs a public commitment cmt and sends it to the verifier.

III. $\mathsf{Challenge}$: on input the public key pk and the commitment cmt, V_1 samples uniformly at random a challenge chal from the challenge space C and sends it to the prover.

IV. $\mathsf{Response}$: on input the secret key sk, the public key pk, the commitment cmt and the challenge chal, P_2 outputs a response resp and sends it to the verifier.

V. Verify: on input a public key pk, the commitment cmt, the challenge chal, and the response resp, V_2 outputs either 1 (accept) if the *transcript* $(\mathsf{cmt}, \mathsf{chal}, \mathsf{resp})$ is valid, or 0 (reject) otherwise.

A Sigma protocol is usually required to satisfy the following properties. First, if the statement is true, an honest prover is always able to convince an honest verifier. This property is called *Completeness*. Secondly, a dishonest prover cannot convince an honest verifier other than with a small probability. This is captured by the *Soundness* property, which also bounds such probability, usually known as *soundness error* or, informally, *cheating probability*. Finally, the protocol has to be *Zero-Knowledge*, i.e. anyone observing the transcript (including the verifier) learns nothing other than the fact that the statement is true. definitions will be included in the full version of this work, but are omitted here in the interest of space.

Definition 5.3. A *Digital Signature scheme* is a protocol between 2 parties: a signer S and a verifier V. The protocol is composed of the following procedures:

I. Keygen: on input the public data (including system parameters), output a secret signing key sk for S and the corresponding public verification key pk.

[1] https://eprint.iacr.org/2022/1559.pdf

II. Sign: on input a secret key sk and a message msg, output a signature σ.

III. Verify: on input a public key pk, a message msg and a signature σ, V outputs either 1 (accept) if the signature is valid, or 0 (reject) otherwise.

Correctness means that an honest signer is always able to get verified. The usual desired security notion for signature schemes is *Unforgeability*, which guarantees computationally infeasible to forge a valid signature without knowing the secret signing key. definitions to the full version of this work.

Definition 5.4. A *Ring Signature scheme* is a protocol between $r+1$ parties: potential signers $\mathsf{S}_1, \dots, \mathsf{S}_r$ (the *ring*) and a verifier V. The protocol is composed of the following procedures:

I. Keygen: on input the public data (including system parameters), output a secret key sk_j for each signer S_j, and the corresponding public key pk_j.

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

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

A crucial feature of a ring signature scheme is that anyone in the ring can produce a valid signature (on behalf on the entire ring), and the verifier can check validity, but cannot learn the identity of the signer. This is achieved by requiring that the scheme satisfies the *Anonymity* property. This captures the idea of protecting the identity of the signer, i.e. it should be impossible for a verifier to tell which secret key was used to sign. Then, *Unforgeability* corresponds to the familiar security notion for signature schemes presented above, extended to include all ring participants. In other words, it should be infeasible to forge a valid signature without knowing at least one of the secret keys in the ring. In addition, *Correctness* is also required, as before.

Linkable ring signature schemes possess additional security features. These protocols are composed of the same three procedures described in Definition 5.4, plus a fourth one, given below:

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

A linkable ring signature scheme is required to satisfy the following security properties. *Linkability* asks that it is computationally infeasible for an adversary to produce more than r unlinked valid signatures, even if some or all of the r public keys are malformed. *Linkable Anonymity* guarantees that, even if an adversary is able to obtain multiple signatures from the same signer, they are still unable to determine which secret key was used. *Non-Frameability* protects the user's identity by requiring that it is computationally infeasible for an adversary to produce a valid signature linked to one produced by an honest party.

Remark 5.1. Note that the linkability property is usually formulated in an alternative way, that is, if more than r valid signatures are produced, then the Link algorithm will output 1 on at least two of them. It is then easy to see, as also pointed out in [27, Remark 2.4], that unforgeability is obtained directly by satisfying linkability and non-frameability.

Finally, we recall the definition of *commitment scheme*, which is a tool necessary for our particular construction of ring signatures. This is a non-interactive function $\mathsf{Com} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{2\lambda}$ mapping a message string of arbitrary length to a commitment string of 2λ bits. The first λ bits of input, chosen uniformly at random, guarantee that the input message is hidden in a very strong sense, as captured in the next definition.

Definition 5.5. Given an adversary A, we define the two following quantities:

$$\mathsf{Adv}^{\mathsf{Bind}}(\mathsf{A}) = \mathsf{Pr}\Big[\,\mathsf{Com}(\mathsf{r}, \mathbf{x}) = \mathsf{Com}(\mathsf{r}', \mathbf{x}') \mid (\mathsf{r}, \mathbf{x}, \mathsf{r}', \mathbf{x}') \leftarrow \mathsf{A}(1^\lambda)\Big];$$

$$\mathsf{Adv}^{\mathsf{Hide}}(\mathsf{A}, \mathbf{x}, \mathbf{x}') = \left| \Pr_{\mathsf{r} \leftarrow \{0,1\}^\lambda}\Big[\mathsf{A}(\mathsf{Com}(\mathsf{r}, \mathbf{x})) = 1\Big] - \Pr_{\mathsf{r} \leftarrow \{0,1\}^\lambda}\Big[\mathsf{A}(\mathsf{Com}(\mathsf{r}, \mathbf{x}')) = 1\Big] \right|.$$

We say that Com is *computationally binding* if, for all polynomial-time adversaries A, the quantity $\mathsf{Adv}^{\mathsf{Bind}}(\mathsf{A})$ is negligible in λ. We say that Com is *computationally hiding* if, for all polynomial-time adversaries A and every pair $(\mathbf{x}, \mathbf{x}')$, the quantity $\mathsf{Adv}^{\mathsf{Hide}}(\mathsf{A})$ is negligible in λ.

5.4 The Matrix Code Equivalence Problem

A $[m \times n, k]$ *matrix code* is a subspace $\mathcal{C}$ of $\mathcal{M}_{m,n}(\mathcal{K})$. These objects are usually measured with the rank metric, where the *distance* between two matrices $\mathbf{A}, \mathbf{B} \in \mathcal{M}_{m,n}(\mathcal{K})$ is defined as $d(\mathbf{A}, \mathbf{B}) = \text{Rank}(\mathbf{A} - \mathbf{B})$. We denote the basis of the subspace by $\langle \mathbf{C_1}, \ldots, \mathbf{C_k} \rangle$, where the $\mathbf{C_i}$'s are linearly independent elements of $\mathcal{M}_{m,n}(\mathcal{K})$. Due to symmetry, without loss of generality, in the rest of the text we will assume $m \leqslant n$.

For a matrix $\mathbf{A} \in \mathcal{M}_{m,n}(\mathcal{K})$, let Vec be a mapping that sends a matrix $\mathbf{a}$ to the vector $\text{Vec}(\mathbf{A}) \in \mathbb{F}_q^{mn}$ obtained by 'flattening' $\mathbf{A}$, i.e.:

$$\text{Vec}: \ \mathbf{A} = \begin{pmatrix} a_{1,1} & \ldots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \ldots & a_{m,n} \end{pmatrix} \mapsto \text{Vec}(\mathbf{A}) = (a_{1,1}, \ldots, a_{1,n}, \ldots, a_{m,1}, \ldots, a_{m,n}).$$

The inverse operation is denoted by Mat, i.e. $\text{Mat}(\text{Vec}(\mathbf{A})) = \mathbf{A}$. Using the map Vec, an $[m \times n, k]$ matrix code can be thought of as an $\mathbb{F}_q$-subspace of

$\mathbb{F}_q^{mn}$, and thus we can represent it with a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times mn}$, in a manner similar to the common representation for linear codes. Indeed, if $\mathcal{C}$ is an $[m \times n, k]$ matrix code over $\mathbb{F}_q$, we denote by $\mathrm{Vec}(\mathcal{C})$ the vectorization of $\mathcal{C}$ i.e.:

$$\mathrm{Vec}(\mathcal{C}) := \{\mathrm{Vec}(\mathbf{A})\colon \mathbf{A} \in \mathcal{C}\}.$$

In this case, $\mathrm{Vec}(\mathcal{C})$ is a k-dimensional $\mathbb{F}_q$-subspace of $\mathbb{F}_q^{mn}$.

Definition 5.6. Let $\mathcal{C}$ and $\mathcal{D}$ be two $[m \times n, k]$ matrix codes over $\mathbb{F}_q$. We say that $\mathcal{C}$ and $\mathcal{D}$ are *equivalent* if there exist two matrices $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$ such that $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$, i.e. for all $\mathbf{C} \in \mathcal{C}$, $\mathbf{A}\mathbf{C}\mathbf{B} \in \mathcal{D}$.

The equivalence between two matrix codes can be expressed using the Kronecker product of $\mathbf{A}^\top$ and $\mathbf{B}$, which we denote by $\mathbf{A}^\top \otimes \mathbf{B}$.

Lemma 5.7. Let $\mathcal{C}$ and $\mathcal{D}$ be two $[m \times n, k]$ matrix codes over $\mathbb{F}_q$. Suppose that $\mathcal{C}$ and $\mathcal{D}$ are equivalent with $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$, with $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$. If $\mathbf{G}$ and $\mathbf{G}'$ are generator matrices for $\mathcal{C}$ and $\mathcal{D}$ respectively, then there exists a $\mathbf{T} \in \mathrm{GL}_k(q)$ such that $\mathbf{G}' = \mathbf{T}\mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B})$.

It is common to write the generator matrices in systematic form (i.e., as a matrix of the shape $(I|M)$); we denote this operation by SF. Following Lemma 5.7, this gives us that $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$ if and only if $\mathsf{SF}(\mathbf{G}') = \mathsf{SF}(\mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B}))$.
To simplify notation, we introduce the following operator:

$$\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}) := \mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B}).$$

We are now ready to describe some hard problems connected to the objects we just introduced. The Matrix Code Equivalence (MCE) problem is formally defined as follows:

Problem 5.2 (Matrix Code Equivalence)**.** MCE $(k, n, m, \mathcal{C}, \mathcal{D})$:
Given: Two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$.
Goal: Determine if there exist $\mathbf{A} \in \mathrm{GL}_m(q)\mathbf{B} \in \mathrm{GL}_n(q)$ such that $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$.

The map $(\mathbf{A}, \mathbf{B}) : \mathbf{C} \mapsto \mathbf{A}\mathbf{C}\mathbf{B}$ is an *isometry* between $\mathcal{C}$ and $\mathcal{D}$, in the sense that it preserves the rank i.e. $\mathrm{Rank}\, \mathbf{C} = \mathrm{Rank}(\mathbf{A}\mathbf{C}\mathbf{B})$. When $n = m$, such isometries can also be extended by transpositions of codewords, however, we choose to work with this smaller set of isometries for simplicity, at no cost to cryptographic security. Note that, although we defined MCE as a decisional problem, our signature construction relies on the computational version of it.

Remark 5.2. We thank Giuseppe D'Alconzo for the following sharp observation: An MCE instance of dimension k with $m \times n$ matrices over $\mathbb{F}_q$ can be viewed as a 3-tensor problem, which is symmetrical in its arguments k, m and n. This means that it is equivalent to an MCE instance of dimension m with $k \times n$ matrices and to an MCE instance of dimension n with $k \times m$ matrices. Switching to equivalent instances is a matter of changing perspective on the $k \times m \times n$ object over $\mathbb{F}_q$ defined by $A_{ijl} = A_{ij}^{(l)}$. In other words, each basis matrix $m \times n$ defines a slice of a cube, and one can take different slices for equivalent instances.

The following related problem is at the basis of the ring signature construction.

Problem 5.3 (Inverse Matrix Code Equivalence)**.** IMCE $(k, n, m, \mathcal{C}, \mathcal{D}_1, \mathcal{D}_2)$:
Given: Three k-dimensional matrix codes $\mathcal{C}, \mathcal{D}_1, \mathcal{D}_2 \subset \mathcal{M}_{m,n}(q)$.
Goal: Determine if there exist $\mathbf{A} \in \mathrm{GL}_m(q)\mathbf{B} \in \mathrm{GL}_n(q)$ such that
$\mathcal{D}_1 = \mathbf{A}\mathcal{C}\mathbf{B}$ and $\mathcal{D}_2 = \mathbf{A}^{-1}\mathcal{C}\mathbf{B}^{-1}$.

Problem 5.3 is closely connected to MCE, in a manner similar to the well-known connection between discrete logarithm and related problems, i.e.:

$$\mathsf{DLOG} \geq \mathsf{DDH} \geq \mathsf{InverseDDH}.$$

Reductions in the opposite direction are not known, as explained for example in [12]. Although this does not provide any further indication about the complexity of such problems, no explicit weaknesses are known either, and the problems are generally considered hard. A more generic overview about hardness of group action-based problems (of which the discrete logarithm is only a particular instantiation) is given in [76], with similar conclusions. For our specific case, we present a discussion about the concrete hardness of IMCE in Section 5.7.

Finally, we present a *multiple-instance* version of MCE, which is at the base of one of the optimizations, using *multiple public keys*, which we will describe in Section 5.6. It is easy to see that this new problem reduces to MCE, as done for instance in [16] for the Hamming case.

Problem 5.4 (Multiple Matrix Code Equivalence)**.** MIMCE $(k, n, m, r, \mathcal{C}, \mathcal{D}_1, \dots, \mathcal{D}_r)$:
Given: $(r+1)$ k-dimensional matrix codes $\mathcal{C}, \mathcal{D}_1, \dots, \mathcal{D}_r \subset \mathcal{M}_{m,n}(\mathcal{K})$.
Goal: Find – if any – $\mathbf{A} \in \mathrm{GL}_m(q)\mathbf{B} \in \mathrm{GL}_n(q)$ such that $\mathcal{D}_i = \mathbf{A}\mathcal{C}\mathbf{B}$ for some $i \in \{1, \dots, r\}$.

The MCE problem has been shown to be at least as hard as the Code Equivalence problem in the Hamming metric [53]. Furthermore, under moderate assumptions, MCE is equivalent to the homogeneous version of the Quadratic Maps Linear Equivalence problem (QMLE) [139], which is considered the hardest among polynomial equivalence problems. An extensive security evaluation will be given in Section 5.7, encompassing an overview of the best attack techniques and concrete security estimates. From this, we infer a choice of parameters in Section 5.8.2.

To conclude, we now lay out the details of the MCE-based group action, given by the action of isometries on k-dimensional matrix codes. That is, the set X is formed by the k-dimensional matrix codes of size $m \times n$ over some base field $\mathcal{K}$, and the group $G = \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$ acts on this set via isometries as follows:

$$\begin{array}{cccc} \star: & G \times X & \to & X \\ & ((\mathbf{A}, \mathbf{B}), \mathcal{C}) & \mapsto & \mathbf{A}\mathcal{C}\mathbf{B} \end{array}$$

We write $\mathcal{G}_{m,n}(q)$ to denote this group of isometries and $\mathbf{M}_{k,m,n}(q)$ for the set of k-dimensional matrix codes; to simplify notation, we drop the indices k, m, n

and q when clear from context. Then, for this MCE-based group action the Vectorization Problem is precisely Problem 5.2. This action is not commutative and in general neither transitive nor free. We can restrict the set $\mathbf{M}$ to a single well-chosen orbit to make the group action both transitive and free. In fact, picking any orbit generated from some starting code $\mathcal{C}$ ensures transitivity, and the group action is free if the chosen code $\mathcal{C}$ has trivial automorphism group $\mathrm{Aut}_{\mathcal{G}}(\mathcal{C}) := \{\phi \in \mathcal{G} : \phi(\mathcal{C}) = \mathcal{C}\}$, where trivial means up to scalars in $\mathbb{F}_q$[2]. The non-commutativity is both positive and negative: although it limits the cryptographical design possibilities, e.g. key exchange becomes hard, it prevents quantum attacks to which commutative cryptographic group actions are vulnerable, such as Kuperberg's algorithm for the dihedral hidden subgroup problem [106].

With regards to efficiency, it is immediate to notice that our group action is very promising, given that the entirety of the operations in the proposed protocols is simple linear algebra; this is in contrast with code-based literature (where complex decoding algorithms are usually required) and other group actions (e.g. isogeny-based) which are burdened by computationally heavy operations. Further details about performance are given in Section 5.8.

5.5 Protocols from Matrix Code Equivalence

The efficient non-commutative cryptographic group action provided by MCE from Section 5.4 yields a promising building block for post-quantum cryptographic schemes. In this section, we obtain a digital signature scheme by first designing a Sigma protocol and then applying the Fiat-Shamir transformation [78]. With a similar procedure, we are able to obtain (linkable) ring signatures as well.

5.5.1 Sigma Protocols and Signatures

The first building block in our work is the Sigma protocol in Figure 5.1, in which a Prover proves the knowledge of an isometry $(\mathbf{A}, \mathbf{B})$ between two equivalent matrix codes. The security result is given in Theorem 5.8.

Theorem 5.8. The Sigma protocol described above is complete, 2-special sound and honest-verifier zero-knowledge assuming the hardness of the MCE problem.

Proof. We prove the three properties separately.

Completeness. An honest prover will always have his response accepted by the verifier. In fact, for the case $c = 0$, we have

$$h' = \mathsf{H}(\mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_0))) = \mathsf{H}(\mathsf{SF}(\pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}(\mathbf{G}_0)) = \mathsf{H}(\tilde{\mathbf{G}}) = h.$$

[2]More accurately, as the action of an isometry (A, B) is only interesting up to scalars $\lambda, \mu \in \mathbb{F}_q$, the group that is acting *freely* is $\mathrm{PGL}_m(q) \times \mathrm{PGL}_n(q)$.

Public Data
$q, m, n, k, \lambda \in \mathbb{N}$.
$\mathsf{H} : \{0,1\}^* \to \{0,1\}^{2\lambda}$.

II. Commit(pk)

1. $\tilde{\mathbf{A}}, \tilde{\mathbf{B}} \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
2. Compute $\tilde{\mathbf{G}} = \mathsf{SF}(\pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}(\mathbf{G}_0))$.
3. Compute $h = \mathsf{H}(\tilde{\mathbf{G}})$.
4. Set $\mathsf{cmt} = h$.
5. Send cmt to verifier.

IV. Response$(\mathsf{sk}, \mathsf{pk}, \mathsf{cmt}, \mathsf{chal})$

1. If $\mathsf{chal} = 0$ set $(\mu, \nu) = (\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$.
2. If $\mathsf{chal} = 1$ set $(\mu, \nu) = (\tilde{\mathbf{A}}\mathbf{A}^{-1}, \mathbf{B}^{-1}\tilde{\mathbf{B}})$.
3. Set $\mathsf{resp} = (\mu, \nu)$.
4. Send resp to verifier.

I. Keygen$()$

1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k \times mn}$ in standard form
2. $(\mathbf{A}, \mathbf{B}) \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
3. Compute $\mathbf{G}_1 = \mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}_0))$.
4. Set $\mathsf{sk} = (\mathbf{A}, \mathbf{B})$ and $\mathsf{pk} = (\mathbf{G}_0, \mathbf{G}_1)$.

III. Challenge$()$

1. $c \xleftarrow{\$} \{0, 1\}$.
2. Set $\mathsf{chal} = c$.
3. Send chal to prover.

V. Verify$(\mathsf{pk}, \mathsf{cmt}, \mathsf{chal}, \mathsf{resp})$

1. If $\mathsf{chal} = 0$ compute
   $h' = \mathsf{H}(\mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_0)))$.
2. If $\mathsf{chal} = 1$ compute
   $h' = \mathsf{H}(\mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_1)))$.
3. Accept if $h' = \mathsf{cmt}$ or reject otherwise.

Figure 5.1: MCE Sigma Protocol

For the case $c = 1$, instead, observe that

$$\pi_{\tilde{\mathbf{A}}\mathbf{A}^{-1},\mathbf{B}^{-1}\tilde{\mathbf{B}}}(\mathbf{G}_1) = \pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}\left(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}(\mathbf{G}_1)\right).$$

Since $\mathbf{G}_1 = \mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}_0))$, then $\mathbf{G}_1 = \mathbf{T}\left(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}_0)\right)$ for some $\mathbf{T} \in \mathrm{GL}_k(q)$. Hence, we have that

$$\pi_{\tilde{\mathbf{A}}\mathbf{A}^{-1},\mathbf{B}^{-1}\tilde{\mathbf{B}}}(\mathbf{G}_1) = \pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}\left(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}\left(\mathbf{T}\left(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}_0)\right)\right)\right) = \mathbf{T}\left(\pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}(\mathbf{G}_0)\right).$$

Now, computing the systematic form and applying H, we again have $h' = h$.

2-Special Soundness. To get the extractor which produces the witness, we prove that having obtained two valid transcripts with the same commitment and a different challenge, we can extract a solution for the underlying MCE problem. This demonstrates that this Sigma protocol is a Proof of Knowledge with soundness error 1/2.

Let $(h, 0, \mathsf{resp}_0)$ and $(h, 1, \mathsf{resp}_1)$ be two valid transcripts, where $\mathsf{resp}_0 = (\mu_0, \nu_0)$ and $\mathsf{resp}_1 = (\mu_1, \nu_1)$. Since both pass verification, we have that

$$\mathsf{H}(\mathsf{SF}(\pi_{\mu_0,\nu_0}(\mathbf{G}_0))) = h = \mathsf{H}(\mathsf{SF}(\pi_{\mu_1,\nu_1}(\mathbf{G}_1))).$$

Then, since we assume that H is collision resistant, it must be that

$$\mathsf{SF}(\pi_{\mu_0,\nu_0}(\mathbf{G}_0)) = \mathsf{SF}(\pi_{\mu_1,\nu_1}(\mathbf{G}_1)).$$

From this, it is easy to see that $\mathsf{SF}(\pi_{\mu^*,\nu^*}(\mathbf{G}_0)) = \mathbf{G}_1$, for an isometry $(\mu^*, \nu^*) = = (\mu_1^{-1}\mu_0, \nu_0\nu_1^{-1})$, i.e. (μ^*, ν^*) is a solution to the MCE problem.

Honest-Verifier Zero-Knowledge. To show this, we provide a simulator S which, without the knowledge of the witness, is able to produce a transcript which is indistinguishable from one obtained after an interaction with an honest verifier. When the challenge is $c = 0$, the prover's response is $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$, and does not involve the witness. Hence, it is obvious that S can obtain a flawless simulation by simply performing the same steps as an honest prover would. Thus,

$$\mathsf{Pr}\Big[\mathsf{V}_2(\mathsf{pk}, \mathsf{cmt}, 0, \mathsf{resp}) = 1 \mid (\mathsf{cmt}, 0, \mathsf{resp}) \leftarrow \mathsf{S}(\mathsf{pk})\Big] = 1.$$

When $c = 1$, the simulator generates two random matrices $(\mathbf{A}^*, \mathbf{B}^*)$, then sets the commitment to $\mathsf{cmt} = \mathsf{H}(SF(\pi_{\mathbf{A}^*,\mathbf{B}^*}(\mathbf{G}_1)))$ and the response to $\mathsf{resp} = (\mathbf{A}^*, \mathbf{B}^*)$. When $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$ is uniformly generated, then the matrices $(\tilde{\mathbf{A}}\mathbf{A}^{-1}, \mathbf{B}^{-1}\tilde{\mathbf{B}})$ are uniformly generated too, and hence follow the same distribution as $(\mathbf{A}^*, \mathbf{B}^*)$. Furthermore, the triple $(\mathsf{cmt}, 1, \mathsf{resp})$ is valid and therefore

$$\mathsf{Pr}\Big[\mathsf{V}_2(\mathsf{pk}, \mathsf{cmt}, 1, \mathsf{resp}) = 1 \mid (\mathsf{cmt}, 1, \mathsf{resp}) \leftarrow \mathsf{S}(\mathsf{pk})\Big] = 1.$$

This concludes the proof. $\square$

$\square$

Applying the Fiat-Shamir transformation gives the signature scheme in Figure 5.2.

Public key and signature size.

We begin by calculating the communication costs for the Sigma protocol of Figure 5.1. Note that, for the case $c = 0$, the response (μ, ν) consists entirely of randomly-generated objects, and is efficiently represented by a single seed (that can be used to generate both matrices). This yields the following cost per round, in bits:

$$\begin{cases} 3\lambda + 1 & \text{if } c = 0 \\ 2\lambda + 1 + (m^2 + n^2)\lceil \log_2(q) \rceil & \text{if } c = 1 \end{cases}$$

remembering that seeds are λ bits and hash digests 2λ to avoid collision attacks.

For the signature scheme we calculate the sizes as follows. First, since the matrix $\mathbf{G}_0$ is random, it can also be represented via a short seed, and therefore can be included in the public key at negligible cost (see Algorithm I. of Figure 5.2). Keeping in mind that the number of rounds t is equal to the value of the desired security level λ, the protocol above yields the following sizes (in bits):

- Public key size: $\lambda + k(mn - k)\lceil \log_2(q) \rceil$
- Average signature size: $t\left(1 + \dfrac{\lambda + (m^2 + n^2)\lceil \log_2(q) \rceil}{2}\right)$.

```
Public Data
$q, m, n, k, t = \lambda \in \mathbb{N}$.
$\mathsf{H} : \{0,1\}^* \to \{0,1\}^t$.

I. Keygen()
  1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k\times mn}$ in standard form
  2. Set $\mathbf{A}_0 = \mathbf{I}_m$, $\mathbf{B}_0 = \mathbf{I}_n$.
  3. $\mathbf{A}_1, \mathbf{B}_1 \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
  4. Compute $\mathbf{G}_1 = \mathsf{SF}(\pi_{\mathbf{A}_1,\mathbf{B}_1}(\mathbf{G}_0))$.
  5. Set $\mathsf{sk} = (\mathbf{A}_1, \mathbf{B}_1)$ and $\mathsf{pk} = (\mathbf{G}_0, \mathbf{G}_1)$.

II. Sign(sk)
  1. For all $i = 0, \dots, t-1$:
       i. $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
       ii. Compute $\tilde{\mathbf{G}}_i = \mathsf{SF}(\pi_{\tilde{\mathbf{A}}_i,\tilde{\mathbf{B}}_i}(\mathbf{G}_0))$.
  2. Compute $h = \mathsf{H}(\tilde{\mathbf{G}}_0, \dots, \tilde{\mathbf{G}}_{t-1}, \mathsf{msg})$.
  3. Parse $h = [h_0|\dots|h_{t-1}]$, $h_i \in \{0,1\}$.
  4. For all $i = 0, \dots, t-1$:
       i. Set $(\mu_i, \nu_i) = (\tilde{\mathbf{A}}_i\mathbf{A}_{h_i}^{-1}, \mathbf{B}_{h_i}^{-1}\tilde{\mathbf{B}}_i)$.
  5. Set $\sigma = (h, \mu_0, \dots, \mu_{t-1}, \nu_0, \dots, \nu_{t-1})$.
  6. Send $\sigma$ to verifier.

III. Verify(pk, msg, $\sigma$)
  1. Parse $h = [h_0|\dots|h_{t-1}]$, $h_i \in \{0,1\}$.
  2. For all $i = 0, \dots, t-1$:
       i. Set $\hat{\mathbf{G}}_i = \mathsf{SF}(\pi_{\mu_i,\nu_i}(\mathbf{G}_{h_i}))$.
  3. Compute $h' = \mathsf{H}(\hat{\mathbf{G}}_0, \dots, \hat{\mathbf{G}}_{t-1}, \mathsf{msg})$.
  4. Accept if $h' = h$ or reject otherwise.
```

Figure 5.2: The basic signature scheme

5.5.2 Ring Signatures

The protocol of Figure 5.1 can be easily adapted to serve as a building block for a ring signature, by providing the ring of users with individual public keys, corresponding to equivalent matrix codes. Any user can then answer the verifier's challenge via the selected private key. The construction crucially utilizes a dedicated primitive known as *Index-Hiding Merkle Tree (IHMT)*. This primitive was first introduced in [27] as a variation on the traditional construction of Merkle trees. With this variant, in fact, the position of a leaf is not revealed, which is necessary to ensure anonymity. This can be accomplished by specifying a different method to construct the tree, based on an alternative ordering (e.g. lexicographic). For further details, we refer the reader to [27]. A visual representation is given in Figure 5.3, where we remind the reader that Algorithm I. Keygen is executed *once for each prover*, Algorithm II. Commit is the same regardless of the chosen prover, and Algorithm IV. Response is instead unique for the specific prover identified by $j^* \in \{1, \dots, r\}$.

It is easy to see that the construction in Figure 5.3 satisfies the Completeness, 2-Special Soundness and Honest-Verifier Zero-Knowledge properties, similarly to the protocol of Figure 5.1. The proof is analogue to the proof of Theorem 5.8, with only minor differences (such as the reversal in the roles of the challenges), that have no noticeable impact; this is therefore omitted in the interest of space. The protocol can then be turned into a ring signature scheme using Fiat-Shamir as usual, i.e. in the same manner as what was done for the protocol of Figure 5.1. Furthermore, we can make the ring signature scheme *linkable*, by means of a few modifications which we explain next. To avoid needless repetition, we avoid presenting the ring signature scheme in its extended form (as in Figure 5.2) and we move on instead to discussing the linkable variant; we will then present the linkable ring signature scheme, in its entirety, to conclude this section.

To begin, recall the group action $\star : G \times X \to X$, formalized in Section 5.4, with $X = \mathcal{M}$ the set of k-dimensional matrix codes, $G = \mathcal{G}$ the group of isometries for such codes, and the action given by $\star : ((\mathbf{A}, \mathbf{B}), \mathcal{C}) \mapsto \mathbf{A}\mathcal{C}\mathbf{B}$. We now require another group action $* : G \times Y \to Y$, satisfying the following properties.

Definition 5.9. Let $\star : G \times X \to X$ and $* : G \times Y \to Y$ be two group actions. We define the following properties for the pair $(\star, *)$:

- *Linkability*: Given $(x, y) \in X \times Y$, it is hard to output $g, g' \in G$ with $g \star x = g' \star x$ and $g * y \neq g' * y$.

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

- *Non-Frameability*: Given $(x, y) \in X \times Y$, $x' = g \star x$ and $y' = g * y$, for g sampled uniformly at random from G, it is hard to output $g' \in G$ with $g' * y = y'$.

Note how these three properties recall the Linkability, Linkable Anonymity and Non-Frameability properties introduced in Section 5.3.2. Indeed, showing that the pair of group actions satisfies the above properties is the key to constructing a secure linkable ring signature scheme. Also, as noted before, one could define unforgeability in a manner similar to the last property (by asking to output $g' \in G$ with $g' \star x = x'$) but this is not necessary, since this is a direct consequence of linkability and non-frameability. Informally, then, one can treat elements $y \in Y$ as "tags", that can be checked to establish the link. To this end, it is convenient to introduce an efficiently computable function $\mathsf{Link} : Y \times Y \to \{0, 1\}$, defined by $\mathsf{Link}(y, y') = 1 \Leftrightarrow y = y'$. For our purposes, we require $Y = X = \mathcal{M}$ and define $*$ as $\star^{-1}$. Thus, $* : ((\mathbf{A}, \mathbf{B}), \mathcal{C}) \mapsto \mathbf{A}^{-1}\mathcal{C}\mathbf{B}^{-1}$. We will then show that the required properties hold for any given code $\mathcal{C}$, with the IMCE problem as a basis for its security.

Theorem 5.10. The pair of group actions $(\star, *)$ described above is linkable, linkably anonymous and non-frameable assuming the hardness of the IMCE problem.

Proof. We prove the three properties separately.

Linkability. Consider a code $\mathcal{C}$ defined by $\mathbf{G}$, together with two isometries $g = (\mathbf{A}, \mathbf{B})$ and $g' = (\mathbf{A}', \mathbf{B}')$. Suppose that $\mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G})) = \mathsf{SF}(\pi_{\mathbf{A}',\mathbf{B}'}(\mathbf{G}))$ or, equivalently, that $\mathsf{SF}(\mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B})) = \mathsf{SF}(\mathbf{G}((\mathbf{A}')^\top \otimes \mathbf{B}'))$. Then, it must be that $\mathbf{A}^\top \otimes \mathbf{B} = (\mathbf{A}')^\top \otimes \mathbf{B}'$, unless there is an automorphism of $\mathcal{C}$ hidden in the product. As noted in [139], it is plausible to assume that a random code $\mathcal{C}$ admits only the trivial automorphism, for all parameter choices we are interested in. If so, such trivial automorphism are pairs of scalar multiples of the identity for scalars $\alpha \in \mathbb{F}_q$. Hence, either we have $(\mathbf{A}^\top \otimes \mathbf{B})^{-1} = ((\mathbf{A}')^\top \otimes \mathbf{B}')^{-1}$; or, $(\mathbf{A}^\top \otimes \mathbf{B})^{-1} = \alpha^{-1}((\mathbf{A}')^\top \otimes \mathbf{B}')^{-1}$. In both cases, we have that $\mathsf{SF}(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}(\mathbf{G})) = \mathsf{SF}(\pi_{(\mathbf{A}')^{-1},(\mathbf{B}')^{-1}}(\mathbf{G}))$ i.e. $g * y = g' * y$.

Linkable Anonymity. This follows directly from the hardness of the IMCE problem. As discussed in Section 5.4, the problem is believed to be only marginally easier than MCE. In Section 5.7, we analyse dedicated attack techniques for IMCE.

Non-Frameability. For this property, an adversary A is given again a code $\mathcal{C}$ defined by $\mathbf{G}$ and codes x' and y' defined by $\mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}))$ and $\mathsf{SF}(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}(\mathbf{G}))$ respectively, where $g = (\mathbf{A}, \mathbf{B})$ is chosen uniformly at random in $\mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$. The adversary A is asked to find $(\mathbf{A}', \mathbf{B}')$ such that

$$\mathsf{SF}(\pi_{(\mathbf{A}')^{-1},(\mathbf{B}')^{-1}}(\mathbf{G})) = y' = \mathsf{SF}(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}(\mathbf{G})).$$

We can use such an adversary to build a distinguisher D for the IMCE problem, as follows. Suppose $(\mathcal{C}, \mathcal{D}_1, \mathcal{D}_2)$ is the given instance of the problem, and let ϵ be the success probability of A. To begin, D calls A on $(\mathcal{C}, \mathcal{D}_1, \mathcal{D}_2)$, and A will reply with $g' = (\mathbf{A}', \mathbf{B}')$ that satisfies the equation above.

$$\mathsf{SF}(\pi_{(\mathbf{A}')^{-1},(\mathbf{B}')^{-1}}(\mathbf{G})) = y' = \mathsf{SF}(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}(\mathbf{G})).$$

From what we have seen in the proof of the Linkability property above, this implies that $((\mathbf{A}')^\top \otimes \mathbf{B}')^{-1} = (\mathbf{A}^\top \otimes \mathbf{B})^{-1}$, modulo trivial automorphisms. Either way, we have that $\mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G})) = \mathsf{SF}(\pi_{\mathbf{A}',\mathbf{B}'}(\mathbf{G}))$. Thus, it is enough for D to compute $\mathsf{SF}(\pi_{\mathbf{A}',\mathbf{B}'}(\mathbf{G}))$ and $\mathsf{SF}(\pi_{(\mathbf{A}')^{-1},(\mathbf{B}')^{-1}}(\mathbf{G})))$, and check whether they define $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. D then answers 1 if this is the case, or 0 otherwise. Since the probability that a randomly-drawn pair $(\mathcal{D}_1, \mathcal{D}_2)$ satisfies this condition is negligible, the probability of success of D is essentially the same as ϵ.

Together, these three properties complete the proof. □ □

Having clarified the nature of the tools at hand, we are now ready to introduce the linkable version of the ring signature scheme. We first explain compactly how to modify the protocol of Figure 5.3, then give a full description in Figure 5.4.

- As part of the public data, choose a collision-resistant hash function $\overline{\mathsf{H}}$.
- The commit procedure (Algorithm $\mathsf{II.}$) now additionally uses a matrix $\mathbf{T} = \mathsf{SF}(\pi_{\mathbf{A}_{j^*}^{-1},\mathbf{B}_{j^*}^{-1}}(\mathbf{G}_0))$; this is the "tag" for prover j^*, which will be trasmitted alongside the commitment to the verifier. The prover then computes $\tilde{\mathbf{T}} = \mathsf{SF}(\pi_{\tilde{\mathbf{A}}^{-1},\tilde{\mathbf{B}}^{-1}}(\mathbf{T}))$. After obtaining the root of the IHMT, the prover obtains $\bar{h} = \overline{\mathsf{H}}(\tilde{\mathbf{T}}, \mathsf{root})$ and the commitment is set to $\bar{h}$ instead of root.
- Algorithm $\mathsf{IV.}$ is identical, except that, in the case $c = 0$, the response has to include also the matrices $\bar{\mu}, \bar{\nu} = \mathbf{A}_{j^*}\tilde{\mathbf{A}}, \tilde{\mathbf{B}}\mathbf{B}_{j^*}$.
- Finally, Algorithm $\mathsf{V.}$ is modified to involve a check on the tag, as well. The case $c = 1$ is trivial, with the verifier essentially repeating the commitment procedure (Algorithm $\mathsf{II.}$) as in the original protocol, only this time checking equality with $\bar{h}$ rather than root. For the case $c = 0$, instead, the verifier additionally computes $\hat{\mathbf{T}} = \mathsf{SF}(\pi_{\bar{\mu}^{-1},\bar{\nu}^{-1}}(\mathbf{G}_0))$, then uses this value to obtain $\overline{\mathsf{H}}(\hat{\mathbf{T}}, \mathsf{root}')$ and verify equality with $\bar{h}$.

It is relatively easy to see that the new protocols satisfy all the necessary security properties. For instance, the ring Sigma protocol, with the above modifications, still satisfies the Completeness, 2-Special Soundness and Honest-Verifier Zero-Knowledge properties. The linkable ring signature scheme of Figure 5.4 is then just an application of the Fiat-Shamir transform. Once again, such proofs are omitted due to space limitations; the interested reader can consult for example [27], where proofs are given in all generality (see e.g. Theorems 4.3, 4.4 and 4.7). We will present computational costs for the (linkable) ring signature scheme in the next section, after discussing optimizations.

5.6 Matrix Equivalence Digital Signature — MEDS

We apply the following optimizations from the literature to the basic Fiat-Shamir-based signature scheme described in Section 5.5, to obtain our Matrix Equivalence Digital Signature (MEDS).

Multiple keys.

The first optimization is a popular one in literature [16, 28, 56], and it consists of utilizing multiple public keys, i.e. multiple equivalent codes $\mathbf{G}_0, \ldots, \mathbf{G}_{s-1}$, each defined as $\mathbf{G}_i = S^{\mathbf{0}}(\pi_{\mathbf{A}_i,\mathbf{B}_i}(\mathbf{G}_0))$ for uniformly chosen secret keys[3] $(\mathbf{A}_i, \mathbf{B}_i)$. This allows to reduce the soundness error from $1/2$ to $1/2^\ell$, where $\ell = \lceil \log_2 s \rceil$. The optimization works by grouping the challenge bits into strings of ℓ bits, which can then be interpreted as binary representations of the indices $\{0, \ldots, s-$

[3]Again, for convenience, we choose $\mathbf{A}_0 = \mathbf{I}_m$, $\mathbf{B}_0 = \mathbf{I}_n$.

1}, thus dictating which public key will be used in the protocol. Security is preserved since the proof of unforgeability can be easily modified to rely on a multi-instance version of the underlying problem: in our case, MIMCE (Problem 5.4). Note that, although in the literature s is chosen to be a power of 2, this does not have to be the case. In this work, we will instead select the value of s based on the best outcome in terms of performance and signature size.

Remark 5.3. This optimization comes at the cost of an s-fold increase in public-key size. As shown for instance in [56], it would be possible to reduce this impact by using Merkle trees to a hash of the tree commitment of all the public keys. This, however, would add some significant overhead to the signature size, because it would be necessary to include the paths for all openings. Considering the sizes of the objects involved, such an optimization is not advantageous in our case.

Partially seeding the public key.

The previous optimization comes at significant cost to the public key, so we propose a new optimization that trades public key size for private key size. This optimization is inspired by the trade-off in the key generation of Rainbow [64] and UOV [26]. It has not been previously used in Fiat-Shamir signatures, but we expect it can be used successfully in any scheme coming from equivalence problems especially the ones using the previous optimization, such as [16, 28, 56]. With this optimization, instead of generating the secret $(\mathbf{A}_i, \mathbf{B}_i)$ from a secret seed and then deriving the public $\mathbf{G}_i$, we generate $\mathbf{G}_i$ partially from a public seed and then use it to find $(\mathbf{A}_i, \mathbf{B}_i)$ and the rest of the public key $\mathbf{G}_i$. In more detail, in order to generate the public $\mathbf{G}_i$ and the corresponding secret $(\mathbf{A}_i, \mathbf{B}_i)$ we perform the following:

- We perform a secret change of basis of $\mathbf{G}_0$ by multiplying it by a secret matrix $\mathbf{T} \in \mathrm{GL}_k(q)$ to obtain $\mathbf{G}_0'$. Assume the codewords from $\mathbf{G}_0'$ are $\mathbf{P}_1^0, \mathbf{P}_2^0, \ldots, \mathbf{P}_k^0$.
- For each $i \in \{1, \ldots, s-1\}$, we generate from a public seed a complete $m \times n$ codeword $\mathbf{P}_1^i$ and the top $m-1$ rows of codeword $\mathbf{P}_2^i$ (depending on the parameters m, n one can get slightly more rows when $m \neq n$).
- Find $\mathbf{A}_i$ and $\mathbf{B}_i$ from the linear relations:

$$\mathbf{P}_1^i \mathbf{B}_i^{-1} = \mathbf{A}_i \mathbf{P}_1^0$$
$$\mathbf{P}_2^i \mathbf{B}_i^{-1} = \mathbf{A}_i \mathbf{P}_2^0$$

  by fixing the first (top left) value of $\mathbf{A}_i$.
- Find $\mathbf{P}_j^i = \mathbf{A}_i \mathbf{P}_j^0 \mathbf{B}_i$ for all $j \in \{3, \ldots, k\}$.
- Construct the public $\mathbf{G}_i$ from $\mathbf{P}_1^i, \mathbf{P}_2^i, \ldots, \mathbf{P}_k^i$.

The public key then is the public seed together with $\mathbf{P}_3^i, \ldots, \mathbf{P}_k^i$. For verification, the complete $\mathbf{G}_i$ are reconstructed using the seed.

Fixed-weight challenges.

Another common optimization is the use of fixed-weight challenges. The idea is to generate the challenge string h with a fixed number of 1s and 0s, i.e. Hamming weight, rather than uniformly at random. This is because, when $h_i = 0$, the response (μ_i, ν_i) consists entirely of randomly-generated objects, and so one can just transmit the seed used for generating them. This creates a noticeable imbalance between the two types of responses, and hence it makes sense to minimize the number of 1 values. To this end, one can utilize a so-called *weight-restricted hash function*, that outputs values in $\mathbb{Z}_{2,w}^t$, by which we denote the set of vectors with elements in $\{0, 1\}$ of length t and weight w. In this way, although the length of the challenge strings increases, the overall communication cost scales down proportionally to the value of w. In terms of security, this optimization only entails a small modification in the statement of the Forking Lemma, and it is enough to choose parameters such that $\log_2 \binom{t}{w} \geq \lambda$. Note that this optimization can easily be combined with the previous one, by mandating hash digests in $\mathbb{Z}_{s,w}^t$ and choosing parameters such that $\log_2 \left(\binom{t}{w}(s-1)^w\right) \geq \lambda$. In practice, this can be achieved with a hash function $\mathsf{H} : \{0, 1\}^* \to \{0, 1\}^\lambda$, by expanding the output to a t-tuple $(h_0, \ldots, h_{t-1})$, $0 \leq h_i < s$ of weight w.

Seed tree.

Finally, the signature size can be optimized again using a *seed tree*. This primitive allows to generate the many seeds used throughout the protocol in a recursive way, starting from a master seed mseed and building a binary tree, via repeated PRNG applications, having t seeds as leaves. When the required $t - w$ values need to be retrieved, it is then enough to reveal the appropriate sequence of nodes. This reduces the space required for the seeds from $\lambda(t-w)$ to λN_{seeds}, where N_{seeds} can be upper bounded by $2^{\lceil \log_2(w) \rceil} + w(\lceil \log_2(t) \rceil - \lceil \log_2(w) \rceil - 1)$, as shown in [94]. We refer the reader to Section 2.7 of [27] for more details. As suggested in [27], we are including a 256-bit salt to ward off multi-target collision attacks and the leaf address as identifier for domain separation in the inputs of the seed-tree hash functions.

To give a complete picture, we present the MEDS protocol in Figure 5.5, in its final form, including all applicable variants. The various parameters control different optimization: for instance s refers to the number of public keys used, whereas w refers to the fixed weight of the challenge hash string. Parameter choices will be thoroughly discussed in Section 5.8.2. Note that some of these optimizations can be applied in an identical way to the (linkable) ring signature scheme; however, we leave an explicit description of such a scheme, as well as a full-fledged implementation of it, to a dedicated future work.

Public key and signature size.

With these various optimizations, we obtain the following public key and signature size for MEDS:

- MEDS public key size: $\lambda + (s-1)((k-2)(mn-k)+n)\lceil \log_2(q) \rceil$
- MEDS signature size:

$$\underbrace{\lambda}_{h} + \underbrace{w(m^2+n^2)\lceil \log_2(q) \rceil}_{\{\mu_i,\nu_i\}_{h_i=1}} + \underbrace{\lambda N_{\text{seeds}}}_{\{\mu_i,\nu_i\}_{h_i=0}} + \underbrace{2\lambda}_{salt}$$

The optimizations described above can also be applied to the ring signature scheme, which gives us the following sizes:

- Ring signature public key size: $\lambda + rk(mn-k)\lceil \log_2(q) \rceil$
- Ring signature size:

$$\underbrace{w\lceil \log_2 t \rceil}_{h} + \underbrace{w((m^2+n^2)\lceil \log_2(q) \rceil + 2\lambda\lceil \log r \rceil + \lambda)}_{\{\mathsf{resp}_i\}_{h_i=0}} + \underbrace{\lambda N_{\text{seeds}}}_{\{\mathsf{resp}_i\}_{h_i=1}} + \underbrace{2\lambda}_{salt}$$

For the linkable variant, the cost of the middle term, i.e. $\{\mathsf{resp}_i\}_{h_i=0}$, is increased to $w(2(m^2+n^2)\lceil \log_2(q) \rceil + 2\lambda\lceil \log r \rceil + \lambda)$, and the signature additionally includes $k(mn-k)\lceil \log_2(q) \rceil$ bits for the tag $\mathbf{T}$.

5.7 Concrete Security Analysis

In this section, we will mostly use the Big O notation $\mathcal{O}$ to express the complexity of algorithms. Where we are not interested in the polynomial factor we will use $\mathcal{O}^*$. We note that despite the notation, the estimates are quite tight and provide a good basis for choosing parameters.

Recall that the goal of an adversary against MCE is to recover the matrices $\mathbf{A}$ and $\mathbf{B}$, given a description of the matrix codes $\mathcal{C}$ and $\mathcal{D}$. The most naïve attack would be to try every $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$ until we find the correct isometry, amounting to a complexity of $\mathcal{O}(q^{n^2+m^2})$. The naïve attack can be improved by noting that once one of the matrices $\mathbf{A}$ or $\mathbf{B}$ is known, the resulting problem becomes easy [53]. Hence, we only need to brute-force one of $\mathbf{A}$ or $\mathbf{B}$, so the complexity becomes $\mathcal{O}^*(q^{\min\{m^2,n^2\}})$.

In the rest of the section, we will see that there exist several non-trivial attacks that perform much better than this upper bound.

5.7.1 Birthday-based graph-theoretical algorithms for solving MCE

Recent works [53, 139] investigate the hardness of MCE by connecting it to other equivalence problems, namely, the Code Equivalence problem in the Hamming metric [53] and the Quadratic Maps Linear Equivalence problem (QMLE) [139]. The latter provides complexity analysis by viewing MCE as an instance of QMLE. We recap their results here. For better understanding, we include the definition of the related QMLE problem.

Problem 5.5. QMLE $(k, N, \mathcal{F}, \mathcal{P})$:
Given: Two k-tuples of multivariate polynomials of degree 2 variables $x_1, \dots, x_N$

$$\mathcal{F} = (f_1, f_2, \dots, f_k),\ \mathcal{P} = (p_1, p_2, \dots, p_k) \in \mathcal{K}[x_1, \dots, x_N]^k.$$

Goal: Find – if any – matrices $\mathbf{S} \in \mathrm{GL}_N(q), \mathbf{T} \in \mathrm{GL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}) = (\mathcal{F}(\mathbf{x}\mathbf{S}))\mathbf{T}.$$

We denote by hQMLE, inhQMLE and BMLE the related problems when the polynomials are homogeneous of degree 2, inhomogeneous and bilinear, respectively. It was shown in [139] that, under the assumption that the two codes $\mathcal{C}$ and $\mathcal{D}$ have trivial automorphism groups (which is believed to be true with overwhelming probability for big enough parameters), MCE $(k, n, m, \mathcal{C}, \mathcal{D})$ is equivalent to hQMLE $(k, N, \mathcal{F}, \mathcal{P})$ where $N = m + n$. Concretely, an MCE instance with a solution $(\mathbf{A}, \mathbf{B})$ is transformed into an hQMLE instance with a solution $(\mathbf{S}, \mathbf{T})$ where $\mathbf{S} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^\top \end{bmatrix}$ and $\mathbf{T}$ corresponds to a change of basis of $\mathcal{D}$. Therefore it is possible to apply algorithms for solving hQMLE to MCE instances such as the graph-theoretic algorithm of Bouillaguet et al. [38].

The algorithm is basically a collision-search algorithm comprised of two steps, as given in Algorithm 3. In the first step we build two lists L_1 and L_2 of size ℓ of elements in $\mathbb{F}_q^{(m+n)}$ that satisfy a predefined distinguishing property $\mathbb{P}$ related to the given systems of polynomials $\mathcal{F}$ and $\mathcal{P}$ and that is preserved under isometry. In the second step, we try to find a collision between the two lists that will lead us to the solution. For the property $\mathbb{P}$, the authors of [38] propose:

$$\mathbb{P}(\mathcal{F}, \mathbf{x}) = \top \Leftrightarrow Dim(Ker(D_{\mathbf{x}}(\mathcal{F}))) = \kappa$$

for a suitably chosen κ, where $D_{\mathbf{x}}(\mathcal{F}) : \mathbf{y} \mapsto \mathcal{F}(\mathbf{x} + \mathbf{y}) - \mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{y})$ is the *differential* of $\mathcal{F}$ at a point $\mathbf{x}$. Clearly, the rank of the differential is preserved under isometry, so this is an appropriate choice of $\mathbb{P}$. Other instantiations are possible as well, as long as they are invariant under isometry, although their success depends on the distribution of elements that satisfy the property for varying κ.

Once a collision $(\mathbf{a}, \mathbf{b})$ is found, it can be used to derive an associated *inhomogeneous* QMLE instance inhQMLE $(k, (m + n), \mathcal{F}', \mathcal{P}')$ as $\mathcal{F}'(\mathbf{x}) = \mathcal{F}(\mathbf{x} + \mathbf{a})$, $\mathcal{P}'(\mathbf{x}) = \mathcal{P}(\mathbf{x} + \mathbf{b})$ on which we call an inhomogeneous solver. Since it can not be directly checked whether a pair is a collision, the solver needs to be called for each pair, similar to the guess and check approach in ISD algorithms [136].

The inhomogeneous instance can be solved much more efficiently than the homogeneous one. Heuristic evidence suggests that solving *random* instances of the inhQMLE problem using an algebraic approach takes $\mathcal{O}((m + n)^9)$ operations [75], however, the derived inhQMLE instances from the collision-search attack are not random enough. These specific instances have a solver with a complexity of $\mathcal{O}(q^\kappa)$ [37]. As κ is typically chosen to be small, this approach is

Algorithm 3 Collision-search algorithm

```
 1: function BuildList($\mathcal{F}, \mathbb{P}$)
 2:     $L \leftarrow \emptyset$
 3:     repeat
 4:         $\mathbf{x} \xleftarrow{\$} \mathbb{F}_q^{(m+n)}$
 5:         if $\mathbb{P}(\mathcal{F}, \mathbf{x})$ then $L \leftarrow L \cup \{\mathbf{x}\}$
 6:     until $|L| = \ell$
 7:     return $L$
 8: function CollisionFind($\mathcal{F}$,$\mathcal{P}$)
 9:     $L_1 \leftarrow$ BuildList($\mathcal{F}, \mathbb{P}$)
10:     $L_2 \leftarrow$ BuildList($\mathcal{P}, \mathbb{P}$)
11:     for all $(\mathbf{x}, \mathbf{y}) \in \{L_1 \times L_2\}$ do
12:         $\varphi \leftarrow$inhQMLE($\mathbf{x}, \mathbf{y}$)
13:         if $\varphi \neq \perp$ then
14:             return solution $\varphi$
15:     return $\perp$
```

still efficient in practice. Following the analysis from [139], the concrete complexity of the algorithm for $k \leqslant 2(m+n)$ follows a birthday argument and is the maximum of the complexity of the two steps, i.e.:

$$\mathsf{max}(\sqrt{q^{(m+n)}/d} \cdot C_{\mathbb{P}}, dq^{(m+n)} \cdot C_{\mathsf{iQ}}), \tag{5.1}$$

with success probability of $\approx 63\%$. Here, $C_{\mathbb{P}}$ denotes the cost of checking whether an element satisfies the property $\mathbb{P}$, d is the proportion of elements satisfying $\mathbb{P}$ and C_{iQ} denotes the cost of a single query to $\mathsf{inhQMLE}$. Note that d can be calculated as $d = 1/\mathcal{O}(q^{\kappa^2+\kappa(k-(m+n))})$ and κ is chosen such that it minimizes Equation (5.1). Asymptotically, the complexity is $\mathcal{O}^*(q^{\frac{2}{3}(m+n)})$ by balancing the steps [139]. The memory complexity is simply the size of the lists.

It is pointed out in [139] that when $k \geq 2(m+n)$, we can no longer assume that we have distinguished elements, so we need to consider *all* elements in the collision search. Thus, we have $d = 1$ and the complexity of the algorithm is simply $\mathcal{O}(q^{m+n})$. In that case, we can consider choosing arbitrarily one element $\mathbf{x}$ and checking for a collision with all other elements $\mathbf{y} \in \mathbb{F}_q^{m+n}$. This approach yields the same complexity as the previous one, but is superior because it is deterministic and has negligible memory requirements. Note that this approach was also proposed in [38], but as an inferior (in terms of *time* complexity) deterministic variant, rather than as a solution for the lack of a distinguishing property. As this attack can be applied to any parameter set, it presents an upper-bound on the complexity of a classical collision-search algorithm.

For a quantum version of Algorithm 3, both BuildList and CollisionFind can be seen as searches of unstructured databases of a certain size, hence Grover's algorithm applies to both: we can build the list L using only $\sqrt{\ell \cdot d^{-1}}$ searches, and we can find a collision using only $\sqrt{|L_1 \times L_2|}$ queries to the solver. This requires both $\mathbb{P}$ and $\mathsf{inhQMLE}$ to be performed in superposition. The balance between both sides remains the same. In total, the complexity of the quantum version becomes $\mathcal{O}^*(q^{\frac{1}{3}(m+n)})$.

Collision-search algorithm using non-trivial roots.

When viewing an MCE instance as an hQMLE instance, it is possible to use certain bilinear properties to improve Algorithm 3. When $n = m$, such instances have approximately q^{2n-k-1} non-trivial roots, which can be used to improve a subroutine of Algorithm 3, and to make it deterministic instead of probabilistic [139]. In practice, such non-trivial roots exist **i)** almost always when $k < 2n$, **ii)** with probability $1/q$ for $k = 2n$, **iii)** with probability $1/q^{k+1-2n}$ for $k > 2n$. The complexity of this approach is $\mathcal{O}^*(q^n)$, if such non-trivial roots exist. This complexity is proven under the assumption that the complexity of the inhomogenous QMLE solver is no greater than $\mathcal{O}(q^n)$, which holds trivially when $k \geq n$ [139], and heuristically when $k < n$. Finding the non-trivial roots can also be done using a bilinear XL algorithm [133]. We do not consider this approach in our analysis, as it is only interesting for a subset of parameters where the systems are (over)determined, i.e. when k is close to $m + n$.

5.7.2 Algebraic attacks

Direct modelling.

Recently, in [139], it was shown that MCE is equivalent to BMLE. which means that any solver for BMLE can be easily adapted to solve MCE. One of the natural attack avenues is thus to model the problem as an algebraic system of polynomial equations over a finite field. This approach was taken in [75], where the general Isomorphism of Polynomials (IP) problem was investigated. Here, we focus specifically on BMLE and perform a detailed complexity analysis.

First, fix arbitrary bases $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ of the codes $\mathcal{C}$ and $\mathcal{D}$ respectively. In terms of the bases, the MCE problem can be rephrased as finding $\mathbf{A} \in \mathrm{GL}_m(q)\mathbf{B} \in \mathrm{GL}_n(q)$ and $\mathbf{T} = (t_{ij}) \in \mathrm{GL}_k(q)$ such that:

$$\sum_{1 \leqslant s \leqslant k} t_{rs}\mathbf{D}^{(s)} = \mathbf{A}\mathbf{C}^{(r)}\mathbf{B}, \quad \forall r, 1 \leqslant r \leqslant k \tag{5.2}$$

The system (5.2) consists of knm equations in the $m^2 + n^2 + k^2$ unknown coefficients of the matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{T}$. The quadratic terms of the equations are always of the form $\gamma a_{ij} b_{i'j'}$ for some coefficients a_{ij} and $b_{i'j'}$ of $\mathbf{A}$ and $\mathbf{B}$ respectively which means the system (5.2) is bilinear. Note that the coefficients of $\mathbf{T}$ appear only linearly. As previously, we can guess the m^2 variables from $\mathbf{A}$, which will lead us to a linear system that can be easily solved. However, we can do better by exploiting the structure of the equations.

For ease of readability of the rest of the paragraph denote by $\mathbf{M}_{i\_}$ and $\mathbf{M}_{\_i}$ the i-th row and i-th column of a matrix $\mathbf{M}$. Note that, in (5.2), for $i \neq j$, the unknown coefficients from two rows $\mathbf{A}_{i\_}$ and $\mathbf{A}_{j\_}$ don't appear in the same equation. Symmetrically, the same holds for $\mathbf{B}_{\_i}$ and $\mathbf{B}_{\_j}$, but we will make use of it for the matrix $\mathbf{A}$. Thus, we can consider only part of the system, and control the number of variables from $\mathbf{A}$. The goal is to reduce the number of variables that we need to guess before obtaining an overdetermined linear system, and we want to do this in an optimal way. Consider the first α rows from $\mathbf{A}$. Extracting the equations that correspond to these rows in (5.2) leads us to the system:

$$\sum_{1 \leqslant s \leqslant k} t_{rs} \mathbf{D}^{(s)}_{i\_} = \mathbf{A}_{i\_} \mathbf{C}^{(r)} \mathbf{B}, \;\; \forall r, i, \; 1 \leqslant r \leqslant k, \; 1 \leqslant i \leqslant \alpha. \tag{5.3}$$

Guessing the αm coefficients from $\mathbf{A}_{i\_}$ $1 \leqslant i \leqslant \alpha$ leads to a linear system of $\alpha k n$ equations in $n^2 + k^2$ variables. Choosing $\alpha = \lceil \frac{n^2+k^2}{kn} \rceil$, the complexity of the approach becomes $\mathcal{O}(q^{m \lceil \frac{n^2+k^2}{kn} \rceil} (n^2 + k^2)^3)$. For the usual choice of $m = n = k$, this reduces to at least $\alpha = 2$ and a complexity of $\mathcal{O}(q^{2n} n^6)$.

Note that, one can directly solve the bilinear system (5.3) using for example XL [50] and the analysis for bilinear systems from [133] (similar results can be obtained from [70]). We have verified, however, that due to the large number of variables compared to the available equations, the complexity greatly surpasses the one of the simple linearization attack presented above.

Improved modelling.

In order to improve upon this baseline algebraic attack, we will model the problem differently and completely avoid the t_{rs} variables. This modelling is in the spirit of the minors modellings of MinRank as in [13, 71].

As previously, let $\mathbf{G}$ and $\mathbf{G}'$ be the $k \times mn$ generator matrices of the equivalent codes $\mathcal{C}$ and $\mathcal{D}$ respectively. Then from Lemma 5.7, $\tilde{\mathbf{G}} = \mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B})$ is a generator matrix of $\mathcal{D}$ for some invertible matrices $\mathbf{A}$ and $\mathbf{B}$. We will take the coefficients of $\mathbf{A}$ and $\mathbf{B}$ to be our unknowns. A crucial observation for this attack is that each row $\tilde{\mathbf{G}}_{i\_}$ of $\tilde{\mathbf{G}}$ is in the span of the rows of $\mathbf{G}'$, since $\mathbf{G}'$ and $\tilde{\mathbf{G}}$ define the same code. This means that adding $\tilde{\mathbf{G}}_{i\_}$ to $\mathbf{G}'$ does not change the code, i.e.,

$$^{(i)}\mathbf{G}' = \begin{pmatrix} \mathbf{G}' \\ \tilde{\mathbf{G}}_{i\_} \end{pmatrix}$$

is not of full rank. From here, all maximal minors $\left| \begin{pmatrix} ^{(i)}\mathbf{G}'_{\_j_1} & ^{(i)}\mathbf{G}'_{\_j_2} & \dots & ^{(i)}\mathbf{G}'_{\_j_{k+1}} \end{pmatrix} \right|$ of $^{(i)}\mathbf{G}'$, for every $\{j_1, j_2, \dots, j_{k+1}\} \subset \{1, 2, \dots, mn\}$, are zero.

Now, as in a minors modeling of MinRank, we can form equations in the unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$ by equating all maximal minors to zero, which amounts to a total of $\binom{mn}{k+1}$ equations. Since the unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$ appear only in the last row of the minors, and only bilinearly, the whole system is also bilinear. Thus we have reduced the problem to solving the bilinear system

$$\left\{ \left| \begin{pmatrix} ^{(i)}\mathbf{G}'_{\_j_1} & ^{(i)}\mathbf{G}'_{\_j_2} & \dots ^{(i)}\mathbf{G}'_{\_j_{k+1}} \end{pmatrix} \right| = 0, \quad \begin{array}{l} \text{for all } i \in \{1, 2, \dots, k\} \text{ and all} \\ \{j_1, j_2, \dots, j_{k+1}\} \subset \{1, 2, \dots, mn\} \end{array} \right. \tag{5.4}$$

in the $m^2 + n^2$ unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$.

At first sight, (5.4) seems to have more than enough equations to fully linearize the system. However, the majority of these equations are linearly dependent. In fact, there are only $(mn - k)k$ linearly independent equations. To see this, fix some i and consider a minor $|\left({}^{(i)}\mathbf{G}'_{\text{-}j_1}\ {}^{(i)}\mathbf{G}'_{\text{-}j_2} \ldots {}^{(i)}\mathbf{G}'_{\text{-}j_{k+1}}\right)|$ of ${}^{(i)}\mathbf{G}'$. Since all rows except the first don't contain any variables, the equation

$$|\left({}^{(i)}\mathbf{G}'_{\text{-}j_1}\ {}^{(i)}\mathbf{G}'_{\text{-}j_2} \ldots {}^{(i)}\mathbf{G}'_{\text{-}j_{k+1}}\right)| = 0$$

basically defines the linear dependence between the columns ${}^{(i)}\mathbf{G}'_{\text{-}j_1}, \ldots {}^{(i)}\mathbf{G}'_{\text{-}j_{k+1}}$. But the rank of the matrix is k, so all columns can be expressed through some set of k independent columns. Thus, in total, for a fixed i we have $mn-k$ independent equations and in total $(mn-k)k$ equations for all i.

Alternatively, we can obtain the same amount of equations from $\tilde{\mathbf{G}}$ and the generator matrix $\mathbf{G}'^{\perp}$ of the dual code of $\mathcal{D}$. Since $\tilde{\mathbf{G}}$ should also be a generator matrix of $\mathcal{D}$, we construct the system:

$$\mathbf{G}'^{\perp} \cdot \tilde{\mathbf{G}}^{\top} = \mathbf{0},$$

which is again a system of $(mn-k)k$ bilinear equations in n^2+m^2 variables.

The complexity of solving the obtained system using either of the modellings strongly depends on the dimension of the code – it is the smallest for $k = mn/2$, and grows as k reduces (dually, as k grows towards mn). In Section 5.8 we give the concrete complexity estimate for solving the system for the chosen parameters using bilinear XL and the analysis from [133].

The attack does not seem to benefit a lot from being run on a quantum computer. Since the costly part comes from solving a huge linear system for which there are no useful quantum algorithms available, the only way is to 'Groverize' an enumeration part of the algorithm. One could enumerate over one set of the variables, either of $\mathbf{A}$ or $\mathbf{B}$, typically the smaller one, and solve a biliner system of less variables. Grover's algorithm could then speed up quadratically this enumeration. However, since in the classical case the best approach is to not use enumeration, this approach only makes sense for quite small values of the field size i.e. only when $q < 4$. In this parameter regime, however, combinatorial attacks perform significantly better, so this approach becomes irrelevant.

5.7.3 Leon-like algorithm adapted to the rank metric

Leon [109] proposed an algorithm against the code equivalence problem in the Hamming metric that relies on the basic property that isometries preserve the weight of the codewords and that the weight distribution of two equivalent codes is the same. Thus, finding the set of codewords of smallest weight in both codes reveals enough information to find a permutation that maps one set to the other, which with high probability is the unknown isometry between the codes. This algorithm is quite unbalanced and heavy on the 'codewords finding' side, since it requires finding all codewords of minimal weight. Beullens [24] proposed to relax the procedure and instead perform a collision based algorithm, much in the spirit of Algorithm 3: Build two lists of elements of the codes of particular weight (the distinguishing property from [24] actually also includes the multiset of entries of a codeword) and find a collision between them. As in Leon's algorithm and Algorithm 3, the 'collision finding' part employs an efficient subroutine for reconstructing the isometry.

The approach from the Hamming metric can be translated to matrix codes and can be used to solve MCE, but some adjustments are necessary. First of all note that finding codewords of a given rank r is equivalent to an instance of MinRank [51, 71] for k matrices of size $m \times n$ over $\mathbb{F}_q$. Depending on the parameters, we have noticed that the Kipnis-Shamir modelling [101] and Bardet's modelling [13] perform the best, so we use both in our complexity estimates.

For the collision part, notice that given two codewords $\mathbf{C}_1$ from $\mathcal{C}$ and $\mathbf{D}_1$ from $\mathcal{D}$, it is not possible to determine the isometry $(\mathbf{A}, \mathbf{B})$, as there are many isometries possible between single codewords. Thus, there is no efficient way of checking that these codewords collide nor finding the correct isometry. On the other hand, a pair of codewords is typically enough. For the pairs $(\mathbf{C}_1, \mathbf{C}_2)$ and $(\mathbf{D}_1, \mathbf{D}_2)$ we can form the system of $2mn$ linear equations

$$\begin{cases} \mathbf{A}^{-1}\mathbf{D}_1 = \mathbf{C}_1\mathbf{B} \\ \mathbf{A}^{-1}\mathbf{D}_2 = \mathbf{C}_2\mathbf{B} \end{cases} \tag{5.5}$$

in the $m^2 + n^2$ unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$. When $m = n$, which is a typical choice, the system is expected to be overdetermined, and thus solved in $\mathcal{O}(n^6)$. In practice, and since $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{D}_1$ and $\mathbf{D}_2$ are low-rank codewords, there are fewer than $2n^2$ linearly independent equations, so instead of a unique solution, we can obtain a basis of the solution space. However, the dimension of the solution space is small enough so that coupling this technique with one of the algebraic modelings in Section 5.7.2 results in a system that can be solved through direct linearization. It is then easy to check whether the obtained isometry maps $\mathcal{C}$ to $\mathcal{D}$. We will thus assume, as a lower bound, that we find collisions between pairs of codewords.

Now, let $C(r)$ denote the number of codewords of rank r in a k-dimensional $m \times n$ matrix code. Then, using a birthday argument, two lists of size $\sqrt{2C(r)}$ of rank r codewords of $\mathcal{C}$ and $\mathcal{D}$ are enough to find two collisions. To detect the two collisions, we need to generate and solve systems as in Equation (5.5) for all possible pairs of elements from the respective lists, so $\binom{\sqrt{2C(r)}}{2}^2$ systems in total. Since $C(r) \approx q^{r(n+m-r)-nm+k}$, the total complexity amounts to

$$\mathcal{O}(q^{2(r(n+m-r)-nm+k)}(m^2+n^2)^{\omega}).$$

Note that a deterministic variant of this approach has the same asymptotic complexity. Choosing two rank r codewords of $\mathcal{C}$ and checking them for a 2-collision against all pairs of rank r codewords of $\mathcal{D}$ requires solving $\binom{C(r)}{2}$ systems.

Finally, we choose r so that both parts – the MinRank and the collision part are as close to a balance as possible. Section 5.8 discusses further the complexity of this approach for the chosen parameters of our scheme.

When considering the quantum version of the algorithm, we apply the same reasoning as in the case of the collision based Algorithm 3, and obtain quadratic speedup in the collision part. Because hybridization is also possible for the MinRank part, it can also benefit from using Grover, especially for larger fields.

5.7.4 Attacks on IMCE

The security of the linkable version of the ring signature scheme relies on the hardness of the IMCE problem. Thus, given three codes $\mathcal{C}$, $\mathcal{D}_1$ and $\mathcal{D}_2$, we need to determine whether they form a triangle of the following form, where the full lines represent the exact mappings the problem asks for, whereas the dashed ones are mappings that if found also break the problem. (The diagram uses loose notation in favor of readability.)

$$\begin{tikzcd}
 & \mathcal{D}_1 \\
\mathcal{C} \arrow[ur, "\pi_{\mathbf{A},\mathbf{B}}"] \arrow[dr, "\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}"] & \\
 & \mathcal{D}_2 \arrow[uu, dashed, "\pi_{\mathbf{A}^2,\mathbf{B}^2}"'] \arrow[ul, dashed, bend left, "\pi_{\mathbf{A},\mathbf{B}}"]
\end{tikzcd}$$

Extended collision-search algorithm.

Our extended collision-search attack consists of finding an isometry between *any* of the three codes. Specifically, finding a collision between $\mathcal{C}$ and $\mathcal{D}_1$ allows us to derive $\pi_{\mathbf{A},\mathbf{B}}$, one between $\mathcal{C}$ and $\mathcal{D}_2$ yields $\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}$, and one between $\mathcal{D}_1$ and $\mathcal{D}_2$ yields $\pi_{\mathbf{A}^2,\mathbf{B}^2}$. We first reduce the problem to an equivalent QMLE-based problem: solve any of the three QMLE instances $(k, m+n, \mathcal{F}, \mathcal{P}_1)$, $(k, m+n, \mathcal{F}, \mathcal{P}_2)$ or $(k, m+n, \mathcal{P}_2, \mathcal{P}_1)$, with

$$\mathcal{P}_1(\mathbf{x}) = (\mathcal{F}(\mathbf{x}\mathbf{S}))\mathbf{T_1},\ \mathcal{P}_2(\mathbf{x}) = (\mathcal{F}(\mathbf{x}\mathbf{S}^{-1}))\mathbf{T_2},\ \mathcal{P}_1(\mathbf{x}) = (\mathcal{P}_2(\mathbf{x}\mathbf{S}^2))\mathbf{T_2}^{-1}\mathbf{T_1}$$

and $\mathbf{S} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^\top \end{bmatrix}$. Then, we apply a modified version of Algorithm 3, where we now build three lists instead of two, and on Line 11, we pick pairs $(\mathbf{x}, \mathbf{y})$ from $\{L_1 \times L_2\} \cup \{L_1 \times L_3\} \cup \{L_2 \times L_3\}$, instead of just $\{L_1 \times L_2\}$.

To derive the complexity of the attack, denote by $N = dq^{m+n}$ the total number of distinguished elements. Then it can be deduced, using a birthday based analysis that the probability of not having a collision when the lists contain ℓ elements each is $P = \prod_{i=1}^{\ell}(1 - \frac{i}{N})(1 - \frac{2i}{N}) \approx e^{-\frac{3\ell^2}{2N}}$. To have a 63% chance of collision, we need to build lists of size $\ell = c_2\sqrt{N}$, with $c_2 = \sqrt{\frac{2}{3}}$. For comparison, performing a similar analysis for the collision-search algorithm for the general MCE problem (where we have only two lists), this constant is $c_1 = \sqrt{2}$. constant is, however, neglected in the complexity analysis because the choice of having 63% chance of success is arbitrary, and an attack with a lower success probability still presents a security threat.

Remark 5.4. For Leon's algorithm for IMCE, the arguments are similar to the extended collision search algorithm. Thus it seems that apart from possibly a constant speed-up, the algorithm does not benefit from the IMCE context.

Extended algebraic attack.

Algebraically, IMCE can be treated in exactly the same manner as the minors modeling of MCE in the previous section. The types of equations are the same, and with the same structure. The only difference is that the algebraic system that we construct has twice as many equations, i.e $2(mn-k)k$, coming from the isometry between $\mathcal{C}$ and D_1 and between $\mathcal{D}_2$ and $\mathcal{C}$ but the amount of variables is the same – $n^2 + m^2$, since the isometry is the same. This means that the cost of the attack is reduced for the same choice of parameters, so adding the linkability property requires larger parameters in the regime where the algebraic attack performs the best.

5.8 Implementation and Evaluation

In this section, we give an assessment of the performance of MEDS. We begin with important considerations about the implementation of the signature scheme. Then we provide concrete parameter choices for MEDS and a first preliminary evaluation of its performance based on a C reference implementation. The source code of our implementation is available at `https://github.com/MEDSpqc/meds`.

5.8.1 Implementation

Besides performance, the security of a cryptographic implementation is of crucial importance. By "implementation security" here we mean the resilience of an implementation against threats such as timing attacks, physical side-channel attacks, and fault injection attacks. While the requirement for side-channel and fault attacks heavily depends on whether in practice physical access is possible for an attacker, it is widely considered best practice to provide protection against timing attacks as baseline for all cryptographic implementations. In order to do this, the implementation must be *constant time*, i.e., timing variations based on secret data must be prevented. This typically means that branching and memory access based on secret data must be avoided. There are generic constructions to achieve timing security for basically any given cryptographic scheme. However, if the design of a cryptographic primitive does not take constant-time requirements into consideration, such generic constructions can be computationally expensive. Therefore, defining a cryptographic scheme such that it supports an efficient protection against timing attacks can make it easier to implement the scheme securely which in turn can make a scheme more secure and efficient in practice.

In the case of MEDS, we need to consider the implementation security of key generation and signing. Verification only operates on public data and hence does not require further protection. The basic operations of MEDS during key generation and signing are:

- field arithmetic,
- matrix multiplication,
- generating random invertible matrices, and
- computing a canonical form of a matrix.

All these operations must be implemented securely.

Field arithmetic.

We are using a relatively small prime field in MEDS. Hence, for finite field addition and multiplication, we can simply perform integer arithmetic followed by a reduction modulo the prime. On most architectures, constant-time operations for adding and multiplying small operands are available. The reduction modulo the prime can be implemented using standard approaches in literature (e.g., by Granlund and Montgomery [92], Barrett [17], or Montgomery [119]). Inversion of field elements can efficiently be implemented in constant time using Fermat's little theorem and optimal addition chains.

In the C reference implementation, we simply use the modulo operation for reduction and Fermat's little theorem for inversion to get a first impression on the performance of the scheme. For using MEDS in practice, the timing side-channel security in particular of the modulo operation needs to be verified.

Matrix multiplication.

The basic schoolbook algorithm for multiplying matrices is not data dependent and hence constant time. More sophisticated approaches like the method by Arlazarov, Dinic, Kronrod, and Faradžev [10] may depend on potentially secret data, but most likely do not significantly improve the performance for the finite field and the matrix sizes used in MEDS. Asymptotically more efficient matrix multiplication algorithms likely are not more efficient for the given matrix dimensions neither. Hence, for the C reference implementation, we are simply using schoolbook multiplication. If a more efficient algorithm is used for optimized implementations, it must be ensured that a constant time algorithm is used.

Random invertible matrix generation.

On several occasions throughout the MEDS operations, we need to generate random invertible matrices: For the partially seeded key generation (cf. Section 5.6) we need to compute an invertible matrix $\mathbf{T} \in \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$ and for signing (cf. step i.. in Figure 5.5) we need to generate potentially secret matrices $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \in \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$. (During verification we need to recompute $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ for cases where $h_i = 0$.)

There are several approaches for generating random invertible matrices:

1. *Trial-and-error approach:* Generate a random matrix $\mathbf{M}$ and attempt to compute its inverse. If $\mathbf{M}$ does not have an inverse, try again with a new random matrix. This approach requires constant-time Gaussian elimination if $\mathbf{M}$ needs to be kept secret and it might require several attempts before a random invertible matrix has been found.

2. *Constructive approach:* Construct a matrix $\mathbf{M}$ that is known to be inveritble. One approach for this is described in [137]: Generate a random lower-left triangular matrix $\mathbf{L}$ with the diagonal all 1 and an upper-right triangular matrix $\mathbf{U}$ with the diagonal all $\neq 0$ as well as a corresponding permutation matrix $\mathbf{P}$ akin to the result of an LUP decomposition. Then compute the random invertible matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{P}^{-1}\mathbf{L}\mathbf{U}$.

   Since generation of and multiplication with $\mathbf{P}$ are expensive to implement in constant time, one can follow the approach of [153], leave out $\mathbf{P}$, and compute $\mathbf{M}$ as $\mathbf{M} = \mathbf{L}\mathbf{U}$ directly. This, however, covers only a subset of $((q-1)/q)^n$ matrices of all invertibe matrices in $\mathbb{F}_q^{n\times n}$. The inverse $\mathbf{M}^{-1}$ of the matrix can then be computed as $\mathbf{M}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$.

The fastest approach in our experiments was the constructive approach following [153]. Hence, we are using the constructive approach in all cases.

Canonical matrix form.

MEDS requires to compute a canonical form of matrices before hashing. However, during signing, this must be computed in constant time. Computing the reduced row-echelon form of a matrix in constant time is expensive. Canonical matrix forms that can be computed in constant time more efficiently include the *systematic form* and the *semi-systematic* form of a matrix, used, e.g., in Classic McEliece [5]. Both the systematic and the semi-systematic form are special cases of the reduced row-echelon form.

For the systematic form, all pivoting elements are required to reside on the left diagonal of the matrix, i.e., a matrix $\mathbf{G} \in \mathbb{F}_q^{k\times mn}$ in systematic form has the shape $(\mathbf{I}_k|\mathbf{G}')$ with $\mathbf{G}' \in \mathbb{F}_q^{k\times(mn-k)}$ and $\mathbf{I}_k$ denoting the $k \times k$ identity matrix. The requirements for the semi-systematic form are more relaxed: Following [5, Sect. 2.2.1], we say that a matrix $\mathbf{G}$ is in (μ, ν)-semi-systematic form if $\mathbf{G}$ has r rows (i.e., no zero rows), the pivot element in row $i \leq r - \mu$ also is in column i and the pivot element in row $i > r - \mu$ is in a column $c \leq i - \mu + \nu$.

However, not all matrices admit a systematic or semi-systematic form. In this case, we need to restart the computation with new random data. The probability that a matrix $\mathbf{G} \in \mathbb{F}_q^{k\times mn}$, $k \leq mn$ is full rank is $\prod_{i=1}^{k}(q^{mn} - q^{i-1})/q^{kmn}$. Therefore, the probability that $\mathbf{G}$ has a systematic form is $\prod_{i=1}^{k}(q^k - q^{i-1})/q^{k^2}$ and the probability that it has a semi-systematic form is $\prod_{i=1}^{k-\mu}(q^k - q^{i-1})/q^{(k-\mu)k} \cdot \prod_{i=1}^{\mu}(q^\nu - q^{i-1})/q^{\mu\nu}$. The probability and the cost of constant-time implementation for the semi-systematic form depend on μ and ν.

This gives us the following three options for avoiding to compute a reduced row-echelon form in constant time for $\tilde{\mathbf{G}}_i$ during signing:

1. *Basis change:* After computing $\tilde{\mathbf{G}}'_i = \pi_{\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i}(\mathbf{G}_0)$, perform a basis change $\tilde{\mathbf{G}}''_i = \mathbf{M}_i \tilde{\mathbf{G}}'_i$ with a secret random invertible matrix $\mathbf{M}_i \in \mathbb{F}_q^{k \times k}$. Then compute a canonical form $\tilde{\mathbf{G}}_i = S^{\mathbf{0}}(\tilde{\mathbf{G}}''_i)$ on a public $\tilde{\mathbf{G}}''_i$ without the need for a constant time implementation. This removes the requirement of a constant time computation of the canonical form but introduces extra cost for the generation of and multiplication with a random invertible matrix $\mathbf{M}_i$. Instead of an invertible matrix $\mathbf{M}_i$, just a random matrix can be used. With low probability (see above), a random matrix does not have full rank and the computation of the canonical form fails. In that case, the process needs to be restarted with a different $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$.

2. *Semi-systematic form:* Compute the semi-systematic form. This requires a less expensive constant-time implementation than for the reduced-row-echelon form. However, computing the canonical form might fail if no semi-systematic form exists, in which case the computation needs to be restarted.

3. *Systematic form:* Compute the systematic form. This can be implemented even more easily and cheaper in constant time than computing the semi-systematic form. However, systemization failure is more frequent and it is more likely that computations need to be restarted.

We performed generic experiments to investigate the performance impact of these variants. Our experiments indicate that the implementation cost for variant 1 is higher than that of the other two. For the specific parameter sets we propose in Section 5.8.2, the probability that $\tilde{\mathbf{G}}_i$ does not have a systematic form is about 0.0015%. Therefore, even though the failure probability can be reduced by computing the semi-systematic form compared to the systematic form with well-chosen μ and ν, the overall overhead (including cost for constant-time implementation) of computing the semi-systematic form (variant 2) is likely higher than the overall overhead for of computing the systematic form (variant 3). Hence, we decided to use the systematic form throughout as canonical form.

5.8.2 Parameter choice and evaluation

A summary of the cost of the three different attacks described in Section 5.7 is given in Table 5.1. First, we decide to set $n = k$, as this seems to be the Goldilocks zone for our scheme. For k larger, the algebraic attack becomes significantly better, and the same is true for Leon's attack when k is smaller. Then, for finite fields of different sizes, we find the smallest value of n that achieves the required security level of 128 bits. We see that Leon's algorithm performs the best in most cases, although the algebraic approach is almost as good. Finally, to determine the optimal value for q, we choose the optimization

$\lceil \log_2 q \rceil$	$n = k$	Birthday	Algebraic	Leon	SIG
9	16	235.29	181.55	131.20	13 296
9	17	249.04	194.55	149.65	16 237
10	15	244.62	174.75	130.50	12 428
11	14	250.79	160.24	131.21	12 519
12	14	272.40	160.24	141.17	13 548
13	**13**	**274.10**	**146.76**	**130.41**	**11 586**
14	13	294.10	146.76	134.41	13 632
20	12	383.75	138.46	135.40	16 320

Table 5.1: Cost of the investigated attacks in log scale, and ‘SIG’ for ‘signature size in bytes. Preferred choice in bold.

parameters (s, t, and w) such that the sizes of the public key and the signature are comparable, and we report the signature size in the last column of Table 5.1. We conclude that the sweet spot for 128-bit security is given for the 13-bit prime $q = 8191$ and $n = k = 13$.

Remark 5.5. Given these parameters, we heuristically assume that the automorphism group of the codes is trivial with overwhelming probability. It is computationally infeasible to compute the automorphism group of codes of this size; however, data on smaller-sized codes shows that the probability of a random code having a trivial automorphism group grows rapidly as q, n, and m increase.

In this setting, we can vary s, t, and w for different trade-offs of public key and signature sizes as well as performance. We also checked the impact of q if we aim for small public keys or small signatures (instead if balancing these two as in Table 5.1). In such cases, both 11-bit and 13-bit primes for q seem to perform similarly well. Hence, we stick to the 13-bit prime $q = 8191$ in our discussion.

Table 5.2 provides an overview of 128-bit security parameters for MEDS, highlighting different performance and key/signature size trade-offs. The best attack for all parameter set based on $q = 8191$, $n = 13$, and $k = 13$ is the Leon-like attack as shown in Table 5.1 with an expected cost of slightly over 2^{130} operations. The best quantum attack is obtained by Groverizing Leon's algorithm and has a cost of around 2^{88} operations. We select s, t, and w such that the probability of an attack on the Fiat-Shamir construction is around 2^{-128}. To improve the efficiency of vectorized implementations using SIMD instructions in the future, we select t as multiple of 16. In general, we are using all optimizations discussed in Section 5.6. However, we provide one parameter set without using the seed tree (without ‘-st’ in the name of the parameter set).

Table 5.3 shows the resulting performance of these parameter sets from our constant-time C reference implementation on an AMD Ryzen 7 PRO 5850U

[4]`https://bench.cr.yp.to/supercop.html`

CPU. The C reference implementation follows the implementation discussion above but does not apply any further algorithmic or platform-specific optimizations. We expect that optimized and vectorized implementations can significantly increase the performance.

TODO: FIX ISSUE WITH SI UNITS

Finally, sets MEDS-42161-st, MEDS-356839-st, and MEDS-716471-st push the public key size to an extreme at the expense of key generation time in the pursue of reducing signature size and computational cost for signing and verification. However, we expect that at least the key generation time can significantly be improved by optimizing the computation of solving the medium-size sparse linear system used for partially seeding the public key.

Overall, Table 5.2 and Table 5.3 highlight the large degree of flexibility offered by the MEDS scheme. All parameter sets are competitive with respect to existing literature, such as the LESS-FM scheme.

Finally, we discuss performance for the ring signature scheme. With the MEDS-2696-st parameter set, the size of a signature is in fact given by approximately $(\log_2 r + 19.26)$ kB; in the linkable case, this increases to $(\log_2 r + 39.22)$ kB. These numbers are close to the lattice instantiation (Falafl) given in [27]; judging by the timings in Table 5.3, we also expect it to be much faster than the isogeny instantiation (Calamari). On the other hand, the work of [15] obtains smaller sizes, but does not propose a linkable variant. Note that both sizes and timings can be improved by choosing dedicated parameters and designing an ad-hoc implementation, which we leave as part of a future work.

5.8.3 Comparison to related signature schemes

Table 5.4 shows a comparison of public key and signature sizes as well as computational performance of our new MEDS scheme with some established schemes and related recent proposals. While the comparison of public key and signature sizes is accurate, the comparison of the performance needs to be taken with a large grain of salt: While we provide numbers in the same performance metric (mega cycles – mcyc.), a direct comparison is still quite hard since not all schemes have received the same degree of optimization and since not all performance data has been obtained on the same CPU architecture.

The performance data from the 'classical' scheme ed25519 as well as from the NIST PQC schemes CRYSTALS-Dilithium [67], Falcon [82], and SPHNICS+[97] has been obtained from the SUPERCOP website[5]. We selected the performance data from the AMD64 Zen CPU, which is an AMD Ryzen 7 1700 from 2017, i.e., the same microarchitecture (but a different CPU) as we used for our measurements of MEDS. We are reporting median cycles directly from the website.

For UOV [26], LESS [16] and Wavelet [11] we list the performance data as reported in the respective papers unless such data was unavailable. In the case of SDitH [2], only reports of performance data in milliseconds on a 3.1 GHz

[5]`https://bench.cr.yp.to/results-sign.html` – amd64; Zen (800f11); 2017 AMD Ryzen 7 1700; 8 x 3000MHz; rumba7, supercop-20220506

Intel Core i9-9990K are available. We computed the corresponding number of cycles from this to enable a rough comparison to the other schemes, but note that this data is therefore not entirely accurate.

Table 5.4 shows that, although code-based schemes do not compete well with pre-quantum or lattice-based PQC schemes, MEDS fills a gap that was not previously available for multivariate or code-based schemes, with a relatively small combined size of public key and signature. Furthermore, its versatility in parameter selection allows for great flexibility for specific applications. In terms of performance, the current implementation of MEDS is still unoptimized. We expect speed-ups of at least one order of magnitude from SIMD parallelization on AVX256 and AVX512 CPUs, since both the data-independent loop of the Fiat-Shamir construction and the matrix arithmetic lend themselves to efficient parallelization. Providing optimized implementations of MEDS for modern SIMD architectures as well as embedded systems is an open task for future work.

```
Public Data
$q, m, n, k, \lambda, r \in \mathbb{N}$.
$\mathsf{Com} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{2\lambda}$.

I. Keygen$(j)$
  1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k\times mn}$ in standard form.
  2. $(\mathbf{A}_j, \mathbf{B}_j) \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
  3. Compute $\mathbf{G}_j = \mathsf{SF}(\pi_{\mathbf{A}_j,\mathbf{B}_j}(\mathbf{G}_0))$.
  4. Set $\mathsf{sk}_j = (\mathbf{A}_j, \mathbf{B}_j)$ and $\mathsf{pk}_j = \mathbf{G}_j$.
  5. Set $\mathsf{pk} = (\mathbf{G}_0, \mathsf{pk}_1, \dots, \mathsf{pk}_r)$

II. Commit$(\mathsf{pk})$
  1. $\tilde{\mathbf{A}}, \tilde{\mathbf{B}} \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
  2. For all $j = 1, \dots, r$:
     i. Sample $\mathsf{r}_j \xleftarrow{\$} \{0,1\}^\lambda$.
     ii. Compute $\tilde{\mathbf{G}}_j = \mathsf{SF}(\pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}(\mathbf{G}_j))$.
     iii. Compute $h_j = \mathsf{Com}(\mathsf{r}_j, \tilde{\mathbf{G}}_j)$.
  3. Build IHMT from $(h_1, \dots, h_r)$ and get $\mathsf{root}$.
  4. Set $\mathsf{cmt} = \mathsf{root}$.
  5. Send $\mathsf{cmt}$ to verifier.

III. Challenge()
  1. $c \xleftarrow{\$} \{0,1\}$.
  2. Set $\mathsf{chal} = c$.
  3. Send $\mathsf{chal}$ to prover.

IV. Response$(\mathsf{sk}_{j^*}, \mathsf{pk}, \mathsf{cmt}, \mathsf{chal})$
  1. If $\mathsf{chal} = 0$:
     i. Set $(\mu, \nu) = (\tilde{\mathbf{A}}\mathbf{A}_{j^*}, \mathbf{B}_{j^*}\tilde{\mathbf{B}})$.
     ii. Get $\mathsf{path}$ corresponding to leaf $j^*$ in IHMT for $(h_1, \dots, h_r)$.
     iii. Set $\mathsf{bits} = \mathsf{r}_{j^*}$.
     iv. Set $\mathsf{resp} = (\mu, \nu, \mathsf{path}, \mathsf{bits})$.
  2. If $\mathsf{chal} = 1$:
     i. Set $(\mu, \nu) = (\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$.
     ii. Set $\mathsf{bits} = (\mathsf{r}_1, \dots, \mathsf{r}_r)$.
     iii. Set $\mathsf{resp} = (\mu, \nu, \mathsf{bits})$.
  3. Send $\mathsf{resp}$ to verifier.

V. Verify$(\mathsf{pk}, \mathsf{cmt}, \mathsf{chal}, \mathsf{resp})$
  1. If $\mathsf{chal} = 0$:
     i. Compute $\hat{\mathbf{G}} = \mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_0))$.
     ii. Compute $h' = \mathsf{Com}(\mathsf{bits}, \hat{\mathbf{G}})$.
     iii. Get $\mathsf{root}'$ from IHMT using $h'$ and $\mathsf{path}$.
  2. If $\mathsf{chal} = 1$:
     i. For all $j = 1, \dots, r$:
        a. Compute $\hat{\mathbf{G}}_j = \mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_j))$.
        b. Compute $h'_j = \mathsf{Com}(\mathsf{r}_j, \hat{\mathbf{G}}_j)$.
     ii. Build IHMT from $(h'_1, \dots, h'_r)$ and get $\mathsf{root}'$.
  3. Accept if $\mathsf{root}' = \mathsf{cmt}$, reject otherwise.
```

Figure 5.3: MCE Ring Sigma Protocol

Public Data
$q, m, n, k, t = \lambda, r \in \mathbb{N}$. $\mathsf{Com} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{2\lambda}$. $\mathsf{H}, \overline{\mathsf{H}} : \{0,1\}^* \to \{0,1\}^t$.

```
II. Sign($\mathsf{sk}_{j^*}$, $\mathsf{pk}$)
  1. Compute the tag
     $\mathbf{T} = \mathsf{SF}(\pi_{\mathbf{A}_{j^*}^{-1}, \mathbf{B}_{j^*}^{-1}}(\mathbf{G}_0))$.
  2. For all $i = 0, \dots, t-1$:
     i. $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
     ii. Set $\tilde{\mathbf{T}}_i = \mathsf{SF}(\pi_{\tilde{\mathbf{A}}_i^{-1}, \tilde{\mathbf{B}}_i^{-1}}(\mathbf{T}))$.
     iii. For all $j = 1, \dots, r$:
        a. Sample $\mathsf{r}_{i,j} \xleftarrow{\$} \{0,1\}^\lambda$.
        b. Set $\tilde{\mathbf{G}}_{i,j} = \pi_{\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i}(\mathbf{G}_j)$.
        c. Set $h_{i,j} = \mathsf{Com}(\mathsf{r}_{i,j}, \mathsf{SF}(\tilde{\mathbf{G}}_{i,j}))$.
     iv. Build IHMT from $(h_{i,1}, \dots, h_{i,r})$ and get $\mathsf{root}_i$.
     v. Compute $\bar{h}_i = \overline{\mathsf{H}}(\tilde{\mathbf{T}}_i, \mathsf{root}_i)$.
  3. Conpute
     $h = \mathsf{H}(\bar{h}_0, \dots, \bar{h}_{t-1}, \mathbf{T}, \mathsf{msg})$.
  4. Parse $h = [h_0 | \dots | h_{t-1}]$, $h_i \in \{0,1\}$.
  5. For all $i = 0, \dots, t-1$:
     i. If $h_i = 0$:
        a. Set $(\mu_i, \nu_i) = (\tilde{\mathbf{A}}_i \mathbf{A}_{j^*}, \mathbf{B}_{j^*} \tilde{\mathbf{B}}_i)$.
        b. Set $(\bar{\mu}_i, \bar{\nu}_i) = (\mathbf{A}_{j^*} \tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \mathbf{B}_{j^*})$.
        c. Get $\mathsf{path}_i$ corresponding to leaf $j^*$ in IHMT for $(h_{i,1}, \dots, h_{i,r})$.
        d. Set $\mathsf{bits}_i = \mathsf{r}_{i,j^*}$.
        e. Set $\mathsf{rsp}_i = \{\mu_i, \nu_i, \bar{\mu}_i, \bar{\nu}_i, \mathsf{path}_i, \mathsf{bits}_i\}$.
     ii. If $h_i = 1$:
        a. Set $(\mu_i, \nu_i) = (\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i)$.
        b. Set $\mathsf{bits}_i = (\mathsf{r}_{i,1}, \dots, \mathsf{r}_{i,r})$.
        c. Set $\mathsf{rsp}_i = \{\mu_i, \nu_i, \mathsf{bits}_i\}$.
  6. Set $\sigma = (h, \mathsf{resp}_0, \dots, \mathsf{resp}_{t-1}, \mathbf{T})$.
  7. Send $\sigma$ to verifier.

I. Keygen($j$)
  1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k \times mn}$ in standard form.
  2. $(\mathbf{A}_j, \mathbf{B}_j) \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
  3. Compute $\mathbf{G}_j = \mathsf{SF}(\pi_{\mathbf{A}_j, \mathbf{B}_j}(\mathbf{G}_0))$.
  4. Set $\mathsf{sk}_j = (\mathbf{A}_j, \mathbf{B}_j)$ and $\mathsf{pk}_j = \mathbf{G}_j$.
  5. Set $\mathsf{pk} = (\mathbf{G}_0, \mathsf{pk}_1, \dots, \mathsf{pk}_r)$

III. Verify($\mathsf{pk}$, $\mathsf{msg}$, $\sigma$)
  1. Parse $h = [h_0 | \dots | h_{t-1}]$, $h_i \in \{0,1\}$.
  2. For all $i = 0, \dots, t-1$:
     i. If $h_i = 0$:
        a. Compute $\hat{\mathbf{G}}_i = \mathsf{SF}(\pi_{\mu_i, \nu_i}(\mathbf{G}_0))$.
        b. Compute $h'_i = \mathsf{Com}(\mathsf{bits}_i, \hat{\mathbf{G}}_i)$.
        c. Get $\mathsf{root}_i$ from IHMT using $h'_i$ and $\mathsf{path}_i$.
        d. Compute
           $\hat{\mathbf{T}}_i = \mathsf{SF}(\pi_{\bar{\mu}_i^{-1}, \bar{\nu}_i^{-1}}(\mathbf{G}_0))$.
     ii. If $h_i = 1$:
        a. For all $j = 1, \dots, r$:
           † Compute
             $\hat{\mathbf{G}}_{i,j} = \mathsf{SF}(\pi_{\mu_i, \nu_i}(\mathbf{G}_j))$.
           ‡ Compute
             $h'_{i,j} = \mathsf{Com}(\mathsf{r}_{i,j}, \hat{\mathbf{G}}_{i,j})$.
        b. Build IHMT from
           $(h'_{i,1}, \dots, h'_{i,r})$ and get $\mathsf{root}'_i$.
        c. Compute
           $\hat{\mathbf{T}}_i = \mathsf{SF}(\pi_{\mu_i^{-1}, \nu_i^{-1}}(\mathbf{T}))$.
     iii. Compute $\bar{h}'_i = \overline{\mathsf{H}}(\hat{\mathbf{T}}_i, \mathsf{root}'_i)$.
  3. Compute
     $h' = \mathsf{H}(\bar{h}'_0, \dots, \bar{h}'_{t-1}, \mathbf{T}, \mathsf{msg})$.
  4. Accept if $h' = h$ or reject otherwise.
```

Figure 5.4: MCE linkable ring signature scheme

```
Public Data
$q, m, n, k, \lambda, t, s, w \in \mathbb{N}$.
$\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\lambda}$.

I. Keygen()
  1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k \times mn}$ in standard form.
  2. Set $\mathbf{A}_0 = \mathbf{I}_m$, $\mathbf{B}_0 = \mathbf{I}_n$.
  3. $\mathbf{T} \xleftarrow{\$} \mathrm{GL}_k(q)$
  4. Compute $\mathbf{G}'_0 = \mathbf{T}\mathbf{G}_0$
  5. Parse the first two rows of $\mathbf{G}'_0$ into $\mathbf{P}_1^0, \mathbf{P}_2^0 \in \mathbb{F}_q^{m \times n}$
  6. For all $j = 1, \dots, s-1$:
      i. $\mathbf{P}_1^j, \mathbf{P}_2^j \xleftarrow{\$} \mathbb{F}_q^{m \times n}$
      ii. Find $\mathbf{A}_j$ and $\mathbf{B}_j$ from:
            $\mathbf{P}_1^j \mathbf{B}_j^{-1} = \mathbf{A}_j \mathbf{P}_1^0$
            $\mathbf{P}_2^j \mathbf{B}_j^{-1} = \mathbf{A}_j \mathbf{P}_2^0$
      iii. Compute $\mathbf{G}_j = S^{\mathbf{0}}(\pi_{\mathbf{A}_j, \mathbf{B}_j}(\mathbf{G}'_0))$.
  7. Set $\mathsf{sk} = (\mathbf{A}_1^{-1}, \mathbf{B}_1^{-1}, \dots, \mathbf{A}_{s-1}^{-1}, \mathbf{B}_{s-1}^{-1})$.
  8. Set $\mathsf{pk} = (\mathbf{G}_0, \mathbf{G}_1, \dots, \mathbf{G}_{s-1})$.

II. Sign(sk)
  1. For all $i = 0, \dots, t-1$:
      i. $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
      ii. Compute $\tilde{\mathbf{G}}_i = S^{\mathbf{0}}(\pi_{\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i}(\mathbf{G}_0))$.
  2. Compute $h = \mathsf{H}(\tilde{\mathbf{G}}_0, \dots, \tilde{\mathbf{G}}_{t-1}, \mathsf{msg})$.
  3. Expand $h$ to $(h_0, \dots, h_{t-1})$, $0 \le h_i < s$.
  4. For all $i = 0, \dots, t-1$:
      i. Set $(\mu_i, \nu_i) = (\tilde{\mathbf{A}}_i \mathbf{A}_{h_i}^{-1}, \mathbf{B}_{h_i}^{-1} \tilde{\mathbf{B}}_i)$.
  5. Set $\sigma = (\mu_0, \dots, \mu_{t-1}, \nu_0, \dots, \nu_{t-1}, h)$.
  6. Send $\sigma$ to verifier.

III. Verify(pk, msg, $\sigma$)
  1. Expand $h$ to $(h_0, \dots, h_{t-1})$, $0 \le h_i < s$.
  2. For all $i = 0, \dots, t-1$:
      i. Set $\hat{\mathbf{G}}_i = S^{\mathbf{0}}(\pi_{\mu_i, \nu_i}(\mathbf{G}_{h_i}))$.
  3. Compute $h' = \mathsf{H}(\hat{\mathbf{G}}_0, \dots, \hat{\mathbf{G}}_{t-1}, \mathsf{msg})$.
  4. Accept if $h' = h$ or reject otherwise.
```

Figure 5.5: The MEDS Protocol

Parameter Set	q	n	m	k	s	t	w	ST	PK	SIG	FS
MEDS-2826-st	8191	13	13	13	2	256	30	✓	2826	18 020	-129.74
MEDS-8445-st-f	8191	13	13	13	4	160	23	✓	8445	13 946	-128.01
MEDS-8445-st	8191	13	13	13	4	464	17	✓	8445	10 726	-128.76
MEDS-8445-st-s	8191	13	13	13	4	1760	13	✓	8445	8702	-128.16
MEDS-11255-st	8191	13	13	13	5	224	19	✓	11 255	11 618	-128.45
MEDS-11255	8191	13	13	13	5	224	19	–	11 255	13 778	-128.45
MEDS-42161-st	8191	13	13	13	16	128	16	✓	42 161	9616	-128.85
MEDS-356839-st	8191	13	13	13	128	80	12	✓	356 839	7288	-129.64
MEDS-716471-st	8191	13	13	13	256	64	11	✓	716 471	6530	-127.37

Table 5.2: Parameters for MEDS, for $\lambda = 128$ bits of classical security. 'ST' for seed tree. 'PK' for 'public key size' and 'SIG' for 'signature size in bytes, 'FS' for 'Fiat-Shamir' probability logarithmic to base 2.

Parameter Set	Key Generation		Signing		Verification	
	(ms)	(mcyc.)	(ms)	(mcyc.)	(ms)	(mcyc.)
MEDS-2826-st	71.13	135.14	102.79	195.30	98.00	186.21
MEDS-8445-st-f	211.45	401.75	63.21	120.09	60.14	114.27
MEDS-8445-st	211.35	401.57	185.68	352.79	178.42	339.01
MEDS-8445-st-s	211.77	402.36	697.00	1324.31	673.19	1279.05
MEDS-11255-st	258.18	490.54	88.12	167.44	84.47	160.48
MEDS-11255	258.99	492.08	88.19	167.56	84.50	160.56
MEDS-42161-st	969.97	1842.95	50.54	96.03	48.42	92.00
MEDS-356839-st	8200.83	15 581.58	31.63	60.10	32.38	61.52
MEDS-716471-st	18 003.07	34 205.83	25.57	48.58	28.94	54.98

Scheme	pk size (byte)	sig size (byte)	key gen (mcyc.)	sign (mcyc.)	verify (mcyc.)
ed25519 (scop)	32	64	0.0484	0.0513	0.182
[67] dilithium2 (scop)	1 312	2 420	0.151	0.363	0.163
[82] falcon512dyn (scop)	897	666	19.5	0.880	0.0856
[97] sphincsf128shake256 (scop)	32	16 976	6.86	220	9.91
[97] sphincss128shake256 (scop)	32	8 080	218	3 500	4.04
[26] UOV ov-Ip	278 432	128	2.90	0.105	0.0903
[16] LESS-I	8 748	12 728	—	—	—
[11] Wavelet	3 236 327	930	7 400	1 640	1.09
[2] SDitH Var3f	144	12 115	—	4.03	3.04
[2] SDitH Var3sss	144	5 689	—	994	969
MEDS-8445-st-f	8 445	13 914	402	120	114
MEDS-11255-st	11 255	11 586	491	168	160
MEDS-42161-st	42 161	9 584	1 840	96.0	92.0
MEDS-716471-st	716 471	6 498	34 200	48.6	55.0

Table 5.4: Performance comparison to other relevant schemes (mcyc. rounded to three significant figures). Data marked with '(scop)' is from the SUPERCOP website. For SPHINCS+ we list results for the 'simple' variant.

Chapter 6

Automorphism group

6.1 Placeholder

The paper will go here.

Considerations

Some considerations on future research on isometries

Part II

Isogenies (CSIDH)

Intro

Some introduction on CSIDH

Chapter 7

Patient Zero & Patient Six

7.1 Placeholder

The paper will go here.

Chapter 8

Disorientation faults

8.1 Placeholder

The paper will go here.

Chapter 9

Projective Radical Isogenies

9.1 Placeholder

The paper will go here.

Chapter 10

Higher Security CSIDH

10.1 Placeholder

The paper will go here.

Chapter 11

Effective Pairings in Isogeny-based Cryptography

11.1 Placeholder

The paper will go here.

Considerations

Some considerations on future research on CSIDH, including ideas we worked on but did not publish.

Part III

Isogenies (SQIsign)

Intro

Some introduction on SQIsign

Chapter 12

AprèsSQI

12.1 Placeholder

The paper will go here.

Chapter 13

SQIsign on M4

13.1 Placeholder

The paper will go here.

Chapter 14

Return of the Kummer

14.1 Placeholder

The paper will go here.

Considerations

Some considerations on future research on SQIsign, given the ideas in Apres, Kummer and current state-of-the-art.

The remainder is not part of part:measuring

add some additional spacing

Part IV

General

Chapter 15

Guide to the design of signature schemes.

Contents of the SOK, e.g. guide to the design.

Bibliography

[1] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and Jurjen Bos. *HQC*. NIST PQC Submission. 2020 (cit. on p. 73).

[2] Carlos Aguilar-Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. *The Return of the SDitH*. Cryptology ePrint Archive, Paper 2022/1645. To appear at Eurocrypt 2023. 2022. URL: https://eprint.iacr.org/2022/1645 (cit. on pp. 103, 109).

[3] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar 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 (cit. on p. 13).

[4] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. "Cryptographic group actions and applications". In: *ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer. 2020, pp. 411–439 (cit. on p. 75).

[5] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. *Classic McEliece*. NIST PQC Submission. 2020 (cit. on pp. 73, 100).

[6] Gianira N Alfarano, Francisco Javier Lobillo, Alessandro Neri, and Antonia Wachter-Zeh. "Sum-rank product codes and bounds on the minimum distance". In: *Finite Fields and Their Applications* 80 (2022), p. 102013 (cit. on p. 56).

[7] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zémor, Valentin Vasseur, and Santosh Ghosh. *BIKE*. NIST PQC Submission. 2020 (cit. on p. 73).

[8] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zemor, Carlos Aguilar Melchor, Slim Bettaieb, Loic Bidoux, Magali Bardet, and Ayoub Otmani. *ROLLO (Rank-Ouroboros, LAKE and LOCKER)*. 2019. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/%20round-2-submissions (cit. on p. 42).

[9] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, and Gilles Zémor. "Low Rank Parity Check Codes: New Decoding Algorithms and Applications to Cryptography". In: *IEEE Transactions on Information Theory* 65 (2019), pp. 7697–7717 (cit. on p. 42).

[10] V. Arlazarov, E. Dinic, M. Kronrod, and I. Faradžev. "On Economical Construction of the Transitive Closure of an Oriented Graph". In: *Doklady Akademii Nauk*. Vol. 194. Russian Academy of Sciences. 1970, pp. 487–488 (cit. on p. 99).

[11] Gustavo Banegas, Thomas Debris-Alazard, Milena Nedeljković, and Benjamin Smith. *Wavelet: Code-based postquantum signatures with fast verification on microcontrollers*. Cryptology ePrint Archive, Paper 2021/1432. 2021. URL: https://eprint.iacr.org/2021/1432 (cit. on pp. 103, 109).

[12] Feng Bao, Robert H. Deng, and Huafei Zhu. "Variations of Diffie-Hellman Problem". In: *ICICS 2003*. Ed. by Sihan Qing, Dieter Gollmann, and Jianying Zhou. Vol. 2836. LNCS. Springer, 2003, pp. 301–312 (cit. on p. 80).

[13] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. "Improvements of Algebraic Attacks for Solving the Rank Decoding and MinRank Problems". In: *ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12491. LNCS. Springer, 2020, pp. 507–536. DOI: 10.1007/978-3-030-64837-4_17. URL: https://doi.org/10.1007/978-3-030-64837-4%5C_17 (cit. on pp. 74, 94, 96).

[14] Alessandro Barenghi, Jean-Francois Biasse, Edoardo Persichetti, and Paolo 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 (cit. on p. 46).

[15] Alessandro Barenghi, Jean-François Biasse, Tran Ngo, Edoardo Persichetti, and Paolo Santini. "Advanced signature functionalities from the code equivalence problem". In: *Int. J. Comput. Math. Comput. Syst. Theory* 7.2 (2022), pp. 112–128 (cit. on pp. 74, 103).

[16] Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. "LESS-FM: fine-tuning signatures from the code equivalence problem". In: *Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021,*

Proceedings 12. Springer. 2021, pp. 23–43 (cit. on pp. 41, 46, 55, 74, 80, 87, 88, 103, 109).

[17] Paul Barrett. “Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor”. In: *CRYPTO’ 86*. Ed. by Andrew M. Odlyzko. Springer Berlin Heidelberg, 1987, pp. 311–323 (cit. on p. 99).

[18] Andrea Basso, Pierrick Dartois, Luca de Feo, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. “SQIsign2D-West The Fast, the Small, and the Safer”. In: (2024) (cit. on p. 34).

[19] Genrich R. Belitskii, Vyacheslav Futorny, Mikhail Muzychuk, and Vladimir V. Sergeichuk. “Congruence of matrix spaces, matrix tuples, and multilinear maps”. In: *Linear Algebra and its Applications* 609 (2021), pp. 317–331. ISSN: 0024-3795. DOI: `https://doi.org/10.1016/j.laa.2020.09.018`. URL: `https://www.sciencedirect.com/science/article/pii/S0024379520304407` (cit. on pp. 42, 52).

[20] Emanuele Bellini, Florian Caullery, Philippe Gaborit, Marcos Manzano, and Víctor Mateu. “Improved Veron Identification and Signature Schemes in the Rank Metric”. In: *2019 IEEE International Symposium on Information Theory (ISIT)* (2019), pp. 1872–1876 (cit. on p. 42).

[21] Thierry P. Berger. “Isometries for rank distance and permutation group of Gabidulin codes”. In: *IEEE Trans. Inf. Theory* 49 (2003), pp. 3016–3019 (cit. on p. 19).

[22] Thierry P. Berger. “Isometries for rank distance and permutation group of Gabidulin codes”. In: *IEEE Trans. Inf. Theory* 49 (2003), pp. 3016–3019 (cit. on p. 42).

[23] Daniel J Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. “Faster computation of isogenies of large prime degree”. In: *Open Book Series* 4.1 (2020), pp. 39–55 (cit. on p. 26).

[24] Ward Beullens. “Not Enough LESS: An Improved Algorithm for Solving Code Equivalence Problems over $\mathbb{F}_q$”. In: *SAC 2020*. Ed. by Orr Dunkelman, Michael J. Jacobson, and Colin O’Flynn. Vol. 12804. LNCS. Springer, 2020, pp. 387–403 (cit. on p. 95).

[25] Ward Beullens. “Not enough LESS: An improved algorithm for solving code equivalence problems over $\mathbb{F}_q$”. In: *International Conference on Selected Areas in Cryptography*. Springer. 2020, pp. 387–403 (cit. on p. 42).

[26] Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jhih Shih, and Bo-Yin Yang. *Oil and Vinegar: Modern Parameters and Implementations*. Cryptology ePrint Archive, Paper 2023/059. 2023. URL: `https://eprint.iacr.org/2023/059` (cit. on pp. 88, 103, 109).

[27] Ward Beullens, Shuichi Katsumata, and Federico Pintore. “Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices”. In: *ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer, 2020, pp. 464–492 (cit. on pp. 74, 78, 84, 87, 89, 103).

[28] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. “CSI-FiSh: Efficient Isogeny Based Signatures Through Class Group Computations”. In: *ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11921. LNCS. Springer, 2019, pp. 227–247 (cit. on pp. 87, 88).

[29] Ward Beullens and Bart Preneel. “Field Lifting for Smaller UOV Public Keys”. In: *Progress in Cryptology – INDOCRYPT 2017*. Ed. by Arpita Patra and Nigel P. Smart. Cham: Springer International Publishing, 2017, pp. 227–246. ISBN: 978-3-319-71667-1 (cit. on p. 41).

[30] Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. “LESS is More: Code-Based Signatures Without Syndromes”. In: *Progress in Cryptology - AFRICACRYPT 2020*. Ed. by Abderrahmane Nitaj and Amr Youssef. Cham: Springer International Publishing, 2020, pp. 45–65. ISBN: 978-3-030-51938-4 (cit. on pp. 41, 46, 74).

[31] Xavier Bonnetain and André Schrottenloher. “Quantum Security Analysis of CSIDH”. In: *EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. LNCS. Springer, 2020, pp. 493–522 (cit. on p. 75).

[32] Giacomo Borin, Edoardo Persichetti, Paolo Santini, Federico Pintore, and Krijn Reijnders. *A Guide to the Design of Digital Signatures based on Cryptographic Group Actions*. Cryptology ePrint Archive, Paper 2023/718. `https://eprint.iacr.org/2023/718`. 2023. URL: `https://eprint.iacr.org/2023/718` (cit. on p. 12).

[33] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. “CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM”. In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2018, pp. 353–367 (cit. on p. 10).

[34] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. “Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem”. In: *2015 IEEE Symposium on Security and Privacy*. 2015, pp. 553–570. DOI: `10.1109/SP.2015.40` (cit. on p. 10).

[35] Wieb Bosma, John Cannon, and Catherine Playoust. “The Magma Algebra System. I. The User Language”. In: *J. Symbolic Comput.* 24.3-4 (1997). Computational algebra and number theory (London, 1993), pp. 235–265 (cit. on p. 70).

[36] C. Bouillaguet, J.-C. Faugère, P.A. Fouque, and L. Perret. “Practical Cryptanalysis of the Identification Scheme Based on the Isomorphism of Polynomial With One Secret Problem”. In: *Public Key Cryptography – PKC 2011*. Vol. 6571. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 441–458 (cit. on pp. 48, 54, 67, 69).

[37] Charles Bouillaguet. “Algorithms for some hard problems and cryptographic attacks against specific cryptographic primitives. (Études d’hypothèses algorithmiques et attaques de primitives cryptographiques)”. PhD thesis. Paris Diderot University, France, 2011. URL: `https://tel.archives-ouvertes.fr/tel-03630843` (cit. on pp. 43, 49, 60, 66, 68–71, 91).

[38] Charles Bouillaguet, Pierre-Alain Fouque, and Amandine 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*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 211–227. DOI: `10.1007/978-3-642-38348-9\_13`. URL: `https://doi.org/10.1007/978-3-642-38348-9%5C_13` (cit. on pp. 43, 49, 58–60, 62, 66–68, 71, 91, 92).

[39] Peter Bruin. “The Tate pairing for abelian varieties over finite fields”. In: *Journal de theorie des nombres de Bordeaux* 23.2 (2011), pp. 323–328 (cit. on p. 35).

[40] David G Cantor. “On arithmetical algorithms over finite fields”. In: *Journal of Combinatorial Theory, Series A* 50.2 (1989), pp. 285–300 (cit. on p. 24).

[41] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. “GeMSS: A Great Multivariate Short Signature”. In: 2017 (cit. on p. 41).

[42] J. W. S. Cassels and E. V. Flynn. *Prolegomena to a Middlebrow Arithmetic of Curves of Genus 2*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1996 (cit. on p. 23).

[43] Wouter Castryck and Thomas Decru. “An Efficient Key Recovery Attack on SIDH”. In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 423–447. DOI: `10.1007/978-3-031-30589-4\_15`. URL: `https://doi.org/10.1007/978-3-031-30589-4%5C_15` (cit. on p. 31).

[44] Wouter Castryck, Marc Houben, Simon-Philipp Merz, Marzio Mula, Sam van Buuren, and Frederik Vercauteren. “Weak instances of class group action based cryptography via self-pairings”. In: *Annual International Cryptology Conference*. Springer. 2023, pp. 762–792 (cit. on pp. 35, 36).

[45] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. “CSIDH: An Efficient Post-Quantum Commutative Group Action”. In: *ASIACRYPT 2018*. Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11274. LNCS. Springer, 2018, pp. 395–427 (cit. on pp. 74, 75).

[46] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. “CSIDH: an efficient post-quantum commutative group action”. In: *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24*. Springer. 2018, pp. 395–427 (cit. on pp. 9, 28).

[47] Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Francisco Rodrıéguez-Henrıéquez, Sina Schaeffler, and Benjamin Wesolowski. *SQIsign: Algorithm specifications and supporting documentation*. National Institute of Standards and Technology. 2023. URL: `https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/sqisign-spec-web.pdf` (cit. on p. 30).

[48] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. CRC press, 2005 (cit. on p. 35).

[49] Maria Corte-Real Santos, Craig Costello, and Benjamin Smith. *Efficient (3,3)-isogenies on fast Kummer surfaces*. Cryptology ePrint Archive, Paper 2024/144. `https://eprint.iacr.org/2024/144`. 2024. URL: `https://eprint.iacr.org/2024/144` (cit. on p. 32).

[50] Nicolas Courtois, Er Klimov, Jacques Patarin, and Adi Shamir. “Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations”. In: *In Advances in Cryptology, Eurocrypt’00, LNCS 1807*. Springer-Verlag, 2000, pp. 392–407 (cit. on p. 94).

[51] Nicolas Tadeusz Courtois. “Efficient zero-knowledge authentication based on a linear algebra problem MinRank”. In: *Advances in Cryptology – ASIACRYPT 2001*. Vol. 2248. LNCS. Springer, 2001, pp. 402–421 (cit. on p. 96).

[52] Jean-Marc Couveignes. *Hard Homogeneous Spaces*. Cryptology ePrint Archive, Paper 2006/291. 2006 (cit. on p. 75).

[53] Alain Couvreur, Thomas Debris-Alazard, and Philippe Gaborit. *On the hardness of code equivalence problems in rank metric*. 2021. arXiv: `2011.04611 [cs.IT]` (cit. on pp. 19, 42, 44, 46, 47, 55, 58, 74, 80, 90).

[54] Ronald Cramer. “Modular design of secure yet practical cryptographic protocols”. In: *Ph. D.-thesis, CWI and U. of Amsterdam* 2 (1996) (cit. on p. 11).

[55] Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. “SQISignHD: new dimensions in cryptography”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2024, pp. 3–32 (cit. on p. 34).

[56] Luca De Feo and Steven D. Galbraith. “SeaSign: Compact Isogeny Signatures from Class Group Actions”. In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Cham: Springer International Publishing, 2019, pp. 759–789. ISBN: 978-3-030-17659-4 (cit. on pp. 41, 74, 87, 88).

[57] Luca De Feo, David Jao, and Jérôme Plût. “Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies”. In: *J. Math. Cryptol.* 8.3 (2014), pp. 209–247. DOI: 10.1515/jmc-2012-0015. URL: https://doi.org/10.1515/jmc-2012-0015 (cit. on p. 31).

[58] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. “SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies”. In: *Advances in Cryptology – ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Cham: Springer International Publishing, 2020, pp. 64–93. ISBN: 978-3-030-64837-4 (cit. on p. 41).

[59] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. “SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies”. 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*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12491. Lecture Notes in Computer Science. Springer, 2020, pp. 64–93. DOI: 10.1007/978-3-030-64837-4_3. URL: https://doi.org/10.1007/978-3-030-64837-4%5C_3 (cit. on p. 30).

[60] Luca De Feo, Antonin Leroux, Patrick Longa, and Benjamin Wesolowski. “New Algorithms for the Deuring Correspondence - Towards Practical and Secure SQISign Signatures”. In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 659–690. DOI: 10.1007/978-3-031-30589-4_23. URL: https://doi.org/10.1007/978-3-031-30589-4%5C_23 (cit. on p. 30).

[61] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. “Wave: A New Family of Trapdoor One-Way Preimage Sampleable Functions Based on Codes”. In: *Advances in Cryptology – ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Cham: Springer International Publishing, 2019, pp. 21–51. ISBN: 978-3-030-34578-5 (cit. on p. 41).

[62] Max Deuring. “Die typen der multiplikatorenringe elliptischer funktionenkörper”. In: *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*. Vol. 14. 1. Springer Berlin/Heidelberg. 1941, pp. 197–272 (cit. on p. 29).

[63] Whitfield Diffie and Martin E Hellman. “New directions in cryptography”. In: *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*. 2022, pp. 365–390 (cit. on p. 8).

[64] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias Kannwischer, and Jacques Patarin. *Rainbow*. Tech. rep. National Institute of Standards and Technology, 2020. URL: `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions` (cit. on p. 88).

[65] Jintai Ding and Dieter Schmidt. “Rainbow, a New Multivariable Polynomial Signature Scheme”. In: *ACNS*. Ed. by John Ioannidis, Angelos D. Keromytis, and Moti Yung. Vol. 3531. Lecture Notes in Computer Science. 2005, pp. 164–175. ISBN: 3-540-26223-7 (cit. on p. 41).

[66] Vivien Dubois, Louis Granboulan, and Jacques Stern. “An efficient provable distinguisher for HFE”. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2006, pp. 156–167 (cit. on pp. 48, 61).

[67] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. “Crystals-dilithium: A lattice-based digital signature scheme”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 238–268 (cit. on pp. 73, 103, 109).

[68] Léo Ducas and Wessel van Woerden. “On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography”. In: *Advances in Cryptology – EUROCRYPT 2022*. Ed. by Orr Dunkelman and Stefan Dziembowski. Cham: Springer International Publishing, 2022, pp. 643–673. ISBN: 978-3-031-07082-2 (cit. on p. 41).

[69] Max Duparc and Tako Boris Fouotsa. *SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies*. Cryptology ePrint Archive, Paper 2024/773. `https://eprint.iacr.org/2024/773`. 2024. URL: `https://eprint.iacr.org/2024/773` (cit. on p. 34).

[70] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. “Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1, 1): Algorithms and complexity”. In: *J. Symb. Comput.* 46.4 (2011), pp. 406–437 (cit. on p. 94).

[71] Jean-Charles Faugère, Françoise Levy-dit-Vehel, and Ludovic Perret. “Cryptanalysis of MinRank”. In: *CRYPTO*. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 280–296 (cit. on pp. 74, 94, 96).

[72] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. "Folding Alternant and Goppa Codes With Non-Trivial Automorphism Groups". In: *IEEE Trans. Inf. Theory* 62.1 (2016), pp. 184–198. DOI: 10.1109/TIT.2015.2493539. URL: https://doi.org/10.1109/TIT.2015.2493539 (cit. on p. 52).

[73] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric Portzamparc, and Jean-Pierre Tillich. "Structural Cryptanalysis of McEliece Schemes with Compact Keys". In: *Des. Codes Cryptography* 79.1 (Apr. 2016), pp. 87–112. ISSN: 0925-1022. DOI: 10.1007/s10623-015-0036-z. URL: https://doi.org/10.1007/s10623-015-0036-z (cit. on p. 52).

[74] Jean-Charles Faugère and Ludovic Perret. "Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects". In: *EUROCRYPT '06*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. Springer, July 5, 2006, pp. 30–47. ISBN: 3-540-34546-9 (cit. on pp. 48, 49, 58, 60, 67, 68).

[75] Jean-Charles Faugère and Ludovic Perret. "Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, 2006, pp. 30–47 (cit. on pp. 91, 93).

[76] Joël Felderhoff. *Hard Homogenous Spaces and Commutative Supersingular Isogeny based Diffie-Hellman*. Internship report. LIX, Ecole polytechnique and ENS de Lyon, Aug. 2019. URL: https://hal.archives-ouvertes.fr/hal-02373179 (cit. on p. 80).

[77] Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. "SCALLOP: scaling the CSI-FiSh". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2023, pp. 345–375 (cit. on p. 9).

[78] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *CRYPTO '86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, 1986, pp. 186–194 (cit. on p. 81).

[79] Amos Fiat and Adi Shamir. "How to prove yourself: Practical solutions to identification and signature problems". In: *Conference on the theory and application of cryptographic techniques*. Springer. 1986, pp. 186–194 (cit. on p. 11).

[80] Amos Fiat and Adi Shamir. "How to prove yourself: practical solutions to identification and signature problems". In: *Proceedings on Advances in cryptology—CRYPTO '86*. Santa Barbara, California, United States: Springer-Verlag, 1987, pp. 186–194 (cit. on pp. 41, 48).

[81] Pierre-Alain Fouque, Louis Granboulan, and Jacques Stern. "Differential Cryptanalysis for Multivariate Schemes". In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 341–353. ISBN: 978-3-540-25910-7 (cit. on p. 48).

[82] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. "Falcon: Fast-Fourier lattice-based compact signatures over NTRU". In: *Submission to the NIST's post-quantum cryptography standardization process* 36.5 (2018), pp. 1–75 (cit. on pp. 73, 103, 109).

[83] Vyacheslav Futorny, Joshua A. Grochow, and Vladimir V. Sergeichuk. "Wildness for tensors". In: *Linear Algebra and its Applications* 566 (2019), pp. 212–244. ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2018.12.022. URL: https://www.sciencedirect.com/science/article/pii/S0024379518305937 (cit. on p. 42).

[84] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. *Swoosh: Efficient Lattice-Based Non-Interactive Key Exchange*. 2024 (cit. on p. 9).

[85] Steven D Galbraith, Florian Hess, and Frederik Vercauteren. "Hyperelliptic pairings". In: *International Conference on Pairing-Based Cryptography*. Springer. 2007, pp. 108–131 (cit. on p. 35).

[86] Steven D. Galbraith. *Advances in Elliptic Curve Cryptography, Chapter IX*. Ed. by Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. Vol. 317. Cambridge University Press, 2005 (cit. on pp. 35, 36).

[87] Steven D. Galbraith. *Mathematics of public key cryptography*. Cambridge University Press, 2012 (cit. on pp. 7, 21).

[88] Theodoulos Garefalakis. "The generalized Weil pairing and the discrete logarithm problem on elliptic curves". In: *LATIN 2002: Theoretical Informatics: 5th Latin American Symposium Cancun, Mexico, April 3–6, 2002 Proceedings 5*. Springer. 2002, pp. 118–130 (cit. on p. 35).

[89] Marc Girault. "A (Non-Practical) Three-Pass Identification Protocol Using Coding Theory". In: *Proceedings of the International Conference on Cryptology on Advances in Cryptology*. AUSCRYPT '90. Sydney, Australia: Springer-Verlag, 1990, pp. 265–272. ISBN: 0387530002 (cit. on p. 41).

[90] Elisa Gorla. "Rank-metric codes". In: *CoRR* abs/1902.02650 (2019). arXiv: 1902.02650. URL: http://arxiv.org/abs/1902.02650 (cit. on p. 17).

[91] Elisa Gorla and Flavio Salizzoni. "MacWilliams' Extension Theorem for rank-metric codes". In: *Journal of Symbolic Computation* 122 (2024), p. 102263 (cit. on p. 18).

[92] Torbjörn Granlund and Peter L. Montgomery. “Division by Invariant Integers Using Multiplication”. In: *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*. PLDI ’94. Orlando, Florida, USA: Association for Computing Machinery, 1994, pp. 61–72 (cit. on p. 99).

[93] Joshua A. Grochow and Youming Qiao. *Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions*. 2019. DOI: `10.48550/ARXIV.1907.00309`. URL: `https://arxiv.org/abs/1907.00309` (cit. on p. 42).

[94] Shay Gueron, Edoardo Persichetti, and Paolo Santini. “Designing a Practical Code-Based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup”. In: *Cryptography* 6.1 (2022), p. 5 (cit. on p. 89).

[95] Ishay Haviv and Oded Regev. “On the lattice isomorphism problem”. In: *SODA 2014*. Ed. by Chandra Chekuri. ACM SIAM, 2014, pp. 391–404 (cit. on p. 74).

[96] Loo-Keng Hua. “A theorem on matrices over a sfield and its applications”. In: *Bulletin of the American Mathematical Society*. Vol. 55. 11. 1949, pp. 1046–1046 (cit. on p. 45).

[97] Andreas Hulsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. *SPHINCS+*. NIST PQC Submission. 2020 (cit. on pp. 73, 103, 109).

[98] Dale Husemöller. *Elliptic Curves, 2nd edition*. Springer, 2004 (cit. on p. 36).

[99] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. *SIKE*. Tech. rep. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions`. National Institute of Standards and Technology, 2022 (cit. on p. 31).

[100] Ernst Kani. “The number of curves of genus two with elliptic differentials.” In: (1997) (cit. on pp. 31, 32).

[101] Aviad Kipnis and Adi Shamir. “Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization”. In: *Advances in Cryptology – CRYPTO ’99*. Vol. 1666. LNCS. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 19–30 (cit. on p. 96).

[102] Neal Koblitz. “Elliptic curve cryptosystems”. In: *Mathematics of computation* 48.177 (1987), pp. 203–209 (cit. on p. 22).

[103] Neal Koblitz. “Hyperelliptic cryptosystems”. In: *Journal of cryptology* 1 (1989), pp. 139–150 (cit. on p. 22).

[104] Bor de Kock. "A non-interactive key exchange based on ring-learning with errors". PhD thesis. Master's thesis. Master's thesis, Eindhoven University of Technology, 2018 (cit. on p. 9).

[105] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. "On the quaternion-isogeny path problem". In: *LMS Journal of Computation and Mathematics* 17.A (2014), pp. 418–432 (cit. on p. 29).

[106] Greg Kuperberg. "Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem". In: *TQC 2013*. Ed. by Simone Severini and Fernando G. S. L. Brandão. Vol. 22. LIPIcs. Schloss Dagstuhl, 2013, pp. 20–34 (cit. on p. 81).

[107] Georg Landsberg. "Ueber eine Anzahlbestimmung und eine damit zusammenhängende Reihe." In: (1893) (cit. on p. 62).

[108] Jeffrey Leon. "Computing automorphism groups of error-correcting codes". In: *IEEE Transactions on Information Theory* 28.3 (1982), pp. 496–511 (cit. on p. 41).

[109] Jeffrey S. Leon. "Computing automorphism groups of error-correcting codes". In: *IEEE Trans. Inf. Theory* 28.3 (1982), pp. 496–510 (cit. on pp. 74, 95).

[110] Florence Jessie MacWilliams. "Combinatorial problems of elementary abelian groups". PhD thesis. 1962 (cit. on p. 18).

[111] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. "A Direct Key Recovery Attack on SIDH". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 448–471. DOI: 10.1007/978-3-031-30589-4_16. URL: https://doi.org/10.1007/978-3-031-30589-4%5C_16 (cit. on p. 31).

[112] Moxie Marlinspike and Trevor Perrin. *The X3DH Key Agreement Protocol*. Signal Specifications. https : / / signal . org / docs / specifications/x3dh/(accessed2022-01-04). 2016 (cit. on p. 10).

[113] R. J. McEliece. "A Public-Key System Based on Algebraic Coding Theory". In: *Jet Propulsion Laboratory, California Institute of Technology* (1978). DSN Progress Report 44, pp. 114–116 (cit. on pp. 41, 73).

[114] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Gilles Zemor, Alain Couvreur, and Adrien Hauteville. *RQC*. 2019. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/%20round-2-submissions (cit. on p. 42).

[115] Ralph C Merkle. "Secure communications over insecure channels". In: *Communications of the ACM* 21.4 (1978), pp. 294–299 (cit. on p. 8).

[116] Victor S Miller. “The Weil pairing, and its efficient calculation”. In: *Journal of cryptology* 17.4 (2004), pp. 235–261 (cit. on p. 35).

[117] Victor S Miller. “Use of elliptic curves in cryptography”. In: *Conference on the theory and application of cryptographic techniques*. Springer. 1985, pp. 417–426 (cit. on p. 22).

[118] Peter L Montgomery. “Speeding the Pollard and elliptic curve methods of factorization”. In: *Mathematics of computation* 48.177 (1987), pp. 243–264 (cit. on p. 22).

[119] Peter L. Montgomery. “Modular multiplication without trial division”. In: *Mathematics of Computation* 44 (1985), pp. 519–521. ISSN: 0025–5718 (cit. on p. 99).

[120] D. Mumford, C. Musili, M. Nori, E. Previato, M. Stillman, and H. Umemura. *Tata Lectures on Theta II: Jacobian theta functions and differential equations*. Modern Birkhäuser Classics. Birkhäuser Boston, 2012. ISBN: 9780817645786. URL: `https://books.google.nl/books?id=xaNCAAAAQBAJ` (cit. on p. 23).

[121] Kohei Nakagawa and Hiroshi Onuki. “QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras”. In: *Cryptology ePrint Archive* (2023) (cit. on p. 34).

[122] Kohei Nakagawa and Hiroshi Onuki. *SQIsign2D-East: A New Signature Scheme Using 2-dimensional Isogenies*. Cryptology ePrint Archive, Paper 2024/771. `https://eprint.iacr.org/2024/771`. 2024. URL: `https://eprint.iacr.org/2024/771` (cit. on p. 34).

[123] National Institute for Standards and Technology. *NIST Workshop on Cybersecurity in a Post-Quantum World*. URL: `http://www.nist.gov/itl/csd/ct/post-quantum-crypto-workshop-2015.cfm` (cit. on p. 41).

[124] Alessandro Neri. “Twisted linearized Reed-Solomon codes: A skew polynomial framework”. In: *arXiv preprint arXiv:2105.10451* (2021) (cit. on p. 56).

[125] P Nguyen and C Wolf. *International Workshop on Post-Quantum Cryptography*. 2006 (cit. on p. 73).

[126] H. Niederreiter. “Knapsack-type cryptosystems and algebraic coding theory.” English. In: *Probl. Control Inf. Theory* 15 (1986), pp. 159–166 (cit. on p. 41).

[127] NIST. *Post-Quantum Cryptography Standardization*. URL: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`. 2017 (cit. on p. 73).

[128] Roberto W Nóbrega and Bartolomeu F Uchôa-Filho. “Multishot codes for network coding using rank-metric codes”. In: *2010 Third IEEE International Workshop on Wireless Network Coding*. IEEE. 2010, pp. 1–6 (cit. on p. 55).

[129] Hiroshi Onuki. "On oriented supersingular elliptic curves". In: *Finite Fields and Their Applications* 69 (2021), p. 101777 (cit. on p. 27).

[130] Jacques Patarin. "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms". In: *EUROCRYPT*. Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1996, pp. 33–48. ISBN: 3-540-61186-X (cit. on pp. 41, 48).

[131] Jacques Patarin. "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms". In: *Advances in Cryptology – EUROCRYPT '96*. Vol. 1070. LNCS. Berlin, Heidelberg: Springer-Verlag, 1996, pp. 33–48 (cit. on pp. 73, 74).

[132] Jacques Patarin, Louis Goubin, and Nicolas Courtois. "Improved Algorithms for Isomorphisms of Polynomials". In: *EUROCRYPT '98*. Vol. 1403. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 1998, pp. 184–200 (cit. on pp. 44, 48).

[133] Ray Perlner and Daniel Smith-Tone. *Rainbow Band Separation is Better than we Thought*. Cryptology ePrint Archive, Paper 2020/702. 2020 (cit. on pp. 93–95).

[134] Ludovic Perret. "A Fast Cryptanalysis of the Isomorphism of Polynomials with One Secret Problem". In: *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*. Vol. 3494. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 354–370 (cit. on pp. 48, 54).

[135] Christiane Peters. "Information-set decoding for linear codes over $\mathbb{F}_q$". In: *International Workshop on Post-Quantum Cryptography*. Springer. 2010, pp. 81–94 (cit. on p. 55).

[136] Eugene Prange. "The use of information sets in decoding cyclic codes". In: *IRE Trans. Information Theory* 8.5 (1962), pp. 5–9 (cit. on p. 91).

[137] Dana Randall. *Efficient Generation of Random Nonsingular Matrices*. Tech. rep. UCB/CSD-91-658. EECS Department, UC Berkeley, 1991 (cit. on p. 100).

[138] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Theory of computing*. Ed. by Harold N. Gabow and Ronald Fagin. ACM, 2005, pp. 84–93 (cit. on p. 73).

[139] Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. *Hardness estimates of the Code Equivalence Problem in the Rank Metric*. **TODO:** FIX. 2022 (cit. on pp. 74, 80, 86, 90–93).

[140] Damien Robert. "Breaking SIDH in Polynomial Time". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 472–503. DOI: 10.1007/978-3-031-30589-4_17. URL: https://doi.org/10.1007/978-3-031-30589-4%5C_17 (cit. on p. 31).

[141] Damien Robert. "Evaluating isogenies in polylogarithmic time". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1068. URL: https://eprint.iacr.org/2022/1068 (cit. on p. 33).

[142] Damien Robert. *The geometric interpretation of the Tate pairing and its applications*. Cryptology ePrint Archive, Paper 2023/177. 2023. URL: https://eprint.iacr.org/2023/177 (cit. on p. 35).

[143] Georg Rosenhain. *Abhandlung über die Functionen zweier Variabler mit vier Perioden: welche die Inversen sind der ultra-elliptischen Integrale erster Klasse*. 65. W. Engelmann, 1895 (cit. on p. 23).

[144] Alexander Rostovtsev and Anton Stolbunov. *PUBLIC-KEY CRYPTOSYSTEM BASED ON ISOGENIES*. Cryptology ePrint Archive, Paper 2006/145. 2006 (cit. on p. 75).

[145] Claus-Peter Schnorr. "Efficient identification and signatures for smart cards". In: *Advances in Cryptology—CRYPTO'89 Proceedings 9*. Springer. 1990, pp. 239–252 (cit. on p. 11).

[146] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Post-Quantum TLS Without Handshake Signatures". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1461–1480. ISBN: 9781450370899. DOI: 10.1145/3372297.3423350. URL: https://doi.org/10.1145/3372297.3423350 (cit. on p. 10).

[147] Nicolas Sendrier. "Finding the permutation between equivalent linear codes: The support splitting algorithm". In: *IEEE Trans. Inf. Theory* 46 (2000), pp. 1193–1203 (cit. on p. 42).

[148] V V SergeıHELPchuk. "CLASSIFICATION PROBLEMS FOR SYSTEMS OF FORMS AND LINEAR MAPPINGS". In: *Mathematics of the USSR-Izvestiya* 31.3 (June 1988), pp. 481–501. DOI: 10.1070/im1988v031n03abeh001086. URL: https://doi.org/10.1070/im1988v031n03abeh001086 (cit. on p. 52).

[149] Peter W. Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332 (cit. on pp. 9, 24).

[150] Joseph H Silverman. "A survey of local and global pairings on elliptic curves and abelian varieties". In: *Pairing-Based Cryptography-Pairing 2010: 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings 4*. Springer. 2010, pp. 377–396 (cit. on p. 35).

[151] Joseph H. Silverman. *The arithmetic of elliptic curves.* Vol. 106. Springer, 2009 (cit. on p. 21).

[152] Gang Tang, Dung Hoang Duong, Antoine Joux, Thomas Plantard, Youming Qiao, and Willy Susilo. "Practical Post-Quantum Signature Schemes from Isomorphism Problems of Trilinear Forms". In: *Advances in Cryptology – EUROCRYPT 2022.* Ed. by Orr Dunkelman and Stefan Dziembowski. Cham: Springer International Publishing, 2022, pp. 582–612. ISBN: 978-3-031-07082-2 (cit. on p. 41).

[153] Gang Tang, Dung Hoang Duong, Antoine Joux, Thomas Plantard, Youming Qiao, and Willy Susilo. "Practical Post-Quantum Signature Schemes from Isomorphism Problems of Trilinear Forms". In: *EUROCRYPT 2022.* Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. LNCS. Springer, 2022, pp. 582–612 (cit. on pp. 74, 100).

[154] John Tate. "Endomorphisms of abelian varieties over finite fields". In: *Inventiones mathematicae* 2.2 (1966), pp. 134–144 (cit. on p. 25).

[155] "Theorie der quadratischen Formen in beliebigen Korpern". In: *J. Reine Angew. Math.* 176 (1937), pp. 31–44 (cit. on p. 49).

[156] "Untersuchungen über quadratische Formen in Korpern der Charakteristik 2, I". In: *J. Reine Angew. Math.* 183 (1941), pp. 148–167 (cit. on p. 49).

[157] Serge Vaudenay. *A Classical Introduction to Cryptography: Applications for Communications Security.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005 (cit. on p. 59).

[158] Jacques Vélu. "Isogénies entre courbes elliptiques". In: *Comptes-Rendus de l'Académie des Sciences* 273 (1971), pp. 238–241 (cit. on p. 26).

[159] John Voight. *Quaternion algebras.* Springer Nature, 2021 (cit. on p. 21).

[160] Zhe-Xian Wan. "A proof of the automorphisms of linear groups over a sfield of characteristic 2". In: *Sci. Sinica* 11 (1962), pp. 1183–1194 (cit. on p. 45).