An Introduction to Code-Based Cryptography

Anna-Lena Horlemann

School of Computer Science, University of St.Gallen, Switzerland

Chapter 1

Introduction

Most of our currently used cryptographic systems rely on either the integer factorization problem or the discrete logarithm problem over an elliptic curve or a finite field. We say that a problem is "hard" for cryptographic purposes if there is no polynomial time algorithm (known) to solve these problems. For the three problems above (integer factorization and two versions of the discrete logarithm problem) this is true for conventional computers. On quantum computers, however, there is a known algorithm that solves these problems in polynomial time. This algorithm is known as *Shor's algorithm*[1]; it was originally formulated for integer factorization, but can be adapted to solve discrete logarithm type problems, as well. This means that the classical systems are not secure and new algorithms are necessary for the future, to ensure secure communication in the time of quantum computers.

With the recent progression of quantum computer development, the cryptographic research community has therefore been actively looking for cryptographic systems that can withstand attacks from quantum computers. This realm of study is referred to as *post-quantum cryptography*.

At the time of writing there are five main streams of post-quantum cryptography:

- code-based cryptography
- lattice-based cryptography
- multivariate cryptography
- hash-based cryptography
- supersingular elliptic curve isogeny cryptography

In 2016, the National Institute of Standards and Technology (NIST) initiated a standardization procedure for post-quantum cryptosystems. These cryptosystems typically derive their foundation from NP-complete problems, primarily for two reasons: NP-complete problems are at least as complex as the most challenging problems within NP, while their solutions can be efficiently verified. In the current NIST standardization project almost all submissions for public-key encryption in the third round are either code or lattice based. In this lecture we will focus on the first, *code-based cryptography*.

[1] `https://en.wikipedia.org/wiki/Shor's_algorithm`

Code-based cryptography relies on the fundamental premise that decoding in a randomly generated linear code constitutes an NP-complete problem, as demonstrated by Berlekamp, McEliece, and Van Tilborg in 1978 [2]. In that same year, McEliece introduced a cryptosystem [14] where a strategically chosen code with an inherent algebraic structure and an efficient decoding algorithm is employed. This code is then masked to appear as a random linear code. The process involves encoding a message into a codeword and encrypting it by introducing an error to the message. With knowledge of the code's algebraic structure, the original message can be retrieved. However, an adversary is confronted with the challenge of decoding an erroneous codeword in a randomly generated linear code.

These lecture notes provide an introduction to code-based cryptography, delving into the mathematical underpinnings of such systems and the challenges associated with designing secure yet practical schemes. We explore the main ideas of code-based public key encryption schemes and the strategies employed to potentially breach these cryptographic systems. Naturally, these lecture notes cannot cover all aspects of code-based cryptography and we refer the interested reader to the nice survey [25] and the references therein for more information.

Chapter 2

Preliminaries

2.1 What is cryptography?

We want to communicate (or store) data in a secure manner, such that no eavesdropper can recover the sent information. This is done with the help of *cryptography*, by *encrypting* the data before sending it, and then *decrypting* it at the receiver's side. An encryption is secure if an eavesdropper, who can intercept the encrypted message, cannot recover the original message from it. Cryptographic security depends on the computing power of the adversary. We call something *cryptographically secure* for a prescribed time t, if the best (known) way of breaking the cryptographic instance needs more than time t, on a given computer (possibly a personal computer or a large scale mega-computer – depending on the application). Moreover, if information is to remain confidential for many years, we have to take the current and future development of computers into account. In particular, new protocols for long-term use should be secure against attacks on a quantum computer.

The original idea of cryptography is to encrypt data such that only the intended receiver can recover the actual message, whereas eavesdroppers cannot. We distinguish two classes of such algorithms: *symmetric* and *asymmetric* encryption schemes. In the former the sender and receiver both use the *same key* in the en- and decryption procedure[1], whereas in the latter the sender encrypts with one key (called the *public key*), and the receiver decrypts with a different key (called the *private/secret key*)[2].

A different, but related, type of algorithm is a digital signature. Such signatures can be used to verify at the receiver's side if the received message was indeed from the sender, or if it was tampered with (or even replaced) by the adversary.

Since symmetric cryptosystems are (so far) not drastically effected by quantum computers, we will focus on asymmetric cryptosystems and digital signatures.

[1] This can be depicted by a normal lock for which both parties hold a key.

[2] This can be depicted with a padlock, where the public key is the actual padlock itself and the secret key is the key to the padlock, see Section 2.2.

2.2 Asymmetric and public key cryptography

Public key cryptography is a major subfield of asymmetric cryptography. In a public key encryption system some sender, say Alice, wants to send an encrypted message to a receiver, say Bob. The cryptosystem is asymmetric, in the sense that Bob publishes a *public key* and secretly stores a *private key*, such that Alice (and everyone else) can use the public key to encrypt her message, and only Bob can decrypt the message (with his private key). For the system to be secure, an attacker should not be able to decrypt the encrypted message without the knowledge of the private key. This is done by using some "hard" mathematical problem.

Digital signatures can also be seen as asymmetric cryptosystems, in particular as reversed versions of public key cryptosystems. Here, Bob uses his private key to sign a message, and Alice (or anyone) can verify his signature with the aid of the public key.

Remark 2.1. Asymmetric cryptography can be depicted with a padlock. If Alice wants to send a message in a treasure chest to Bob, he can send her an open padlock (the public key) that only he has a key to (the private key). She can use the lock on the chest and send it to Bob. He can use his private key to unlock the chest.

Now, of course, the question remains, how to construct such padlocks mathematically.

Mathematical idea (one-way function): Find "something" that is easy to compute, but where it is very (very very) hard to compute the inverse.
$\Longrightarrow$ Then the outcome of the computation is the public key and the preimage is the private key.

As an example we will look at the RSA cryptosystem, which is based on the integer factorization problem. Here we use the fact that it is easy to compute the product of two integers, but it is not easy to find the factors of a given product (of very large numbers). The algorithm is described in Algorithm 1.

Let's have a look at why the underlying hard problem of RSA is the integer factorization problem – if an attacker manages to factor n into p and q, they can recover the private key and hence decrypt any ciphertext. In fact, almost all currently used asymmetric cryptosystems use one of the following three mathematical problems:

- integer factorization
- discrete logarithm problem
- elliptic curve discrete logarithm problem

As we will see, these three problems are not hard to solve on quantum computers and hence need to be replaced by other mathematical problems in quantum secure cryptosystems.

Algorithm 1 RSA Algorithm

- Choose two large prime numbers p and q.
- Compute the product $n = p \cdot q$.
- Pick any number e that is relatively prime to $(p-1)(q-1)$.
- Compute $d \equiv e^{-1} \mod (p-1)(q-1)$.
- **Private key:** The number d.
- **Public key:** The numbers n and e.
- **Encryption:** Represent the message as a number m. It is encrypted as

$$c \equiv m^e \mod n.$$

- **Decryption:** Compute

$$m \equiv c^d \mod n.$$

Exercise 2.2. Create an RSA signature scheme, by first using the private key for signing, and the public key for validating a signature. Is Alice's signature always the same or does it depend on the message?

Exercise 2.3. Assume that Alice wants to send a secret message to Bob in a chest/box with Bob's open padlock. If the eavesdropper Eve is the mail woman and transports the open padlock of Bob, as well as the locked chest of Alice – how can she trick Alice and Bob and read the message?

Exercise 2.4. Assume that Alice and Bob still want to communicate in a secure manner by sending a message in a chest/box to each other. Assume furthermore that there is no way of providing (reliably) a public open padlock. However, Alice and Bob both have a padlock with a key. How can Alice still send a message secretly to Bob?

2.3 Attacks and cryptanalysis

To be pessimistic up front: any cryptosystem (no matter if digital or analog) can be broken. Full stop. However, when we speak of *security* we assume that it is extremely unlikely that an attacker can break a system. For this we *measure* the security of a system in the number of operations an attacker needs (worst case or on average) to break a system.[3] If the number of operations needed for an attack supersedes the number of atoms in the Universe (ca. 2^{80}) we consider this a first level of security (for non-sensitive information). In most real-world applications, however, we rather use systems with a security level of 256 or more bits (i.e., 2^{256} necessary operations for an attack).

We distinguish two main types of theoretical attacks on an encryption scheme (and similarly on a signature scheme):

[3]This is more general than considering the actual time a specific computer would need to break the system.

- **Message recovery attacks:** The attacker can directly decrypt a ciphertext but does not recover the used (private) key.
- **Key recovery attacks:** The attacker recovers the (private) key and can thus also decrypt any ciphertext.

Both are equally important[4] and need to be considered when analyzing a cryptosystem. However, they often use different techniques and tools and are therefore considered separately.

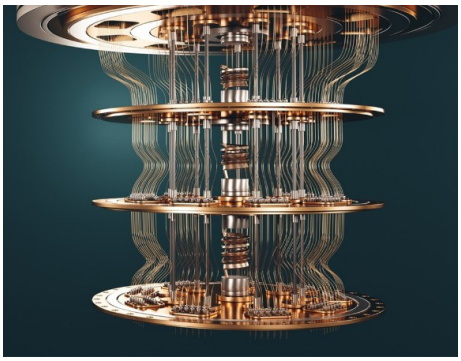
A special type of – and commonly used – key recovery attack is a *distinguisher attack*. In this case the attacker uses publicly known (mathematical) properties of the private key (that distinguishes it from a random element of the ambient set) to recover the private key from the public key. We will look at some distinguisher attacks in the following chapter.

Exercise 2.5. There are several security considerations for the setup of RSA. Here are some examples:

1. You should always choose two distinct primes $p \neq q$. How can Eve attack RSA if $p = q$?
2. When sending the same message to several parties, you should always use different moduli n. How can Eve recover the message m if she intercepted two ciphertexts $c_1 = m^{e_1} \mod n$ and $c_2 = m^{e_2} \mod n$?

2.4 The issue with quantum computers

Quantum computers are one of the most exciting and challenging research objects in today's communication and computation research. Due to their different functionality, they promise big advantages in many computational applications such as simulations in biotechnology, chemistry, pharmaceutical research, etc., and optimization tasks in finance, energy management, or logistics.



On the other hand, quantum computers pose one of the biggest threats in cybersecurity, in particular to public key cryptosystems and digital signatures. Currently the most commonly used public key cryptosystems are RSA and Diffie-Hellman (in classical and elliptic curve variants). All these methods are

[4] To be precise, the latter usually has a more severe impact on the long-term use of the cryptosystem, but both types need to be infeasible for a system to be secure.

insecure if quantum computers capable of running Shor's algorithm[5] become available, which is likely to happen in the next few decades. The importance of this threat is further stressed by the NSA's statement in 2015 to transition to post-quantum secure algorithms for their Suite B family of cryptographic algorithms and the National Institute of Standards and Technology's (NIST) initiation of the international Post-Quantum Cryptography Standardization project[6] in 2016.

Remark 2.6. It is not true that quantum computers will speed up every type of algorithm. In fact, there are only very few algorithms known where the quantum technology will lead to a significant speed up.[7]

The expected time frame for quantum computers to be powerful enough to break these implementations is between 10 to 30 years, however there are two major reasons why we should transition to post-quantum cryptography already now:

1. Adversaries can store captured encrypted data for many years and decrypt it when the development of quantum computers enables the adversary to do so. This is a particular threat for highly sensitive data that needs to be secured over a long period of time, such as diplomatic data or trade secrets.

2. Communication systems in aircrafts, trains or ships (among others) are often in operation for more than 30 years and are hardly ever updated due to operational reasons. It is thus of paramount importance that communication algorithms on such systems are quickly changed to post-quantum secure systems. Otherwise these systems will be vulnerable and become accessible to adversaries.

Many proposals for post-quantum secure encryption and digital signatures have been made in the past, many of which have been broken by now. Currently, the National Institute of Standards and Technology (NIST) is considering a number of proposals for post-quantum secure encryption and signatures to be made US (and most likely also international) standards. The goal is to find cryptosystems of comparable size and computational complexity to RSA (or Diffie-Hellman), but that are secure against attacks run on quantum computers.

Remark 2.7. Since Shor's algorithm is currently the only known algorithm that leads to a speed up from exponential to polynomial run time on quantum computers, post-quantum cryptography can also be reformulated as the Shor-resistant cryptography.

As already mentioned in the introduction there are currently five main streams of *post-quantum cryptography*, based on different hard mathematical problems:

[5] https://en.wikipedia.org/wiki/Shor's_algorithm

[6] https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization

[7] https://en.wikipedia.org/wiki/Quantum_algorithm

- code-based cryptography
- lattice-based cryptography
- multivariate cryptography
- hash-based cryptography
- supersingular elliptic curve isogeny cryptography

We will focus on *code-based cryptography*, which is based on tools from *coding theory* or *error-correcting codes*, which we will introduce in the following section.

2.5 Error-correcting codes

Since code-based cryptography uses a lot of *coding theory*, we start by briefly explaining *error correcting codes*. This will later on be used to explain the basic ideas of code-based cryptography.

Error correcting codes are classically used for communication over a noisy channel. One adds redundancy to the data to be sent, so that the noise (or the errors) can be filtered out on the receiver's side. Depending on the channel, the noise type varies and different codes have to be constructed for the different channel types. The differences can lie in the choice of the underlying alphabet, as well as the choice of the coding metric that is used. We will introduce the alphabets and metrics that will be used in these lecture notes in the following. For more details we refer the reader to the books [11, 13].

Denote by $\mathbb{F}_q$ the finite field with q elements where q is a prime power.

Definition 2.8. A classical *block code* is a set of vectors of a fixed length n over some finite field $\mathbb{F}_q$. A block code in $\mathbb{F}_q^n$ is called *linear*, if it is a linear subspace of $\mathbb{F}_q^n$.

A linear code $C \subseteq \mathbb{F}_q^n$ of dimension k can be represented by a *generator matrix* $G \in \mathbb{F}_q^{k\times n}$ such that $C = \langle G\rangle$ – where $\langle G\rangle$ denotes the row space of a matrix G – or a *parity check matrix* $H \in \mathbb{F}_q^{(n-k)\times n}$ such that C is the kernel of H.

Even though these are the most studied families, error-correcting codes do not have to consist of vectors. For example, in *linear network coding* codewords are matrices over $\mathbb{F}_q$, or they are the row spaces of those matrices. In the former case, the codewords are still elements of the (matrix) vector space $\mathbb{F}_q^{m\times n}$, for some $m, n \in \mathbb{N}$, and hence linearity of such matrix codes can be defined in the usual sense. For our lecture notes the notion above (where codewords are vectors) is sufficient.

Various metrics can be used for different coding theoretic applications. In this lecture we will consider the Hamming and the rank metric. We will explain these metrics in the following.

Definition 2.9.

- The *Hamming distance* d_H on $\mathbb{F}_q^n$ is defined as

$$d_H((u_1,\dots,u_n),(v_1,\dots,v_n)) := |\{i \mid u_i \neq v_i\}|$$

for any $(u_1, \ldots, u_n), (v_1, \ldots, v_n) \in \mathbb{F}_q^n$. Note that this metric can be defined for vectors over any underlying alphabet.

- A *rank metric code* is a subset of the matrix space $\mathbb{F}_q^{m \times n}$. The *rank distance* d_R is defined as the rank of the difference of the corresponding matrices over $\mathbb{F}_q$, i.e., for $A, B \in \mathbb{F}_q^{m \times n}$,

$$d_R(A, B) := \operatorname{rank}(A - B).$$

Note that these codes can also be represented in $\mathbb{F}_{q^m}^n$, via a vector space isomorphism $\mathbb{F}_q^m \cong \mathbb{F}_{q^m}$.

For both of the above metrics d_* (where $* \in \{H, R\}$) and the corresponding codes C we define the *minimum distance* of C as

$$d_*(C) := \min\{d_*(u, v) \mid u, v \in C, u \neq v\}.$$

For both distances we consider additive error models, i.e., when sending a codeword $c \in C$ over the considered communication channel we receive

$$r = c + e,$$

for some error vector (or matrix) e that lives in the same space as c. We define the *weight* of a codeword to be its distance to the all-zero codeword. Then the distance between r and c is equal to the weight of e. If e is an error vector of weight at most $(d_*(C) - 1)/2$, then there is a unique closest codeword to r – namely c – with respect to the chosen metric. The process of finding the closest codeword c to a given received word r is called *(minimum distance) decoding*. It follows that we can correct (or decode) any error of weight at most $(d_*(C)-1)/2$; hence we say that C has *error-correction capability* $\lfloor (d_*(C) - 1)/2 \rfloor$.

The various metrics for error correction arise from different applications with different noise behaviors. In general, given an application with a prescribed abstract error model, one tries to find a metric such that the most likely sent codeword is the closest codeword to a received word with respect to the metric. For example, in the classical telecommunication channel, where information is transmitted via periodic waveforms, the Hamming metric is used for *orthogonal modulation*, whereas the Lee metric is used for *phase modulation* [22]. In other applications we consider the complete loss of a symbol instead of added noise. In these *erasure channels* the Hamming metric is again the usual choice for recovering the codeword and the corresponding message.

One of the main applications for the rank metric is *linear network coding*. In this setting we consider broadcast communication over a network with one sender and several receivers who all want to get the same information. One can show that the capacity of such channels can be achieved by splitting the information into several packets which are simultaneously injected into the network via separate edges and letting the inner nodes of the network linearly combine their incoming information before forwarding it. The codewords can now be represented by stacking the packets that are sent simultaneously as rows in a matrix. However, the linear combinations inside the network may lead to a propagation of the errors during transmission over large parts of the network, which makes the Hamming metric unsuitable for this setting. It turns out that,

if the operations of the network are known, the number of actual errors is best measured by the rank metric [3].

Many code constructions are known for the Hamming and the rank metric, of which we will use

- Reed-Solomon codes
- Goppa codes
- Gabidulin codes

in our lecture notes. We will define those in the following.

Definition 2.10. Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ be pairwise distinct. Moreover, let $v_1, \ldots, v_n \in \mathbb{F}_q^*$. Denote $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $v = (v_1, \ldots, v_n)$. Then

$$\mathrm{RS}_{n,k,q}(\alpha, v) := \{(v_1 f(\alpha_1), \ldots, v_n f(\alpha_n)) \mid f(x) \in \mathbb{F}_q[x], \deg f < k\}$$

is called a *generalized Reed-Solomon code* of length n and dimension k.

Theorem 2.11. *A generalized Reed-Solomon code of length n and dimension k has minimum Hamming distance $n-k+1$ and is therefore an MDS (maximum distance separable) code.*[8]

Definition 2.12. Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{q^m}^*$ be pairwise distinct. Moreover, let $G(x) \in \mathbb{F}_{q^m}[x]$ be such that $G(\alpha_i) \neq 0$ and $v_i = \prod_{j \neq i}(\alpha_j - \alpha_i) G(\alpha_i)^{-1}$ for $i = 1, \ldots, n$. Denote $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $v = (v_1, \ldots, v_n)$. Then

$$RS_{n,k,q^m}(\alpha, v) \cap \mathbb{F}_q^n$$

is called a *q-ary Goppa code* of length n.

Note that with the definition above Goppa codes are subfield subcodes of Reed-Solomon codes. Originally (non-binary) Goppa codes were defined differently, however, the definitions are equivalent.

There is no closed formula for the dimension and the minimum Hamming distance of Goppa codes, however there are lower bounds on these parameters. For these lecture notes we do not need these lower bounds and refer the interested reader to [7].

Definition 2.13. Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{q^m}$ be linearly independent over $\mathbb{F}_q$ and write $\alpha = (\alpha_1, \ldots, \alpha_n)$. Denote by $\mathcal{L}_q[x]$ the set of $\mathbb{F}_q$-linearized polynomials[9] over $\mathbb{F}_{q^m}$. Then

$$\mathrm{Gab}_{n,k}(\alpha) := \{(f(\alpha_1), \ldots, f(\alpha_n)) \mid f(x) \in \mathcal{L}_q[x], \deg f < k\}$$

is called a *Gabidulin code* of length n and dimension k.

Theorem 2.14. *A Gabidulin code of length n and dimension k has minimum rank distance $n-k+1$ and is therefore an MRD (maximum rank distance) code.*[10]

[8] MDS codes are optimal in the sense that they reach the upper bound $n-k+1$ on the minimum Hamming distance.

[9] Polynomials of the form $f(x) = \sum_i u_i x^{q^i}$ with $u_i \in \mathbb{F}_{q^m}$.

[10] MRD codes are optimal in the sense that they reach the upper bound $n-k+1$ on the minimum rank distance.

Lastly we need to know what the isometries (i.e., the distance preserving automorphisms) for the Hamming metric and the rank metric are. We denote by $\mathrm{GL}_n(q)$ the general linear group of invertible matrices in $\mathbb{F}_q^{n\times n}$ and by S_n the group of $n \times n$ permutation matrices.

Theorem 2.15. *1. The linear isometries for the metric space $(\mathbb{F}_q^n, d_H)$ are represented by the monomial matrices, i.e., matrices of the form*

$$\begin{pmatrix} v_1 & 0 & \dots & 0 \\ 0 & v_2 & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & v_n \end{pmatrix} P$$

for some $P \in S_n$ and non-zero $v_1, \dots, v_n \in \mathbb{F}_q$.

2. The linear isometries for the metric space $(\mathbb{F}_{q^m}^n, d_R)$ are represented by the elements of $\mathrm{GL}_n(q)$.

These linear isometries can be extended to semi-linear isometries by combining them (coordinate-wise) with the Galois group of $\mathbb{F}_q$, respectively $\mathbb{F}_{q^m}$.

Chapter 3

Code-Based Cryptosystems

3.1 General setup for public key encryption

A completely different application of error-correcting codes in any ambient metric space can be found in *code-based cryptography*, one of the main streams of *post-quantum cryptography*. Here the main focus is to secure data against eavesdroppers, in contrast to the channel coding setup above, where the reliability of data was the main objective.

The main idea of code-based cryptography is to use the *syndrome decoding problem (SDP)* as the underlying hard mathematical problem. The computational version of this problem can be stated as follows:

Syndrome Decoding Problem: Given some $t \in \mathbb{N}$, a parity check matrix $H \in \mathbb{F}_q^{r \times n}$ and a syndrome $s \in \mathbb{F}_q^r$, find a vector $e \in \mathbb{F}_q^n$ of weight at most t that fulfills $eH^\top = s$.

Note that the weight in our formulation can be any general weight; historically the SDP was first studied with respect to the Hamming metric, and thereafter for the rank metric and other metrics/weights. Its decisional version (i.e., the problem of deciding if such a vector e exists, without explicitly finding it) is known to be NP complete for any additive weight:

Theorem 3.1. *Let* $\mathrm{wt} : \mathbb{F}_q \to \mathbb{R}_{\geq 0}$ *be a function that satisfies the following properties:*

1. $\mathrm{wt}(0) = 0$,
2. $\mathrm{wt}(1) = 1$ *and* $\mathrm{wt}(x) \geq 1$ *for all* $x \neq 0$.

If wt *is extended additively to* $\mathbb{F}_q^n$ *(i.e., by adding the weights of the coordinates) then the SDP with respect to* wt *is NP-complete.*

Remark 3.2. Most known coding weights fulfill the conditions above, however, the rank metric does not, since it is not additive on the coordinates. It is currently not known if the rank metric version of the SDP is NP-complete, however, there are indicators that it should be the case. (In particular, there is a probabilistic reduction of an NP-complete problem to the rank metric SDP.)

Remark 3.3. From a theoretical point of view it is one of the big selling points of code-based cryptography that the SDP is proven to be NP-complete. E.g., the

integer factorization problem, which has been used in public-key cryptography in abundance over the last decades, is not NP-complete (under the assumption that N is not equal to NP).

There are two main general cryptosystems which are based on error-correcting codes – the *McEliece* [14] and the *Niederreiter system* [17]. Both of these have many variants, depending on which type of code one wants to use. Since both are equivalent from a security point of view (see [10]) we focus on the McEliece system in this proposal. For implementation purposes however, and for the construction of digital signatures, the Niederreiter system is of great interest, as well.

Originally, the McEliece cryptosystem was based on binary Goppa codes. The underlying idea in its general form, using an arbitrary linear block code, is as follows.

Algorithm 2 Generalized McEliece cryptosystem

- The receiver chooses a code C with generator matrix G and an efficient decoding algorithm. Moreover, they need a disguising function ϕ that is a near-isometry.[1]
- **Private key:** G and the decoding algorithm
- **Public key:** $\phi(G)$ together with the error correction capability of the code generated by $\phi(G)$, say $\hat{t}$
- **Encryption:** Choose a random error vector e of weight at most $\hat{t}$ and encrypt the message m as

$$c = m\ \phi(G) + e.$$

- **Decryption:** Compute $\phi^{-1}(c)$ and decode this in the secret code C to recover m.

Similarly the general framework of the Niederreiter system is described in Algorithm 3.

In both cryptosystems an attacker is not able to recover the message m without knowing ϕ, respectively the secret code C. As a brute force attack they can try to decode in the public code $\phi(C)$, but this code has no structure and hence no efficient decoding algorithm, and decoding in such a "random" code is too difficult.

Remark 3.4. One can show that the McEliece and the Niederreiter cryptosystems are equivalent from a security point of view, i.e., that if one can break one of them (with given parameters and codes) then one can also break the other.

The advantage of Niederreiter is that the ciphertext is smaller than in McEliece, whereas the disadvantage is that mapping the message to an error vector is not straight-forward.

[1] A near-isometry is a relaxation of the concept of isometry, namely a function on the vectors that changes the weight by at most a given value. To be precise, the function ϕ needs to have more properties than just being a near-isometry. However, we will not go into the technical details at this point.

Algorithm 3 Generalized Niederreiter algorithm

- The receiver chooses a code C with parity check matrix H and an efficient decoding algorithm. Moreover, they need a disguising function ϕ that is a near-isometry.
- **Private key:** H and the decoding algorithm
- **Public key:** $\phi(H)$ together with the error correction capability of the code which is the kernel of $\phi(H)$, say $\hat{t}$
- **Encryption:** Represent the message as an error vector e of weight at most $\hat{t}$ and encrypt it as
$$c = e\,\phi(H)^\top.$$
- **Decryption:** Compute $\phi^{-1}(c)$ and decode this in the secret code C to recover e and hence the message.

For the above cryptosystems to be efficient and secure, the following have to be fulfilled:

- We need a code C that has efficient encoding and decoding algorithms.
- We assume that the code family used is known, hence we require this family to have enough elements to prevent brute force attacks.
- The error correction capability needs to be large enough such that a brute force attack on the possible errors can be prevented.
- We need a disguising function such that the original code cannot be found from the public generator matrix.
- Any generic decoding[2] should be infeasible.

Naturally, one could choose extremely long codes to increase the security parameters. However, this affects the efficiency and the key size of the cryptosystem. Generally, code-based cryptography suffers from large key sizes. To be secure against generic decoding attacks, the generator matrix of the secret code, and hence also the generator matrix of the public key, needs to be quite large. E.g., to achieve 96 bits security[3], we need approximately 10^6 bits in the public key size of the classical McEliece system. For comparison, RSA needs a public key of less than 2048 bits to achieve the same security level. On the other hand, the encryption and decryption times are very fast compared to other cryptosystems, which makes code-based cryptosystems very promising for many applications. One of the main research goals is hence to find codes and disguising functions that allow smaller key sizes than currently known variants.

Exercise 3.5. If $C \subseteq \mathbb{F}_q^n$ is a linear code of dimension k, what are the sizes of the public key and the ciphertext in the McEliece cryptosystem (in $q$bits or bits)?

[2]Generic decoding means a general decoding procedure that works for any (random or unstructured) code of the prescribed parameters.

[3]Meaning that an attack needs at least 2^{96} operations.

What are the respective sizes for the Niederreiter system? Can you compress the public key sizes by sharing more public information?

3.2 Overview of some variants

Over the last decades many variants of the McEliece (or Niederreiter) cryptosystem have been proposed. Recall that the original paper introducing the McEliece cryptosystem proposed to use binary Goppa codes. This system still remains unbroken. Since then, many other block codes have been studied in this system, e.g., Reed–Solomon, Reed–Muller, algebraic geometry, low density parity check (LDPC), wild Goppa, and polar codes. However, most of these codes are too structured, and the private key can be found efficiently by the attacker.

In the last years, some new code classes without known structural attacks were proposed to be used in code-based cryptosystems, e.g., quasi-cyclic MDPC (medium density parity check) codes [15]. This class results in significantly smaller key sizes compared to the original McEliece cryptosystem.

Another way of achieving smaller key sizes is to use rank metric codes, which was first suggested by Gabidulin, Paramonov, and Tretjakov (GPT) [6], since known generic decoding algorithms in the rank metric are less efficient than in the Hamming metric [1, 19]. Until recently, all proposed variants in the rank metric used Gabidulin codes; the main difference of the variants is the respective disguising function. Many of these variants have again been broken by now, mostly due to structural attacks, see e.g., [18, 9, 8]. However, recently some new variants were proposed, using other classes of codes, such as low rank parity check (LRPC) codes [20] or twisted Gabidulin codes [21]. Furthermore, other disguising functions have been proposed, e.g. in [12].

In the following sections we will study some of these variants and explain their strengths and weaknesses.

3.3 Variants in the Hamming metric

3.3.1 The original McEliece system (Goppa codes)

As already mentioned, the original proposal by McEliece employs binary Goppa codes as secret codes. The generator matrix is then disguised by multiplying with an invertible matrix on the left (i.e., choosing a different basis) and multiplying with a permutation matrix on the right (i.e., applying an isometry on the code).

We state the complete description in Algorithm 4.

Let us now have a look at why this works: Note that in the decryption process we implicitly use the fact that

$$cP^{-1} = (mAGP + e)P^{-1} = \underbrace{mAG}_{\text{another codeword}} + \underbrace{eP^{-1}}_{\text{same weight as } e},$$

i.e., eP^{-1} is another uniquely decodable error and we hence recover the codeword mAG (and thus the corresponding message vector mA) with any decoder for C. Since A is known to the recipient, they can recover the original message m from mA.

Algorithm 4 Original McEliece cryptosystem

- The receiver chooses a binary Goppa code $C \subseteq \mathbb{F}_2^n$ of dimension $k = n - mt$ with generator matrix G and minimum distance $2t + 1$. Moreover, they choose a random $A \in \mathrm{GL}_k(2)$ and $P \in S_n$.
- **Private key:** G and (A, P)
- **Public key:** $G' = AGP$
- **Encryption:** Choose a random error vector e of weight at most t and encrypt the message m as

$$c = m\,G' + e.$$

- **Decryption:** Compute $c' = cP^{-1}$ and decode this in C to recover mA. Recover m by multiplying with A^{-1}.

Example 3.6. Bob chooses the small binary Goppa code of length 8 and dimension 2 with generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

as secret code. This code has minimum distance 5 and can therefore correct $t = 2$ errors. Moreover, he randomly picks

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \text{ and } P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Then he publishes (or sends to Alice) the public key

$$G' = AGP = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Alice wants to send the message $m = \begin{pmatrix} 1 & 0 \end{pmatrix}$ and encrypts it as

$$\begin{aligned} c = mG' + e &= \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}, \end{aligned}$$

where e is randomly chosen among the vectors of weight at most $t = 2$. She sends c to Bob, who will then decrypt it by first inverting the operation of P,

$$c' = cP^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

and then decoding it to the closest codeword in C, which is

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

The corresponding message vector is $\begin{pmatrix}1 & 0\end{pmatrix}$ which leads to the message via

$$m = \begin{pmatrix}1 & 0\end{pmatrix} A^{-1} = \begin{pmatrix}1 & 0\end{pmatrix}.$$

Remark 3.7. The above example is of course not secure since the code's parameters are very small. In general, the code parameters have to be chosen such that a generic decoder for the public code would not be able to find the message by just decoding in this (random-looking) public code.

It is a big part of code-based cryptographic research to determine the computational complexity of the best generic decoder. In the Hamming metric most of the best known decoders are *information set decoders (ISD)* or *birthday (paradox) decoders*. For an overview of such decoders we refer to [25].

Exercise 3.8. Set up a Niederreiter system with the same parameters as in Example 3.6. What is Alice's ciphertext and how would Bob recover the message?

3.3.2 McEliece systems based on Reed-Solomon codes

One question that easily comes up when considering the McEliece cryptosystem is why it uses binary Goppa codes. A natural idea is to use MDS (maximum distance separable) codes, due to their optimality and hence most compact representation for a given error correction capability. Therefore, we could easily replace the code C in Algorithm 4 with a (generalized) Reed-Solomon code. However, it turns out that Reed-Solomon codes give rise to a *distinguisher* (a certain type of key recovery) *attack*.

For this notice that Reed-Solomon codes distinguish themselves from random linear codes by the following property.

Definition 3.9. For $u, v \in \mathbb{F}_q^n$ define the *Schur/star/coordinate-wise product* as

$$u \star v := \begin{pmatrix}u_1 v_1, \ldots, u_n v_n\end{pmatrix}.$$

Then define the *(star) square code* of a linear code $C \subseteq \mathbb{F}_q^n$ to be

$$C^{\star 2} := \langle \{ c \star d \mid c, d \in C \} \rangle.$$

Theorem 3.10. *[4, Theorem 2.3] If $C \subseteq \mathbb{F}_q^n$ is a random linear code then by high probability*

$$\dim(C^{\star 2}) = \min\left\{ \binom{k+1}{2}, n \right\},$$

whereas if C is a (generalized) Reed-Solomon code, then

$$\dim(C^{\star 2}) = \min\{2k-1, n\}.$$

The above theorem presents an easily computable distinguisher for (generalized) Reed-Solomon codes. In fact, the (star product) square code can also be used for the following code families:

- low-codimensional subcodes of GRS codes,
- Reed-Muller codes,
- Polar codes,
- special types of Goppa codes,
- high rate alternant codes,
- algebraic geometry codes.

Now that we have a distinguisher it is still not straight-forward how to use this in a key recovery attack.

In Wieschebrink's paper [27], the star product is used to identify for a certain subcode C' of a GRS code a possible pair (x, y) of code locators and column multipliers. This is achieved by computing $C'^{\star 2}$ which turns out to be $C^{\star 2}$. The Sidelnikov-Shestakov algorithm [23] is then used on $C'^{\star 2}$ to recover suitable code locators and column multipliers of the original generalized Reed-Solomon code.

The idea of a distinguisher attack becomes more apparent when looking at Wieschebrink's cryptosystem [26] and the corresponding attack. Wieschebrink's system differs from the classical McEliece system with Reed-Solomon codes by using a different disguising function. We describe it in Algorithm 5.

Algorithm 5 Wieschebrink cryptosystem

- The receiver chooses a Reed-Solomon code $C \subseteq \mathbb{F}_q^n$ of dimension k with generator matrix G. They also choose $C_1, \ldots, C_r \in \mathbb{F}_q^k$, $S \in \mathrm{GL}_k(q)$ and $P \in S_{n+r}$ uniformly at random. Let $\bar{G}$ be the matrix obtained by concatenating G and the columns $C_1, \ldots, C_r$.
- **Private key:** G and (S, P)
- **Public key:** $G' = S^{-1}\bar{G}P^{-1}$
- **Encryption:** Choose a random error vector e of weight at most t and encrypt the message m as

$$c = m\,G' + e.$$

- **Decryption:** Compute $c' = cP^{-1}$, erase the last r coordinates and decode this in C to recover mS^{-1}. Recover m by multiplying with S.

To attack this system we use the fact that by puncturing the public generator matrix in random positions and computing the dimension of the star square code, we can identify the inserted columns $C_1, \ldots, C_r$. The attack goes as follows:

- Choose a random $1 \leq i \leq n + r$ and shorten G' in the ith position (i.e., erase the ith column in G').
- Compute the dimension of the square code of this new generator matrix.
- If the dimension of the square is

$$\begin{cases} 2k + r - 2 & \text{then the erased column was probably from } \{C_1, \ldots, C_r\} \\ 2k + r - 1 & \text{then the erased column was probably from the GRS code.} \end{cases}$$

- Identify all random columns like above to recover a generator matrix for the original GRS code.
- Apply Sidelnikov-Shestakov to recover the code locators and column multipliers to be able to decode.

Exercise 3.11. Prove Theorem 3.10.

Exercise 3.12. Prove that shortening a GRS code results in another GRS code. What are the dimension and the length of the shortened code?

Exercise 3.13. Prove that the square code of the public code generated by G' has dimension $2k + r$ by high probability.

3.3.3 McEliece system with LDPC/MDPC codes

A really different approach for disguising the private key can be done with low density parity check (LDPC) codes, as first proposed in [16]. In this case the decoding algorithm depends on a sparse parity check matrix of the code. We therefore do not need to hide the code used but only the sparse parity check matrix. We describe the algorithm in Algorithm 6.

Algorithm 6 McEliece with LDPC codes

- Choose an efficiently decodable LDPC code with sparse parity check matrix $H \subseteq \mathbb{F}_q^{r \times n}$.
- **Private key:** H
- **Public key:** G any generator matrix for the code defined by H
- **Encryption:** Choose a random error vector e of weight at most t and encrypt the message m as

$$c = m\,G + e.$$

- **Decryption:** Decode c in C (with the help of H) to recover m.

Recall that the security of this system does not lie in hiding the code itself, but rather in the representation of the code that gives rise to an efficient decoder. Therefore, this system is not despicable to a distinguisher attack. However, there are key recovery attacks for this system, namely by finding low weight

codewords in the dual code, which can then be used to design a sparse parity check matrix which in turn leads to an efficient decoder.

In return, when this attack based on finding low weight codewords was analyzed it was found that there is a regime of sparsity where the attack is not feasible any more, but the cryptosystem still works. The corresponding codes are now called *MDPC (medium/moderate density parity check) codes*[4] and the NIST standardization project finalist BIKE is partly based on this idea.

3.4 Variants in the rank metric

3.4.1 The original GPT system

The Gabidulin-Paramonov-Tretjakov (GPT) cryptosystem was introduced in [6] and is another variant of the McEliece cryptosystem, using codes and near-isometries for the rank metric instead of for the Hamming metric.

To understand the near-isometry that is used as the disguising function, we need to define one more concept:

Definition 3.14.
1. Let $X \in \mathbb{F}_{q^m}^{k\times n}$. We define the *column rank* of X to be the $\mathbb{F}_q$-dimension of the $\mathbb{F}_q$-space spanned by the columns of X.
2. Let $X \in \mathbb{F}_{q^m}^{k\times n}$ be a matrix of rank k and column rank t and $V \in \mathbb{F}_{q^m}^{k\times t}, U \in \mathbb{F}_q^{t\times n}$ such that $X = VU$. We call $\langle U\rangle$ the *Grassmann support* of X which will be denoted by $\langle U\rangle = \mathrm{supp}_{\mathrm{Gr}}(X)$.

Example 3.15. Consider $\mathbb{F}_4 = \mathbb{F}_2[\alpha]$ and the matrix

$$M = \begin{pmatrix} 1 & 0 & \alpha & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

in $\mathbb{F}_4^{2\times 4}$. Its rank over $\mathbb{F}_4$ is 2, but its column rank is 3, since the first and third column are not $\mathbb{F}_2$-multiples of each other.

We can decompose M into

$$M = \begin{pmatrix} 1 & 0 & \alpha \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Then the Grassmann support of M is the row space of

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The GPT cryptosystem is described in Algorithm 7.

Similarly to Reed-Solomon codes, also Gabidulin codes suffer from distinguishing properties that make most of the cryptosystems using them vulnerable to key recovery attacks. Where we used the star product for distinguishing Reed-Solomon codes, Gabidulin codes can be distinguished by considering the intersection with itself under the coordinate-wise Frobenius map:[5]

[4] Usually, a binary code is called MDPC if there exists a parity check matrix whose rows have weight $\mathcal{O}(\sqrt{n\log(n)})$.

[5] This map – denoted by $x^{(q)}$ – raises every entry to its qth power when working over $\mathbb{F}_{q^m}$.

Algorithm 7 GPT Cryptosystem

- The receiver chooses a Gabidulin code of minimum rank distance $2t+1$ with generator matrix $G \subseteq \mathbb{F}_{q^m}^{k\times n}$. Moreover, they randomly choose $S \in \mathrm{GL}_k(q^m)$ and $X \in \mathbb{F}_{q^m}^{k\times n}$ of column rank at most $s < t$.
- **Private key:** G and S
- **Public key:** $G' = SG + X$
- **Encryption:** Choose a random error vector e of rank weight at most $t-s$ and encrypt the message m as

$$c = m\,G' + e.$$

- **Decryption:** Decode c in the Gabidulin code to recover mS. Multiply with S^{-1} to recover m.

Theorem 3.16. *[5] If $C \subseteq \mathbb{F}_{q^m}^n$ is a random linear code of dimension $0 < k < n$ then by high probability*

$$\dim(C \cap C^{(q)}) = \max\{2k-n, 0\},$$

whereas if C is a Gabidulin code, then

$$\dim(C \cap C^{(q)}) = k-1.$$

Note that this behavior differs as soon as $1 < k < n-1$, i.e., for any non-trivial Gabidulin code.

As before in the Hamming metric case, the question is now how to exploit this distinguishing property in an attack. We will explain such an attack in the following and last part of these lecture notes:

Distinguisher (key recovery) attack on the GPT cryptosystem based on rank 1 codewords

Consider the Gabidulin code $\mathrm{Gab}_{n,k}(\alpha)$ with dimension $1 < k < n$ and generator matrix SG, where $S \in \mathrm{GL}_k(\mathbb{F}_{q^m})$ and

$$G = \begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^q & \alpha_2^q & \dots & \alpha_n^q \\ & & \vdots & \\ \alpha_1^{q^{k-1}} & \alpha_2^{q^{k-1}} & \dots & \alpha_n^{q^{k-1}} \end{pmatrix}. \tag{3.1}$$

Then $\mathrm{Gab}_{n,k}(\alpha)^{(q)} \cap \mathrm{Gab}_{n,k}(\alpha)$ is the Gabidulin code $\mathrm{Gab}_{n,k-1}(\alpha^{(q)})$. Iterating with this new Gabidulin code, we can eventually obtain a code of dimension 1, which is generated by $\alpha^{(q^{k-1})}$. If we take some non-zero element of this space, it has the form $\beta\alpha^{(q^{k-1})}$, for some $\beta \in \mathbb{F}_{q^m}$. Applying the Frobenius map coordinate-wise $m-k+1$ times, we obtain an element of the form $\beta^{q^{m-k+1}}\alpha$.

Using this element, we can construct a generator matrix, BG, for $\mathrm{Gab}_{n,k}(\alpha)$ which will have the form

$$BG = \begin{pmatrix} \beta^{q^{m-k+1}} & & & \\ & \beta^{q^{m-k+2}} & & \\ & & \ddots & \\ & & & \beta \end{pmatrix} \begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^q & \alpha_2^q & \dots & \alpha_n^q \\ & & \vdots & \\ \alpha_1^{q^{k-1}} & \alpha_2^{q^{k-1}} & \dots & \alpha_n^{q^{k-1}} \end{pmatrix}.$$

The change of basis from SG to BG is then given by BS^{-1}. For a message $m \in \mathbb{F}_{q^m}^k$, encoded as mSG, we can now decode with respect to $\mathrm{Gab}_{n,k}(\beta^{q^{m-k+1}}\alpha)$ to obtain mSB^{-1}. Then, applying BS^{-1}, we can recover m.

To set up our attack we must be able to find the elements of rank one in a linear rank metric code efficiently. To accomplish this, we only need to find the codewords that have all coordinates in $\mathbb{F}_q$ (all other rank one codewords are multiples of these). The following lemma shows how these codewords in $\mathbb{F}_q^n$ can be computed.

Lemma 3.17. *[8] Let $G \in \mathbb{F}_{q^m}^{k\times n}$ be in reduced row echelon form and denote by G_i the ith row of G. Then the solutions to*

$$\sum_{i=1}^{k} a_i(G_i^{(q)} - G_i) = 0, \tag{3.2}$$

for variables $a_i \in \mathbb{F}_q$, represent the codewords of $\langle G\rangle$ in $\mathbb{F}_q^n$.

When expanded over $\mathbb{F}_q$, Equation (3.2) gives rise to a linear system of equations with k variables, which can efficiently be solved with standard methods.

Now back to our attack to break the GPT cryptosystem, which extends Overbeck's attack [18] to cryptanalyze the system for all suggested parameters. Recall that the public key generator matrix is of the form

$$G' = SG + X \in \mathbb{F}_{q^m}^{k\times n},$$

where G is a generator matrix of a Gabidulin code $\mathrm{Gab}_{n,k}(\alpha)$, $X \in \mathbb{F}_{q^m}^{k\times n}$ is a matrix of column rank s, and $S \in \mathrm{GL}_k(q^m)$. For simplicity we will from now on assume that X cannot be decomposed into a Moore matrix[6] plus a matrix of less column rank. This assumption will simplify our explanation of the attack. The interested reader is referred to [9, 8] for a general explanation of the attack on any X, depending on the so-called *Moore decomposition* of X.

Note that, as an attacker, we do not have a priori knowledge of the parameter s. We can generally assume $s = t$, or else start with $s = 1$ and increase the value up to t until the attack succeeds.

We need one more preliminary lemma before getting to the main results used in our attack:

Lemma 3.18. *[8] Let $G \in \mathbb{F}_{q^m}^{k\times n}$ and X be as above. Then all elements of rank one in*

$$\sum_{i=0}^{s} \langle G + X\rangle^{(q^i)}$$

exactly span $\mathrm{supp}_{\mathrm{Gr}}(X)$.

[6] I.e., a matrix of the form (3.1).

Proof. Let $\mathcal{U}$ be the subspace spanned by all elements of rank one in

$$\sum_{i=0}^{s} \langle G + X \rangle^{(q^i)}.$$

Let $H \in \mathbb{F}_q^{(n-s)\times n}$ be a parity check matrix for $\mathrm{supp}_{\mathrm{Gr}}(X)$. We have

$$d_{\mathrm{R}}\left(\sum_{i=0}^{s} \langle G + X \rangle^{(q^i)} H^\top\right) = d_{\mathrm{R}}\left(\sum_{i=0}^{s} \langle G \rangle^{(q^i)} H^\top\right) \geq 2t + 1 - 2s = 2(t-s) + 1 \geq 3.$$

Since H is a matrix over $\mathbb{F}_q$, we get $\mathrm{wt}_{\mathrm{R}}(x) \geq \mathrm{wt}_{\mathrm{R}}(xH)$, and therefore we must have that all elements of rank one must be from a Frobenius power of X. It follows that $\mathcal{U} = \mathrm{supp}_{\mathrm{Gr}}(X)$. □

Theorem 3.19. *[8] Consider a GPT cryptosystem as defined above. Suppose an adversary can find a full rank matrix $U \in \mathbb{F}_q^{s\times n}$ satisfying*

$$\langle U \rangle = \langle X \rangle,$$

then an encrypted message can be recovered in polynomial time.

Proof. Let $H \in \mathbb{F}_q^{(n-s)\times n}$ be a parity check matrix for $\langle U \rangle$. Applying H to the public key generator matrix yields

$$G'H^\top = (SG + X)H^\top = SGH^\top.$$

Then, it follows that $\langle G \rangle H^T$ has minimum rank distance at least $n - k + 1 - s$. Moreover, $GH^\top$ is a Moore matrix.

From the minimum distance we know that there are $n - s$ independent columns in this matrix, which generate a Gabidulin code of minimum distance $n - s - k + 1$, $\mathrm{Gab}_{n-s,k}(\gamma)$, for some $\gamma \in \mathbb{F}_{q^m}^{n-s}$. From the results above, we can recover a decoding algorithm for $\mathrm{Gab}_{n-s,k}(\gamma)$ with respect to the submatrix formed by these $n - s$ columns. The error correction capability of $\mathrm{Gab}_{n-s,k}(\gamma)$ is

$$\left\lfloor \frac{n-s-k}{2} \right\rfloor = \left\lfloor t - \frac{s}{2} \right\rfloor \geq t - s \geq \mathrm{rank}(e) \geq \mathrm{rank}(eH^\top),$$

where the last inequality follows from the fact that H is a matrix over $\mathbb{F}_q$. For an encrypted message $m(SG + X) + e$, we have

$$(m(SG + X) + e)H^\top = mS(GH^\top) + eH^\top.$$

When we restrict this to the above chosen independent columns, we can uniquely decode in the respective code $\mathrm{Gab}_{n-s,k}(\gamma)$ and can therefore recover m. □

We can now use the previous result to attack and break the GPT cryptosystem.

Corollary 3.20. *[8] Consider a GPT cryptosystem as defined above with public key generator matrix $G' = SG + X \in \mathbb{F}_{q^m}^{k\times n}$. For any such cryptosystem, an encrypted message can be recovered in polynomial time.*[7]

[7] The attack needs $O(k^2nm^2(s^2+k))$ operations over $\mathbb{F}_q$, plus the operations needed for the Gabidulin code decoding algorithm. E.g., the decoding algorithm of [24] needs $O(m^3 \log m)$ operations over $\mathbb{F}_q$.

Proof. We first note that

$$\left\lfloor \frac{n-k}{2} \right\rfloor = t > s.$$

By Corollary 3.18, all the elements of rank one in $\sum_{i=0}^{s} \langle G + S^{-1}X \rangle^{(q^i)}$ belong to $\mathrm{supp}_{\mathrm{Gr}}(S^{-1}X) = \mathrm{supp}_{\mathrm{Gr}}(X)$. With Lemma 3.17 we can find a basis matrix $U \in \mathbb{F}_q^{s \times n}$ for these elements of rank one in polynomial time. Then we can use Theorem 3.19 to recover the encrypted message. □

Exercise 3.21. Show that the decryption procedure in Algorithm 7 recovers the message m.

Exercise 3.22. The rank decomposition from Definition 3.14 is generally not unique. For a given rank decomposition, how can you create all other options? Why is the Grassmann support still well defined?

Exercise 3.23. Prove Lemma 3.17.

3.4.2 Other variants in the rank metric

There are many other variants in the rank metric, as already mentioned in the introduction to this chapter. They mostly differ in the choice of the disguising function (when using Gabidulin codes), or in using a similar idea as for LDPC codes, namely LRPC (low rank parity check) codes. We refer the interested reader to [25] and/or the references in Section 3.2 for further information on these systems.

3.5 Other metrics, other alphabets, other cryptosystems (outlook)

We gave a first overview of different variants of public key code-based encryption schemes, using codes and decoding algorithms in the Hamming or the rank metric. Historically, these were also the first two metrics studied in code-based cryptography. However, recently other metrics such as the sum-rank or the Lee metric have been studied for their use in McEliece type cryptosystems and promising first analyses were made. Furthermore, one needs not restrict to finite fields, but can also set up and analyze code-based cryptosystems over other alphabets, e.g., finite rings.

Remember that in the beginning we said that any public key encryption scheme can be transformed into a digital signature scheme. Theoretically, this is also true, but practically both the McEliece and the Niederreiter system suffer from the problem of turning the decoding procedure into a signing procedure. A few attempts have been made, but to no satisfactory result so far. It remains an interesting open question to design a practical code-based signature scheme (that is not of the form below).

Another avenue to design digital signatures is via a well-known transform – called the Fiat-Shamir transform – from zero knowledge identification schemes. There are code-based versions of these schemes and they can (practically) be used to create digital signatures. However, they suffer from large signature sizes, which is a problem that is inherent in the setup with zero knowledge identification schemes. Furthermore, it is questionable if these schemes are

really code-based cryptography, since they do not make use of any decoding algorithms (which is at the heart of coding theory). On the other hand, they rely on the NP-hardness of the syndrome decoding problem and can therefore be called code-based.

Bibliography

[1] N. Aragon, P. Gaborit, A. Hauteville, and J.-P. Tillich. Improvement of generic attacks on the rank syndrome decoding problem. *Preprint, available at* `https://hal.archives-ouvertes.fr/hal-01618464`, 2017.

[2] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Information Theory*, IT-24(3):384–386, 1978.

[3] N. Cai and R. W. Yeung. Network error correction, part 2: Lower bounds. *Communications in Information and Systems*, 6(1):37–54, 2006.

[4] I. Cascudo, R. Cramer, D. Mirandola, and G. Zémor. Squares of random linear codes. *IEEE Transactions on Information Theory*, 61(3):1159–1173, 2015.

[5] A. Couvreur, P. Gaborit, V. Gauthier-Umaña, A. Otmani, and J.-P. Tillich. Distinguisher-based attacks on public-key cryptosystems using Reed–Solomon codes. *Des. Codes Cryptogr.*, 73:641–666, 2014.

[6] E. M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and their application in cryptology. In *Advances in Cryptology, EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 482–489. Springer Berlin Heidelberg, 1991.

[7] V. D. Goppa. Algebraic-geometric codes. *Izv. Akad. Nauk SSSR Ser. Mat.*, 46(4):762–781, 896, 1982.

[8] A.-L. Horlemann-Trautmann, K. Marshall, and J. Rosenthal. Considerations for rank-based cryptosystems. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT) 2016*, July 2016.

[9] A.-L. Horlemann-Trautmann, K. Marshall, and J. Rosenthal. Extension of Overbeck's attack for Gabidulin based cryptosystems. *Des. Codes Cryptogr.*, 86:319–340, 2018.

[10] Y. X. Li, R. H. Deng, and X. M. Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1):271–273, 1994.

[11] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and their Applications*. Cambridge University Press, Cambridge, London, 1994. Revised edition.

[12] P. Loidreau. A new rank metric codes based encryption scheme. In *Proceedings of International Workshop on Post-Quantum Cryptography*, pages 3–17, 2017.

[13] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North Holland, Amsterdam, 1977.

[14] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical report, DSN Progress report # 42-44, Jet Propulsion Laboratory, Pasadena, Californila, 1978.

[15] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. SLM Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 2069–2073. IEEE, 2013.

[16] C. Monico, J. Rosenthal, and A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. In *Proceedings of the 2000 IEEE International Symposium on Information Theory*, page 215, Sorrento, Italy, 2000.

[17] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory 15*, 1(6):159–166, 1986.

[18] R. Overbeck. Structural attacks for public key cryptosystems based on Gabidulin codes. *J. Cryptology*, 21(2):280–301, 2008.

[19] O. Ruatta P. Gaborit and J. Schrek. On the complexity of the rank syndrome decoding problem. *IEEE Transactions on Information Theory*, 62(2):1006–1019, 2016.

[20] O. Ruatta P. Gaborit, G. Murat and G. Zémor. Low rank parity check codes and their application to cryptography. In *Proceedings of the Workshop on Coding and Cryptography WCC*, 2013.

[21] J. Renner S. Puchinger and A. Wachter-Zeh. Twisted Gabidulin codes in the GPT cryptosystem. In *Proceedings of International Workshop on Algebraic and Combinatorial Coding Theory (ACCT)*, 2018.

[22] M. Schwartz. *Information Transmission, Modulation, and Noise: A Unified Approach to Communication Systems*. McGraw-Hill, 1980.

[23] V. M. Sidelnikov and S. O. Shestakov. On an encoding system constructed on the basis of generalized Reed-Solomon codes. *Diskret. Mat.*, 4(3):57–63, 1992.

[24] A. Wachter-Zeh, V. Afanassiev, and V. Sidorenko. Fast decoding of Gabidulin codes. *Designs, Codes and Cryptography*, 66(1):57–73, 2013.

[25] V. Weger, N. Gassner, and J. Rosenthal. A survey on code-based cryptography. *https://arxiv.org/abs/2201.07119*, 2022.

[26] C. Wieschebrink. Two NP-complete problems in coding theory with an application in code based cryptography. In *2006 IEEE International Symposium on Information Theory*, pages 1733–1737, 2006.

[27] C. Wieschebrink. Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes. In *PQCrypto 2010, Lecture Notes in Computer Science*, volume 6061, page 61–72, 2010.