FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Zuzana Parýzková

Cryptosystems based on coding theory

Department of Algebra

Supervisor of the master thesis:	doc. Mgr. et Mgr. Jan Žemlička, Ph.D.
Study programme:	Mathematics
Study branch:	Mathematics for Information Technologies

Prague 2023

I would like to thank my supervisor doc. Mgr. et Mgr. Jan Žemlička, Ph.D. for introducing the topic of cryptography based on coding theory to me. I appreciate the useful consultations and feedback I have been given. I would also like to thank my husband and my whole family for supporting me in my studies

Title: Cryptosystems based on coding theory

Author: Zuzana Parýzková

Department: Department of Algebra

Supervisor: doc. Mgr. et Mgr. Jan Žemlička, Ph.D., Department of Algebra

Abstract: Nowadays public-key cryptosystems such as RSA are threatened by quantum computing. Therefore, a post-quantum standardization process was initiated by NIST in 2017. As of today, several cryptosystems have been selected for standardization and several still remain in the process. A cryptosystem based on coding theory - Classic MeEliece - is one of the cryptosystems that might be standardized. This thesis covers McEliece and Niederreiter cryptosystems as well as their rank-metric variants (GGPT cryptosystem). Sidelnikov-Shestakov's attack is explained in detail and an example of the attack is given. Stern's and Overbeck's attacks are discussed as well. Furthermore, a new polynomial-time attack against GGPT without distortion matrix X is given.

Keywords: coding theory, public-key cryptography, post-quantum cryptography, McEliece cryptosystem

Contents

Introduction

In 1994 Peter Shor introduced a quantum computer algorithm that solves factorization and discrete logarithm problems in polynomial time. There is no known polynomial-time algorithm for either of the problems for classic computers. Therefore, recent public-key cryptography is based on hardness of factorization (RSA) or discrete logarithm (Diffie-Hellman key exchange). Moreover, these cryptosystems are used in NIST (National Institute of Standards and Technology) standards. Of course, the threat of quantum computers is not immediate. For example, NIST suggests using RSA with 2048-bit keys ([NIoST15]). However, nowadays quantum computers are nowhere near to factoring 2048-bit numbers. Of course, we need to be aware of the threat of attackers decrypting nowadays ciphertexts in the future. Thus, in 2017 NIST initiated a post-quantum standardization process (a post-quantum cryptography is a cryptography that resists quantum computing). As of today, one algorithm for public-key encryption and key-establishment and three digital signature algorithms have been selected for standardization. Furthermore, there are more four public-key encryption and key-establishment algorithms submitted for fourth round of the process. It is possible that some of them will be standardized. One of the fourth-round submissions is Classic McEliece cryptosystem ([ABC$^+$]) based on the cryptosystem proposed by Robert J. McEliece in 1978 ([McE78]).

In this master thesis we discuss different versions of the original McEliece cryptosystem from 1978 and several attacks against them. The cryptosystem is based on theory of error correcting codes. It takes advantage of general decoding problem being NP-hard. The cryptosystem uses a linear error correcting code for which an efficient decoding algorithm is known. The legitimate user keeps a secret generator matrix of the code which gives them the power to use the efficient decoding algorithm. On the other hand, an attacker knows only the public key, i.e., scrambled generator matrix of a different code, which leaves them with solving the general decoding problem.

In Chapter 1 we give necessary preliminaries regarding cryptography and error-correcting codes. Then in Chapter 2 we define the original McEliece cryptosystem as well as the somewhat dual Niederreiter cryptosystem that was proposed by Harrald Niederreiter in [Nie86] in 1986.

Chapter 3 deals with algorithm for finding low weight codeword in linear binary error correcting code. The algorithm was proposed by Jacques Stern in 1988 in [Ste88]. We also show how to use the algorithm to construct an attack against McEliece and Niederreiter cryptosystems. That was proposed in 2008 by Daniel J. Bernstein, Tanja Lange and Christiane Peters. They also proposed a newer version of the algorithm and stated that it could break McEliece cryptosystem with the original parameters from 1978 in one week.

The original McEliece proposal used Goppa codes, whereas the original Niederreiter proposal used GRS codes. The latter was proved insecure in 1992. V. M. Sidelnikov and S. O. Shestakov described a polynomial-time attack against it in [SS92]. On the other hand, using Goppa codes has not been proved vulnerable. In Chapter 4 we prove correctness of Sidelnikov-Shestakov's attack as well as show how to use it against McEliece cryptosystem using GRS codes. Furthermore, we give an example of the attack computed by a SageMath program the code of which can be found in Appendix A.1.

All cryptosystems and attacks that have so far been mentioned use error-correcting codes with Hamming metric. In Chapter 5 we focus on GPT cryptosystem using Gabidulin codes. These codes use rank metric and were proposed by Ernst M. Gabidulin

1985. Later in 1991 E. M. Gabidulin together with A. V. Paramonov and O. V. Tretjakov proposed GPT cryptosystem. The cryptosystem takes advantage of same principles as the original McEliece one. There have been several attacks found against the cryptosystem which resulted in newer version called GGPT proposed by Raphael Overbeck in 2005. In this chapter we give the necessary background of Gabidulin codes and rank metric. Then we define GGPT cryptosystem and its corresponding Niederreiter version. We briefly describe Overbeck's attack from [Ove05]. In the last section we introduce a new attack against GGPT cryptosystem without distortion matrix X. We also pinpoint similarities between GGPT without distortion matrix X and Niederreiter cryptosystem using GRS codes.

1. Preliminaries

1.1 Used Notation

If not stated otherwise, let throughout the thesis $n, m, k \in \mathbb{N}$ and p be a prime.

Notation used in the thesis:

- $\mathbb{P}$ denotes the set of prime numbers.
- $\mathbb{F}_p$ denotes the finite field with p elements.
- $\mathbb{F}_{p^m}$ denotes the finite field with p^m elements with characteristic p.
- $\mathbb{F}_p^n$ denotes the n-dimensional vector space over $\mathbb{F}_p$.
- $\underline{v}$ denotes a row vector. That is the usual approach in the theory of error correcting codes (ECC).
- v_i denotes the i-th coordinate of a vector $\underline{v}$.
- $\underline{o}$ denotes the row zero vector (its size will always be properly stated).
- $\underline{1}$ denotes the row vector $(1, \ldots, 1)$, its size will always be properly stated as well.
- Given a vector $\underline{v} \in \mathbb{F}^n$ and a set of indices $I = (i_1, \ldots, i_k) \subseteq \{1, n, \ldots,\}$ then $\underline{v}_{\restriction I} = (v_{i_1}, \ldots, v_{i_k})$.
- $\mathcal{C}_G$ denotes the linear error correcting code generated by a matrix G.
- $d(\mathcal{C}_G)$ denotes the minimum distance of the code $\mathcal{C}_G$.
- A^{-1} denotes the inverse matrix of a square nonsingular matrix A.
- A_{*i} denotes the j-th column of a matrix A. Similarly $A_{*\{1,\ldots,j\}}$ denotes the submatrix formed by first j columns of a matrix A.
- In the same manner A_{i*} denotes the i-th row of a matrix A and $A_{\{1,\ldots,i\}*}$ the submatrix formed by first i rows of a matrix A.
- Similarly, combining the notation, we denote by $A_{\{\{1,i,\ldots,\}\}\{\{1,j,\ldots,\}\}}$ the submatrix of A formed by its first i rows and j columns.
- The linear span of column vectors $(\underline{v_1}^\top, \ldots, \underline{v_n}^\top)$ over finite field $\mathbb{F}$ is denoted by $\operatorname{Span}\left(\underline{v_1}^\top, \ldots, \underline{v_n}^\top\right) = \{\sum_{i=1}^n \lambda_i v_i \mid \lambda_i \in \mathbb{F}\}$.
- Similarly, the column space of a matrix A is denoted by $\operatorname{Span}(A)$.

1.2 Cryptosystems

In this thesis we examine multiple cryptosystems based on error correcting codes. We first give the definition of a cryptosystem.

Definition 1. A cryptosystem $\mathcal{C}$ is defined as a quintuplet $(\mathcal{P}, \mathcal{C}, \mathcal{K}, e, d)$ where

$$\begin{aligned} &\mathcal{P}_P \text{ is set of all possible plaintexts,}\\ &\mathcal{P}_C \text{ is set of all possible ciphertext,}\\ &\mathcal{K} \text{ is set of all possible keys,}\\ \text{Enc}:\ &\mathcal{P}_P \times \mathcal{K} \to \mathcal{P}_C \text{ is the encryption mapping}\\ \text{and Dec}:\ &\mathcal{P}_C \times \mathcal{K} \to \mathcal{P}_P \text{ is the decryption mapping.} \end{aligned}$$

Furthermore, the following holds

$$\forall x \in \mathcal{P}_P\ \forall k \in \mathcal{K}\ \exists \tilde{k} \in \mathcal{K}:\ \text{Dec}_{\tilde{k}}((\text{Enc}_k(x)) = x.$$

We further distinguish public-key (asymmetric) cryptosystems and symmetric cryptosystems. In this thesis we will focus on public-key cryptosystems (PKCs).

A public-key cryptosystem has public key $\mathcal{K}_{pub}$ and private key $\mathcal{K}_{priv}$. Public key is distributed and can be easily computed from private key. On the other hand, private key is kept secret and it is computationally impossible to obtain private key from public key. Public key is used for encrypting and private key for decrypting.

If quantum computers were powerful enough they could factor large numbers, which would threaten PKCs currently used (such as RSA). The numbers used in nowadays RSA PKC are usually 4096 bit. Nowadays quantum computers can factor much smaller numbers, thus the threat is not real yet. However, quantum computers might one day in future decrypt messages send these days. Therefore, in order to be prepared NIST has in 2017 started standardization process for post-quantum cryptography (PQC) Standard. Note that PQC is cryptography that resists quantum computers. One of the round 4 submissions is Classic McEliece (see [ABC$^+$]), a PKC based on McEliece PKC. The original McEliece's proposal will be described in Chapter 2.

1.3 Basic Properties of Error Correcting Codes

As McEliece PKC is based on error correcting codes (ECC), we shall first give basic properties of ECC. We will namely work with linear codes. In what follows we will by code always mean linear error correcting code. Such codes form a vector space over the corresponding finite field and have a generator and parity-check matrix. Furthermore, every linear code is described by a triple (n, k, d) where n stands for the length of codewords, k is the dimension of the code, d is the minimal distance of the code. If d is not important in the moment, we shall omit it and write (n, k)-code instead.

Namely, let $\mathbb{F}$ be an arbitrary finite field. An (n, k, d)-code $\mathcal{C}$over $\mathbb{F}$ is a k-dimensional subspace of $\mathbb{F}^n$. Its generator matrix is any matrix $G \in \mathbb{F}^{k\times n}$ such that

$$\begin{aligned} \forall \underline{c} \in \mathcal{C}\ \exists \underline{m} \in \mathbb{F}^k:\ \underline{m}G = \underline{c} \text{ and}\\ \text{Rank}\,(G) = k. \end{aligned}$$

On the other hand, its parity-check matrix is any matrix $H \in \mathbb{F}^{n-k \times n}$ such that

$$\begin{aligned} \forall \underline{c} \in \mathbb{F}^n : \ \underline{c}H^\top = \underline{o} \iff \underline{c} \in \mathcal{C} \text{ and} \\ \operatorname{Rank}(H) = n - k. \end{aligned}$$

The minimal distance of the code is defined with respect to a given metric. In this thesis we will use Hamming metric and Rank metric. The latter will be defined in Section 5.1.1.

Definition 2. The Hamming distance of two vectors $\underline{v}$, $\underline{u} \in \mathbb{F}^n$ is defined as the number of entries in which they differ. It will be denoted by $d_H(\underline{v}, \underline{u},)$. Similarly, $w_H(\underline{v}) := d_H(\underline{v}, \underline{o},)$ denotes the Hamming weight of $\underline{v}$.

Remark. It can be easily verified that Hamming distance is a metric.

The minimal distance of a code is bounded by Singleton bound.

Theorem 1. *Let $\mathcal{C}$ be an (n, k, d)-code. Then $d \leq n - k + 1$.*

ECC are used as a reliable method to transfer data over channels with error. Every code can correct number of errors. However, if the number of errors exceeds the limitation of the code, the transmitted message cannot be corrected.

Theorem 2. *[Mac77, p. 10] An $(n, k, d)-$code can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors.*

After receiving the encoded message, we wish do decode it to obtain the original one. There is always the brute-force way but we wish to have an efficient algorithm.

Definition 3. Let us have an (n, k)-code $\mathcal{C}$ over an arbitrary finite field $\mathbb{F}$ with a generator matrix G such that it can correct up to to t errors. We say that $\mathcal{D}_\mathcal{C}$ is an efficient decoding algorithm of $\mathcal{C}$ if it runs in polynomial time and

$$\forall \underline{m} \in \mathbb{F}_q^k \ \forall \underline{e} \in \mathbb{F}_q^n \text{ such that } w_H(\underline{e}) \leq t : \ \mathcal{D}_\mathcal{C}(\underline{m}G + \underline{e}) = \underline{m}.$$

Definition 4. Let $\underline{c}$ be a codeword in code with parity-check matrix H. Let $\underline{e}$ be an error correctable in the code. Furthermore, assume that $\underline{\tilde{c}} = \underline{c} + \underline{e}$. Then

$$\underline{\tilde{c}}H^\top = \underline{c} + \underline{e}H^\top = \underline{c}H^\top + \underline{e}H^\top = \underline{o} + \underline{e}H^\top$$

is called syndrome of $\underline{\tilde{c}}$.

Definition 5. Let $\mathcal{C}$ be an (n, k, d)-code that can correct up to t errors. Let H be its parity-check matrix. A syndrome decoding algorithm is an algorithm that upon input $\underline{s}$ returns

- $\underline{e}$ such that $\underline{s} = \underline{e}H^\top$ and $w_H(\underline{e}) \leq t$ if such $\underline{e}$ exists,
- *fail* otherwise.

The algorithm will be denoted by $\mathcal{S}_H$. An efficient syndrome decoding algorithm is such that runs in polynomial time.

Remark. A syndrome decoding algorithm has its role in decoding of the received message. Suppose that we have received $\underline{\tilde{c}} = \underline{c} + \underline{e}$, a corrupted codeword of some linear error-correcting code with generator matrix G and parity-check matrix H. We first multiply the word by the parity-check matrix

$$\underline{\tilde{c}}H^\top = \underline{c}H^\top + \underline{e}H^\top = \underline{e}H^\top.$$

Suppose that upon calling the syndrome decoding algorithm, we obtain $\underline{e}$ (i.e. we do not obtain *fail*). Now we can easily compute the original message as

$$(\underline{\tilde{c}} - \underline{e})\tilde{G}$$

where $\tilde{G}$ is a right inverse to G.

2. Original McEliece and Niederreiter PKC

The McEliece Public-Key Cryptosystem was originally introduced by Robert J. McEliece in [McE78]. It is a PKC based on coding theory originally using Goppa Codes. Over the years there have been many proposal using same scheme but different codes. However, Goppa codes still remain most secure variant.

In [Nie86] Harald G. Niederreiter proposed a PKC similar to the McEliece one. The Niederreiter cryptosystem is somewhat dual to the McEliece PKC and will be described later in the chapter. Both cryptosystems use codes with Hamming weight.

We will omit the definition and construction of Goppa codes and focus rather on the general properties of the cryptosystem.

2.1 Description of McEliece PKC

We defined the cryptosystem independently of the used code.

Parameters

- Choose $p \in \mathbb{P}$ and $m, n, t \in \mathbb{N}$ such that $t << n$.

Key generation

1. Choose $G_{priv} \in \mathbb{F}_{p^m}^{k \times n}$ to be a generator matrix of a linear (n, k)-code over $\mathbb{F}_{p^m}$ such that it can correct up to t errors and that there exists an efficient decoding algorithm $\mathcal{D}_{G_{priv}}$. Here k is chosen to be maximal given n, t.

2. Choose random nonsingular matrix $S \in \mathbb{F}_{p^m}^{k \times k}$ and a random permutation matrix $P \in \mathbb{F}_{p^m}^{n \times n}$.

3. Compute $G_{pub} := S G_{priv} P$.

4. Let
$$\mathcal{K}_{pub} = (G_{pub}, t)$$
be a public key and
$$\mathcal{K}_{priv} = (G_{priv}, S, P)$$
be a private key.

Encryption The sending party generates a random vector $\underline{e} \in \mathbb{F}_{p^m}^n$ such that $w_H(\underline{e}) = t$ and then encrypts a plaintext $\underline{m} \in \mathbb{F}_{p^m}^k$ as

$$\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{m}) = \underline{m} G_{pub} + \underline{e}.$$

Decryption After receiving the ciphertext $\underline{c}$, the legitimate user decrypts it as

$$\mathrm{Dec}_{\mathcal{K}_{priv}}(\underline{c}) = \left(\mathcal{D}_{G_{priv}}\left(\underline{c}P^{-1}\right)\right) S^{-1}.$$

Theorem 3. *Using the notation as above,* $\mathrm{Dec}_{\mathcal{K}_{priv}}(\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{m})) = \underline{m}$.

Proof. Let us denote $\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{m})$ by $\underline{c}$. We know that $\exists \underline{e} \in \mathbb{F}_{p^m}^n,\ w_H(\underline{e}) = t$ such that

$$\begin{aligned} \underline{c} &= \underline{m}G_{pub} + \underline{e} \\ &= \underline{m}SG_{priv}P + \underline{e}. \end{aligned}$$

Thus,

$$\underline{c}P^{-1} = \underline{m}SG_{priv} + \underline{e}P^{-1}.$$

Note that $w_H(\underline{e}) = w_H(\underline{e}P^{-1})$, therefore $\mathcal{D}_{G_{priv}}(\underline{c}P^{-1}) = \underline{m}S$. Multiplying the right-hand side by S^{-1} yields $\underline{m}$. □

2.2 Security of McEliece PKC

The McEliece scheme is based on the fact that General Decoding Problem is NP-complete (proved in [BMvT78]).

Definition 6 (General Decoding Problem [BMvT78])**.** Let C be an (n, k)-binary linear code. Let $\underline{y}$ be the received word, H be an $(n-k) \times n$ parity-check matrix and $\underline{y}H^\top$ be the syndrome of $\underline{y}$. Find the minimum-weight $\underline{z_0}$ such that $\underline{y}H^\top = \underline{z_0}H^\top$.

If the given linear code is not an arbitrary code, but has some useful structure, there are known efficient decoding algorithms. An example of such codes are alternant codes or GRS codes.

Theorem 4. *[Gab01, SEction 2.2] There are known efficient decoding algorithms for GRS codes.*

Definition 7. [Gab01, p. 20] Let C be an (n, k)-GRS (Generalized Reed-Solomon) code over $\mathbb{F}_{p^m}$. Consider the subcode C_a consisting of all codewords with all coordinates in the base field $\mathbb{F}_p$. The code C_a is an alternant code.

There are mainly two algorithms for efficient decoding: Peterson algorithm and Berlekamp-Massey algorithm.

Example of alternant codes are Goppa codes ([Gab01, p. 19]). Interestingly, binary Goppa codes give us better minimal distance then general Goppa codes (see [Gab01, p. 23-24]). That is one of the reasons we choose (irreducible) binary Goppa codes to use in the McEliece scheme. Even the Classic McEliece ([ABC$^+$]), a candidate for NIST PQC standardization, is based on binary Goppa codes.

After receiving the ciphertext $\underline{c} = \underline{m}G_{pub} + \underline{e}$, the legitimate user possessing the secret matrix G_{priv} can easily and efficiently compute the original plaintext $\underline{m}$. On the other hand, the adversary knowing only the matrix G_{pub} has very little information about the used code and has to solve general decoding problem to recover the plaintext.

2.3 Further Comments on McEliece PKC

In the McEliece PKC we multiply the private matrix G_{priv} from left by a nonsingular matrix S and from right by a permutation matrix P. We will now discuss the importance and cryptographic impact of both multiplications. This section is inspired by [EOS06, p. 15].

2.3.1 Matrix S

Let us first focus on the nonsingular matrix S. This multiplication obviously has no impact on the generated code, because matrices G_{priv} and SG_{priv} generate the same error-correcting code.

Theorem 5. *Let $G_{priv} \in \mathbb{F}_{p^m}^{k\times n}$ and let $S \in \mathbb{F}_{p^m}^{k\times k}$ be a nonsingular matrix. Then matrices G_{priv} and SG_{priv} generate the same error correcting code.*

Proof. Suppose $\underline{c} = \underline{x}SG_{priv}$ is a codeword in $\mathcal{C}_{SG_{priv}}$, where $\underline{x} \in \mathbb{F}_{p^m}^k$. Then obviously $\underline{c} = (\underline{x}S)G_{priv}$ is also a codeword in $\mathcal{C}_{G_{priv}}$.

On the other hand, assume that $\underline{c'} = \underline{y}G_{priv}$ is a codeword in $\mathcal{C}_{G_{priv}}$, where $\underline{y} \in \mathbb{F}_{p^m}^k$. Matrix S is nonsigular, therefore there exists $\underline{z} \in \mathbb{F}_{p^m}^k$ such that $\underline{z} = \underline{y}S^{-1}$. Now we see that $\underline{c} = \underline{y}SG_{priv}$ is also a codeword in $\mathcal{C}_{SG_{priv}}$. □

We might therefore think that the multiplication by S is not necessary. But imagine if the generator matrix G was in standard form.

Definition 8. [Rom96, p. 201] A $k \times n$ matrix G is in (left) standard form if it has the form

$$G = \begin{pmatrix} I_k & \tilde{G} \end{pmatrix}$$

where I_k is an identity matrix of size k and $\tilde{G}$ is a matrix of size $k \times (n-k)$

If it was the case, the encrypted text would be

$$\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{m}) = \underline{m}G_{pub} + \underline{e} = (m_1, m_2, \ldots, m_k, \tilde{g}_{k+1}, \ldots, \tilde{g}_n) + \underline{e}.$$

Where $\underline{m} = (m_1, \ldots, m_k)$ is the plaintext the user wishes to securely transfer and $\tilde{g}_{k+1}, \ldots, \tilde{g}_n$ are arbitrary (for our purposes) values. Adding $\underline{e}$ changes only $t << k$ values. Note that $t = \lfloor \frac{d-1}{2} \rfloor \leq \frac{n-k}{2}$. If $n < 3k$ then $\frac{n-k}{2} < k$. Thus, if $n < 3k$, using matrix in standard form reveals at least one position of the plaintext.

Of course, the matrix could be in somewhat ”nearly standard form” and might reveal part of the plaintext too. To avoid such threats, we multiply the original generator matrix by random nonsingular matrix. This way we do not change the code and its properties, but mask unuseful properties of the matrix G_{priv}.

McEliece ([McE78]) suggests the nonsingular matrix S to be dense. Intuitively multiplying by dense matrix hides more information then multiplying by sparse matrix.

2.3.2 Not Every Dense Matrix Is a Good Matrix

Let G be a generator matrix in standard form of an (n, k)-code. Let T be a nonsingular dense matrix. Suppose that the legitimate user generates a private matrix $G_{priv} = TG$. Further assume that the user generates a nonsingular dense matrix S such that $S = T^{-1}$. The user then publishes what they think is a scrambled generator matrix $G_{pub} = STG = G$. The encryption now reveals first k indices of the plaintext.

2.3.3 Matrix P

The permutation matrix P is the one that is cryptographically more important. Although mathematically, codes generated by G_{priv} and $G_{priv}P$ are by definition equivalent (which means they differ only in the order of the symbols - see [Mac77, p. 24]). The purpose of this multiplication is to change the code so that the attacker cannot recover the plaintext.

2.3.4 Without Matrix P

Let us investigate what would happen if there were no matrix P. Or equivalently, imagine what would happen if $P = I_n$.

Let G_{priv} be a generator matrix of an (n, k)-binary Goppa code, $G_{pub} = SG_{priv}I_n$ be the scrambled generator matrix. Let $\underline{m}$ be the plaintext and $\underline{c} = \underline{m}G_{pub} + \underline{e}$ be the ciphertext.

The adversary can easily use one of the fast decoding algorithms because as we know, the code generated by G_{priv} is the same as the one generated by SG_{priv}. Thus, the adversary has the generator matrix of the right code.

2.4 Description of Niederreiter PKC

In this section we will introduce the Niederreiter PKC. As mentioned before, it resembles the McEliece one. However, instead of using generator matrix we use a parity-check matrix to encrypt an error vector. Hence, ciphertexts are syndromes of an error correcting code.

The original Niederreiter's proposal ([Nie86]) used Generalized Reed Solomon (GRS) codes instead of Goppa codes. Unfortunately, using GRS codes was proved to be insecure in 1992. In [SS92] V. M. Sidelnikov and S. O. Shestakov described an algorithm that breaks the cryptosystem in polynomial time. The attack is described in Chapter 4.1. When used with Goppa codes, the Niederreiter scheme is believed to be secure. In fact, in this case the security of McEliece and Niederreiter cryptosystems is equivalent ([EOS06, p. 8]).

Parameters

- Choose $p \in \mathbb{P}$ and $m, n, t \in \mathbb{N}$ such that $t << n$.

Key generation

1. Choose $H_{priv} \in \mathbb{F}_{p^m}^{(n-k)\times n}$ to be a parity-check matrix of a linear (n, k)-code over $\mathbb{F}_{p^m}$ such that it can correct up to t errors and that there exists an efficient syndrome decoding algorithm $\mathcal{S}_{H_{priv}}$. Again, k is chosen to be maximal given n, t.

2. Choose random nonsingular matrix $M \in \mathbb{F}_{p^m}^{(n-k)\times(n-k)}$ and a random permutation matrix $P \in \mathbb{F}_{p^m}^{n\times n}$.

3. Compute $H_{pub} := MH_{priv}P$.

4. Let
$$\mathcal{K}_{pub} = (H_{pub}, t)$$
and
$$\mathcal{K}_{priv} = (M, H_{priv}, P)$$
be a public and a private key, respectively.

Encryption The sending party wishing to securely send a plaintext $\underline{e} \in \mathbb{F}_{p^m}^n$ of weight t sends

$$\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{e}) = \underline{e}H_{pub}^{\top}.$$

Decryption Upon receiving the ciphertext $\underline{c}$, the legitimate user decrypts it as

$$\mathrm{Dec}_{\mathcal{K}_{priv}}(\underline{c}) = \left(\mathcal{S}_{H_{priv}}\left(\underline{c}\cdot\left(M^{\top}\right)^{-1}\right)\right)\cdot\left(P^{\top}\right)^{-1}.$$

Theorem 6. *The decryption always returns the original ciphertext, i.e.*

$$\mathrm{Dec}_{\mathcal{K}_{priv}}(\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{e})) = \underline{e}.$$

Proof. Let us denote $\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{e})$ by $\underline{c}$. We know that

$$\underline{c} = \underline{e}H_{pub}^{\top}$$

and

$$H_{pub} = MH_{priv}P.$$

Thus,

$$\begin{aligned}\underline{c}\left(M^{\top}\right)^{-1} &= \underline{e}H_{pub}^{\top}\left(M^{\top}\right)^{-1}\\ &= \underline{e}P^{\top}H_{priv}^{\top},\\ \mathcal{S}_{H_{priv}}(\underline{e}P^{\top}H_{priv}^{\top}) &= \underline{e}P^{\top}.\end{aligned}$$

Multiplying $\underline{e}P^{\top}$ by $\left(P^{\top}\right)^{-1}$ gives us the desired $\underline{e}$. □

Remark. We could wonder why the plaintext needs to have Hamming weight equal to t. For the encryption and decryption to work it is sufficient to have $w_H(\underline{e}) \leq t$. However, if the weight was too small, there would be an easy brute-force attack. Imagine the extreme case when $w_H(\underline{e}) = 1$. The attacker only needs to go through n possible plaintexts and check if they equal the ciphertext. Thus, we want to use the maximal possible weight. Of course, if $w_H(\underline{e}) = n - 1$, there would similarly be an easy attack. However, the Hamming weight of $\underline{e}$ is bounded by t which is significantly smaller then n.

3. Stern's Attack

In 1989 Jacques Stern proposed a probabilistic algorithm that finds a codeword of small weight in a given linear binary code. We will briefly comment the algorithm that can be found in [Ste88]. Afterwards, we show how to use the algorithm for breaking special cases of McEliece and Niederreiter scheme. That part will be based on [BLP08].

3.1 Stern's Algorithm

Let $\mathcal{C}_G$ be a linear binary (n, k, d)-code with a parity-check matrix $H \in \mathbb{F}_q^{(n-k)\times n}$. Let $\omega \in \mathbb{N}$ be the desired small weight. Then the algorithm takes (H, ω) as input and outputs a codeword $\underline{x} \in \mathcal{C}_G$ such that $w_H(\underline{x}) = \omega$. The algorithm takes two parameters p and l.

Algorithm 1: Basic Stern's Attack

```
   Input: $H \in \mathbb{F}_2^{(n-k)\times n}$, $\omega \in \mathbb{N}$
   Parameters: $p \leq \frac{\omega}{2}$, $l \leq n-k$
   Output: $\underline{x} \in \mathbb{F}_2^n$ such that $\underline{x}H^\top = \underline{o}$ and $w_H(\underline{x}) = \omega$
 1 $K := \emptyset, M := \emptyset, i := 1$
 2 Set $H' := H$, while $|K| < n-k$ do
 3     randomly choose $m_i \in \{1, \dots, n\} \setminus M$,
 4     if $\exists k_i \in \{1, \dots, n-k\} \setminus K$ such that $H_{k_i m_i}$ is a pivot then
 5         $M = M \cup \{m_i\}$,
 6         $K = K \cup \{k_i\}$,
 7         find the row equivalent matrix of $H'$ with the $m_i$-th column of weight
             equal to 1,
 8         denote the row equivalent matrix by $H'$,
 9         i := i+1.
10 Set $I := \{1, \dots, n\} \setminus M, X := \emptyset, Y := \emptyset$.
11 for $i \in I$ do
12     put the index $i$ randomly (uniformly) and independently either into $X$ or $Y$.
13 Choose randomly a set of row indices $J \subseteq \{1, \dots, n-k\}$ such that $|J| = l$.
14 for all $A \subseteq X$ and all $B \subseteq Y$ such that $|A| = |B| = p$ do
15     $\pi(A) := H'_{JA} \cdot \underline{1}^\top$,
16     $\pi(B) := H'_{JB} \cdot \underline{1}^\top$.
17 for all pairs $A, B$ do
18     if $\pi(A) = \pi(B)$ then
19         $V := H'_{*A\cup B} \cdot \underline{1}^\top \in \mathbb{F}_2^{n-k}$
20         if $w_H(V) = \omega - 2p$ then
21             define $\underline{y} \in \mathbb{F}_2^n$ such that $\forall m \in \{1, \dots, n\} : y_m = 1 \iff m \in A \cup B$
22             define $\underline{z} \in \mathbb{F}_2^n$ such that $\forall m \in \{1, \dots, n\} : z_m = 1 \iff$
                 $(\exists i \in \{1, \dots, n\}$ such that $m = m_i \in M$ and $V_{k_i} = 1)$
23             return $\underline{x} = \underline{y} + \underline{z} \in \mathbb{F}_2^n$
24 return failure
```

In what follows we will use the notation used in Algorithm 1.

Observation. *We assume* $\operatorname{rank}(H) = n - k$, *thus the while loop terminates at last. The algorithm always stops after finitely many steps. Furthermore, if H is a parity-check matrix of an MDS code, then any $n-k$ columns are linearly independent, thus the Gauss-Jordan elimination always succeeds.*

Lemma 7. *If $\underline{x} \in \mathcal{C}_G$ then $\underline{x}(H')^\top = \underline{o}$ for every H' we obtain in the algorithm.*

Proof. Any matrix H' was obtained from H by elementary row operations, thus there exists a nonsingular matrix $R \in \mathbb{F}_2^{(n-k)\times(n-k)}$ such that $H' = RH$. Thus, $\underline{x}(H')^\top = \underline{x}H^\top R^\top = \underline{o}$. □

Lemma 8. *[Ste88, proposition 1] If the algorithm returns $\underline{x} \in \mathbb{F}_2^n$, then $\underline{x} \in \mathcal{C}_G$ and $w_H(\underline{x}) = \omega$.*

Proof. We first prove that $\underline{x}$ returned by the algorithm is a codeword of $\mathcal{C}_G$. Let $\underline{x} = \underline{y} + \underline{z}$ be as defined in Steps 21-22 of the algorithm. It is clear that

$$\underline{x}H'^\top = \underline{y}H'^\top + \underline{z}H'^\top.$$

From definition of $\underline{y}$ we obtain that

$$\sum_{m=1}^{n} y_m \cdot (H')_{*m} = V.$$

Similarly, it is easily seen that

$$\sum_{m=1}^{n} z_m \cdot (H')_{*m} = V.$$

Altogether $\underline{x}H'^\top = \underline{o}$. Lemma 7 then implies that $\underline{x} \in \mathcal{C}_G$.

Obviously,

$$w_H(\underline{x}) = w_H\left(\underline{y}\right) + w_H(\underline{z}) - |\{m \in \{1,\dots,n\} |\ y_m = 1 = z_m\}|.$$

From definition of $\underline{y}$ and $\underline{z}$ we see that

$$\begin{aligned} w_H\left(\underline{y}\right) &= |A \cup B| = 2p, \\ w_H(\underline{z}) &= w_H(V) = \omega - 2p. \end{aligned}$$

Furthermore, assume that $m \in \{1,\dots,n\} :\ y_m = 1 = z_m$. Then again by definition we have

$$\begin{aligned} y_m = 1 &\iff m \in A \cup B \Rightarrow m \notin M, \\ z_m = 1 &\iff \exists i \in \{1,\dots,n\} :\ m = m_i \text{ and } V_{k_i} = 1 \Rightarrow m \in M. \end{aligned}$$

Thus, $|\{m \in \{1,\dots,n\} |\ y_m = 1 = z_m\}| = 0$ and $w_H(\underline{x}) = \omega$. □

Theorem 9. *[Ste88, proposition 2] Let $\underline{\tilde{x}} \in \mathcal{C}_G$ and J, X, Y be the sets of indices chosen in Steps 11-13 of Algorithm 1. If $\exists \tilde{A} \subseteq X,\ |\tilde{A}| = p$ and $\exists \tilde{B} \subseteq Y,\ |\tilde{B}| = p$ such that*

(P.1) $w_H(\underline{\tilde{x}}) = \omega$,

(P.2) $\forall i \in X: \left(\underline{\tilde{x}}_i = 1 \iff i \in \tilde{A}\right)$,

(P.3) $\forall i \in Y: \left(\underline{\tilde{x}}_i = 1 \iff i \in \tilde{B}\right)$,

(P.4) $\forall m_i \in N_J: \ \tilde{x}_{m_i} = 0$ *where* $N_J := \{m_i \mid i \in \{1, \ldots, n-k\}, k_i \in J\}$

then Algorithm 1 succeeds.

Proof. Assume that we are in Step 17 of the algorithm and

$$\begin{aligned} A &= \tilde{A}, \\ B &= \tilde{B}. \end{aligned}$$

We will show that then the algorithm return $\underline{\tilde{x}}$. Note that the algorithm might return different codeword if it finds it first. However, the important thing is that the algorithm succeeds.

Let us first show the algorithm proceeds into Step 19. Or equivalently that $\pi(\tilde{A}) = \pi(\tilde{B})$. We know that $\underline{\tilde{x}} H^\top = \underline{\tilde{x}} \left(H'\right)^\top = \underline{o}$ due to Lemma 7. Thus,

$$\sum_{m \in \tilde{A} \cup \tilde{B}} H'_{*m} \tilde{x}_m + \sum_{m \in (X \cup Y) \setminus (\tilde{A} \cup \tilde{B})} H'_{*m} \tilde{x}_m + \sum_{m \notin X \cup Y} H'_{*m} \tilde{x}_m = \underline{o}$$

and we furthermore obtain

$$\begin{aligned} H'_{*\tilde{A}\cup\tilde{B}} \underline{1}^\top &= \sum_{m \in \tilde{A} \cup \tilde{B}} H'_{*m} \tilde{x}_m \\ &= \sum_{m \notin X \cup Y} H'_{*m} \tilde{x}_m + \sum_{m \in (X \cup Y) \setminus (\tilde{A} \cup \tilde{B})} H'_{*m} \tilde{x}_m \\ &= \sum_{m \notin X \cup Y} H'_{*m} \tilde{x}_m + \underline{o} \\ &= \sum_{i=1}^{n-k} H'_{*m_i} \tilde{x}_{m_i} \end{aligned}$$

The first equality holds due to (P.2) and (P.3) and the last equality is just reindexing of the sum.

It is sufficient to show that

$$\begin{aligned} \pi(\tilde{A}) + \pi(\tilde{B}) &= H_{J\tilde{A}} \cdot \underline{1}^\top + H_{J\tilde{B}} \cdot \underline{1}^\top \\ &= H_{J\tilde{A}\cup\tilde{B}} \cdot \underline{1}^\top \\ &= \sum_{i=1}^{n-k} H'_{Jm_i} \tilde{x}_{m_i} \\ &= \underline{o}. \end{aligned}$$

The last equality holds due to (P.4). Note that over $\mathbb{F}_2$ we have $\pi(\tilde{A}) + \pi(\tilde{B}) = \underline{o} \iff \pi(\tilde{A}) = \pi(\tilde{B})$.

Let us now compute the Hamming weight of $V := H'_{*\tilde{A}\cup\tilde{B}} \cdot \underline{1}^\top$ to show that the algorithm proceeds into Step 21. A trivial verification shows that

$$\begin{aligned} w_H(V) &= w_H\left(\sum_{i=1}^{n-k} H'_{*m_i} \tilde{x}_{m_i}\right) \\ &= |\{m_i | i \in \{1, \ldots, n-k\}, \tilde{x}_{m_i} = 1\}| \\ &= w_H(\underline{\tilde{x}}) - |\tilde{A}| - |\tilde{B}| \\ &= \omega - 2p. \end{aligned}$$

Here we took advantage of the fact that the columns H'_{*m_i} have Hamming weight equal to 1.

Finally, we get that $\underline{y} + \underline{z} = \underline{\tilde{x}}$ and thus the algorithm returns $\underline{\tilde{x}}$. We shall divide the indices $\{1, \dots, n\}$ into three disjoint sets

$$\{1, \dots, n\} = (\tilde{A} \cup \tilde{B}) \cup \Big((X \cup Y) \setminus (\tilde{A} \cup \tilde{B})\Big) \cup M$$

where $M = \{m_1, \dots, m_{n-k}\}$ and examine the equality of the two vectors on each set individually. By definition

$$\begin{aligned} \underline{\tilde{x}}_{\restriction \tilde{A} \cup \tilde{B}} &= \underline{1}, \\ \underline{y}_{\restriction \tilde{A} \cup \tilde{B}} &= \underline{1}, \\ \underline{z}_{\restriction \tilde{A} \cup \tilde{B}} &= \underline{o} \end{aligned}$$

and

$$\begin{aligned} \underline{\tilde{x}}_{\restriction (X \cup Y) \setminus (\tilde{A} \cup \tilde{B})} &= \underline{o}, \\ \underline{y}_{\restriction (X \cup Y) \setminus (\tilde{A} \cup \tilde{B})} &= \underline{o}, \\ \underline{z}_{\restriction (X \cup Y) \setminus (\tilde{A} \cup \tilde{B})} &= \underline{o}. \end{aligned}$$

Thus

$$\begin{aligned} \underline{\tilde{x}}_{\restriction \tilde{A} \cup \tilde{B}} &= \underline{y}_{\restriction \tilde{A} \cup \tilde{B}} + \underline{z}_{\restriction \tilde{A} \cup \tilde{B}}, \\ \underline{\tilde{x}}_{\restriction (X \cup Y) \setminus (\tilde{A} \cup \tilde{B})} &= \underline{y}_{\restriction (X \cup Y) \setminus (\tilde{A} \cup \tilde{B})} + \underline{z}_{\restriction (X \cup Y) \setminus (\tilde{A} \cup \tilde{B})} \end{aligned}$$

and it suffices to show the same for indices from M. From definition of $\underline{y}$ we see

$$\underline{y}_{\restriction M} = \underline{o}.$$

Let us divide $M = N_J \cup (M \setminus N_J)$. From (P.4) we know that

$$\underline{\tilde{x}}_{\restriction N_J} = \underline{o}.$$

Obviously, if $m_i \in N_J$ then $k_i \in J$ and $V_{k_i} = \pi(\tilde{A}) + \pi(\tilde{B}) = 0$. Therefore,

$$\underline{z}_{\restriction N_J} = \underline{o}.$$

Finally, we need to examine $\underline{z}_{\restriction M \setminus N_J}$ and $\underline{\tilde{x}}_{\restriction M \setminus N_j}$. By definition we know

$$w_H\left(\underline{z}\right) = w_H\left(V\right) = \omega - 2p = w_H\left(\underline{z}_{\restriction M \setminus N_J}\right).$$

The last equality comes from what we have so far examined. Similarly, by (P.1) and by what we have so far obtained, we know that

$$w_H\left(\underline{\tilde{x}}_{\restriction M \setminus N_j}\right) = \omega - |\tilde{A} \cup \tilde{B}| = \omega - 2p.$$

Moreover, note for $m \in M \setminus N_J$ the following holds:

$$z_m = 0 \Rightarrow V_{k_i} = 0 \Rightarrow m \notin \tilde{A} \cup \tilde{B} \Rightarrow \tilde{x}_m = 0.$$

The last equality holds due to (P.2) and (P.3). Altogether,

$$\begin{aligned} \underline{\tilde{x}}_{\restriction N_j} &= \underline{y}_{\restriction N_j} + \underline{z}_{\restriction N_j}, \\ .\underline{\tilde{x}}_{\restriction M \setminus N_j} &= \underline{y}_{\restriction M \setminus N_j} + \underline{z}_{\restriction M \setminus N_j}. \end{aligned}$$

□

Definition 9. Let $\underline{x} \in \mathbb{F}_2^n$ such that $\underline{x}H^\top = \underline{o}$ and $w_H(\underline{x}) = \omega$. Then (X, Y, N_J) is said to be a favourable triple with respect to $\underline{x}$ if

(S.1) $w_H\left(\underline{x}_{\restriction X}\right) = p$,

(S.2) $w_H\left(\underline{x}_{\restriction Y}\right) = p$,

(S.3) $\underline{x}_{\restriction N_J} = \underline{o}$ where $N_J := \{m_i \mid i \in \{1, \ldots, n-k\}, k_i \in J\}$.

We now wish to compute the probability of success of the algorithm. We could compute the probability that the random choices in the algorithm choose desired triple X, Y, N_J. However, we only estimate it by the probability that uniformly randomly chosen triple (X, Y, N) is a favourable triple with respect to $\underline{x}$.

Theorem 10. *[Ste88, theorem 1] Let $\underline{x}$ such that $\underline{x}H^\top = \underline{o}$ and $w_H(\underline{x}) = \omega$. Let X, Y, N be an uniformly randomly chosen triple such that*

- *$X, Y, N \subseteq \{1, \ldots, n\}$,*
- *X, Y, N are disjoint,*
- *$|X \cup Y| = |X| + |Y| = k$,*
- *$|N| = l$.*

Then the probability that (X, Y, N) is favourable w.r.t $\underline{x}$ is equal to

$$\frac{\binom{\omega}{2p}\binom{n-\omega}{k-2p}}{\binom{n}{k}} \cdot \frac{\binom{2p}{p}}{2^{2p}} \cdot \frac{\binom{n-k-\omega+2p}{l}}{\binom{n-k}{l}}.$$

Proof. Note that (S.1) and (S.2) are mutually independent events from (S.3).

The first factor is the probability that $X \cup Y$ is chosen such that $w_H\left(\underline{x}_{\restriction X \cup Y}\right) = 2p$. The total number of possibilities how to choose $X \cup Y$ is $\binom{n}{k}$, thus the denominator. The Hamming weight of $\underline{x}$ is ω. Therefore, we need to choose exactly $2p$ indices out of ω and exactly $k - 2p$ indices out of $n - \omega$.

Once we assume $w_H\left(\underline{x}_{\restriction X \cup Y}\right) = 2p$, the events (S.1) and (S.2) are equivalent. The probability of such conditioned event is then the second factor from the theorem. We want to choose X out of $X \cup Y$ such that $w_H\left(\underline{x}_{\restriction X}\right) = p$.

Finally, the probability that the right N is chosen is the third factor. We need to choose l indices out of $n - k - (\omega - 2p)$ indices on which $\underline{x}$ is zero. $\square$

Theorem 11. *[Ste88, Chapter 4] The computational complexity of the basic attack is approximately*

$$\frac{1}{2}(n-k)^3 + k(n-k)^2 + 2lp\binom{\frac{k}{2}}{p} + 2p(n-k)\frac{\binom{\frac{k}{2}}{p}^2}{2^l}.$$

J. Stern showed that for binary code with parameters

$$\begin{aligned} n &= 300, \\ k &= 150, \\ \omega &= 20, \\ p &= 3, \\ l &= 16 \end{aligned}$$

the time complexity would be approximately $6 \cdot 10^7$ operations of processor.

R. J. McEliece in his [McE78] proposed to use binary Goppa codes with parameters

$$\begin{aligned} n &= 1024, \\ k &= 524, \\ t &= 50. \end{aligned}$$

In [BLP08] a new version of Stern's attack is presented. It is stated in Chapter 6 of [BLP08] that the new attack would break McEliece cryptosystem with parameters originally proposed by McEliece in one week.

3.2 Stern's Attack on McEliece PKC

In this section we will show how to use Stern's attack to break McEliece cryptosystem over $\mathbb{F}_2$. Note that the success of the attack depends on success of Stern's attack. Thus, only cryptosystems based on codes with smaller parameters are vulnerable. The attack is based on [BLP08, p. 35].

Assume that the legitimate user had generated matrices G_{priv}, M, P and published the McEliece public key (G_{pub}, t) where $G_{pub} = SG_{priv}P \in \mathbb{F}_2^{k\times n}$. Here G_{priv} and G_{pub} are generator matrices of (n, k, d)-code and $t := \lfloor \frac{d-1}{2} \rfloor$.

Suppose the sending party wants to send a message $\underline{m}$ which they encrypt as

$$\underline{y} = \underline{m}G_{pub} + \underline{e} = \underline{c} + \underline{e}.$$

Here $\underline{e}$ is the error generated by sending party such that $w_H(\underline{e}) = t$.

Attacker then eavesdrops $\underline{y}$ and knows the public key (G_{pub}, t). The algorithm of the attack is then as follows.

Algorithm 2: Stern-McEliece Attack

```
   Input: $G_{pub}, t, \underline{y}$
   Output: $\underline{c}$ such that $\underline{y} = \underline{m}G_{pub} + \underline{e} = \underline{c} + \underline{e}$ and $w_H(\underline{e}) = t$
1  Extend the generator matrix $G_{pub}$ by vector $\underline{y}$ to obtain
       $G_y = \begin{pmatrix} G_{pub} \\ \underline{y} \end{pmatrix} \in \mathbb{F}_2^{(k+1)\times n}$.
2  Compute the parity-check matrix $H'$ of $\mathcal{C}_{G_y}$.
3  Run Stern's algorithm to find a codeword $\underline{x} \in \mathcal{C}_{G_y}$ such that $w_H(\underline{x}) = t$.
4  return $\underline{y} - \underline{x} \in \mathcal{C}_{G_{pub}}$.
```

We shall now discuss the correctness of the algorithm.

Observation. *The newly obtained generator matrix G_y generates code*

$$\mathcal{C}_{G_y} = \mathcal{C}_{G_{pub}} + \{\underline{o}, \underline{y}\}.$$

Lemma 12. *The codeword $\underline{x}$ found in the Algorithm 2 equals $\underline{y} - \underline{c}$.*

Proof. We first show that $\underline{y} - \underline{c}$ could be the desired $\underline{x}$. We need to show that $\underline{y} - \underline{c} \in \mathcal{C}_{G_y}$ and that $w_H\left(\underline{y} - \underline{c}\right) = t$. The first condition is obviously fulfilled. Furthermore, $w_H\left(\underline{y} - \underline{c}\right) = w_H\left(\underline{e}\right) = t$. Thus, the second conditions is fulfilled too.

Secondly, we prove that this solution is unique. For contradiction, suppose there exists another solution than $\underline{x} = \underline{y} - \underline{c}$. Obviously, $\underline{x} \in \mathcal{C}_{G_y} \setminus \mathcal{C}_{G_{pub}}$ because $d(\mathcal{C}_{G_{pub}}) = 2t + 1 > t$. Let

$$\underline{x} = \underline{y} - \underline{c} \in \mathcal{C}_{G_{pub}},$$
$$\underline{x}' = \underline{y} - \underline{c}' \in \mathcal{C}_{G_{pub}}$$

where $\underline{c}' \in \mathcal{C}_{G_{pub}}$ such that $w_H\left(\underline{x}'\right) = t$ and $\underline{x} \neq \underline{x}'$. It is evident that

$$\underline{x} - \underline{x}' = (\underline{y} - \underline{c}) - (\underline{y} - \underline{c}') = \underline{c}' - \underline{c} \in \mathcal{C}_{G_{pub}}.$$

However, using the triangle inequality we see

$$w_H\left(\underline{c}' - \underline{c}\right) = w_H\left(\underline{c}' + \underline{c}\right) \leq w_H\left(\underline{c}'\right) + w_H\left(\underline{c}\right) = 2t < 2t + 1.$$

That implies $w_H\left(\underline{c}' - \underline{c}\right) = 0$ and we thus get $\underline{c}' = \underline{c}$. The equality implies that $\underline{x} = \underline{x}'$ which contradicts our assumption. □

Corollary 13. *Given $G_{pub}, t, \underline{y}$ the Algorithm 2 returns $\underline{c}$ such that $\underline{y} = \underline{c} + \underline{e}$ and $\exists \underline{m} : \underline{m}G_{pub} = \underline{c}$.*

Observation. *After performing the attack, attacker knows $\underline{c}, G_{pub}, t$. They want to find $\underline{m}$ such that $\underline{m}G_{pub} = \underline{c}$. They only need to compute $X \in \mathbb{F}_2^{n \times k}$ such that $G_{pub}X = I_k$, which can be done in polynomial time using Gaussian elimination. Then they can easily compute $\underline{c}X = \underline{m}G_{pub}X = \underline{m}$.*

Note that solving a linear system of at most n equations and n indeterminates by Gaussian elimination has complexity $\mathcal{O}(n^3)$ (see [BT, p. 65]). Therefore, the computation of X has complexity $\mathcal{O}(kn^3)$.

3.3 Stern's Attack on Niederreiter PKC

We can attack Niederreiter PKC similar way as McEliece PKC. Let (H_{pub}, t) be the public key of a Niederreiter PKC. Thus, H_{pub} is a parity-check matrix of an (n, k, d)-binary code $\mathcal{C}_{G_{pub}}$. Let $t := \lfloor \frac{d-1}{2} \rfloor$ be the maximum number of errors the code can correct. Let $\underline{e} \in \mathbb{F}_2^n$, $w_H\left(\underline{e}\right) = t$ be the Niederreiter plaintext and $\underline{c} = \underline{e}H_{pub}^\top$ be the Niederreiter ciphertext. The attacker knows H_{pub}, t and overhears $\underline{c}$.

Observation. *Let $H_{pub}, G_{pub}, \underline{e}, \underline{c}$ be as stated above. Assume that $\underline{y} \in \mathbb{F}_2^n$ such that $\underline{c} = \underline{y}H_{pub}^\top$. Then $\left(\underline{y} - \underline{e}\right) H_{pub}{}^\top = \underline{o}$, thus*

$$\underline{y} - \underline{e} \in \mathcal{C}_{G_{pub}}.$$

Furthermore, we know that

$$\exists! \underline{m} \in \mathbb{F}_2^k : \ \underline{m}G_{pub} = \underline{y} - \underline{e}.$$

Therefore, $\underline{y} - \underline{m}G_{pub} = \underline{e}$.

Using the observation we can construct the following attack which is briefly described at [BLP08, p. 35].

Algorithm 3: Stern-Niederreiter Attack

```
   Input: $H_{pub}, t, \underline{c}$
   Output: $\underline{e}$ such that $w_H(\underline{e}) = t$ and $\underline{c} = \underline{e}H_{pub}{}^\top$
1  Compute the generator matrix $G_{pub}$.
2  Find an $\underline{y} \in \mathbb{F}_2^n$ such that $\underline{y}H_{pub}^\top = \underline{c}$.
3  Run Stern-McEliece attack with input $G_{pub}, t, \underline{y}$ to obtain $\underline{m}G_{pub}$.
4  return $\underline{y} - \underline{m}G_{pub}$.
```

We shall now discuss the correctness of the attack. Computation of G_{pub} is doable by means of basic linear algebra in polynomial time. So is finding $\underline{y}$ as we only need to solve a system of $n-k$ linear equations with n variables. Stern-McEliece attack in Step 3 returns $\underline{m}G_{pub}$ such that $\underline{y} = \underline{m}G_{pub} + \underline{e'}$ where $w_H(\underline{e'}) = t$. However, we showed in the observation that then $\underline{e'} = \underline{e}$. Therefore,

$$\underline{y} - \underline{m}G_{pub} = \underline{e}$$

and

$$\underline{e}H^\top = \left(\underline{y} - \underline{m}G_{pub}\right) H^\top = \underline{y}H^\top = \underline{c}.$$

4. Sidelnikov-Shestakov's Attack

In this chapter we will first describe Sidelnikov-Shestakov's attack on the original Niederreiter PKC based on GRS codes. Then we will show how to use the attack to break McEliece PKC based on GRS codes. Finally, we will give an example of the attack against Niederreiter PKC.

For brevity of notation, we will throughout the chapter denote H_{priv} by H and H_{pub} by K. Moreover, let throughout the chapter q be a power of a prime number.

4.1 Sidelnikov-Shestakov's Attack on the Original Niederreiter PKC

In this section we will describe the found attack by V. M. Sidelnikov and S. O. Shestakov. The section is based on [SS92], [EOS06] and partially on [Hru14].

It is crucial for the attack that the Niederreiter scheme is being used with the GRS code. V. M. Sidelnikov and S. O. Shestakov in 1992 thought that Goppa codes could be attacked by similar algorithm. However, it has been 30 years, and nobody has found a way to apply this attack to Goppa codes. The reason might be that Goppa codes are constructed over $\mathbb{F}_{p^m}$ but interpreted over $\mathbb{F}_p$ for $p \in \mathbb{P}$ and $m \in \mathbb{N}$.

4.1.1 Brief Overview of the Attack

Throughout the section we will focus on (n, k, d)-GRS codes. Parity-check matrices of such codes are of size $(s + 1) \times n$. Where $s = n - k - 1$.

The attack finds an alternative private key in $\mathcal{O}(s^4 + sn)$ steps which makes it an efficient attack. We will show that an alternative private key is sufficient to decrypt any intercepted message.

Definition 10. Let $\mathcal{K}_{priv}$ be the private key belonging to the legitimate user. We say that $\mathcal{K}'_{priv}$ is an alternative private key if it produces the same public key as $\mathcal{K}_{priv}$. An alternative private key might or might not be the same as the original one.

The parity check matrix of an (n, k, d)-GRS code over $\mathbb{F}_q$ depends on values $\underline{\alpha} = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_q$ and $\underline{z} = (z_1, \dots, z_n) \in \left(\mathbb{F}_q^*\right)^n$ (see Definition 15). Assume that the legitimate user has generated private and public keys using $\underline{\alpha}, \underline{z}$ and matrices P, M. The attacker finds alternatives $\underline{\beta}, \underline{y}$ and a matrix $\tilde{M}$. Note that the attacker does not have to look for the matrix P. The reasons are discussed just before Lemma 15.

The structure of the attack is as follows.

1. The very first step of the attack is to extend $\mathbb{F}_q$ into $\mathbb{F}_\infty = \mathbb{F}_q \cup \{\infty\}$. The construction is based on projective line $\mathbb{P}^1(\mathbb{F}_q)$. See Section 4.1.2.

2. We state the notation of parity-check matrix and extend the definition over $\mathbb{F}_\infty$. We show that knowing the alternative private key is sufficient for breaking the cryptosystem. See Section 4.1.3.

3. In Section 4.1.4 we show that we can find the private key up to birational transformation over $\mathbb{F}_\infty$.

4. In Section 4.1.5 we finally give an algorithm that finds values $\gamma_1, \dots, \gamma_n \in \mathbb{F}_\infty$. We transform these values to $\beta_1, \dots, \beta_n \in \mathbb{F}_q$ in Section 4.1.6.

5. In Sections 4.1.7, 4.1.8 and 4.1.9 we will not need $\mathbb{F}_\infty$ anymore. By means of basic linear algebra we find values $y_1, \dots, y_{s+2}$, then the matrix $\tilde{M}$ and finally the remaining values $y_{s+3}, \dots, y_n$.

4.1.2 Defining $\mathbb{F}_\infty$

In this section we will define $\mathbb{F}_\infty = \mathbb{F}_q \cup \{\infty\}$. The motivation (which we will explain) for defining such set comes from the theory of projective space. More on the subject can be found e.g. in [Mor91].

Projective line $\mathbb{P}^1(\mathbb{F}_q)$

Definition 11. The projective line over $\mathbb{F}_q$ denoted by $\mathbb{P}^1(\mathbb{F}_q)$ consists of all equivalence classes of pairs $(\xi_x, \xi_y) \in \mathbb{F}_q^2$ with at least one of ξ_x, ξ_y nonzero. The equivalence relation is given by

$$(\xi_x, \xi_y) \sim (\zeta_x, \zeta_y) \iff \exists \lambda \in \mathbb{F}_q^* : \ (\lambda\xi_x, \lambda\xi_y) = (\zeta_x, \zeta_y).$$

The equivalence class of (ξ_x, ξ_y) is denoted by so called homogeneous coordinates $(\xi_x : \xi_y)$.

Remark. Points of the form $(\xi_x : 1) \in \mathbb{P}^1(\mathbb{F}_q)$ are called affine points of the projective line. On the other hand, the point $(1 : 0) \in \mathbb{P}^1(\mathbb{F}_q)$ is called the point at infinity.

Definition 12. Let $f(x) = \sum_{i=0}^{\deg f} a_i x^i$ be a polynomial over $\mathbb{F}_q$ with $\deg f \leq s$. The corresponding homogenized polynomial will be denoted by

$$\bar{f}(x, y) = \sum_{i=0}^{\deg f} a_i x^i y^{\deg f - i}.$$

We shall define a corresponding homogenized polynomial of degree s as

$$\tilde{f}(x, y) = y^{s - \deg f} \bar{f}(x, y).$$

Observation. *Let $f(x)$ be as in the definition above. Let $\xi = (\xi_x, 1)$ be a representant of $(\xi_x : 1) \in \mathbb{P}^1(\mathbb{F}_q)$. Then*

$$\tilde{f}(\xi_x, 1) = 1^{s-\deg f} \cdot \bar{f}(\xi_x, 1) = \sum_{i=0}^{\deg f} a_i \cdot \xi_x^i \cdot 1^{\deg f - i} = f(\xi_x).$$

On the other hand, let $\xi = (1, 0)$ be a representant of the point at infinity of $\mathbb{P}^1(\mathbb{F}_q)$. Then

$$\tilde{f}(1, 0) = 0^{s-\deg f} \cdot \bar{f}(1, 0) = 0^{s-\deg f} \cdot \sum_{i=0}^{\deg f} a_i \cdot 1^i \cdot 0^{\deg f - i} = \begin{cases} a_s & \text{if } \deg f = s, \\ 0 & \text{otherwise.} \end{cases}$$

Correspondence between $\mathbb{P}^1(\mathbb{F}_q)$ and $\mathbb{F}_\infty$

Definition 13. Define $\mathbb{F}_\infty = \mathbb{F}_q \cup \{\infty\}$ such that

$$\begin{aligned} \forall x \in \mathbb{F}_q^* :\ \frac{x}{0} &= \infty, \\ \forall x \in \mathbb{F}_q^* :\ \frac{x}{\infty} &= 0, \\ \forall f(x) = \sum_{i=0}^{s} a_i x^i \in \mathbb{F}_q[x] :\ f(\infty) &= a_s. \end{aligned}$$

Remark. There is a bijection between projective line $\mathbb{P}^1(\mathbb{F}_q)$ and $\mathbb{F}_\infty$ as follows:

$$\begin{aligned} \pi :\ \mathbb{P}^1(\mathbb{F}_q) &\to \mathbb{F}_\infty \\ \forall \xi_x \in \mathbb{F}_q :\ (\xi_x : 1) &\mapsto \xi_x, \\ (1:0) &\mapsto \infty. \end{aligned}$$

Thus, the infinity corresponds to the projective point at infinity of the projective line over $\mathbb{F}_q$. We can also see, that the value of ∞ in $f(x)$ is well defined as

$$f(\infty) = \tilde{f}(1,0).$$

Remark. From now on we will identify $\mathbb{P}^1(\mathbb{F}_q)$ with $\mathbb{F}_\infty$ in means of the previous remark.

4.1.3 Parity-check Matrix and Notation

Definition 14. [Mac77, p. 333] Let $\underline{\alpha} \in \mathbb{F}_q^n$ have pairwise distinct entries and $\underline{v} \in \mathbb{F}_q^n$ have nonzero entries. Then

$$G_1 = \begin{pmatrix} v_1 & v_2 & \dots & v_n \\ v_1\alpha_1 & v_2\alpha_2 & \dots & v_n\alpha_n \\ \vdots & & \ddots & \vdots \\ v_1\alpha_1^i & v_2\alpha_2^i & \dots & v_n\alpha_n^i \\ \vdots & & \ddots & \vdots \\ v_1\alpha_1^{k-1} & v_2\alpha_2^{k-1} & \dots & v_n\alpha_n^{k-1} \end{pmatrix}$$

is a generator matrix of an (n,k)-GRS code.

Lemma 14. *[Mac77, p. 333] Let $\mathcal{C}_{G_1}$ be as in the definition above. Then exists $\underline{z} \in \mathbb{F}_q^n$ with nonzero entries such that*

$$G_2 = \begin{pmatrix} z_1 & z_2 & \dots & z_n \\ z_1\alpha_1 & z_2\alpha_2 & \dots & z_n\alpha_n \\ \vdots & & \ddots & \vdots \\ z_1\alpha_1^i & z_2\alpha_2^i & \dots & z_n\alpha_n^i \\ \vdots & & \ddots & \vdots \\ z_1\alpha_1^{n-k-1} & z_2\alpha_2^{n-k-1} & \dots & z_n\alpha_n^{n-k-1} \end{pmatrix}$$

is a parity-check matrix of $\mathcal{C}_{G_2}$.

Observation. *Assume G_2 is the matrix defined in the previous lemma. Then G_2 is a generator matrix of an $(n, n-k)$-GRS code.*

Definition 15. Let $\underline{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_\infty^n$ and $\underline{z} = (z_1, \ldots, z_n) \in \mathbb{F}_q^n$ such that

$$\forall i, j \in \{1, \ldots, n\} : \; i \neq j \Rightarrow \alpha_i \neq \alpha_j,$$
$$\forall i \in \{1, \ldots, n\} : \; z_i \neq 0.$$

If $j \in \{1, \ldots, n\}$ such that $\alpha_j \in \mathbb{F}_q$ then the j=th column of a parity-check matrix of a GRS code defined by $\underline{\alpha}$ and $\underline{z}$ will be denoted by

$$H(\underline{\alpha}, \underline{z})_{*j} = \begin{pmatrix} z_j \alpha_j^0 \\ z_j \alpha_j^1 \\ \vdots \\ z_j \alpha_j^s \end{pmatrix}.$$

Consistently with Definition 13 if $j \in \{1, \ldots, n\} : \; \alpha_j = \infty$, then

$$H(\underline{\alpha}, \underline{z})_{*j} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ z_j \end{pmatrix}.$$

Remark. From now on when working with $H(\underline{\alpha}, \underline{z})$ we will implicitly assume that $\underline{\alpha}$ has pairwise distinct values and $\underline{z}$ has nonzero values.

Remark. Suppose that the legitimate user had generated a parity-check matrix $H(\underline{\tilde{\alpha}}; \underline{\tilde{z}}) \in \mathbb{F}_q^{(s+1)\times n}$, a nonsingular matrix $M \in \mathbb{F}_q^{(s+1)\times(s+1)}$ and a permutation matrix $P \in \mathbb{F}_q^{n\times n}$ and published the product $K = MH(\underline{\tilde{\alpha}}; \underline{\tilde{z}})P$.

The aim of the attack is to find an alternative private key given the public key K. Hence, the attacker wants to find an alternative decomposition

$$K = \tilde{M} H(\underline{\beta}; \underline{y}) \tilde{P},$$

where $\tilde{M} \in \mathbb{F}_q^{(s+1)\times(s+1)}$ is nonsingular, $H(\underline{\beta}; \underline{y}) \in \mathbb{F}_q^{(s+1)\times n}$ is a parity-check matrix of an (n, k, d)-GRS code and $\tilde{P} \in \mathbb{F}_q^{n\times n}$ is a permutation matrix.

However, there is no need to find the matrix $\tilde{P}$. The reason is that there exists a GRS code equivalent to the original one (the one the legitimate user used) such that $H(\tilde{\alpha}_i; \tilde{z}_i)P$ is its parity-check matrix. The equivalence is defined by the permutation given by the matrix P. In particular, the attacker only needs to find $\tilde{M}, \underline{\beta}$ and $\underline{y}$.

Lemma 15. *Suppose K is the known public key, $(M, H(\underline{\tilde{\alpha}}, \underline{\tilde{z}}), P)$ is the secret private key. Assume the attacker has found an alternative decomposition of the secret key $K = \tilde{M} H(\underline{\beta}, \underline{y})$. Then after intercepting a ciphertext $\underline{c}$, the attacker can decipher it to obtain the original plaintext.*

Proof. The sending party wishes to send $\underline{m}$ and encrypts it as $\underline{c} := \underline{m}K^\top$. After eavesdropping on the conversation, the attacker knows $\underline{c}$. They multiply it by $\left(\tilde{M}^\top\right)^{-1}$ to obtain $\underline{m}\left(H(\underline{\beta}, \underline{y})\right)^\top$. Now they only need to proceed with the syndrome decoding algorithm to obtain $\underline{m}$. □

Remark. It might seem that the attacker has a quicker way of obtaining the plaintext than the legitimate party. However, the legitimate party may use the syndrome decoding algorithm with matrix $H(\underline{\tilde{\alpha}}, \underline{\tilde{z}})P$ instead of $H(\underline{\tilde{\alpha}}, \underline{\tilde{z}})$.

Notation. From now on let K be the fixed public key, let $(M, H(\underline{\tilde{\alpha}}, \underline{\tilde{z}}), P)$ be the original private key and let $H(\underline{\alpha}, \underline{z}) = H(\underline{\tilde{\alpha}}, \underline{\tilde{z}})P$

4.1.4 Birational Transformations

Definition 16. Let $a, b, c, d \in \mathbb{F}_q$ such that $ad - bc \neq 0$. Then

$$\begin{aligned} \phi :\ & \mathbb{P}^1(\mathbb{F}_q) \to \mathbb{P}^1(\mathbb{F}_q) \\ & (\xi_x : \xi_y) \mapsto (a\xi_x + b\xi_y : c\xi_x + d\xi_y) \end{aligned}$$

is called a birational transformation.

Lemma 16. *The birational transformation on $\mathbb{F}_\infty$ is well defined and*

$$\begin{aligned} \forall \xi \in \mathbb{F}_q :\ \phi(\xi) &= \frac{a\xi + b}{c\xi + d}, \\ \phi(\infty) &= \frac{a}{c}. \end{aligned}$$

Proof. The proof easily follows from the bijection $\pi :\ \mathbb{P}^1(\mathbb{F}_q) \to \mathbb{F}_\infty$. First, assume that $c\xi_x + d \neq 0$, then we obtain

$$\begin{aligned} \forall \xi_x \in \mathbb{F}_q :\ (\xi_x, 1) \overset{\phi}{\mapsto} (a\xi_x + b, c\xi_x + d) \sim \left(\frac{a\xi_x + b}{c\xi_x + d}, 1\right) \overset{\pi}{\mapsto} \frac{a\xi_x + b}{c\xi_x + d} \in \mathbb{F}_q, \\ (1, 0) \overset{\phi}{\mapsto} (a, c) \sim \left(\frac{a}{c}, 1\right) \overset{\pi}{\mapsto} \frac{a}{c}. \end{aligned}$$

Second, assume that $c\xi_x + d = 0$. That can happen only in two cases.

- Either $d = \xi_x = 0, c \neq 0$, then we see that

$$(0, 1) \overset{\phi}{\mapsto} (b, 0) \sim (1, 0) \overset{\pi}{\mapsto} \infty.$$

- Or $d \neq 0, c \neq 0, \xi_x = -\frac{d}{c}$. In this case we have

$$\left(\frac{-d}{c}, 1\right) \overset{\phi}{\mapsto} \left(-\frac{ad}{c} + b, 0\right) \sim (1, 0) \overset{\pi}{\mapsto} \infty$$

□

Definition 17. Remember that K is the given fixed public key. If there exist nonsingular matrix $\tilde{M}$, $\underline{\beta} = (\beta_1, \ldots, \beta_n) \in \mathbb{F}_\infty^n$ and $\underline{y} = (y_1, \ldots, y_n) \in \left(\mathbb{F}_q^*\right)^n$ such that

$$K = \tilde{M} H(\underline{\beta}, \underline{y})$$

then we say that $(\underline{\beta}, \underline{y}, \tilde{M})$ is a solution. The values $(\beta_1, \ldots, \beta_n) = \underline{\beta}$ are called a part of a solution.

Birational Transformations and Solutions

Recall that $K \in \mathbb{F}_q^{(s+1)\times n}$ is the shared public key and $(M, H(\underline{\alpha}, \underline{z}))$ is the original private key (the permutation matrix being hidden in the parity-check matrix) such that

$$K = MH(\underline{\alpha}, \underline{z}).$$

Attacker wishes to recover an alternative private key $(\tilde{M}, H(\underline{\beta}, \underline{y}))$. However, the attacker first finds

$$\underline{\gamma} = (1, 0, \infty, \gamma_4, \dots, \gamma_n) \in \mathbb{F}_\infty^n$$

such that there exist $\hat{M} \in \mathbb{F}_q^{(s+1)\times(s+1)}, \underline{\hat{y}} \in \left(\mathbb{F}_q^*\right)^n$ fulfilling

$$K = \hat{M} H(\underline{\gamma}, \underline{\hat{y}}).$$

After that the attacker birationally transforms $\underline{\gamma} \in \mathbb{F}_\infty^n$ to $\underline{\beta} \in \mathbb{F}_q^n$ and the rest of the attack continues over $\mathbb{F}_q$.

In this section we explain that we can without loss of generality search for a part of a solution of the form $\underline{\gamma} = (\gamma_1, \dots, \gamma_n)$ where

$$\begin{aligned} \gamma_1 &= 1, \\ \gamma_2 &= 0, \\ \gamma_3 &= \infty, \\ \forall i \in \{4, \dots, n\} :\ \gamma_i &\neq \infty \end{aligned}$$

We will do that by showing that "being a solution" is invariant under birational transformations.

Lemma 17. *Given a birational transformation*

$$\phi :\ x \mapsto \frac{ax+b}{cx+d}, \quad (ad - bc) \neq 0, c \neq 0$$

on $\mathbb{F}_\infty$ then there exist $a_1, a_2 \in \mathbb{F}_q^*$ and $b_1, b_2 \in \mathbb{F}_q$ such that $\phi = \varphi_{a_1,b_1} \circ \varphi \circ \varphi_{a_2,b_2}$. The mappings are defined as follows

$$\begin{aligned} \varphi_{a_1,b_1} :&\ x \mapsto a_1 x + b_1, \\ \varphi_{a_2,b_2} :&\ x \mapsto a_2 x + b_2, \\ \varphi :&\ x \mapsto \frac{1}{x}. \end{aligned}$$

We omitted $c = 0$ in the lemma, because the case is far simpler.

Observation. *Suppose $c = 0$ and $ad - bc \neq 0$, we obtain a birational transformation of the form*

$$\phi :\ x \mapsto \frac{ax+b}{d} = \frac{a}{d}x + \frac{b}{d}.$$

It can easily be seen that $\phi = \varphi_{\frac{a}{d},\frac{b}{d}}$.

For brevity of notation, we introduce the following definition.

Definition 18. Let $n \in \mathbb{N}$. Let us extend the definition of any mapping

$$\rho :\ \mathbb{F}_\infty \to \mathbb{F}_\infty$$

to

$$\begin{aligned} \underline{\rho} :\ \mathbb{F}_\infty^n &\to \mathbb{F}_\infty^n, \\ (\omega_1, \dots, \omega_n) &\mapsto (\rho(\omega_1), \dots, \rho(\omega_n)). \end{aligned}$$

Sidelnikov and Shestakov show that $\forall a_1, a_2 \in \mathbb{F}_q^* \; \forall b_1, b_2 \in \mathbb{F}_q$ transformations $\varphi_{a_1,b_1}, \varphi_{a_2,b_2}$ and φ preserve "being a solution". The precise meaning follows in the theorems.

Theorem 18. *Let $a \in \mathbb{F}_q^*$, $b \in \mathbb{F}_q$ and*

$$\begin{aligned} \varphi_{a,b} : \mathbb{F}_q &\to \mathbb{F}_q, \\ x &\mapsto ax + b. \end{aligned}$$

Let us define matrix $T_{a,b} = (t_{ij}) \in \mathbb{F}_q^{(s+1)\times(s+1)}$. Where t_{ij} fulfil

$$(ax+b)^i = \sum_{j=0}^{s} t_{ij} x^j.$$

Put

$$\forall i \in \{1, \ldots, n\} : \; d(\alpha_i) = \begin{cases} 1 & \text{if } \alpha_i \neq \infty, \\ 0 & \text{otherwise.} \end{cases}$$

Then $T_{a,b} H(\underline{\alpha}, (d(\alpha_1)z_1, \ldots, d(\alpha_n)z_n)) = H(\underline{\varphi}_{a,b}(\underline{\alpha}), \underline{z})$.

Remark. From the binomial expansion we can easily see that the entries of the matrix $T_{a,b}$ look like

$$t_{i,j} = \begin{cases} \binom{i}{j} a^j b^{i-j} & \text{if } j \leq i, \\ 0 & \text{if } j > i. \end{cases}$$

In particular, $T_{a,b}$ is lower triangular with nonzero elements in the diagonal. Therefore, $T_{a,b}$ is nonsingular.

Corollary 19. *If $(\underline{\alpha}, \underline{z}, M)$ is a solution, then*

$$\left(\underline{\varphi}_{a,b}(\underline{\alpha}), \left(\frac{z_1}{d(\alpha_1)}, \ldots, \frac{z_n}{d(\alpha_n)} \right), MT_{a,b}^{-1} \right)$$

is also a solution.

Proof. From Theorem 18 we see that

$$H(\underline{\alpha}, \underline{z}) = T_{a,b}^{-1} H\left(\underline{\varphi}_{a,b}(\underline{\alpha}), \left(d(\alpha_1)^{-1} z_1, \ldots, d(\alpha_n)^{-1} z_n \right) \right).$$

Multiplying both sides by M yields

$$MH(\underline{\alpha}, \underline{z}) = MT_{a,b}^{-1} H\left(\underline{\varphi}_{a,b}(\underline{\alpha}), (d(\alpha_1)^{-1} z_1, \ldots, d(\alpha_n)^{-1} z_n) \right).$$

Which by assumption equals K. Obviously, $MT_{a,b}^{-1}$ is nonsingular as it is a product of nonsingular matrices. We need to verify that $\forall i \in \{1, \ldots, n\} : \; z_i d(\alpha_i)^{-1} \neq 0$. We know that

$$\frac{z_i}{d(\alpha_i)} = 0 \iff (z_i = 0 \text{ xor } d(\alpha_i) = \infty).$$

However, by assumptions and definition of $d(\cdot)$ we know that

$$\begin{aligned} &\forall i \in \{1, \ldots, n\} : z_i \neq 0, \\ &\forall i \in \{1, \ldots, n\} : d(\alpha_i) \in \{0, 1\}. \end{aligned}$$

Thus, we have obtained a new solution □

An analogous theorem holds for $\varphi : \ x \mapsto \frac{1}{x}$.

Theorem 20. *Let us define*

$$T = \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ & & \ddots & & \\ 1 & 0 & \dots & 0 & 0 \end{pmatrix}$$

and

$$\forall i \in \{1, \dots, n\} : \ e(\alpha_i) := \begin{cases} \alpha_i^{-s} & \text{if } \alpha_i \notin \{0, \infty\}, \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$TH(\underline{\alpha}, (z_1 e(\alpha_1), \dots, z_n e(\alpha_n))) = H(\underline{\varphi}(\underline{\alpha}), \underline{z}).$$

Corollary 21. *If $(\underline{\alpha}, \underline{z}, M)$ is a solution, then*

$$\left(\underline{\varphi}(\underline{\alpha}), \left(\frac{z_1}{e(\alpha_1)}, \dots, \frac{z_n}{e(\alpha_n)}\right), MT\right)$$

is a solution too.

Proof. As in the previous case we again see that

$$H(\underline{\alpha}, \underline{z}) = T^{-1} H(\underline{\varphi}(\underline{\alpha}), (z_1 e(\alpha_1)^{-1}, \dots, z_n e(\alpha_n)^{-1})).$$

Multiplying by M and taking advantage of the fact that $T^{-1} = T$ gives us

$$MH(\underline{\alpha}, \underline{z}) = MTH(\underline{\varphi}(\underline{\alpha}), (z_1 e(\alpha_1)^{-1}, \dots, z_n e(\alpha_n)^{-1})).$$

Which again by definition equals K. We see that MT is a nonsingular matrix. We know that

$$\forall i \in \{1, \dots, n\} : \ z_i \neq 0,$$
$$\forall i \in \{1, \dots, n\} : \ e(\alpha_i) \neq \infty.$$

Which implies that $\forall i \in \{1, \dots, n\} : \ \frac{z_i}{e(\alpha_i)} \neq 0$. Therefore, the previous gives a solution. □

Theorem 22. *Let $\phi : \ \mathbb{F}_\infty \to \mathbb{F}_\infty$ be a birational transformation. If $(\underline{\alpha}, \underline{z}, M)$ is a solution, then there exists $\tilde{M} \in \mathbb{F}_q^{(s+1)\times(s+1)}$ and $\tilde{\underline{z}} \in \mathbb{F}_\infty^n$ such that*

$$\left((\underline{\phi}(\underline{\alpha})), \tilde{\underline{z}}, \tilde{M}\right)$$

is also a solution.

Proof. Let $a_1, a_2 \in \mathbb{F}_q^*$ and $b_1, b_2 \in \mathbb{F}_q$ such that $\phi = \varphi_{a_1,b_1} \circ \varphi \circ \varphi_{a_2,b_2}$. Existence of such a_1, a_2, b_1, b_2 emerges from Lemma 17.

Corollary 19 gives us that

$$\left(\underline{\varphi}_{a_2,b_2}(\underline{\alpha}), (z_1 d(\alpha_1)^{-1}, \dots, z_n d(\alpha_n)^{-1}), MT_{a_2,b_2}^{-1}\right)$$

is a solution.

Now we can proceed using Corollary 21 to obtain that

$$\left(\underline{\varphi}(\underline{\varphi}_{a_2,b_2}(\underline{\alpha})), \left(\frac{z_1}{d(\alpha_1)e(\varphi_{a_2,b_2}(\alpha_1))}, \dots, \frac{z_n}{d(\alpha_n)e(\varphi_{a_2,b_2}(\alpha_n))}\right)^\top, MT_{a_2,b_2}^{-1}T\right)$$

is a solution.

Finally, we transform $\underline{\varphi}(\underline{\varphi}_{a_2,b_2}(\underline{\alpha})) \to \underline{\varphi}_{a_1,b_1}(\underline{\varphi}(\underline{\varphi}_{a_2,b_2}(\underline{\alpha}))) = \underline{\phi}(\underline{\alpha})$ and using Corollary 19 again we see that

$$\left(\underline{\phi}(\underline{\alpha}), \left(\dots, \frac{z_i}{d(\alpha_i)e(\varphi_{a_2,b_2}(\alpha_i))d(\varphi(\varphi_{a_2,b_2}(\alpha_i)))}, \dots\right), MT_{a_2,b_2}^{-1}TT_{a_1,b_1}^{-1}\right)$$

is a solution too.

To complete the proof, it is sufficient to put

$$\tilde{M} := MT_{a_2,b_2}^{-1}TT_{a_1,b_1}^{-1},$$
$$\forall i \in \{1, \dots, n\} : \ \tilde{z}_i := \frac{z_1}{d(\alpha_1) \cdot e(\varphi_{a_2,b_2}(\alpha_1)) \cdot d(\varphi(\varphi_{a_2,b_2}(\alpha_1)))}.$$

□

Remark. As an attacker, we can search for an alternative private key up to birational transformation on $\underline{\alpha}$.

4.1.5 Finding Pairwise Distinct Values $\gamma_1, \dots, \gamma_n \in \mathbb{F}_\infty$

In the beginning the legitimate party chose

$$M, \underline{\alpha} = (\alpha_1, \dots, \alpha_n), \underline{z} = (z_1, \dots, z_n)$$

in order to generate a public key K. The attacker first recovers

$$\underline{\gamma} = (\gamma_1, \dots, \gamma_n) \in (\mathbb{F}_\infty^*)^n .$$

In this section we show how to recover $\underline{\gamma}$.

Theorem 23. *For any distinct $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n \in \mathbb{F}_q \subseteq \mathbb{F}_\infty$ there exists a birational transformation*

$$\phi : \mathbb{F}_\infty \to \mathbb{F}_\infty,$$
$$x \mapsto \frac{ax+b}{cx+d}, \ (ad-bc) \neq 0,$$

satisfying

$$\begin{aligned} \phi(\alpha_1) &= \gamma_1 = 1, \\ \phi(\alpha_2) &= \gamma_2 = 0, \\ \phi(\alpha_3) &= \gamma_3 = \infty, \\ \forall j > 3 : \ \phi(\alpha_j) &= \gamma_j \notin \{1, 0, \infty\}. \end{aligned}$$

Proof. The equalities give us four requirements to be fulfilled. They are

$$\begin{align}
\phi(\alpha_1) = 1 &\Rightarrow a\alpha_1 + b = c\alpha_1 + d \neq 0, \tag{4.1}\\
\phi(\alpha_2) = 0 &\Rightarrow a\alpha_2 + b = 0 \text{ and } c\alpha_2 + d \neq 0, \tag{4.2}\\
\phi(\alpha_3) = \infty &\Rightarrow a\alpha_3 + b \neq 0 \text{ and } c\alpha_3 + d = 0, \tag{4.3}\\
ad - bc &\neq 0. \tag{4.4}
\end{align}$$

It can easily be proved that

$$ad - bc \neq 0 \Rightarrow \forall i \in \{1,2,3\} : \ ((a\alpha_i + b \neq 0) \text{ or } (c\alpha_i + d \neq 0)).$$

Let us for contradiction assume that

$$ad - bc \neq 0 \text{ and } \exists i \in \{1,2,3\} : \ a\alpha_i + b = 0 = c\alpha_i + d.$$

Obviously, $a = 0 \Rightarrow b = 0$ and similarly $c = 0 \Rightarrow d = 0$. Both cases would be in contradiction with $ad - bc \neq 0$. Thus, we can assume that $a, c \neq 0$. Therefore, we can write

$$-\frac{b}{a} = \alpha_i = -\frac{d}{c}.$$

Which leads to $cb = ad$ and subsequently to $ad - bc = 0$. We have a contradiction again.

Thus, we can equivalently rewrite the four requirements as follows:

$$\begin{align}
a\alpha_1 + b - (c\alpha_1 + d) &= 0, \tag{4.5}\\
a\alpha_2 + b &= 0, \tag{4.6}\\
c\alpha_3 + d &= 0, \tag{4.7}\\
ad - bc &\neq 0. \tag{4.8}
\end{align}$$

Let us first focus on equalities 4.5, 4.6 and 4.7. These equalities give us system of linear equations with a, b, c, d as unknowns:

$$\begin{pmatrix} \alpha_1 & 1 & -\alpha_1 & -1 \\ \alpha_2 & 1 & 0 & 0 \\ 0 & 0 & \alpha_3 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

We can easily solve the matrix and write down the kernel of the matrix with $t \in \mathbb{F}_q$ as a parameter. Let us denote the matrix by A, then

$$\operatorname{Ker}(A) = \left\{ t \begin{pmatrix} 1 \\ -\alpha_2 \\ \frac{\alpha_2 - \alpha_1}{\alpha_3 - \alpha_1} \\ \alpha_3 \frac{\alpha_1 - \alpha_2}{\alpha_3 - \alpha_1} \end{pmatrix} ; t \in \mathbb{F}_q \right\}.$$

Note that $\alpha_1, \alpha_2, \alpha_3$ are pairwise distinct, thus none of the denominators is zero. Also note, that

$$\forall (a,b,c,d)^\top \in \operatorname{Ker}(A) : \ (a,b,c,d)^\top = \underline{o} \iff t = 0.$$

We will now show that $\forall (a,b,c,d)^\top \in \operatorname{Ker}(A) \setminus \{\underline{o}\}$ we have $ad - bc \neq 0$, i.e. for all nontrivial solutions of the system, the nonequality 4.8 holds. Let

$$(a,b,c,d)^\top = t\left(1, -\alpha_2, \frac{\alpha_2-\alpha_1}{\alpha_3-\alpha_1}, \alpha_3\frac{\alpha_1-\alpha_2}{\alpha_3-\alpha_1}\right)^\top \in \operatorname{Ker}(A) \setminus \{\underline{o}\}$$

be an arbitrary nontrivial solution. We have

$$\begin{aligned} ad - bc &= t^2\left(\alpha_3\frac{\alpha_1-\alpha_2}{\alpha_3-\alpha_1} - \alpha_2\frac{\alpha_1-\alpha_2}{\alpha_3-\alpha_1}\right) \\ &= t^2\left(\frac{\alpha_1-\alpha_2}{\alpha_3-\alpha_1}\right)(\alpha_3-\alpha_2) \\ &\neq 0 \end{aligned}$$

where the last inequality holds due to $\alpha_1, \alpha_2, \alpha_3$ being pairwise distinct and t being nonzero. □

Corollary 24. *If $(\underline{\alpha}, \underline{z}, M)$ is a solution, then there exist a birational transformation ϕ, a nonsingular matrix M and a vector $\underline{w} \in (\mathbb{F}_\infty^*)^n$ such that*

$$\left((1, 0, \infty, \phi(\alpha_4), \ldots, \phi(\alpha_n)), \underline{w}, \tilde{M}\right)$$

is a solution

Proof. The previous theorem gives us the existence of such a birational transformation and from Theorem 22 we know that the birational transformation gives as a solution. □

Remark. In other words, we can fix $\gamma_1, \gamma_2, \gamma_3$ to specific values without loss of generality. In what follows, we fix $(\gamma_1, \gamma_2, \gamma_3) := (1, 0, \infty)$. Furthermore, $\underline{\gamma}$ is a part of a solution such that $(\gamma_1, \gamma_2, \gamma_3) := (1, 0, \infty)$. From now on let us assume that $(\underline{\gamma}, \underline{w}, \tilde{M})$ is a solution.

Lemma 25. *If $(\underline{\gamma}, \underline{w}, \tilde{M})$ is a solution then there exist unique polynomials $f_1, \ldots, f_{s+1}$ of degree less than or equal to s such that the matrix K can be expressed as*

$$K = \begin{pmatrix} w_1 f_1(\gamma_1) & w_2 f_1(\gamma_2) & \ldots & w_n f_1(\gamma_n) \\ w_1 f_2(\gamma_1) & w_2 f_2(\gamma_2) & \ldots & w_n f_2(\gamma_n) \\ & & \ddots & \\ w_1 f_{s+1}(\gamma_1) & w_2 f_{s+1}(\gamma_2) & \ldots & w_n f_{s+1}(\gamma_n) \end{pmatrix}.$$

Proof. Set $\tilde{M} = (m_{ij}) \in \mathbb{F}_q^{(s+1)\times(s+1)}$. Let us define

$$\forall i \in \{1, \ldots, s+1\}: \; f_i(x) := \sum_{k=1}^{s+1} m_{ik} x^{k-1}.$$

We shall prove the lemma by examining the j-th column of the matrix K.

First, suppose that $j \in \{1, \ldots, n\}: \; \gamma_j \in \mathbb{F}_q$. Then

$$K_{*j} = \left(\tilde{M} H(\underline{\gamma}, \underline{w})\right)_{*j} = \tilde{M} \cdot \left(H(\underline{\gamma}, \underline{w})\right)_{*j} = \tilde{M} \cdot \begin{pmatrix} w_j \gamma_j^0 \\ w_j \gamma_j^1 \\ \vdots \\ w_j \gamma_j^s \end{pmatrix}$$

$$= \begin{pmatrix} w_j \cdot \sum_{k=1}^{s+1} m_{1k}\gamma_j^{k-1} \\ w_j \cdot \sum_{k=1}^{s+1} m_{2k}\gamma_j^{k-1} \\ \vdots \\ w_j \cdot \sum_{k=1}^{s+1} m_{s+1,k}\gamma_j^{k-1} \end{pmatrix} = \begin{pmatrix} w_j \cdot f_1(\gamma_j) \\ w_j \cdot f_2(\gamma_j) \\ \vdots \\ w_j \cdot f_{s+1}(\gamma_j) \end{pmatrix}.$$

Second, suppose that $l \in \{1, \dots, n\} : \ \gamma_l = \infty$. Then the l-th column of K equals

$$K_{*l} = \tilde{M} \cdot \left(H(\underline{\gamma}, \underline{w})\right)_{*l} = \tilde{M} \cdot \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ w_l \end{pmatrix} = \begin{pmatrix} w_l \cdot m_{1,s+1} \\ w_l \cdot m_{2,s+1} \\ \vdots \\ w_l \cdot m_{s+1,s+1} \end{pmatrix}.$$

Remember that by Definition 13:

$$\forall i \in \{1, \dots, s+1\} : \ f_i(\infty) = m_{i,s+1}.$$

Thus,

$$K_{*l} = \begin{pmatrix} w_l \cdot f_1(\infty) \\ w_l \cdot f_2(\infty) \\ \vdots \\ w_l \cdot f_{s+1}(\infty) \end{pmatrix} = \begin{pmatrix} w_l \cdot f_1(\gamma_l) \\ w_l \cdot f_2(\gamma_l) \\ \vdots \\ w_l \cdot f_{s+1}(\gamma_l) \end{pmatrix}$$

□

Remark. Previous lemma holds for any solution.

We show how to compute the values of $\gamma_4, \dots, \gamma_n \in \mathbb{F}_n$. The proof is constructive and thus in the end gives us Algorithm 4.

Theorem 26. *Let $\underline{\gamma} = (1, 0, \infty, \gamma_4, \dots, \gamma_n)$ be a part of a solution. Define set of indices*

$$\begin{aligned} J_1 &= \{1, s+2, \dots, 2s\}, \\ J_2 &= \{2, s+2, \dots, 2s\}, \\ J_3 &= \{1, 3, \dots, s+1\}, \\ J_4 &= \{2, 3, \dots, s+1\}, \end{aligned}$$

then

$$\forall l \in \{1, \dots, 4\} : \ \exists \underline{c_l} \in \mathbb{F}_q^{s+1} \setminus \{\underline{o}\} \text{ such that } \underline{c_l} K_{*J_l} = \underline{o}.$$

Take an arbitrary index

$$r \in \{1, \dots, n\} \setminus (J_1 \cup J_2 \cup \{3\}),$$

and define

$$\begin{aligned} \underline{p_l} &:= \underline{c_l} K, \\ \forall j \in \{1, \dots, n\} \setminus (J_1 \cup J_2 \cup \{3\}) : \ b_j &:= \frac{p_{1j}}{p_{2j}}, \\ \forall j \in \{1, \dots, n\} \setminus (J_3 \cup J_4) : \ b_j &:= \frac{p_{3j}}{p_{4j}} \cdot \frac{p_{4r}}{p_{3r}} \cdot b_r, \end{aligned}$$

Then

$$\forall j \in \{4, \dots, n\} : \ \gamma_j := \frac{b_3}{b_3 - b_j}.$$

Proof. We first show that $\forall l \in \{1,\dots,4\} : \underline{c_l} \neq \underline{o}$ exists. It is sufficient to note that $|J_l| = s$. Thus, $\underline{c_l}K_{*J_l} = \underline{o}$ is a system of s equations and $s+1$ variables, which always has a nontrivial solution.

Let $f_1,\dots,f_{s+1}$ be the unique polynomials from Lemma 25. Let us construct polynomials

$$\forall l \in \{1,\dots,4\} : \; g_l(x) := \sum_{i=1}^{s+1} c_{li} f_i(x) \in \mathbb{F}_q[x].$$

All four polynomials are nonzero. For contradiction, assume that $\exists l \in \{1,\dots,4\} : g_l(x) = 0 \in \mathbb{F}_q$. Then the polynomials $f_1,\dots,f_{s+1} \in \mathbb{F}_q[x]$ would be linearly dependent. Hence, the rows of K would be linearly dependent, which is a contradiction with $\operatorname{Rank}(K) = n-k$. Obviously, the degree of all polynomials is restricted by the degrees of $f_1,\dots,f_n$. Thus, $\forall l \in \{1,\dots,4\} : \deg g_l \leq s$.

From the definition of $f_1,\dots,f_n$ it can be easily seen that:

$$\forall l \in \{1,\dots,4\} \forall j \in \{1,\dots,n\} : \; w_j \cdot g_l(\gamma_j) = \underline{c_l}K_{*j}. \tag{4.9}$$

As $w_1,\dots,w_n$ are nonzero, we obtain

$$\forall l \in \{1,\dots,4\} \forall j \in J_l : \; g_l(\gamma_j) = 0.$$

Note that that gives us s roots for both polynomials.

Let us now focus only on g_1 and g_2. Obviously, there are no other roots due to the degree restriction. We have obtained factorization of both polynomials:

$$\forall l \in \{1,2\} : \; g_l(x) = a_l \cdot \prod_{j \in J_l} (x - \gamma_j).$$

Note that there is only one value in which J_1 and J_2. differ. Thus, we obtain

$$\begin{aligned}\frac{g_1(x)}{g_2(x)} &= \frac{a_1 \cdot \prod_{j\in J_1}(x-\gamma_j)}{a_2 \cdot \prod_{j\in J_2}(x-\gamma_j)} \\ &= \frac{a_1}{a_2} \cdot \frac{x-\gamma_1}{x-\gamma_2} \\ &= \frac{a_1}{a_2} \cdot \frac{x-1}{x}.\end{aligned}$$

Where $a_1, a_2 \in \mathbb{F}_q$ are the leading coefficients of g_1, g_2, respectively. From the Definition 13 and Equation 4.9 we know that

$$\forall l \in \{1,2\} : \; a_l = g_l(\infty) = g_l(\gamma_3) = \frac{1}{w_3}\underline{c_l}K_{*3} = p_{l3}.$$

That gives us new equation in which only γ_j is unknown:

$$\begin{aligned}\forall j \in \{1,\dots,n\} \setminus (J_1 \cup J_2 \cup \{3\}) : \; \frac{g_1(\gamma_j)}{g_2(\gamma_j)} &= \frac{p_{13}}{p_{23}} \cdot \frac{\gamma_j - 1}{\gamma_j}, \\ \frac{p_{1j}}{p_{2j}} &= \frac{p_{13}}{p_{23}} \cdot \frac{\gamma_j - 1}{\gamma_j}, \\ \gamma_j &= \frac{\frac{p_{13}}{p_{23}}}{\frac{p_{13}}{p_{23}} - \frac{p_{1j}}{p_{2j}}}.\end{aligned}$$

Using the b_j, we obtain

$$\forall j \in \{1, \dots, n\} \setminus (J_1 \cup J_2 \cup \{3\}) : \ \gamma_j = \frac{b_3}{b_3 - b_j}$$

We shall now focus on polynomials g_3 and g_4. This time $3 \in J_3 \cap J_4$, thus $g_3(\gamma_3) = g_4(\gamma_3) = 0$. However, we assumed that $\gamma_3 = \infty$. Thus, Definition 13 gives us that the coefficients associated with x^s are zero in both $g_3(x)$ and $g_4(x)$. Which means that $\forall l \in \{3, 4\} : \ \deg(g_l) \leq s - 1$. Furthermore, both polynomials have $s - 1$ roots from $\mathbb{F}_q$. We have again obtained a factorization of both polynomials:

$$\begin{aligned} g_3(x) &= a_3 \cdot \prod_{j \in J_3 \setminus \{3\}} (x - \gamma_j), \\ g_4(x) &= a_4 \cdot \prod_{j \in J_4 \setminus \{3\}} (x - \gamma_j). \end{aligned}$$

As in the previous proof, let us examine the fraction of these polynomials:

$$\begin{aligned} \frac{g_3(x)}{g_4(x)} &= \frac{a_3}{a_4} \cdot \frac{\prod_{j \in J_3 \setminus \{3\}} (x - \gamma_j)}{\prod_{j \in J_4 \setminus \{3\}} (x - \gamma_j)} \\ &= \frac{a_3}{a_4} \cdot \frac{x - \gamma_1}{x - \gamma_2} \\ &= \frac{a_3}{a_4} \cdot \frac{x - 1}{x}. \end{aligned} \tag{4.10}$$

This time we do not know the values of leading coefficients a_3 and a_4. Taking the chosen r we obtain

$$\begin{aligned} \frac{g_3(\gamma_r)}{g_4(\gamma_r)} &= \frac{a_3}{a_4} \cdot \frac{\gamma_r - 1}{\gamma_r}, \\ \frac{p_{3r}}{p_{4r}} &= \frac{a_3}{a_4} \cdot \frac{\gamma_r - 1}{\gamma_r}, \\ \frac{a_3}{a_4} &= \frac{p_{3r}}{p_{4r}} \cdot \frac{\gamma_r}{\gamma_r - 1}. \end{aligned} \tag{4.11}$$

By definition

$$\begin{aligned} \gamma_r &= \frac{b_3}{b_3 - b_r}, \\ b_3 - b_r &= \frac{b_3}{\gamma_r}, \\ b_r &= b_3 - \frac{b_3}{\gamma_r} = \frac{b_3(\gamma_r - 1)}{\gamma_r}. \end{aligned}$$

That easily yields

$$\frac{b_r}{b_3} = \frac{\frac{b_3(\gamma_r - 1)}{\gamma_r}}{b_3} = \frac{\gamma_r - 1}{\gamma_r}. \tag{4.12}$$

Now combining 4.11, 4.12 and 4.10 we obtain

$$\begin{aligned}
\frac{p_{3j}}{p_{4j}} &= \frac{a_3}{a_4} \cdot \frac{\gamma_j - 1}{\gamma_j} \\
&= \frac{p_{3r}}{p_{4r}} \cdot \frac{\gamma_r}{\gamma_r - 1} \cdot \frac{\gamma_j - 1}{\gamma_j} \\
&= \frac{p_{3r}}{p_{4r}} \cdot \frac{b_3}{b_r} \cdot \frac{\gamma_j - 1}{\gamma_j}, \\
\frac{p_{3j}}{p_{4j}} \cdot \frac{p_{4r}}{p_{3r}} \cdot b_r &= \frac{\gamma_j - 1}{\gamma_j} \cdot b_3, \\
b_j &= \frac{\gamma_j - 1}{\gamma_j} \cdot b_3, \\
\gamma_j &= \frac{b_3}{b_3 - b_j}.
\end{aligned}$$

□

```
Algorithm 4: Find $\gamma_4, \dots, \gamma_n$
  Input: $K$
  Output: $\gamma_4, \dots, \gamma_n$
1  $J_1 := \{1, s+2, \dots, 2s\}$
2  $J_2 := \{2, s+2, \dots, 2s\}$
3  $J_3 := \{1, 3, \dots, s+1\}$
4  $J_4 := \{2, 3, \dots, s+1\}$
5  for $l \in \{1, \dots, 4\}$ do
6      Find $\underline{c_l} \in \mathbb{F}_q^{s+1}$ such that $\underline{c_l} K_{*J_l} = \underline{o}$.
7      $\underline{p_l} := \underline{c_l} K \in \mathbb{F}_q^n$
8  $b_3 := \frac{p_{13}}{p_{23}}$
9  for $j \in \{4, \dots, s+1, 2s+1, \dots, n\}$ do
10     $b_j := \frac{p_{1j}}{p_{2j}}$
11     $\gamma_j := \frac{b_3}{b_3 - b_j}$
12 choose $r \in \{4, \dots, n\} \setminus (J_1 \cup J_2 \cup J_3 \cup \{3\})$
13 for $j \in \{s+2, \dots, 2s\}$ do
14     $b_j := \frac{p_{4r}}{p_{3r}} \cdot \frac{p_{3j}}{p_{4j}} \cdot b_r$
15     $\gamma_j := \frac{b_3}{b_3 - b_j}$
16 return $(\gamma_4, \dots, \gamma_n)$
```

Theorem 27. *If there exists a solution, then algorithm 4 always terminates and finds $\gamma_4, \dots, \gamma_n$ such that*

$$(1, 0, \infty, \gamma_4, \dots, \gamma_n)$$

is a part of a solution.

Proof. According to Theorem 23 for any $\alpha_1, \dots, \alpha_n \in \mathbb{F}_n$ there exists a birational transformation ϕ such that

$$\begin{aligned}\phi(\alpha_1) &= 1,\\ \phi(\alpha_2) &= 0,\\ \phi(\alpha_3) &= \infty,\\ \forall i \in \{4,\dots,n\} : \ \phi(\alpha_i) &\notin \{1,0,\infty\}.\end{aligned}$$

Theorem 22 then gives us that if $(\alpha_1,\dots,\alpha_n)$ is a part of a solution, then $(1,0,\infty,\phi(\alpha_4),\dots,\phi(\alpha_n))$ is a part of a solution too.

In the constructive proof of Theorem 26 we showed that if $(1,0,\infty,\gamma_4,\dots,\gamma_n)$ is a part of a solution, then Algorithm 4 finds $\gamma_4,\dots,\gamma_n$.

Note that in Theorem 26 we showed that $\forall l \in \{1,\dots,4\}$ $\underline{c_l}$ always exist. Thus, the algorithm always terminates and returns $\gamma_4,\dots,\gamma_n$. □

Comments on Algorithm 4

Remark. Note that the value of γ_r must be found in the first part of the algorithm. That means we need $\{4,\dots,n\}\setminus(J_1\cup J_2\cup J_3\cup\{3\}) = \{2s+1,\dots,n\}$ to be nonempty. In other words, we need $2s+1 \le n$. The inequality can be modified as follows (using Theorem 2):

$$\begin{aligned}n &\ge 2s+1\\ &= 2(d-2)+1\\ &= 2(2t+1-2)+1\\ &= 4t-1.\end{aligned}$$

However, we are focusing on the codes with $t << n$, therefore we can implicitly assume that $4t-1 \ge n$. Note that codes with bigger t would have smaller k and thus might be more vulnerable to brute-force attacks.

For example, the original proposal of parameters given by McEliece in [McE78] was

$$\begin{aligned}n &= 1024,\\ k &= 524,\\ t &= 50.\end{aligned}$$

And we see that in this case $4t-1 = 200-1 \le 1024 = n$. Thus, we can safely assume that such r always exists.

We should examine if the algorithm is deterministic. In Step 6 we ask for any c_l. We show in the following lemma, that the result of the algorithm does not depend on the chosen c_l.

Lemma 28. *The result of the algorithm does not depend on the chosen $\underline{c_l}$, $l \in \{1,\dots,4\}$.*

Proof. Let $i \in \{1,\dots,4\}$. Recall that $K_{*J_i} = \tilde{M}\left(H(\underline{\gamma},\underline{w})\right)_{*J_i}$. By definition, $\forall i \in \{1,2\}$: $\left(H(\underline{\gamma},\underline{w})\right)_{\{1,\dots,s\}J_i}$ are transposed Vandermonde matrices. Thus, $\forall i \in \{1,2\}$: $\operatorname{rank}\left(H(\underline{\gamma},\underline{w})\right)_{*J_i} = s$.

For $i \in \{3,4\}$ the matrix $\left(H(\underline{\gamma},\underline{w})\right)$ obtains the third column relevant to $\gamma_3 = \infty$ $(0,\dots,w_3)^\top$. Thus, $\left(H(\underline{\gamma},\underline{w})\right)_{\{1\}\cup\{3,\dots,s+1\}J_i}$ is a transposed Vandermonde matrix with one substituted column. The column is linearly independent from the others, thus again $\forall i \in \{3,4\}:\ \operatorname{rank}\left(H(\underline{\gamma}, \underline{w})\right)_{*J_i} = s$.

Altogether, suppose $l \in \{1,\dots,4\}, \underline{c_l}, \underline{\tilde{c}_l} \in \mathbb{F}_q^s$ then $\underline{c_l} K_{*J_l} = \underline{o} = \underline{\tilde{c}_l} K_{*J_l} \Rightarrow \exists \lambda \in \mathbb{F}_q :$ $\underline{c_l} = \lambda \underline{\tilde{c}_l}$.

Suppose we ran the algorithm twice. Let us denote the values from the first run as in the description of the algorithm. And let us denote the values from the second run by tilde. We will now show, that $\forall j \in \{4,\dots,n\}:\ \gamma_j = \tilde{\gamma}_j$.

Let $l \in \{1,\dots,4\}$. Assume that in the first run we chose in Step 5 $\underline{c_l}$. Then in the second run we chose $\underline{\tilde{c}_l} = \lambda \underline{c_l}$. Subsequently, we computed $\underline{p_l}$ and $\underline{\tilde{p}_l} = \lambda \underline{p_l}$. In Step 8 we computed b_3 and $\tilde{b_3} = \frac{\tilde{p}_{13}}{\tilde{p}_{23}} = \frac{\lambda p_{13}}{\lambda p_{23}} = b_3$.

Now for all $j \in \{4,\dots,s+1, 2s+1,\dots,n\}$:

$$\tilde{b}_j = \frac{\tilde{p}_{1j}}{\tilde{p}_{2j}} = \frac{\lambda p_{1j}}{\lambda p_{2j}} = b_j.$$

And for all $j \in \{s+2,\dots,2s\}$:

$$\tilde{b}_j = \frac{\tilde{p}_{4r}}{\tilde{p}_{3r}} \frac{\tilde{p}_{3j}}{\tilde{p}_{4j}} \tilde{b}_r = \frac{\lambda p_{4r}}{\lambda p_{3r}} \frac{\lambda p_{3j}}{\lambda p_{4j}} b_r = b_j.$$

We now easily see that $\forall j \in \{4,\dots,n\}:\ \tilde{\gamma}_j = \gamma_j$. □

4.1.6 Transforming $\underline{\gamma} \in \mathbb{F}_\infty^n$ To Obtain $\underline{\beta} \in \mathbb{F}_q^n$

Although we have recovered $\gamma_1,\dots,\gamma_n$, it is obvious that $\gamma_3 \notin \mathbb{F}_q$. Fortunately, we can map γ_3 into $\mathbb{F}_q$ using a birational transformation.

Lemma 29. *Let $\underline{\gamma} = (1, 0, \infty, \gamma_4, \dots, \gamma_n) \in \mathbb{F}_\infty^n$ be a part of a solution. Then there exists $a \in \mathbb{F}_q$ such that $a \notin \{\gamma_1,\dots,\gamma_n\}$. Define*

$$\forall i \in \{1,\dots,n\}:\ \beta_i := \frac{1}{a - \gamma_i}.$$

Then $\underline{\beta} = (\beta_1,\dots,\beta_n)$ is a part of a solution. Furthermore, $\underline{\beta} \in \mathbb{F}_q^n$.

Proof. In the extreme case $n = q$ but $\gamma_3 = \infty \notin \mathbb{F}_q$, thus there is always at least one a such that $\forall j \in \{1,\dots,n\}:\ a \neq \gamma_j$.
The mapping $\gamma_i \mapsto \frac{1}{a-\gamma_i}$ is a composition of mappings from Theorem 18 and Theorem 20. Therefore, $\underline{\beta}$ surely is a part of a solution. Furthermore,

$$\frac{1}{a-\gamma_3} = \frac{1}{a-\infty} = 0,$$
$$\gamma_i \in \mathbb{F}_q \Rightarrow \frac{1}{a-\gamma_i} \in \mathbb{F}_q.$$

Hence, $\forall i \in \{1,\dots,n\}:\ \beta_i \in \mathbb{F}_q$. □

4.1.7 Finding Nonzero Values $y_1, \dots, y_{s+2} \in \mathbb{F}_q^*$

From now on we will work with $\underline{\beta} = (\beta_1, \dots, \beta_n) \in \left(\mathbb{F}_q^*\right)^n$. We need to find $\underline{y} = (y_1, \dots, y_n)$ and the nonsingular matrix $\tilde{M}$ such that $(\underline{\beta}, \underline{y}, \tilde{M})$ is a solution. We have already discussed that such $\underline{y}$ and $\tilde{M}$ exist.

In this section we will show how to recover $(y_1, \dots, y_{s+2})$. Let us define the matrix

$$Y = \operatorname{diag}(y_1, \dots, y_n)$$

and the set of indices

$$J_5 := \{1, 2, \dots, s+2\}.$$

Note that $|J_5| = s + 2$.

Observation. *From now on let us without loss of generality assume that $y_1 = 1$. We can do so because*

$$\begin{aligned} K &= \tilde{M} H(\underline{\beta}, \underline{1}) Y \\ &= y_1 \tilde{M} H(\underline{\beta}, \underline{1}) \frac{1}{y_1} Y \\ &= \hat{M} H(\underline{\beta}, \underline{1}) \tilde{Y}. \end{aligned}$$

For

$$\begin{aligned} \hat{M} &= y_1 \tilde{M}, \\ \tilde{Y} &= {y_1}^{-1} \cdot Y. \end{aligned}$$

Note that the matrix $\hat{M}$ is nonsingular.

Observation. *Let $A \in \mathbb{F}_q^{(s+1) \times n}$ such that $K = AY$. Then $A = \hat{M} H(\underline{\beta}, \underline{1})$.*

Proof. Lemma 25 gives us the existence of matrix A. By definition,

$$K = \hat{M} H(\underline{\beta}, \underline{1}) Y.$$

Matrix Y is nonsingular, thus $A = \hat{M} H(\underline{\beta}, \underline{1})$. □

Remark. Especially the following holds

$$A_{J_5 *} = \hat{M} \cdot H(\underline{\beta}, \underline{1})_{J_5 *}. \tag{4.13}$$

Definition 19. Let $\underline{u} = (u_1, \dots, u_{s+2}) \in \mathbb{F}_q^{s+2}$ such that $K_{* J_5} \cdot \underline{u}^\top = \underline{o}^\top$. Define $U = \operatorname{diag}(u_1, \dots, u_{s+2})$.

Observation. *Such $\underline{u}$ always exists as it is a solution to a system of $s+1$ equations and $s+2$ variables. Furthermore, if there were $i \in \{1, \dots, s+2\} :\ u_i = 0$ then the remaining $s+1$ columns have to be linearly dependent. However, that would be in contradiction with $d = s+2$ (see [Mac77, p. 33]). Thus, all coordinates of $\underline{u}$ are nonzero and U is nonsingular.*

We can easily see that

$$K_{J_5*}\underline{u}^\top = (AY)_{J_5*}\underline{u}^\top = A_{J_5*}U\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{s+2}\end{pmatrix}$$

and therefore

$$K_{J_5*}\underline{u}^\top = \underline{o}^\top \iff A_{J_5*}U\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{s+2}\end{pmatrix} = \underline{o}^\top.$$

If we knew the entries of A_{J_5*} we would now by linear algebra obtain the values of $y_2, \dots, y_{s+2}$ (remember we chose $y_1 = 1$). However, as an attacker we only know the entries of K_{J_5*} and are not able to "separate" the entries of A_{J_5*} from it. Luckily, we can multiply the latter equation by M^{-1} and take the advantage of Equation 4.13. This way we obtain

$$A_{J_5*}U\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{s+2}\end{pmatrix} = \underline{o}^\top \iff H(\underline{\beta}, \underline{1})_{J_5*}U\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{s+2}\end{pmatrix} = \underline{o}^\top.$$

We only need to change the equation slightly as we already know the value of y_1:

$$H(\underline{\beta}, \underline{1})_{J_5*}U\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{s+2}\end{pmatrix} = \underline{o}^\top$$

$$\iff$$

$$H(\underline{\beta}, \underline{1})_{J_6*} \cdot \operatorname{diag}(u_2, \dots, u_{s+2})\begin{pmatrix} y_2 \\ y_3 \\ \vdots \\ y_{s+2}\end{pmatrix} = -y_1 \cdot \begin{pmatrix} u_1\beta_1^0 \\ u_1\beta_1^1 \\ \vdots \\ u_1\beta_1^s\end{pmatrix}.$$

Where $J_6 = J_5 \setminus \{1\} = \{2, \dots, s+2\}$.

Hence, to obtain $y_1, \dots, y_{s+2}$, we only need to set $y_1 = 1$ and solve the system of linear equations

$$\begin{pmatrix} u_2 \cdot \beta_2^0 & u_3 \cdot \beta_3^0 & \dots & u_{s+2} \cdot \beta_{s+2}^0 \\ u_2 \cdot \beta_2^1 & u_3 \cdot \beta_3^1 & \dots & u_{s+2} \cdot \beta_{s+2}^1 \\ \vdots & & \ddots & \vdots \\ u_2 \cdot \beta_2^s & u_3 \cdot \beta_3^s & \dots & u_{s+2} \cdot \beta_{s+2}^s \end{pmatrix} \cdot \begin{pmatrix} y_2 \\ y_3 \\ \vdots \\ y_{s+2}\end{pmatrix} = \begin{pmatrix} -u_1\beta_1^0 \\ -u_1\beta_1^1 \\ \vdots \\ -u_1\beta_1^s\end{pmatrix}.$$

Lemma 30. *The system above always has a solution.*

Proof. The matrix $H(\underline{\beta},\underline{1})_{J6*}\cdot \operatorname{diag}(u_2,\dots,u_{s+2})$ is nonsingular, because its determinant is equal to

$$\prod_{i=2}^{s+2} u_i \cdot \det(H(\underline{\beta},\underline{1})_{J6*}) \neq 0$$

The inequality holds due to $\underline{u}$ having all coordinates nonzero and $H(\underline{\beta},\underline{1})_{J6*}$ having no $s+1$ linearly dependent columns. $\square$

4.1.8 Finding Matrix $\hat{M}$

After successfully recovering $y_1,\dots,y_{s+2}$ we find the matrix $\hat{M}$. Note that

$$k_{ij} = \sum_{k=1}^{s+1} y_j m_{ik}\beta_j^{k-1},$$

where m_{ik} are the entries of $\hat{M}$.

Let us fix i. Then we obtain a system of $s+1$ linear equations and $s+1$ unknowns:

$$\begin{pmatrix} \beta_1^0 & \beta_1^2 & \dots & \beta_1^s \\ \beta_2^0 & \beta_2^2 & \dots & \beta_2^s \\ & & \ddots & \\ \beta_{s+1}^0 & \beta_{s+1}^2 & \dots & \beta_{s+1}^s \end{pmatrix} \cdot \begin{pmatrix} m_{i1} \\ m_{i2} \\ \vdots \\ m_{is+1} \end{pmatrix} = \begin{pmatrix} y_1^{-1}k_{i1} \\ y_2^{-1}k_{i2} \\ \vdots \\ y_{s+1}^{-1}k_{is+1} \end{pmatrix}.$$

The matrix on the left-hand side is a Vandermonde matrix. We chose $\beta_1,\dots,\beta_n$ to be distinct. Hence, the matrix is nonsingular, and the system has a unique solution. Solving the system for all $i \in \{1,\dots,s+1\}$ gives all entries of $\hat{M}$.

4.1.9 Finding Nonzero Values $y_{s+3},\dots,y_n \in \mathbb{F}_q$

The last step of the attack is to recover the rest of $\underline{y}$. To obtain values $y_{s+3},\dots,y_n$ we only need to multiply the equation

$$K = \hat{M}H(\underline{\beta},\underline{y})$$

by $\hat{M}^{-1}$. That yields

$$\hat{M}^{-1}K = H(\underline{\beta},\underline{y}).$$

Focusing on the first row of the equation we obtain

$$\forall i \in \{1,\dots,n\}: \; y_i = \sum_{j=1}^{s+1} m'_{1j}k_{ji}.$$

Here m'_{1j} are the entries of the first row of $\hat{M}^{-1}$. Computing the equation for $i \geq s+3$ will give us $y_{s+3},\dots,y_n$. Note that we also need to compute the inverse of $\hat{M}$.

4.2 Sidelnikov-Shestakov's Attack on McEliece PKC Based on GRS Codes

In this section we will show how to break McEliece based on GRS codes using the attack described in the previous chapter.

4.2.1 Brief Overview of the Attack

Let $\mathcal{K}_{pub} = (G_{pub}, t)$ and $\mathcal{K}_{priv} = (S, G_{priv}, P)$ be the public and private key of McEliece PKC, respectively. Suppose the cryptosystem uses (n, k, d)-GRS codes over $\mathbb{F}_q$. Thus, there exist pairwise distinct $\alpha_1, \dots, \alpha_n \in \mathbb{F}_q$ and nonzero $z_1, \dots, z_n \in \mathbb{F}_q$ such that

$$\begin{aligned} G_{priv} &= \begin{pmatrix} \alpha_1^0 z_1 & \alpha_2^0 z_n & \dots & \alpha_n^0 z_n \\ \alpha_1^1 z_1 & \alpha_2^1 z_n & \dots & \alpha_n^1 z_n \\ & & \vdots & \\ \alpha_1^{k-1} z_1 & \alpha_2^{k-1} z_n & \dots & \alpha_n^{k-1} z_n \end{pmatrix} \in \mathbb{F}_q^{n \times k}, \\ G_{pub} &= S G_{priv} P \end{aligned}$$

where

$$\begin{aligned} S &\in \mathbb{F}_q^{n \times n} \text{ is a random nonsingular matrix,} \\ P &\in \mathbb{F}_q^{k \times k} \text{ is a random permutation matrix.} \end{aligned}$$

We will show how to find an alternative private key i.e., $\tilde{G}_{priv}, \tilde{S}$ such that $G_{pub} = \tilde{S}\tilde{G}_{priv}$. We do not have to search for an alternative permutation matrix as a permutation of GRS code is still a GRS code (see remark for the parity-check matrix on page 25).

Note that the sending party encrypts the message $\underline{m} \in \mathbb{F}_q^k$ as $\underline{c} = \underline{m}G + \underline{e} \in \mathbb{F}_q^n$. Where $\underline{e} \in \mathbb{F}_q^n$ is an error vector of $w_H(\underline{e}) = t$. We will show that an alternative private key is sufficient for decrypting any intercepted ciphertext.

Lemma 31. *Let (G_{pub}, t) be the public key. Suppose the attacker has found an alternative private key $(\tilde{S}, \tilde{G}_{priv})$. Then after intercepting a ciphertext $\underline{c} = \underline{m}G_{pub} + \underline{e}$ the attacker can recover $\underline{m}$.*

Proof. We can rewrite the ciphertext as

$$\underline{c} = \underline{m}\left(\tilde{S}\tilde{G}_{priv}\right) + \underline{e}.$$

Matrix $\tilde{G}_{priv}$ is a generator matrix of an GRS code, thus there exists an efficient decoding algorithm $\mathcal{D}_{\tilde{G}_{priv}}$ that on input $\underline{c}$ returns $\underline{m}\tilde{S}$. Now from $\tilde{S}$ we can compute $\tilde{S}^{-1}$ and subsequently obtain $\underline{m}$. □

4.2.2 Attacking McEliece PKC

We shall now demonstrate how to find an alternative private key knowing only G_{pub}.

Observation. *Let G be a generator matrix of an (n, k, d)-GRS code. By definition of generator and parity-check matrices of a GRS code (see Lemma 14) there exists an $(n, n-k, k+1)$-GRS code such that G is its parity-check matrix.*

We can therefore perform the attack as if we were attacking the Niederreiter version. The attack will then proceed as follows.

- Let $G_{pub} \in \mathbb{F}_q^{k \times n}$ be the known generator matrix.
- Set $K := G_{pub}$.

- Perform the attack from the previous chapter to reveal $(\underline{\beta}, \underline{y}, \hat{M})$ such that

$$K = \hat{M}H(\underline{\beta}, \underline{y}).$$

The matrix $H(\underline{\beta}, \underline{y})$ is as well a generator matrix of an (n, k, d)-GRS code.

- Put $\tilde{S} := \hat{M}$ and $\tilde{G}_{priv} := H(\underline{\beta}, \underline{y})$.

Lemma 32. *Knowing $\tilde{S}, \tilde{G}_{priv}$ computed as above the attacker can recover $\underline{m}$ from $\underline{c} = \underline{m}G_{pub} + \underline{e}$.*

Proof. We know that $G_{pub} = K = \hat{M}H(\underline{\beta}, \underline{y}) = \tilde{S}\tilde{G}_{priv}$. Thus, the attacker can successfully decrypt $\underline{c}$ as shown in proof of Lemma 31. □

4.3 Example of Sidelnikov-Shestakov's Attack

In this section we demonstrate Sidelnikov-Shestakov's attack on a simple example. We chose the parameters so that the example is not trivial, but a reader can still easily follow.

The example was computed by SageMath program that we attach as Appendix A.1.

4.3.1 Initialization

Parameters Our cryptosystem will be over the finite field

$$\mathbb{F}_3[\eta] = \frac{\mathbb{F}_3[X]}{(X^2 + X + 2)} \text{ with } \eta^2 + 2\eta + 2 = 0.$$

We will choose $n = 8$. We need to choose $k < n$ to fulfil $2s + 1 = 2(n - k - 1) \leq n$ (see 16). We choose $k = 4$.

Altogether the parameters of the code are

$$\begin{aligned} q &= 9, \\ n &= 8, \\ k &= 4, \\ d &= 5, \\ s &= 3. \end{aligned}$$

Key generation The legitimate user chooses vectors

$$\begin{aligned} \underline{\alpha} &= (\eta, \eta + 1, 2\eta + 1, 2, 2\eta, 2\eta + 2, \eta + 2, 1), \\ \underline{z} &= (2\eta, 2\eta + 2, \eta + 1, \eta + 1, 2\eta, \eta, 2, 2\eta + 2) \end{aligned}$$

and generates the secret parity-check matrix

$$H(\underline{\alpha}, \underline{z}) = \begin{pmatrix} 1 & 1 & 2\eta & 2\eta + 2 & 2 & \eta & \eta + 2 & 2\eta + 2 \\ \eta & \eta + 1 & 1 & \eta + 1 & \eta & \eta + 2 & 2\eta + 2 & 2\eta + 2 \\ \eta + 1 & 2 & 2\eta + 1 & 2\eta + 2 & 2\eta + 2 & 2\eta & 2\eta & 2\eta + 2 \\ 2\eta + 1 & 2\eta + 2 & 2\eta + 2 & \eta + 1 & 2\eta + 1 & 2\eta + 1 & 2 & 2\eta + 2 \end{pmatrix}.$$

The legitimate user then generates the permutation matrix

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{F}_9^{8\times 8}$$

which defines the permutation

$$(\ 1\ 5\ 6\ 2\ 8\ 3\ 7\ )(\ 4\ ).$$

The user also generates a nonsingular matrix

$$M = \begin{pmatrix} \eta+1 & 1 & \eta+2 & 2\eta+1 \\ 2 & 2 & \eta+1 & 2\eta \\ 1 & \eta+2 & 0 & \eta+1 \\ 2\eta & 0 & \eta+1 & \eta+1 \end{pmatrix} \in \mathbb{F}_9^{4\times 4}.$$

The legitimate user now computes the public parity-check matrix

$$\begin{aligned} K &= MHP \\ &= \begin{pmatrix} \eta+1 & 2\eta & \eta & 2\eta+2 & 2\eta+1 & 2\eta & 2\eta & 1 \\ 2\eta & \eta+1 & 2\eta+1 & \eta & 2\eta+1 & 1 & 2\eta+2 & 2\eta \\ 2\eta & \eta & 2\eta+1 & 1 & 2\eta+2 & \eta+2 & 2\eta+2 & 0 \\ 1 & 2\eta & 0 & 2\eta+1 & 2\eta+1 & \eta & \eta+2 & 2 \end{pmatrix}. \end{aligned}$$

The legitimate user computes $\lfloor \frac{d-1}{2} \rfloor = 2$ and publishes the public key

$$\mathcal{K}_{pub} = (K, 2)\,.$$

4.3.2 Attack

As an attacker we only know the public key $\mathcal{K}_{pub}$. We will now follow the algorithm shown in Section 4.1.

Recovering $\gamma_4, \dots, \gamma_8$

Let us follow Algorithm 4. Put

$$\begin{aligned} (\gamma_1, \gamma_2, \gamma_3) &:= (1, 0, \infty), \\ J_1 &:= \{1, 5, 6\}, \\ J_2 &:= \{2, 4, 5\}, \\ J_3 &:= \{1, 3, 4\}, \\ J_4 &:= \{2, 3, 4\}. \end{aligned}$$

For $l \in \{1, \dots, 4\}$ find $\underline{c_l} \in \mathbb{F}_9^5$ such that $\underline{c_l} K_{*J_l} = \underline{o}$. Let $l = 1$, then

$$K_{*J_1} = \begin{pmatrix} \eta+1 & 2\eta+1 & 2\eta \\ 2\eta & 2\eta+1 & 1 \\ 2\eta & 2\eta+2 & \eta+2 \\ 1 & 2\eta+1 & \eta \end{pmatrix}.$$

Gaussian elimination of $K_{*J_1}^\top$ gives us the solution $\underline{c_1} = (0, 1, 2\eta+2, 2\eta+2)$. Analogously we find

$$\begin{aligned} \underline{c_2} &= (1, 2\eta, 2\eta, \eta+1), \\ \underline{c_3} &= (1, \eta, 1, \eta), \\ \underline{c_4} &= (1, 2\eta+1, 2\eta, 2\eta+1) \end{aligned}$$

and compute

$$\begin{aligned} \underline{p_1} &= (0, \eta+1, 1, \eta+2, 0, 0, \eta+1, 1), \\ \underline{p_2} &= (\eta+1, 0, \eta+2, 2\eta+1, 0, 0, \eta+2, 1), \\ \underline{p_3} &= (0, \eta, 0, 0, \eta+1, 2\eta, 2\eta+2, \eta), \\ \underline{p_4} &= (\eta+1, 0, 0, 0, 2\eta, \eta+2, 2, \eta+1), \\ b_3 &= \frac{1}{\eta+2} = \eta. \end{aligned}$$

We can now easily find the values of b_j, γ_j for $j \in \{4, \dots, s+1, 2s+1, \dots, n\} = \{4, 7, 8\}$:

$$\begin{aligned} b_4 &= \frac{\eta+2}{2\eta+1} = 2 \Rightarrow \gamma_4 = \frac{\eta}{\eta-2} = \eta+2, \\ b_7 &= 2\eta+1 \qquad \Rightarrow \gamma_7 = 2\eta+1, \\ b_8 &= 1 \qquad \Rightarrow \gamma_8 = \eta+1. \end{aligned}$$

Analogously, we compute b_j, γ_j for $j \in \{s+2, \dots, 2s\} = \{5, 6\}$:

$$\begin{aligned} b_5 &= 2\eta+2 \Rightarrow \gamma_5 = 2\eta+2, \\ b_6 &= \eta+2 \Rightarrow \gamma_6 = \eta. \end{aligned}$$

We have obtained a part of a solution

$$\underline{\gamma} = (1, 0, \infty, \eta+2, 2\eta+1, \eta+1, 2\eta+2, \eta).$$

Transformation $\underline{\gamma} \in \mathbb{F}_\infty^8 \mapsto \underline{\beta} \in \mathbb{F}_9^8$

We will follow Lemma 29. Let us choose $a \in \mathbb{F}_q$ such that

$$a \notin \{1, 0, \infty, \eta+2, 2\eta+1, \eta+1, 2\eta+2, \eta\}.$$

We will, for instance, take $a = 2$. Now we can transform the obtained part of a solution as follows

$$\forall i \in \{1, \dots, 8\} : \ \gamma_i \mapsto \beta_i = \frac{1}{2 - \gamma_i}.$$

We obtained a new part of a solution $\underline{\beta} = (1, 2, 0, 2\eta+1, \eta+2, \eta+1, 2\eta+2, 2\eta)$.

Recover $y_1, \dots, y_5$

Assuming that $\left(\underline{\beta}, \underline{y}, \tilde{M}\right)$ is a solution we can recover $y_1, \dots, y_5$ as described in Section 4.1.7. Without loss of generality, we choose $y_1 = 1$.

We define $J_5 := \{1,2,3,4,5\}$ and find $\underline{u}$ such that $K_{*J_5}\underline{u}^\top = \underline{o}^\top$. Gaussian elimination gives us the solution $\underline{u} = (1, 2\eta+2, \eta+1, 2, \eta)$. Then solving the system

$$\begin{pmatrix} 2\eta+2 & \eta+1 & 2 & \eta \\ \eta+1 & 0 & \eta+2 & 1 \\ 2\eta+2 & 0 & \eta+1 & \eta+2 \\ \eta+1 & 0 & 2\eta & 2\eta+2 \end{pmatrix} \begin{pmatrix} y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 2\eta+1 \\ 0 \\ \eta \end{pmatrix}$$

gives us $(y_2, \dots, y_5) = (\eta+1, 2\eta+1, \eta+1, 2\eta)$.

Recover $\hat{M}$

Knowing the values $y_1, \dots, y_5$ we can recover the matrix $\hat{M}$ such that $\left(\underline{\gamma}, \underline{y}, \hat{M}\right)$ is a solution. Solving the three systems of linear equations as mentioned in Section 4.1.8, we obtain the entries of the matrix

$$\hat{M} = \begin{pmatrix} 2\eta+2 & 2\eta & \eta+2 & 2\eta \\ 1 & 2 & \eta+1 & \eta+2 \\ 1 & 2\eta & 0 & 2 \\ 0 & 2\eta+1 & \eta+1 & 2 \end{pmatrix}.$$

Recover y_6, y_7, y_8

As discussed in Section 4.1.9, we now set

$$\begin{aligned} y_6 &:= \sum_{j=1}^{4} m'_{1j} k_{j6}, \\ y_7 &:= \sum_{j=1}^{4} m'_{1j} k_{j7}, \\ y_8 &:= \sum_{j=1}^{4} m'_{1j} k_{j8}. \end{aligned}$$

To do so, we first need to compute the inverse of $\hat{M}$. This can be done in polynomial time and gives us

$$(\hat{M})^{-1} = \begin{pmatrix} 0 & \eta+1 & 2\eta+1 & 2\eta+2 \\ 2 & 2\eta+1 & 2\eta+2 & 2 \\ \eta & \eta+2 & 2\eta & 2\eta+1 \\ 2\eta & 1 & 2\eta+1 & 2\eta+1 \end{pmatrix}.$$

Thus,

$$\begin{aligned} y_6 &= 2, \\ y_7 &= 2, \\ y_8 &= \eta. \end{aligned}$$

Alternative private key

Altogether, we have found matrices

$$H(\underline{\gamma}, \underline{y}) = \begin{pmatrix} 1 & \eta+1 & 2\eta+1 & \eta+1 & 2\eta & 2 & 2 & \eta \\ 1 & 2\eta+2 & 0 & 2\eta & 2 & 2\eta+2 & \eta+1 & 2\eta+2 \\ 1 & \eta+1 & 0 & 1 & 2\eta+1 & 1 & 1 & 2\eta+1 \\ 1 & 2\eta+2 & 0 & 2\eta+1 & \eta+1 & \eta+1 & 2\eta+2 & 1 \end{pmatrix}$$

and

$$\hat{M} = \begin{pmatrix} 2\eta+2 & 2\eta & \eta+2 & 2\eta \\ 1 & 2 & \eta+1 & \eta+2 \\ 1 & 2\eta & 0 & 2 \\ 0 & 2\eta+1 & \eta+1 & 2 \end{pmatrix}.$$

We can easily verify that $\hat{M}H(\underline{\gamma}, \underline{y}) = K$, thus we have successfully recovered an alternative private key.

4.4 Modified Niederreiter PKC

Niederreiter PKC using GRS codes can be modified in order to resist the Sidelnikov Shestakov attack. In this section we will describe the modification from [Gab01, p. 41].

In this section we will return to notation H_{priv} and H_{pub} of the private and public matrix, respectively.

4.4.1 Description of the PKC

Parameters

- Choose $p \in \mathbb{P}$ and $m, n, t \in \mathbb{N}$ such that $t << n$.

Key generation

1. Choose $H_{priv} \in \mathbb{F}_q^{(n-k)\times n}$ to be a parity-check matrix of a linear (n, k)-code over $\mathbb{F}_q$ such that it can correct up to t errors and that there exists an efficient syndrome decoding algorithm $\mathcal{S}_{H_{priv}}$. Here k is chosen to be maximal given n and t.

2. Choose a random nonsingular matrix $S \in \mathbb{F}_q^{k\times k}$, random $\underline{b} \in \mathbb{F}_q^n$ and random $\underline{e} \in \mathbb{F}_q^n$ such that $w_H(\underline{e}) = d - 1$.

3. Set $\underline{a} := \underline{e}H_{priv}^\top$ and $X := \underline{a}^\top \underline{b}$.

4. Compute $H_{pub}^{mod} := S(H_{priv} + X)$.

5. Let $\mathcal{K}_{pub} = (H_{pub}^{mod}, t)$ be a public key and $\mathcal{K}_{priv} = (H_{priv}, S, \underline{a})$ be a private key.

Encryption The sending party encrypts a plaintext $\underline{m} \in \mathbb{F}_q^k$ such that $w_H(\underline{m}) < t$ as

$$\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{m}) = \underline{m}\left(H_{pub}^{mod}\right)^\top.$$

Decryption After receiving the ciphertext $\underline{c}$, the legitimate user knowing $\mathcal{K}_{priv}$ decrypts it using the following algorithm.

Algorithm 5: Decryption of Modified Niederreiter PKC

```
   Input: $\mathcal{K}_{pub}, \mathcal{K}_{priv}, \underline{c}$
   Output: $\underline{m}$ such that $\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{m}) = \underline{c}$
1  Compute $S^{-1}$ from $S$.
2  $\underline{\tilde{c}} := \underline{c}\left(S^{\top}\right)^{-1}$
3  for $\tilde{\lambda} \in \mathbb{F}_q$ do
4      if $\mathcal{S}_{H_{priv}}(\underline{\tilde{c}} - \tilde{\lambda}\underline{a})$ does not fail then
5          return $\underline{m} := \mathcal{S}_{H_{priv}}(\underline{\tilde{c}} - \tilde{\lambda}\underline{a})$
```

Theorem 33. *Set $\lambda := \underline{m} \cdot \underline{b}^{\top}$. $\mathcal{S}_{H_{priv}}(\underline{\tilde{c}} - \tilde{\lambda}\underline{a})$ returns fail if and only if $\lambda \neq \tilde{\lambda}$.*

Proof. First note that any $d-1$ columns of H_{priv} are linearly independent. Thus, $\underline{a} = \underline{e}H_{priv}^{\top} \neq \underline{o}$. Second note that $\underline{\tilde{c}}$ can be written as

$$\begin{aligned}\underline{\tilde{c}} &= \underline{m}\left(H_{pub}^{mod}\right)^{\top}\left(S^{\top}\right)^{-1} \\ &= \underline{m}\left(H_{priv} + X\right)^{\top} S^{\top}\left(S^{\top}\right)^{-1} \\ &= \underline{m}\left(H_{priv} + X\right)^{\top} \\ &= \underline{m}H_{priv}^{\top} + \underline{m}X^{\top} \\ &= \underline{m}H_{priv}^{\top} + \underline{m} \cdot \underline{b}^{\top}\underline{a} \\ &= \underline{m}H_{priv}^{\top} + \lambda\underline{a}.\end{aligned}$$

Suppose that $\lambda \neq \tilde{\lambda}$. Then

$$\begin{aligned}\underline{\tilde{c}} - \tilde{\lambda}\underline{a} &= \underline{m}H_{priv}^{\top} + \lambda\underline{a} - \tilde{\lambda}\underline{a} \\ &= \underline{m}H_{priv}^{\top} + \left(\lambda - \tilde{\lambda}\right)\underline{a} \\ &= \underline{m}H_{priv}^{\top} + \left(\lambda - \tilde{\lambda}\right)\underline{e}H_{priv}^{\top} \\ &= \left(\underline{m} + (\lambda - \tilde{\lambda})\underline{e}\right)H_{priv}^{\top}.\end{aligned}$$

Obviously,

$$\begin{aligned}w_H\left(\underline{m} + (\lambda - \tilde{\lambda})\underline{e}\right) &\geq w_H\left((\lambda - \tilde{\lambda})\underline{e}\right) - w_H\left(\underline{m}\right) \\ &\geq d - 1 - (t-1) \\ &\geq (2t+1) - 1 - t + 1 \\ &= t + 1 \\ &> t.\end{aligned}$$

Therefore, $\mathcal{S}_{H_{priv}}(\underline{\tilde{c}} - \tilde{\lambda}\underline{a})$ returns *fail*.

On the other hand, suppose that $\lambda = \tilde{\lambda}$. Then $\underline{\tilde{c}} - \tilde{\lambda}\underline{a} = \underline{m}H_{priv}^{\top}$. We can easily see that $\mathcal{S}_{H_{priv}}(\underline{\tilde{c}} - \tilde{\lambda}\underline{a}) = \underline{m}$. □

Corollary 34. *The decryption algorithm is correct, i.e. using the notation as above,* $\mathrm{Dec}_{\mathcal{K}_{priv}}\left(\mathrm{Enc}_{\mathcal{K}_{pub}}(\underline{m})\right) = \underline{m}$.

Remark. In [Gab01, p. 43] E. M. Gabidulin states that depending on the matrix X the PKC could still be vulnerable to Sidelnikov-Shestakov's attack.

5. GPT PKC

Another cryptosystem based on coding theory is the GPT cryptosystem introduced in 1991 by E. M. Gabidulin, A. V. Paramonov and O. V. Tretjakov ([GPT91]). It is a version of the McEliece PKC, but the authors modified the scheme to use codes with rank metric instead of Hamming metric. Of course, similar modification can be used to obtain a rank metric version of the Niederreiter PKC.

The used codes are so called Rank codes. Having been discovered by E. M. Gabidulin in 1985, they are also called Gabidulin codes. There has been a great amount of research regarding Gabidulin codes. An overview of the results can be found e.g., in [Gab21].

In this chapter we first give the necessary theory of rank metric and Gabidulin codes. We also define both the McEliece and the Niederreiter version of GPT PKC. The McEliece version will be introduced as GGPT PKC. Here GGPT stands for "Generalized GPT" which describes the original proposal as well as the column scrambler variant. The column scrambler variant was proposed by R. Overbeck in [Ove06] in order to make the PKC resistant to Gibson's attack described in [Gib96]. The Niederreiter version will be introduced as NGPT, which is an abbreviation that we introduce for the purposes of this thesis only. Then we briefly describe Overbeck's attack on GGPT PKC. Afterwards, we introduce a polynomial-time attack that we have found against GGPT PKC without distortion matrix X.

5.1 Gabidulin Codes

In this section we give the necessary background of rank metric and Gabidulin Codes. Note that $p \in \mathbb{P}$ and $m \in \mathbb{N}$.

5.1.1 Rank Metric

Rank metric is based on interpretation of $\mathbb{F}_{p^m}$ as vector space over $\mathbb{F}_p$.

Theorem 35. *[LN97, p. 31] The finite field $\mathbb{F}_{p^m}$ is an m-dimensional vector space over its prime subfield $\mathbb{F}_p$.*

Definition 20. Let $B = (b_1, \ldots, b_m) \in \mathbb{F}_{p^m}^m$ be a basis of $\mathbb{F}_{p^m}$ over $\mathbb{F}_p$. Let $x \in \mathbb{F}_{p^m}$ and $(\lambda_1, \ldots, \lambda_m) \in \mathbb{F}_p^m$ such that

$$x = \sum_{i=1}^{m} \lambda_i \cdot b_i.$$

Then $(\lambda_1, \ldots, \lambda_m)$ is said to be the coordinate vector of x with respect to B denoted by $[x]_B$.

Definition 21. Let $\underline{x} \in \mathbb{F}_{p^m}^n$. We define

$$\mathrm{Span}_{\mathbb{F}_p}(\underline{x}) := \{\sum_{i=1}^{n} \lambda_i x_i \mid \lambda_i \in \mathbb{F}_p\}.$$

Definition 22. Let B be a basis of $\mathbb{F}_{p^m}$ over $\mathbb{F}_p$. We define the rank of $\underline{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{p^m}^n$ over $\mathbb{F}_p$ as

$$\mathrm{Rank}_{\mathbb{F}_p}(\underline{x}) := \mathrm{Rank}\left(\left([x_1]_B^\top | \ldots | [x_n]_B^\top\right)\right).$$

Remark. By definition $\mathrm{Rank}_{\mathbb{F}_p}(\underline{x}) = \dim\left(\mathrm{Span}_{\mathbb{F}_p}(\underline{x})\right)$. Therefore, the value of $\mathrm{Rank}_{\mathbb{F}_p}(\underline{x})$ does not depend on the choice of the basis and is well defined.

Definition 23. We define the rank distance of $\underline{x}, \underline{y} \in \mathbb{F}_{p^m}^n$ as

$$d_R\left(\underline{x}, \underline{y}\right) = \mathrm{Rank}_{\mathbb{F}_p}\left(\underline{x} - \underline{y}\right).$$

Lemma 36. *The mapping $d_R : \mathbb{F}_{p^m}^n \times \mathbb{F}_{p^m}^n \to \mathbb{N}_0$ is a metric and does not depend on the choice of the basis.*

Proof. The independency of the choice of the basis is a straightforward corollary of the previous remark.

To prove that $d_R(\cdot, \cdot)$ is a metric, we prove the four properties of a metric.

1. Obviously, rank is alwys nonnegative, thus

$$\forall \underline{x}, \underline{y} \in \mathbb{F}_{p^m}^n : \ d_R\left(\underline{x}, \underline{y}\right) \geq 0.$$

2. We can easily see that

$$\forall \underline{x} \in \mathbb{F}_{p^m}^n : \ d_R(\underline{x}, \underline{x}) = 0.$$

Furthermore,

$$\forall \underline{x}, \underline{y} \in \mathbb{F}_{p^m}^n : \ d_R\left(\underline{x}, \underline{y}\right) = 0 \Rightarrow \underline{x} = \underline{y}.$$

3. To prove symmetry, we just need to realize that $-1 \in \mathbb{F}_p$ and therefore

$$\forall \underline{x}, \underline{y} \in \mathbb{F}_{p^m}^n : \ \mathrm{Rank}_{\mathbb{F}_p}\left(\underline{x} - \underline{y}\right) = \mathrm{Rank}_{\mathbb{F}_p}\left(\underline{y} - \underline{x}\right).$$

Hence, $d_R\left(\underline{x}, \underline{y}\right) = d_R\left(\underline{y}, \underline{x}\right)$.

4. Finally, we can see that

$$\forall U, V \in \mathbb{F}_p^{m \times n} : \ \mathrm{Rank}(U) + \mathrm{Rank}(V) \geq \mathrm{Rank}(U + V)$$

and therefore

$$\forall \underline{u}, \underline{v} \in \mathbb{F}_{p^m}^n : \ d_R(\underline{u}, \underline{o}) + d_R(\underline{v}, \underline{o}) \geq d_R(\underline{u} + \underline{v}, \underline{o}).$$

Especially for $\underline{u} := \underline{x} - \underline{y}$ and $\underline{v} := \underline{y} - \underline{z}$ we have

$$\begin{aligned} d_R\left(\underline{x} - \underline{y}, \underline{o}\right) + d_R\left(\underline{y} - \underline{z}, \underline{o}\right) &= d_R\left(\underline{x}, \underline{y}\right) + d_R\left(\underline{y}, \underline{z}\right) \\ &\geq d_R(\underline{x}, \underline{z}). \end{aligned}$$

□

Definition 24. We define the rank weight of $\underline{x} \in \mathbb{F}_{p^m}^n$ as $w_R(\underline{x}) := d_R(\underline{x}, \underline{o})$.

Definition 25. [Mar21, Definícia 1] Subsequently, the column $\mathbb{F}_p$-rank of an arbitrary matrix $A = (\underline{a_1}|\underline{a_2})|\dots|\underline{a_n}) \in \mathbb{F}_{p^m}^{k \times n}$ is defined as

$$\mathrm{Rank}_{\mathbb{F}_p}^{|}(A) = \dim\left(\mathrm{Span}_{\mathbb{F}_p}\left(\underline{a_1}, \dots, \underline{a_n}\right)\right).$$

5.1.2 $\mathbb{F}_p$-isomorphism and Notation

For brevity of notation let us define the following.

Definition 26. Let $\alpha \in \mathbb{F}_{p^m}$ and $i \in \mathbb{N}_0$. We define $\alpha^{[i]} := \alpha^{p^i}$.

Observation. *Note that*

$$\begin{aligned} \mathbb{F}_{p^m} &\to \mathbb{F}_{p^m}, \\ a &\mapsto a^{[i]} \end{aligned}$$

is an $\mathbb{F}_p$-isomorphism on $\mathbb{F}_{p^m}$.

Definition 27. Let

$$\begin{aligned} \varphi_i : \ \mathbb{F}_{p^m} &\to \mathbb{F}_{p^m}, \\ x &\mapsto xa^{[i]} \end{aligned}$$

We shall define $a^{[-i]}$ as the preimage of a under φ_i.

Observation.

$$\begin{aligned} \mathbb{F}_{p^m} &\to \mathbb{F}_{p^m}, \\ a &\mapsto a^{[-i]} \end{aligned}$$

is the inverse map to

$$\begin{aligned} \mathbb{F}_{p^m} &\to \mathbb{F}_{p^m}, \\ a &\mapsto a^{[i]} \end{aligned}$$

Observation. *Let $i, j \in \mathbb{Z}$ and $a \in \mathbb{F}_{p^m}$. Then*

$$\left(a^{[i]}\right)^{[j]} = \left(a^{p^i}\right)^{p^j} = a^{p^i \cdot p^j} = a^{p^{i+j}} = a^{[i+j]}.$$

Moreover, due to Lagrange's group theorem we know that in $\mathbb{F}_{p^m}$:

$$a^{[m]} = a^{p^m} = a.$$

That implies that

$$a^{[i]} = a^{[i \bmod m]}.$$

Definition 28. Let $i \in \mathbb{N}_0$ and $M \in \mathbb{F}_{p^m}^{k \times n}$ be an arbitrary matrix. We define

$$M^{[i]} := \begin{pmatrix} m_{11}^{[i]} & \cdots & m_{n1}^{[i]} \\ \vdots & \ddots & \vdots \\ m_{k1}^{[i]} & \cdots & m_{kn}^{[i]} \end{pmatrix}.$$

Let similarly $\underline{v} \in \mathbb{F}_{p^m}^k$ be an arbitrary vector. We define

$$\underline{v}^{[i]} := (v_1^{[i]}, \ldots, v_k^{[i]}).$$

5.1.3 Maximum Rank Distance Codes

If an (n, k, d)-Hamming distance code reaches the Singleton bound, we call it an MDS (maximum distance separable) code. An analogous family of codes exists for rank metric codes, they are called MRD (maximum rank distance) codes. It can be shown that Gabidulin codes are MRD codes.

Lemma 37 (Singleton-style Bound)**.** *[Gab01, Lemma 2] Let $n \leq m$ and $\mathcal{C}$ be an (n, k, d)-rank metric code over $\mathbb{F}_{p^m}$. Then $k \leq n - d + 1$.*

Definition 29. [Gab01, Lemma 2] Rank metric codes that reach the Singleton-style bound are called MRD (maximum rank distance) codes.

Definition 30. Let $\underline{g} = (g_1, \dots, g_n) \in \mathbb{F}_{p^m}^n$. We define

$$\langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,k} := \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^{[1]} & g_2^{[1]} & \cdots & g_n^{[1]} \\ \vdots & & \ddots & \vdots \\ g_1^{[k-1]} & g_2^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix} \in \mathbb{F}_{p^m}^{k \times n}.$$

Definition 31. [Ove05, Definition 3] Let $n \leq m$. Let $\underline{g} = (g_1, \dots, g_n) \in \mathbb{F}_{p^m}^n$ be a vector with entries linearly independent over $\mathbb{F}_p$. Then

$$\langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,k} \in \mathbb{F}_{p^m}^{k \times n}$$

is a generator matrix of a Gabidulin code over $\mathbb{F}_{p^m}$ of length n and dimension k.

Theorem 38. *[Gab21, Theorem 2.3] Any Gabidulin code over $\mathbb{F}_{p^m}$ is an MRD code.*

Corollary 39. *An (n, k)-Gabidulin code corrects $\lfloor \frac{n-k}{2} \rfloor$ errors.*

Theorem 40. *[Gab21, Theorem 2.3] Given a generator matrix $G = \langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,k}$ of an (n, k)-Gabidulin code, there exists a vector $\underline{h} = (h_1, \dots, h_n) \in \mathbb{F}_{p^m}^n$ such that*

$$\langle \underline{h} \rangle_{\mathbb{F}_{p^m}}^{n,n-k} \in \mathbb{F}_{p^m}^{(n-k) \times n}$$

is a parity-check matrix of the given code.

There exist efficient decoding and syndrome decoding algorithms for Gabidulin codes. An overview can be found e.g. in [Gab21, Chapter 4].

5.2 McEliece Version of GPT

We shall now describe the McEliece version of GPT PKC. This section is mainly based on Chapter 3 of [Ove05]. As mentioned before, the PKC will be denoted by GGPT which stands for ”Generalized GPT”.

System parameters

- $p \in \mathbb{P}$,
- $k, n, m \in \mathbb{N}$ such that $k < n \leq m$,
- $s, t \in \mathbb{N}$ such that $s \leq t < n - k - 1$.

Key generation

1. The legitimate user chooses
   - $G_{priv} := \langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,k} \in \mathbb{F}_{p^m}^{k\times n}$ a generator matrix of an (n,k)-Gabidulin code,
   - a random matrix $X \in \mathbb{F}_{p^m}^{k\times t}$ such that
   $$\begin{aligned}\text{Rank}|_{\mathbb{F}_p}(X) &= t,\\ \text{Rank}(X) &= s,\end{aligned}$$
   - a random nonsingular $S \in \mathbb{F}_{p^m}^{k\times k}$ (the row scrambler),
   - a random nonsingular $T \in \mathbb{F}_{p}^{n\times n}$ (the column scrambler).
2. The legitimate user now computes
$$G_{pub} = S\left(\left(X|0_{k\times(n-t)}\right) + G_{priv}\right)T \in \mathbb{F}_{p^m}^{k\times n}$$
3. The legitimate user chooses e such that $1 \le e \le \frac{n-k-t}{2}$.
4. The legitimate user publishes a public key
$$\mathcal{K}_{pub} = (G_{pub}, e)$$
   and keeps a private key
$$\mathcal{K}_{priv} = (G_{priv}, S, T).$$

Encryption The sending party wishes to send $\underline{m} \in \mathbb{F}_{p^m}^k$. They generate uniformly randomly an error vector $\underline{z} \in \mathbb{F}_{p^m}^n$ with $w_R(\underline{z}) = e$ Then they encrypt the message using the public key as
$$\underline{c} = \underline{m} \cdot G_{pub} + \underline{z}.$$

Decryption Upon receiving the above ciphertext $\underline{c}$, the legitimate party decrypts it as follows.

1. Compute $\hat{\underline{c}} := \underline{c} \cdot T^{-1}$.
2. Decode $\hat{\underline{c}}_{\restriction\{t+1,\dots,n\}}$ to obtain $\underline{m}S$.
3. Multiply the previous from right by S^{-1} (which the legitimate user can easily compute from knowing S) and obtain $\underline{m}$.

Theorem 41. *The decryption works, i.e., after decrypting $\underline{c}$ the legitimate user obtains the original plaintext $\underline{m}$.*

Proof. Let us use the same notation as in the description of decryption. The structure of the ciphertext is
$$\begin{aligned}\underline{c} &= \underline{m} \cdot G_{pub} + \underline{z}\\ &= \underline{m}\left(S\left(\left(X|0_{k\times(n-t)}\right) + G_{priv}\right)T\right) + \underline{z}\\ &= \underline{m}\left(\left(SX|0_{k\times(n-t)}\right)T + SG_{priv}T\right) + \underline{z}\\ &= \left(\underline{m}\cdot SX|0_{k\times(n-t)}\right)T + \underline{m}\cdot SG_{priv}T + \underline{z}.\end{aligned}$$

Multiplication by T^{-1} yields

$$\begin{aligned}
\underline{\hat{c}} &= \underline{c} \cdot T^{-1} \\
&= \underline{m}\left(SX|0_{k\times(n-t)}\right) + \underline{m}SG_{priv} + \underline{z}T^{-1}, \\
\underline{\hat{c}}_{\restriction\{1,\dots,t\}} &= \underline{m}SX + \underline{m}S\left((G_{priv})_{*\{1,\dots,t\}}\right) + \underline{z}\left(\left(T^{-1}\right)_{*\{1,\dots,t\}}\right) \in \mathbb{F}_{p^m}^{t}, \\
\underline{\hat{c}}_{\restriction\{t+1,\dots,n\}} &= \underline{m}S\left((G_{priv})_{*\{t+1,\dots,n\}}\right) + \underline{z}\left(\left(T^{-1}\right)_{*\{t+1,\dots,n\}}\right) \in \mathbb{F}_{p^m}^{n-t}.
\end{aligned}$$

We now wish to work with a code given by

$$\begin{aligned}
\hat{G} &= (G_{priv})_{*\{t+1,\dots,n\}} \\
&= \begin{pmatrix} g_{t+1} & g_2 & \cdots & g_n \\ g_{t+1}^{[1]} & g_2^{[1]} & \cdots & g_n^{[1]} \\ \vdots & & \ddots & \vdots \\ g_{t+1}^{[k-1]} & g_2^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix} \in \mathbb{F}_{p^m}^{k\times n-t}.
\end{aligned}$$

Such code is an $(n-t,k)$-Gabidulin code, thus it can correct $\lfloor\frac{n-t-k}{2}\rfloor$ errors. Note that vector $\underline{z}$ was chosen to have $w_R(\underline{z}) = e \le \frac{n-k-t}{2}$. Matrix $T \in \mathbb{F}_p^{n\times n}$ was chosen to be nonsingular, thus

$$w_R(\underline{z}) = w_R\left(\underline{z}T^{-1}\right) = e.$$

Therefore, $\underline{z}T^{-1}$ is an error that can be corrected in the code defined by $\hat{G}$.

The decoding algorithm returns $\mathcal{D}_{\hat{G}}(\underline{\hat{c}}_{\restriction\{t+1,\dots,n\}}) = \underline{m}S$. Obviously $\underline{m}$ can now be retrieved. □

In Section 5.5 we show that if the distortion matrix X was omitted, the cryptosystem would be breakable in polynomial time. However, we first show in Section 5.4 that under given assumptions, the PKC can be broken in polynomial time by Overbeck's attack.

5.3 Niederreiter Version of GPT

Analogously we can construct a version of Niederreiter cryptosystem that uses Gabidulin codes. We describe the version that can be found in Section 3.1 of [Ove05]. The cryptosystem will be denoted by NGPT for purposes of this thesis.

System parameters

- $p \in \mathbb{P}$,
- $l, k, n, m \in \mathbb{N}$ such that $l < k < n \le m$.

Key generation

1. The legitimate user chooses
   - a parity-check matrix $H_{priv} := \langle \underline{h} \rangle_{\mathbb{F}_{p^m}}^{n,n-k} \in \mathbb{F}_{p^m}^{n-k\times n}$ of an (n,k)-Gabidulin code,

- a random matrix $A \in \mathbb{F}_{p^m}^{l \times n}$ of full rank,
- a random nonsingular matrix $S \in \mathbb{F}_{p^m}^{(n-k+l) \times (n-k+l)}$.

2. The legitimate user computes

$$H_{pub} = S \begin{pmatrix} H_{priv} \\ A \end{pmatrix},$$
$$e = \left\lfloor \frac{n-k}{2} \right\rfloor .$$

3. The legitimate user publishes a public key

$$\mathcal{K}_{pub} = (H_{pub}, e)$$

and keeps a private key

$$\mathcal{K}_{priv} = (H_{priv}, S) .$$

Encryption The sending party wishes to send the plaintext $\underline{m}$ with $\mathrm{Rank}_{\mathbb{F}_p}(\underline{m}) = e$. They encrypt it as $\underline{c} = \underline{m} \cdot {H_{pub}}^{\top}$.

Decryption Upon receiving the ciphertext $\underline{c}$, the legitimate user computes

$$\hat{\underline{c}} := \underline{c} \left(S^{\top}\right)^{-1} .$$

To decrypt the message, it is sufficient to run the syndrome decoding algorithm on input $\hat{\underline{c}}_{\restriction\{1,\dots,n-k\}}$.

Theorem 42. *The decryption of NGPT works, i.e. the legitimate user obtains the original plaintext* $\underline{m}$.

Proof. The hidden structure of the ciphertext is

$$\underline{c} = \underline{m} \cdot \left(H_{priv}^{\top} | A^{\top}\right) S^{\top}.$$

Thus, $\hat{\underline{c}} = \underline{m} \cdot \left(H_{priv}^{\top} | A^{\top}\right)$. Obviously, $\hat{\underline{c}}_{\restriction\{1,\dots,n-k\}} = \underline{m} H_{priv}^{\top}$ is a syndrome in the code with parity-check matrix H_{priv}. Therefore, $\mathcal{S}_{H_{priv}}(\hat{\underline{c}}_{\restriction\{1,\dots,n-k\}})$ returns $\underline{m}$. $\square$

5.4 Overbeck's attack on GGPT PKC

In this section we will briefly describe Overbeck's attack on GGPT PKC. The chapter is based on Overbeck's [Ove06] as well as on M. Marko's Bachelor Thesis [Mar21]. More detailed descriptions of the attack can be found in these publications. Marko in his thesis works with a version of GGPT cryptosystem which is under natural assumption more general, we shall denote it by MGGPT.

5.4.1 MGGPT PKC

System parameters

- $p \in \mathbb{P}$,
- $k, n, m \in \mathbb{N}$ such that $k < n \leq m$,
- $s, t \in \mathbb{N}$ such that $s \leq k$ and $s \leq t$.

Key generation

1. The legitimate user chooses
   - $G_{priv} := \langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,k} \in \mathbb{F}_{p^m}^{k \times n}$ a generator matrix of an (n, k)-Gabidulin code,
   - a random matrix $X \in \mathbb{F}_{p^m}^{k \times t}$ such that
   $$\begin{aligned} \mathrm{Rank}_{\mathbb{F}_p}^{|} (X) &= t, \\ \mathrm{Rank} (X) &= s, \end{aligned}$$
   - a random nonsingular $S \in \mathbb{F}_{p^m}^{k \times k}$ (the row scrambler),
   - a random nonsingular $T \in \mathbb{F}_p^{(n+t) \times (n+t)}$ (the column scrambler).
2. The legitimate user now computes
$$G_{pub} = S \left(X | G_{priv} \right) T \in \mathbb{F}_{p^m}^{k \times (n+t)}.$$
3. The legitimate user chooses e such that $1 \leq e \leq \frac{n-k}{2}$.
4. The legitimate user publishes a public key
$$\mathcal{K}_{pub} = (G_{pub}, e)$$
and keeps a private key
$$\mathcal{K}_{priv} = (G_{priv}, S, T).$$

Encryption The sending party wishes to send $\underline{m} \in \mathbb{F}_{p^m}^k$. They generate uniformly randomly an error vector $\underline{z} \in \mathbb{F}_{p^m}^n$ with $w_R (\underline{z}) = e$. Then they encrypt the message using the public key as follows
$$\underline{c} = \underline{m} \cdot G_{pub} + \underline{z}.$$

Decryption Upon receiving the ciphertext $\underline{c}$, the legitimate party decrypts it as follows.

1. Compute $\hat{\underline{c}} := \underline{c} \cdot T^{-1}$.
2. Decode $\hat{\underline{c}}_{\{t+1,\dots,n+t\}}$ to obtain $\underline{m}S$.
3. Multiply the previous from right by S^{-1} (which the legitimate user knows from having S) and obtain $\underline{m}$.

Remark. The MGGPT PKC is more general version of GGPT PKC introduced in Section 5.2. In GGPT the first t columns of

$$\left(\left(X|0_{k\times(n-t)}\right)+G_{priv}\right)$$

equal

$$X+\left(G_{priv}\right)_{*\{1,\dots,t\}}.$$

In MGGPT PKC the first t columns of $(X|G_{priv})$ equal X. In other words, in GGPT PKC we mask the structure of $\left(G_{priv}\right)_{*\{1,\dots,t\}}$ by adding X. However, in MGGPT PKC we take only X (which is an unstructured random matrix).

Observation. *Let*

$$\begin{aligned} S &\in \mathbb{F}_{p^m}^{k\times k},\\ \tilde{\tilde{X}} &\in \mathbb{F}_{p^m}^{k\times t},\\ T &\in \mathbb{F}_{p}^{n\times n},\\ G_{priv} &\in \mathbb{F}_{p^m}^{k\times n},\\ \tilde{\tilde{G}}_{pub} &= S\left(\left(\tilde{\tilde{X}}|0_{k\times(n-t)}\right)+G_{priv}\right)T \end{aligned}$$

to be matrices used in a GGPT PKC. If

$$\begin{aligned} \operatorname{Rank}|_{\mathbb{F}_p}\left(\tilde{\tilde{X}}+G_{*\{1,\dots,t\}}\right) &= t,\\ \operatorname{Rank}\left(\tilde{\tilde{X}}+G_{*\{1,\dots,t\}}\right) &= s \end{aligned}$$

then the GGPT PKC can be written as MGGPT PKC.

Consider MGGPT PKC with matrices

$$\begin{aligned} S &\in \mathbb{F}_{p^m}^{k\times k},\\ X &= \tilde{\tilde{X}}+\left(G_{priv}\right)_{*\{1,\dots,t\}} \in \mathbb{F}_{p^m}^{k\times t},\\ T &\in \mathbb{F}_{p}^{n\times n},\\ G_{priv} &\in \mathbb{F}_{p^m}^{k\times n},\\ G_{pub} &= S\left(X|G_{priv}\right)T. \end{aligned}$$

We can easily see that

$$\begin{aligned} G_{pub} &= S\left(\tilde{\tilde{X}}+\left(G_{priv}\right)_{*\{1,\dots,t\}}|G_{priv}\right)T\\ &= S\left(\left(\tilde{\tilde{X}}|0_{k\times(n-t)}\right)+G_{priv}\right)T\\ &= \tilde{\tilde{G}}_{pub}. \end{aligned}$$

Furthermore, $\operatorname{Rank}|_{\mathbb{F}_p}(X)=t$ *and* $\operatorname{Rank}(X)=s$.

Remark. Note that the assumptions

$$\begin{aligned} \operatorname{Rank}|_{\mathbb{F}_p}\left(\tilde{\tilde{X}}+G_{*\{1,\dots,t\}}\right) &= t,\\ \operatorname{Rank}\left(\tilde{\tilde{X}}+G_{*\{1,\dots,t\}}\right) &= s \end{aligned}$$

are natural ones. If the ranks were smaller, the PKC would be weaker. In the extreme case we could get zero columns in the public matrix $\tilde{\tilde{G}}_{pub}$. The assumption ensures the best behaviour.

5.4.2 Preliminaries

Definition 32. Let I be a subset of $\mathbb{N}$. Let us define mapping $\pi_I : I \to \mathbb{N}$ that maps $i \in I$ to its position in I when ordered by the standard less-than-or-equal relation.

Definition 33. Let $M \in \mathbb{F}_{p^m}^{l\times n}$ and $f \in \mathbb{N}$. We define

$$\Lambda_f(M) := \begin{pmatrix} M \\ M^{[1]} \\ M^{[2]} \\ \vdots \\ M^{[f]} \end{pmatrix} \in \mathbb{F}_{p^m}^{(f+1)l\times n}.$$

Lemma 43. *[Ove05, Lemma 1], [Mar21, Lemma 31] If $G = \langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,k}$ (i.e. it is a generator matrix of a Gabidulin code) and $f \le n-k-1$ then the rows of $\Lambda_f(G)$ generate an $(n, k+f)$-Gabidulin code.*

The attack takes advantage of the structure of $\Lambda_f(G)$ for G a generator matrix of a Gabidulin code. An attacker knows the pulic key

$$\mathcal{K}_{pub} = (G_{pub}, e)$$

and similarly as in the case of Sidelnikov-Shestakov's attack they wish to find an alternative private key

$$\mathcal{K}_{priv}' = \left(\tilde{G}_{priv}, \tilde{S}, \tilde{T}\right).$$

After obtaining the alternative private key, they can recover the secret plaintext from any intercepted ciphertext.

Assume that the attacker has recovered an alternative private key

$$\begin{aligned} \tilde{G}_{priv} &\in \mathbb{F}_{p^m}^{k\times n}, \\ \tilde{S} &\in \mathbb{F}_{p^m}^{k\times k}, \\ \tilde{T} &\in \mathbb{F}_{p}^{k\times k}. \end{aligned}$$

Upon intercepting the ciphertext $\underline{c}$ the attacker multiplies it to obtain

$$\hat{\underline{c}} := \underline{c} \cdot \left(\tilde{T}\right)^{-1}.$$

They then treat $\hat{\underline{c}}$ as a word in the code $\mathcal{C}_{\tilde{G}_{priv}}$ and decode it to obtain $\underline{m}\tilde{S}$. Finally, they multiply the vector by $\left(\tilde{S}\right)^{-1}$ to obtain $\underline{m}$.

Observation. *It is obvious that the attacker's decryption works. The only nontrivial part is that $w_R\left(\underline{z}(\tilde{T})^{-1}\right) = w_R(\underline{z})$ which holds by definition due to the $T \in \mathbb{F}_p^{n\times n}$ being nonsingular.*

5.4.3 Overbeck's Attack

The algorithm we describe can be found in [Mar21, Chapter 3].

Algorithm 6: Overbeck's Attack

```
Input: generator matrix of a Gabidulin code $G_{pub} \in \mathbb{F}_{p^m}^{n \times k}$
Output: a generator matrix $\tilde{G}_{priv} \in \mathbb{F}_{p^m}^{k \times n}$ of an $(n, k)$-Gabidulin code,
matrix $\tilde{X} \in \mathbb{F}_{p^m}^{k \times t}$,
nonsingular matrix $\tilde{T} \in \mathbb{F}_p^{n \times n}$ such that $G_{pub} = (\tilde{G}_{priv}|\tilde{X}) \cdot \tilde{T}$
1 Compute $\hat{G}_{pub} := \Lambda_{n-k-1}(G_{pub})$.
2 Compute parity-check matrix $\underline{u}$ of the code generated by $\hat{G}_{pub}$.
3 Choose $I \subseteq \{1, \dots, n+t\}$ such that $|I| = n$ and $w_R\left(\underline{u}_{\restriction I}\right) = n$.
4 Set $J := \{1, \dots, n+t\} \setminus I$.
5 Compute matrix $\hat{T} = (\hat{t}_{ij})_{i,j=1}^{n+t}$ such that $\underline{u}_{\restriction J} = \underline{u}_{\restriction I}\hat{T}$.
6 Compute matrix $\tilde{T}^{-1} = (d_{ij})_{i,j=1}^{n+t}$ where
  $$d_{ji} = \begin{cases} 1 & \text{if } j \in J, i = \pi_J(j) \text{ or } j \in I, i = \pi_I(j) + t, \\ \hat{t}_{(i-t)\pi_J(j)} & \text{if } j \in J, i > t, \\ 0 & \text{otherwise.} \end{cases}$$
7 Compute $\tilde{T}$.
8 Compute
  $$\left(\tilde{X}|\tilde{G}_{priv}\right) := G_{pub} \cdot \tilde{T}^{-1}.$$
9 Return $(\tilde{G}_{priv}, \tilde{T})$.
```

Theorem 44. *[Mar21, p. 21] Let $\underline{g} \in \mathbb{F}_{p^m}^n$ such that $G = \langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,k}$. Then there exist matrices $Y \in \mathbb{F}_{p^m}^{(n-k-1)(k-1) \times t}$, $Z \in \mathbb{F}_{p^m}^{(n-1) \times t}$ and nonsingular matrices $\tilde{S}, R \in \mathbb{F}_{p^m}^{(n-k)k \times (n-k)k}$*

$$\hat{G}_{pub} = \hat{S} \begin{pmatrix} Z & G_{n-1} \\ Y & 0_{(n-k-1)(k-1) \times n} \end{pmatrix} T,$$

where $G_{n-1} = \langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,n-1}$.

Theorem 45. *[Mar21, Lemma 32 and p. 22-23] Let us use the notation from the previous lemma. If* $\operatorname{Rank}(Y) = t$ *then $\exists \underline{u} \in \mathbb{F}_{p^m}^{n+t}, \exists I \subseteq \{1, \dots, n+t\}, |I| = n$ such that*

- $w_R\left(\underline{u_{\restriction I}}\right) = n$,
- $w_R\left(\underline{u}\right) = n$,
- $\underline{u} = (\underline{o} \mid \underline{u}_{\restriction I})(\tilde{T}^{-1})^{\top}$ *is parity-check matrix of the code generated by $\hat{G}_{pub}$.*

Remark. The matrix $(\tilde{T}^{-1})$ was constructed in the algorithm in the way so that the theorem would hold.

Theorems so far have showed that if $\operatorname{Rank}(Y) = t$, the algorithm proceeds to Step 8. Obviously, in such case the algorithm terminates in Step 9. Moreover, we can easily see that

$$\left(\tilde{X}|\tilde{G}_{priv}\right) \cdot \tilde{T} = \left(G_{pub} \cdot \tilde{T}^{-1}\right) \cdot \tilde{T} = G_{pub}.$$

Therefore, if the algorithm terminates, it returns an alternative private key.

It can be shown that if $\operatorname{Rank}(Y) = t$ the algorithm terminates in polynomial time.

Theorem 46. *[Ove05, Theorem 3], [Mar21, p. 24] If* $\operatorname{Rank}(Y) = t$ *(using the same notation as in Theorem 44) then the attack succeeds in $\mathcal{O}((n+t)^3)$ operations over $\mathbb{F}_{p^m}$.*

5.5 GGPT PKC Without Distortion Matrix X

In this section we describe a deterministic algorithm for breaking GGPT PKC without the distortion matrix X.

Let us use the notation from the previous chapter. Assume that we chose

$$X = 0_{k\times n}.$$

Then we obtain

$$\begin{aligned} G_{pub} &= SG_{priv}T, \\ \mathcal{K}_{pub} &= (G_{pub}, e), \\ \mathcal{K}_{priv} &= (G_{priv}, S, T). \end{aligned}$$

The enciphered message now looks like

$$\underline{c} = \underline{m} \cdot G_{pub} + \underline{z} = \underline{m} \cdot SG_{priv}T + \underline{z}.$$

Definition 34. We shall for brevity denote such PKC by GGPTX PKC.

R. Overbeck states in [Ove05, p. 52] that such version of the cryptosystem would be vulnerable to attack similar to the one by Sidelnikov and Shestakov ([SS92]) described in Section 4.1 of this thesis.

We tried constructing such attack but could not proceed. The reason is described in Section 5.5.2. Instead we constructed an attack slightly similar to Overbeck's attack from [Ove05] (described in Section 5.4 of this thesis), that can be found in Section 5.5.3. The attack breaks the PKC by finding an alternative private key.

5.5.1 Properties of GGPTX PKC

We first discuss basic properties of GGPTX PKC. The properties are very similar to those of Niederreiter PKC based on GRS codes discussed in Section 4.1.

Theorem 47. *Let matrices G_{priv} and T be as defined in GGPTX cryptosystem above, i.e.,*

$$\begin{aligned} &G_{priv} = \langle \underline{g} \rangle^{n,k}_{\mathbb{F}_{p^m}} \in \mathbb{F}^{k\times n}_{p^m}, \\ &T \in \mathbb{F}^{n\times n}_{p}, \text{ such that } \operatorname{Rank}(T) = n. \end{aligned}$$

Then there exists a vector $\underline{\hat{g}} \in \mathbb{F}^n_{p^m}$ such that $G_{priv}T = \langle \underline{\hat{g}} \rangle^{n,k}_{\mathbb{F}_{p^m}}$.

Proof. To prove the theorem, we just need to examine the entries of $G_{priv} \cdot T$. Let us denote the entries of T by t_{ij}. We obtain

$$\begin{aligned} \forall i \in \{1,\dots,k\} \forall j \in \{1,\dots,n\} : \ (G_{priv}\cdot T)_{ij} &= \sum_{l=1}^{n} g_l^{[i-1]} \cdot t_{lj} \\ &= \sum_{l=1}^{n} (g_l \cdot t_{lj})^{[i-1]} \\ &= \left(\sum_{l=1}^{n} g_l \cdot t_{lj} \right)^{[i-1]}. \end{aligned}$$

In the last two equalities we used the first observation from Section 5.1.2. Set $\hat{g}_j := \sum_{l=1}^{n} g_l \cdot t_{lj} : \ \forall j \in \{1,\dots,n\}$. Then $G_{priv} \cdot T = \langle \underline{\hat{g}} \rangle^{n,k}_{\mathbb{F}_{p^m}}$. □

Observation. *Using the same notation as above, the matrices G_{priv} and $G_{priv} \cdot T$ are both generator matrices of an (n, k, d)-Gabidulin code. Hence, multiplication by T changes the code but not the parameters.*

Remark. Note that Niederreiter PKC based on GRS codes has similar property. In that case matrices H and HP are both parity-check matrices of an (n, k)-GRS code (see discussion on page 25).

Remark. From now on we will not consider matrix T as part of the private key of the PKC. The reason is same as discussed in Section 4.1.3.

Theorem 48. *Let $\mathcal{K}_{pub} = (G_{pub}, e)$ and $\mathcal{K}_{priv} = (G_{priv}, S, T)$ be the keys of GGPTX PKC. Suppose that $\underline{c} = \underline{m}G_{pub} + \underline{z} \in \mathbb{F}_{p^m}^n$ is a ciphertext that the attacker has intercepted. Furthermore, assume that the attacker knows an alternative private key $\tilde{\mathcal{K}}_{priv} = (\tilde{G}_{priv}, \tilde{S}, I_n)$ such that $G_{pub} = \tilde{S}\tilde{G}_{priv}$. The attacker can then decipher $\underline{c}$ to obtain $\underline{m}$.*

Proof. The ciphertext can be viewed as

$$\begin{aligned} \underline{c} &= \underline{m}G_{pub} + \underline{z} \\ &= \underline{m}(\tilde{S}\tilde{G}_{priv}) + \underline{z} \\ &= (\underline{m}\tilde{S})\tilde{G}_{priv} + \underline{z}. \end{aligned}$$

Vector $\underline{z}$ is an error that can be corrected, thus $\mathcal{D}_{\tilde{G}_{priv}}(\underline{c})$ returns $\underline{m}\tilde{S}$. Matrix $\tilde{S}$ is nonsingular, therefore the attacker can easily obtain $\underline{m}$. $\square$

Remark. The aim of the attack we will construct is for the attacker to retrieve an alternative private key from the known public key.

5.5.2 Comments on Attack Based on Sidelnikov-Shestakov's Attack

We tried to construct an attack similar to Sidelnikov-Shestakov's one. However, we were not able to proceed. Namely, we could not convert the mapping

$$\begin{aligned} \varphi :\ \mathbb{F}_{p^m} &\to \mathbb{F}_{p^m}, \\ x &\mapsto \frac{1}{x} \end{aligned}$$

to a linear mapping. Let us, $\forall k \in \mathbb{N}$ define the extended mapping as

$$\begin{aligned} \underline{\varphi} :\ \mathbb{F}_{p^m}^k &\to \mathbb{F}_{p^m}^k, \\ (x_1, \dots, x_k) &\mapsto (\varphi(x_1), \dots, \varphi(x_k). \end{aligned}$$

Suppose that $\underline{g} \in \mathbb{F}_{p^m}^n$ such that $\mathrm{Rank}_{\mathbb{F}_p}\left(\underline{g}\right) = n$. In order to use the principles of Sidelnikov-Shestakov's attack we wish to find a nonsingular matrix $S \in \mathbb{F}_{p^m}^{k \times k}$ such that

$$S\langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,k} = \langle \underline{\varphi}(\underline{g}) \rangle_{\mathbb{F}_{p^m}}^{n,k}.$$

Let us show an example when such matrix S does not exist.

Example. Consider a finite field

$$\mathbb{F} = \mathbb{F}_2[a] = \frac{\mathbb{F}_2[X]}{X^3 + X + 1} \text{ with } a^3 + a + 1 = 0.$$

Suppose that $\underline{g} = (1, a, a^2)$ and consider a generator matrix of a $(3, 2)$-Gabidulin code

$$G = \langle \underline{g} \rangle_{\mathbb{F}}^{3,2} = \begin{pmatrix} 1 & a & a^2 \\ 1 & a^2 & a^2 + a \end{pmatrix}.$$

We wish to find a nonsingular matrix $S \in \mathbb{F}^{2\times 2}$ such that

$$SG = \begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix} \cdot \begin{pmatrix} 1 & a & a^2 \\ 1 & a^2 & a^2 + a \end{pmatrix} = \begin{pmatrix} 1 & a^2+1 & a^2+a+1 \\ 1 & a^2+a+1 & a+1 \end{pmatrix} = \langle \underline{\varphi}(\underline{g}) \rangle_{\mathbb{F}}^{3,2}$$

Note that φ denotes an inversion on $\mathbb{F}_{p^m}$.

That gives us two systems of linear equations with $s_{11}, s_{12}, s_{21}, s_{22}$ as unknown variables:

$$\begin{pmatrix} 1 & 1 \\ a & a^2 \\ a^2 & a^2+a \end{pmatrix} \begin{pmatrix} s_{11} \\ s_{12} \end{pmatrix} = \begin{pmatrix} 1 \\ a^2+1 \\ a^2+a+1 \end{pmatrix},$$
$$\begin{pmatrix} 1 & 1 \\ a & a^2 \\ a^2 & a^2+a \end{pmatrix} \begin{pmatrix} s_{21} \\ s_{22} \end{pmatrix} = \begin{pmatrix} 1 \\ a^2+a+1 \\ a+1 \end{pmatrix}.$$

Matrices on the left-hand side can be reduced to row echelon form which gives us

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} s_{11} \\ s_{12} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} s_{21} \\ s_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Obviously, these systems of linear equations have no solution. Therefore, such $s_{11}, s_{12}, s_{21}, s_{22}$ do not exist.

▲

Instead of constructing an attack that would be similar to Sidelnikov-Shestakov's one, we found an attack slightly similar to Overbeck's attack.

5.5.3 Attack on GGPTX PKC

Preliminaries

Throughout this section, let us follow the usual notation. That is

$$\begin{aligned} G_{priv} &= \langle \underline{g} \rangle_{\mathbb{F}_{p^m}}^{n,k} && \text{is a private generator matrix of a Gabidulin code,} \\ S &\in \mathbb{F}_{p^m}^{k\times k} && \text{is a private nonsingular matrix,} \\ G_{pub} &= SG_{priv} && \text{is a public generator matrix.} \end{aligned}$$

Remark. As in Section 5.1.2 we will define $M^{[i]}, \underline{v}^{[i]}$ for any matrix and vector over $\mathbb{F}_{p^m}$.

Notation. Let us denote by K a parity-check matrix of the code defined by G_{pub}.

Observation. *As mentioned before,*

$$\begin{aligned} \mathbb{F}_{p^m} &\to \mathbb{F}_{p^m}, \\ a &\mapsto a^{[i]} \end{aligned}$$

is an $\mathbb{F}_p$-isomorphism on $\mathbb{F}_{p^m}$. Thus

$$\begin{aligned} \forall i \in \mathbb{Z} : \ K^{[i]} {G_{pub}^{[i]}}^\top &= 0_{n-k \times k}, \\ \forall i \in \mathbb{Z} : \qquad G_{pub}^{[i]} &= S^{[i]} G_{priv}^{[i]}. \end{aligned}$$

Having defined the basics, the attack can be written as follows.

```
Algorithm 7: Attack on GGPTX
  Input: generator matrix of a Gabidulin code $G_{pub} \in \mathbb{F}_{p^m}^{n \times k}$
  Output: nonsingular matrix $\hat{S} \in \mathbb{F}_{p^m}^{k \times k}$,
  generator matrix of a Gabidulin code $\tilde{G}_{priv} \in \mathbb{F}_{p^m}^{n \times k}$ such that $\hat{S}\tilde{G}_{priv} = G_{pub}$
1 Find a parity-check matrix $K$ of the code defined by the public matrix $G_{pub}$.
2 Compute matrix $\Lambda_{k-1}(K)$.
3 Find $\underline{\tilde{g}}$ such that $\Lambda_{k-1}(K)\underline{\tilde{g}}^\top = \underline{o}$.
4 for $i \in \{0, \dots, k-1\}$ do
5     Find $\underline{\tilde{s}}_{k-i}$ such that $\underline{\tilde{g}}^{[-i]} = \underline{\tilde{s}}_{k-i} G_{pub}$.
6 Define
      $$\tilde{S} = \begin{pmatrix} \underline{\tilde{s}}_1 \\ \vdots \\ \underline{\tilde{s}}_k \end{pmatrix}.$$
7 Compute $\hat{S} := \left(\tilde{S}\right)^{-1}, \tilde{G}_{priv} := \tilde{S} G_{pub}$.
8 return $\hat{S}, \tilde{G}_{priv}$
```

Correctness of the Attack

Notation. Let us for brevity of notation denote the subspaces of $\mathbb{F}_{p^m}^n$ by

$$\begin{aligned} \forall i \in \{0, \dots, k-1\} : \ \mathcal{P}_{pub_i} &:= \operatorname{Span}\left(\left(G_{pub}^{[i]}\right)^\top\right), \\ \forall i \in \{0, \dots, k-1\} : \ \mathcal{P}_{priv_i} &:= \operatorname{Span}\left(\left(G_{priv}^{[i]}\right)^\top\right), \\ \mathcal{P}_{pub} &:= \bigcap_{i=0}^{k-1} \operatorname{Span}\left(\left(\mathcal{P}_{pub_i}\right)^\top\right), \\ \mathcal{P}_{priv} &:= \bigcap_{i=0}^{k-1} \operatorname{Span}\left(\left(\mathcal{P}_{priv_i}\right)^\top\right). \end{aligned}$$

Lemma 49. *Assume that $M \in \mathbb{F}_{p^m}^{k\times k}$ is an arbitrary matrix and $i \in \mathbb{Z}$. Then M is nonsingular if and only if, $M^{[i]}$ is nonsingular.*

Proof. It is sufficient to examine the determinants of both matrices. By definition of a determinant and the fact that exponentiation to $[i]$ is a homomorphism on $\mathbb{F}_{p^m}$ we obtain

$$\det(M^{[i]}) = (\det(M))^{[i]}.$$

Therefore,

$$\det(M) = 0 \iff \det(M^{[i]}) = 0.$$

□

Corollary 50. *Let $M \in \mathbb{F}_{p^m}^{k\times k}$ and $i \in \mathbb{Z}$. Then* $\operatorname{Rank}\left(M^{[i]}\right) = \operatorname{Rank}(M)$.

Proof. To prove the corollary it is sufficient to recall that the rank of an arbitrary matrix is equal to the number of rows of its biggest nonsingular submatrix (see e.g. [Par20, p. 88]). □

Corollary 51. *Let S be the nonsingular matrix from GGPTX PKC. Then $S^{[i]}$ is nonsingular $\forall i \in \mathbb{Z}$.*

Lemma 52. *For all $i \in \{0, \dots, k-1\}$ we have $\mathcal{P}_{pub_i} = \mathcal{P}_{priv_i}$.*

Proof. As shown before, we know that

$$\forall i \in \{0, \dots, k-1\}: \; G_{pub}^{[i]} = S^{[i]} G_{priv}^{[i]}.$$

Furthermore, as $S^{[i]}$ is nonsingular (see the previous lemma), we have by definition:

$$\begin{aligned}
\mathcal{P}_{pub_i} &= \operatorname{Span}\left(\left(G_{pub}^{[i]}\right)^\top\right) \\
&= \operatorname{Span}\left(\left(S^{[i]} G_{priv}^{[i]}\right)^\top\right) \\
&= \operatorname{Span}\left(\left(G_{priv}^{[i]}\right)^\top\right) \\
&= \mathcal{P}_{priv_i}.
\end{aligned}$$

□

Remark. A straightforward corollary is that $\mathcal{P}_{priv} = \mathcal{P}_{pub}$.

Lemma 53. *Suppose that K is a parity-check matrix of the code generated by public generator matrix G_{pub}. Then $\forall i \in \mathbb{N}: \; K^{[i]}$ is a parity-check matrix of the code generated by $G_{pub}^{[i]}$.*

Proof. First, suppose that $\underline{c} \in \mathbb{F}_{p^m}^n$ such that $K\underline{c}^\top = \underline{o}^\top$. Then because exponentiation to $[i]$ is an isomorphism on $\mathbb{F}_{p^m}$ and $\operatorname{Rank}\left(K^{[i]}\right) = \operatorname{Rank}(K) = n-k$, we know that

$$K^{[i]}\left(\underline{c}^{[i]}\right)^\top = \underline{o}^\top \iff \underline{c}^{[i]} \text{ is a codeword in the code generated by } G_{pub}^{[i]}.$$

□

Theorem 54. *Assume that $\underline{\tilde{g}} \in \mathbb{F}_{p^m}^n$ such that $\Lambda_{k-1}(K)\underline{\tilde{g}}^\top = \underline{o}$. Then*

$$\forall i \in \{0, \cdots k-1\}\ \exists \underline{\tilde{s}}_{k-i} \in \mathbb{F}_{p^m}^k$$

such that

$$\underline{\tilde{g}} = \underline{\tilde{s}}_{k-i}^{[i]} G_{pub}^{[i]}.$$

Proof. By definition of $\Lambda_{k-1}(K)$ we know that

$$\forall i \in \{0, \dots, k-1\} :\ K^{[i]}\underline{\tilde{g}}^\top = \underline{o}^\top.$$

Therefore, Lemma 52 and Lemma 53 give us that

$$\underline{\tilde{g}} \in \mathcal{P}_{pub} = \mathcal{P}_{priv}.$$

It implies that

$$\forall i \in \{0, \dots, k-1\}\ \exists \underline{m}_i \in \mathbb{F}_{p^m}^k :\ \underline{\tilde{g}} = \underline{m}_i G_{priv}^{[i]} = \underline{m}_i \left(S^{-1}\right)^{[i]} G_{pub}^{[i]}.$$

Setting $\underline{\tilde{s}}_{k-i} := \underline{m}_i^{[-i]} S^{-1}$ completes the proof. □

Lemma 55. *Let us denote the first row of G_{priv} by $\underline{g}$. Let $l \leq k-1$. Then*

$$\bigcap_{i=0}^{l} \mathcal{P}_{priv_i} = \operatorname{Span}\left(\begin{pmatrix} g_1^{[l]} & \cdots & g_n^{[l]} \\ \vdots & \ddots & \vdots \\ g_1^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}^\top\right)$$

and $\dim\left(\bigcap_{i=0}^{l} \mathcal{P}_{priv_i}\right) = k - l$.

Proof. We will prove the lemma by induction. First, let $l = 0$. Then by definition

$$\begin{aligned}
\bigcap_{i=0}^{0} \operatorname{Span}\left(\left(\mathcal{P}_{priv_i}\right)^\top\right) &= \mathcal{P}_{priv_0} \\
&= \operatorname{Span}\left(\left(G_{priv}^{[0]}\right)^\top\right) \\
&= \operatorname{Span}\left(\left(G_{priv}\right)^\top\right) \\
&= \operatorname{Span}\left(\begin{pmatrix} g_1^{[0]} & \cdots & g_n^{[0]} \\ \vdots & \ddots & \vdots \\ g_1^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}^\top\right)
\end{aligned}$$

and $\dim(\mathcal{P}_{priv_0}) = \operatorname{Rank}(G_{priv}) = k$. Thus, the statement is true.
Second, suppose the statement holds for $l = j \leq k-1$. Then we shall prove it for $l = j+1$:

$$\begin{aligned}
\bigcap_{i=0}^{j+1} \mathcal{P}_{priv_i} &= \left(\bigcap_{i=0}^{j} \mathcal{P}_{priv_i}\right) \cap \mathcal{P}_{priv_{j+1}} \\
&= \operatorname{Span}\left(\begin{pmatrix} g_1^{[j]} & \cdots & g_n^{[j]} \\ \vdots & \ddots & \vdots \\ g_1^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}^\top\right) \cap \operatorname{Span}\left(\begin{pmatrix} g_1^{[j+1]} & \cdots & g_n^{[j+1]} \\ \vdots & \ddots & \vdots \\ g_1^{[k-1+j+1]} & \cdots & g_n^{[k-1+j+1]} \end{pmatrix}^\top\right) \qquad (5.1) \\
&= \operatorname{Span}\left(\begin{pmatrix} g_1^{[j+1]} & \cdots & g_n^{[j+1]} \\ \vdots & \ddots & \vdots \\ g_1^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}^\top\right).
\end{aligned}$$

The dimension can be computed as (see e.g. [BT, p. 180])

$$\begin{aligned}\dim\left(\bigcap_{i=0}^{j+1}\mathcal{P}_{{priv}_i}\right) &= \dim\left(\left(\bigcap_{i=0}^{j}\mathcal{P}_{{priv}_i}\right)\cap\mathcal{P}_{{priv}_{j+1}}\right)\\ &= \dim\left(\bigcap_{i=0}^{j}\mathcal{P}_{{priv}_i}\right)+\dim\left(\mathcal{P}_{{priv}_{j+1}}\right)-\dim\left(\left(\bigcap_{i=0}^{j}\mathcal{P}_{{priv}_i}\right)+\mathcal{P}_{{priv}_{j+1}}\right).\end{aligned}$$

The induction hypothesis gives us

$$\dim\left(\bigcap_{i=0}^{j}\mathcal{P}_{{priv}_i}\right)=k-j.$$

As discussed before, exponentiation to $[i]$ is an $\mathbb{F}_p$-isomorphism on $\mathbb{F}_{p^m}$, thus

$$\begin{aligned}\dim\left(\mathcal{P}_{{priv}_{j+1}}\right) &= \dim\left(\text{Span}\left(\left(G_{priv}^{[j+1]}\right)^\top\right)\right)\\ &= \dim\left(\text{Span}\left(G_{priv}^\top\right)\right)\\ &= k.\end{aligned}$$

Finally, from line 5.1 we can easily see that

$$\dim\left(\left(\bigcap_{i=0}^{j}\mathcal{P}_{{priv}_i}\right)+\mathcal{P}_{{priv}_{j+1}}\right)=(k-1-j+1)+(j+1)=k+1.$$

Altogether,

$$\dim\left(\bigcap_{i=0}^{j+1}\mathcal{P}_{{priv}_i}\right)=(k-j)+k-(k+1)=k-j-1.$$

□

Theorem 56. *Any $\underline{\tilde{g}}\in\mathbb{F}_{p^m}^n$ such that $\Lambda_{k-1}(K)\underline{\tilde{g}}^\top=\underline{o}^\top$ is an $\mathbb{F}_{p^m}$-multiple of $\underline{g}^{[k-1]}$. Here $\underline{g}$ is the first row of G_{priv}.*

Proof. We know that

$$\begin{aligned}\{\underline{\tilde{g}}\in\mathbb{F}_{p^m}^n \mid \Lambda_{k-1}(K)\underline{\tilde{g}}^\top=\underline{o}^\top\} &= \mathcal{P}_{pub}\\ &= \mathcal{P}_{priv}\\ &= \bigcap_{i=0}^{k-1}\text{Span}\left(\left(\mathcal{P}_{{priv}_i}\right)^\top\right)\\ &= \text{Span}\left(\left(g_1^{[k-1]} \quad \dots \quad g_n^{[k-1]}\right)^\top\right).\end{aligned}$$

The last equality holds due to Lemma 55. □

Remark. Note that $\underline{g}^{[k-1]}$ is by definition the last row of matrix G_{priv}.

Corollary 57. *Any nonzero $\underline{\tilde{g}}\in\left(\mathbb{F}_{p^m}^n\right)^*$ such that $\Lambda_{k-1}(K)\underline{\tilde{g}}^\top=\underline{o}^\top$ has $w_R\left(\underline{\tilde{g}}\right)=n$.*

Proof. We suppose that $w_R\left(\underline{g}\right) = n$. That means

$$\operatorname{Rank}\left(([g_1]_B^\top, \ldots, [g_n]_B^\top)\right) = n$$

for B a basis of $\mathbb{F}_{p^m}$ over $\mathbb{F}_p$. Because exponentiation to $[k-1]$ is an isomorphism on $\mathbb{F}_{p^m}$

$$B' := \{b^{[k-1]} \mid b \in B\}$$

is a basis of $\mathbb{F}_{p^m}$ over $\mathbb{F}_p$ too. If $g_1 = \sum_{i=1}^m \lambda_i b_i$ for $\lambda_i \in \mathbb{F}_p$ then

$$\begin{aligned} g_1^{[k-1]} &= \sum_{i=1}^m \lambda_i^{[k-1]} b_i^{[k-1]} \\ &= \sum_{i=1}^m \lambda_i b_i^{[k-1]}. \end{aligned}$$

In other words, we have obtained the following matrix equality

$$([g_1]_B^\top, \ldots, [g_n]_B^\top) = ([g_1^{[k-1]}]_{B'}^\top, \ldots, [g_n^{[k-1]}]_{B'}^\top).$$

Therefore, $w_R\left(\underline{g}^{[k-1]}\right) = n$. Since $\forall a \in \mathbb{F}_{p^m}^*$

$$\begin{aligned} \mathbb{F}_{p^m}^n &\to \mathbb{F}_{p^m}^n, \\ \underline{u} &\mapsto a \cdot \underline{u} \end{aligned}$$

forms an $\mathbb{F}_p$-isomorphism, we can in the same manner conclude

$$\forall a \in \mathbb{F}_{p^m}^* : \; w_R\left(a \cdot \underline{g}^{[k-1]}\right) = n.$$

$\square$

Theorem 58. *Let us define $\tilde{S} \in \mathbb{F}_{p^m}^{k\times k}$ and $\tilde{G}_{priv} \in \mathbb{F}_{p^m}^{k\times n}$ as in Algorithm 7, i.e.*

$$\begin{aligned} \tilde{G}_{priv} &= \tilde{S} G_{pub} \\ &= \begin{pmatrix} \underline{\tilde{s}}_1 \\ \vdots \\ \underline{\tilde{s}}_k \end{pmatrix} G_{pub}. \end{aligned}$$

Then $\forall i \in \{1, \ldots, k\}$ the i-th row of matrix $\tilde{G}_{priv}$ equals

$$\tilde{G}_{priv_{i*}} = \left(\underline{\tilde{g}}^{[-k]}\right)^{[i]} = \underline{\tilde{g}}^{[i-k]}.$$

Proof. From Theorem 54 we know that

$$\forall j \in \{0, \ldots, k-1\} : \; \underline{\tilde{g}} = \underline{\tilde{s}}_{k-j}^{[j]} G_{pub}^{[j]}.$$

Or equivalently

$$\forall j \in \{0, \ldots, k-1\} : \; \underline{\tilde{g}}^{[-j]} = \underline{\tilde{s}}_{k-j} G_{pub}.$$

Let $i := k - j \; \forall j \in \{0, \ldots, k-1\}$ then $i \in \{1, \ldots, k\}$ and

$$\underline{\tilde{g}}^{[i-k]} = \underline{\tilde{s}}_i G_{pub}.$$

Furthermore, by definition of matrix $\tilde{G}_{priv}$ we know that

$$\forall i \in \{1, \ldots, k\} : \; \tilde{G}_{priv_{i*}} = \underline{\tilde{s}}_i G_{pub}.$$

$\square$

Corollary 59. *Matrix $\tilde{G}_{priv}$ is a generator matrix an (n,k)-Gabidulin code.*

Proof. From Theorem 56 we know that $w_R\left(\tilde{\underline{g}}\right) = n$ and from the previous theorem we see that $\tilde{G}_{priv} = \langle \tilde{\underline{g}} \rangle^{n,k}_{\mathbb{F}_{p^m}}$. □

Lemma 60. *Matrix $\tilde{S}$ is nonsingular.*

Proof. Let us for contradiction assume that

$$\exists \underline{a} \in \mathbb{F}^k_{p^m},\ \underline{a} \neq \underline{o} \text{ such that } \underline{a}S = \underline{o}.$$

That implies

$$\begin{aligned}\underline{o} &= \underline{a}SG_{pub}\\ &= \underline{a}\tilde{G}_{priv}\\ &= \left(\sum_{i=1}^{k} a_i \tilde{g}_1^{[i-1]}, \dots, \sum_{i=1}^{k} a_i \tilde{g}_n^{[i-1]}\right).\end{aligned}$$

In other words, $\tilde{g}_1, \dots, \tilde{g}_n$ are roots of polynomial $f(x) = \sum_{i=1}^{k} a_i x^{[i-1]} \in \mathbb{F}_{p^m}[x]$. Furthermore, let $v \in \mathbb{F}_p^n$. Then

$$\begin{aligned}0 &= \sum_{i=1}^{n} v_i f(\tilde{g}_i)\\ &= \sum_{i=1}^{n} f(v_i \cdot \tilde{g}_i).\end{aligned}$$

Thus any $\mathbb{F}_p$-linear combination of $\tilde{g}_1, \dots, \tilde{g}_n$ is a root of $f(x)$. Note that

$$\mathrm{Rank}_{\mathbb{F}_p}\left((\tilde{g}_1, \dots, \tilde{g}_n)\right) = n,$$

thus $f(x)$ is a polynomial of degree p^{k-1} with p^n roots. Which is a contradiction. □

Corollary 61. *Matrices $\left(\tilde{S}\right)^{-1}$ and $\tilde{G}_{priv}$ form an alternative private key for the known public key G_{pub}.*

Proof. We know that matrix $\tilde{G}_{priv}$ is a generator matrix of an (n,k)-Gabidulin code. Matrix $\tilde{S}$ is nonsingular, thus there exists matrix $\left(\tilde{S}\right)^{-1}$. Furthermore, we know that

$$G_{pub} = \left(\tilde{S}\right)^{-1} \tilde{G}_{priv}.$$

□

5.5.4 Time Complexity

In order to show that the attack described in Algorithm 7 is efficient, we need to examine its time-complexity. We shall compute the complexity as number of operations over $\mathbb{F}_{p^m}$. In other words we view operations over $\mathbb{F}_{p^m}$ as elementary.

Note that by definition of the PKC,

$$k < n \leq m.$$

Theorem 62. *[BT, p. 65] Let $b \in \mathbb{N}$. The time complexity of solving a linear system of at most b equations and b indeterminates by Gaussian elimination is $\mathcal{O}(b^3)$.*

Step 2 Knowing K, to compute the matrix $\Lambda_{k-1}(K)$ we first need to compute

$$k_{ij}^{[1]} \ \forall i \in [n-k] \ \forall j \in \{1, \dots, n\}.$$

Then we shall iterate and compute

$$\begin{aligned} k_{ij}^{[2]} &:= (k_{ij}^{[1]})^{[1]}, \\ k_{ij}^{[3]} &:= (k_{ij}^{[2]})^{[1]}, \\ &\vdots \\ k_{ij}^{[k-1]} &:= (k_{ij}^{[k-2]})^{[1]}. \end{aligned}$$

Therefore, in total the complexity is

$$(n-k) \cdot n \cdot (k-1) = \mathcal{O}(n^2 k).$$

Step 3 Step 3 of the algorithm can be performed as simple Gaussian elimination of a system of $k(n-k)$ equations. Its complexity is therefore

$$\mathcal{O}(k^3(n-k)^3) = \mathcal{O}(k^3 n^3).$$

Steps 4 and 5 In Steps 4 and 5 we need to compute $\underline{\tilde{g}}^{[-1]}, \dots, \underline{\tilde{g}}^{[-k+1]}$. However, over $\mathbb{F}_{p^m}$ we know that

$$\tilde{g}^{[m]} = \tilde{g}^{[0]}$$

(see Observation on page 52). Therefore, we can instead compute $\underline{\tilde{g}}^{[m-1]}, \dots, \underline{\tilde{g}}^{[-k+1+m]}$. Again, we can first compute $\underline{\tilde{g}}^{[m-k+1]}$ and then proceed iteratively. That gives us $k-1$ operations over $\mathbb{F}_{p^m}$.

We need to perform Gaussian elimination on the matrix G_{pub} just once, that has complexity of

$$\mathcal{O}(n^3).$$

Finally, we need to perform the back substitution k times (in each iteration), that has complexity of

$$k \cdot \mathcal{O}(n^2) = \mathcal{O}(kn^2).$$

Step 7 Computing the inverse of $\tilde{S}$ in Step 6 can be viewed as performing Gaussian elimination on k systems with k equation. Therefore, its time complexity is

$$k \cdot \mathcal{O}(k^3) = \mathcal{O}(k^4).$$

Finally, computation of matrix $\tilde{G}_{priv} = \tilde{S} G_{pub}$ is actually $n \cdot k^2$ multiplications and $n \cdot k \cdot (k-1)$ addings in $\mathbb{F}_{p^m}$. That gives us time complexity of

$$n \cdot k^2 + k \cdot (k-1) \cdot n = \mathcal{O}(nk^2).$$

Altogether We see that the overall complexity is

$$\mathcal{O}(k^3 n^3) = \mathcal{O}(n^6)$$

which is polynomial in the length of the code.

Conclusion

The aim of the thesis was to study different versions of McEliece cryptosystem and attacks against them. It was intended to focus on cryptosystems using different metrics and to potentially find a new attack.

We discussed both Hamming metric and rank metric variants of McEliece and Niederreiter cryptosystems. In Section 2.3 we commented the purpose of matrices P and S in McEliece cryptosystem.

In Chapter 3 we covered the special case of cryptosystems using binary codes that can be attacked by Stern's attack. We reformulated [Ste88, proposition 2] as Theorem 9 and gave a comprehensive proof. We gave algorithms for Stern-McEliece and Stern-Niederreiter attacks and proved their correctness. Both attacks were originally described in [BLP08, p. 35].

We explained in detail Sidelnikov-Shestakov's attack on cryptosystems based on GRS codes and proved the correctness of defining $\mathbb{F}_\infty$ which the original work [SS92] does not cover. For better understanding of the attack, we showed an example in Section 4.3.

In Chapter 5 we focused on GGPT - a cryptosystem using rank metric. We explained why an attack against GGPTX (GGPT cryptosystem without distortion matrix X) conjectured by R. Overbeck based on the principles of Sidelnikov-Shestakov's attack cannot work. We found similarities between cryptosystems based on GRS codes and GGPTX cryptosystem and took inspiration from Overbeck's attack on GGPT to construct a polynomial-time attack on GGPTX.

Cryptosystems based on coding theory is a vast area with ongoing research. It would be interesting to go on with examining the similarities between McEliece based on GRS codes and GGPTX. We might as well examine their more secure variants - Modified Niederreiter (briefly mentioned in Section 4.4) and GGPT cryptosystem.

Bibliography

[ABC$^+$] 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. `https://classic.mceliece.org/`.

[BLP08] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and Defending the McEliece Cryptosystem. In *Post-Quantum Cryptography*, pages 31–46, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[BMvT78] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (Corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[BT] Libor Barto and Jiří Tůma. Lineární algebra. The page numbers refer to the version found at `https://www2.karlin.mff.cuni.cz/~stanovsk/vyuka/LinAlg/skripta22.pdf`.

[EOS06] D. Engelbert, R. Overbeck, and A. Schmidt. A Summary of McEliece-Type Cryptosystems and their Security. 2006.

[Gab01] Ernst Gabidulin. Public-Key Cryptosystems Based on Linear Codes. 2001.

[Gab21] Gabidulin, Ernst M. *Rank Codes*. TUM.University Press, München, 2021.

[Gib96] Gibson, Keith. The Security of the Gabidulin Public Key Cryptosystem. In *Advances in Cryptology — EUROCRYPT '96*, pages 212–223, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[GPT91] 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*, pages 482–489, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[Hru14] Tereza Hrubešová. Klasický strukturturální útok na Niederreiterův kryptosystém vytvořený nad GRS kódy. 2014.

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite Fields*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 1997.

[Mac77] F. J. MacWilliams. *The theory of error-correcting codes*. North-Holland mathematical library ; 16. North-Holland, Amsterdam, 1977.

[Mar21] Marek Marko. Kryptosystémy založené na kódoch s hodnostnou metrikou, 2021.

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. 1978.

[Mor91] Carlos J. Moreno. *Algebraic curves over finite fields* . Cambridge tracts in mathematics ; 97. Cambridge University Press, Cambridge, 1991.

[Nie86] H. Niederreiter. Knapsack-type cryptosystem and algebraic coding theory. *Problems of Control and Information coding Theory*, 1986.

[NIoST15] National Institute of Standards and Technology. Recommendation for key management, part 3: Application-specific key management guidance. Technical Report SP 800-57 Part 3 Rev. 1, U.S. Department of Commerce, Washington, D.C., 2015.

[Ove05] Overbeck, Raphael. A New Structural Attack for GPT and Variants. In *Progress in Cryptology – Mycrypt 2005*, pages 50–63, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[Ove06] Overbeck, Raphael. Extending Gibson's Attacks on the GPT Cryptosystem. In *Coding and Cryptography*, pages 178–188, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Par20] Gunadhar Paria. *Linear algebra.* New Central Book Agency Ltd, London, 2020.

[Rom96] Steven Roman. *Introduction to Coding and Information Theory.* Undergraduate Texts in Mathematics. Springer New York, 1996.

[SS92] V. M. Sidelnikov and S. O. Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Matematics and Applications*, 2(4):439–444, 1992.

[Ste88] Jacques Stern. A method for finding codewords of small weight. In *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988.

A. Attachments

A.1 Sidelnikov-Shestakov's Attack on Original Niederreiter

[1]:
```
class Niederreiter:
def __init__(self, F, k, M = None, P = None, pointsAlpha = None,␣
 ↪pointsZ = None):
    self.k = k
    self.n = len(F)-1
    self.s = self.n-1-k
    self.F = F
    self.Flist = [i for i in self.F]
    self.Flist.remove(0)
    self.M = M
    self.P = P
    self.pointsAlpha = pointsAlpha
    self.pointsZ = pointsZ
    if self.pointsZ == None:
        self.pointsZ = []
        for i in range(self.n):
            self.pointsZ += random.sample(self.Flist, 1)
    if self.pointsAlpha == None:
        self.pointsAlpha = self.Flist
    self.GRS = codes.GeneralizedReedSolomonCode(self.pointsAlpha, k,␣
 ↪self.pointsZ)
    self.H = self.GRS.parity_check_matrix()
    if self.M == None:
        self.M = random_matrix(self.F,self.s+1,self.s+1)
        while self.M.is_singular():
            self.M = random_matrix(self.F,self.s+1,self.s+1)
    if self.P == None:
        self.P = Permutations(self.n).random_element().to_matrix()
    #
    self.K = self.M*self.H*self.P
```

Parameters

[2]:
```
import random
q = 9
F.<eta> = GF(q, name='eta', modulus=x^2 + 2*x +2 )
print("k has to be greater or equal to ", float((q-1)/2))
```

```
k has to be greater or equal to  4.0
```

[3]:
```
k = int(input())
```

```
4
```

```
[4]: n = q-1
     s = n-k-1
```

```
[5]: print("n = ", n)
     print("k = ", k)
     print("s = ", s)
```

```
n =  8
k =  4
s =  3
```

Generate keys

```
[6]: M = matrix([[ eta + 1, 1, eta + 2, 2*eta + 1],
             [2,2,eta + 1,2*eta],
             [1,eta + 2,0,eta + 1],
             [2*eta,0,eta + 1,eta + 1]])
     l = F(1)
     P = matrix([[0, 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, 0, 1, 0, 0, 0, 0],
            [1, 0, 0, 0, 0, 0, 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]])
     pointsAlpha = [eta, eta + 1, 2*eta + 1, 2, 2*eta, 2*eta + 2, eta + 2, 1]
     pointsZ = [2*eta, 2*eta + 2, eta + 1, eta + 1, 2*eta, eta, 2, 2*eta + 2]
```

```
[7]: NiederreiterPKC = Niederreiter(F,k,M,P,pointsAlpha,pointsZ)
     K = NiederreiterPKC.K
     print("private matrix H ="),
     print(NiederreiterPKC.H, "\n")
     print("public matrix K ="),
     print(K)
```

```
private matrix H =
[        1         1     2*eta 2*eta + 2         2       eta   eta + 2 2*eta + 2]
[      eta   eta + 1         1   eta + 1       eta   eta + 2 2*eta + 2 2*eta + 2]
[  eta + 1         2 2*eta + 1 2*eta + 2 2*eta + 2     2*eta     2*eta 2*eta + 2]
[2*eta + 1 2*eta + 2 2*eta + 2   eta + 1 2*eta + 1 2*eta + 1         2 2*eta + 2]

public matrix K =
[  eta + 1   2*eta       eta 2*eta + 2 2*eta + 1   2*eta     2*eta     1]
[    2*eta eta + 1 2*eta + 1       eta 2*eta + 1       1 2*eta + 2 2*eta]
[    2*eta     eta 2*eta + 1         1 2*eta + 2 eta + 2 2*eta + 2     0]
[        1   2*eta         0 2*eta + 1 2*eta + 1     eta   eta + 2     2]
```

Attack

Recover $gamma_4, ..., gamma_8$

[8]:
```
maxindex = min(2*s-1,n-1)

J1 = [0]+list(range(s+1,maxindex+1))
K1 = K.matrix_from_columns([i for i in J1])
c1 = K1.left_kernel().basis()[0]
pi1 = c1*K
print("c_1 = ", c1)
print("pi_1 = ", pi1)

J2 = [1]+list(range(s+1,maxindex+1))
K2 = K.matrix_from_columns([i for i in J2])
c2 = K2.left_kernel().basis()[0]
pi2 = c2*K
print("\nc_2 = ", c2)
print("pi_2 = ", pi2)
```

```
c_1 =  (0, 1, 2*eta + 2, 2*eta + 2)
pi_1 =  (0, eta + 1, 1, eta + 2, 0, 0, eta + 1, 1)

c_2 =  (1, 2*eta, 2*eta, eta + 1)
pi_2 =  (eta + 1, 0, eta + 2, 2*eta + 1, 0, 0, eta + 2, 1)
```

[9]:
```
a1 = pi1[2]
a2 = pi2[2]
blist = [None]*n
blist[2] = a1/a2

gammalist = list([None]*n)

for j in range(3,(s)+1):
blist[j] = pi1[j]/pi2[j]
gammalist[j] = blist[2]/(blist[2]-blist[j])

for j in range(2*s,(n-1)+1):
blist[j] = pi1[j]/pi2[j]
gammalist[j] = blist[2]/(blist[2]-blist[j])

print("(gamma_1, ..., gamma_8) = ", gammalist)
```

```
(gamma_1, ..., gamma_8) =  [None, None, None, eta + 2, None, None, 2*eta
↪+ 1, eta + 1]
```

[10]:
```
J3 = [0]+list(range(2,(s)+1))
K3 = K.matrix_from_columns([i for i in J3])
c3 = K3.left_kernel().basis()[0]
pi3 = c3*K
print("c_3 = ", c3)
print("pi_3 = ", pi3)

J4 = [1]+list(range(2,(s)+1))
K4 = K.matrix_from_columns([i for i in J4])
c4 = K4.left_kernel().basis()[0]
pi4 = c4*K
print("\nc4 = ", c4)
print("pi_4 = ", pi4)
```

```
c_3 =  (1, eta, 1, eta)
pi_3 =  (0, eta, 0, 0, eta + 1, 2*eta, 2*eta + 2, eta)

c4 =  (1, 2*eta + 1, 2*eta, 2*eta + 1)
pi_4 =  (eta + 1, 0, 0, 0, 2*eta, eta + 2, 2, eta + 1)
```

[11]:
```
r = random.randint(2*s,n-1)
for j in range(0,n):
if j not in set(J3+J4+[2]):
    if gammalist[j] == None:
        blist[j] = (pi4[r]*pi3[j]*blist[r])/(pi3[r]*pi4[j])
        gammalist[j] = blist[2]/(blist[2]-blist[j])
print("(gamma_1, ..., gamma_8) = ", gammalist)
```

```
(gamma_1, ..., gamma_8) =  [None, None, None, eta + 2, 2*eta + 2, eta,␣
↪2*eta + 1, eta + 1]
```

Transformation $\mathbb{F}_\infty^8 \to \mathbb{F}_9^8$

[12]:
```
gammalist[0] = 1
gammalist[1] = 0

for i in F:
if i not in gammalist:
    a = i
    break

betalist = gammalist[:]
for j in range(3,n):
gammanew = 1/(a-gammalist[j])
betalist[j] = gammanew
betalist[0] = 1/(a-1)
betalist[1] = 1/(a)
```

```
betalist[2] = 0
print("(beta_1, ..., beta_8) = ", betalist)
```

```
(beta_1, ..., beta_8) =  [1, 2, 0, 2*eta + 1, eta + 2, eta + 1, 2*eta +␣
↪2, 2*eta]
```

Recover $y_1, \ldots, y_5$

[13]:
```
J5 = list(range(0,(s+1)+1))
K5 = K.matrix_from_columns([i for i in J5])
c5 = K5.right_kernel().basis()[0]

ylist = [None] * n
ylist[0] = 1

V = (matrix.vandermonde(betalist[:(s+1)+1]).transpose()).
 ↪matrix_from_rows([i for i in range(0,s+1)])

B = V.matrix_from_columns([i for i in range(1,(s+1)+1)])
C = matrix.diagonal(c5[1:])
BC = B*C
solve = vector(-1 * c5[0] * V.matrix_from_columns([0]))
y2 = BC.solve_right(solve)

i = 1
for j in y2:
ylist[i] = j
i += 1
print("(y_1, ..., y_8) = ", ylist)
```

```
(y_1, ..., y_8) =  [1, eta + 1, 2*eta + 1, eta + 1, 2*eta, None, None,␣
↪None]
```

Recover $\hat{M}$

[14]:
```
V = matrix.vandermonde(betalist[:s+1])
Mhat = [[None]*(s+1)]*(s+1)

ytmp = vector(ylist[:s+1])
yinv = vector([yi^(-1) for yi in ytmp])

for i in range(s+1):
solve = yinv[:]
for j in range(s+1):
    solve[j] *= K[i][j]
Mhat[i] = V.solve_right(solve)
Mhat = matrix(Mhat)
```

```
print("\\hat{M} = ")
print(Mhat)
```

```
\hat{M} =
[2*eta + 2     2*eta eta + 2   2*eta]
[        1         2 eta + 1 eta + 2]
[        1     2*eta       0       2]
[        0 2*eta + 1 eta + 1       2]
```

Recover y_6, y_7, y_8

[15]:
```
Mhatinv = Mhat.inverse()
for i in range(s+2,n):
tmpsum = 0
for j in range(s+1):
    tmpsum += Mhatinv[0][j]*K[j][i]
ylist[i] = tmpsum
print("(y_1, ..., y_8) = ", ylist)
```

```
(y_1, ..., y_8) =  [1, eta + 1, 2*eta + 1, eta + 1, 2*eta, 2, 2, eta]
```

Check if the attack was succesful.

[16]:
```
Halt = matrix.vandermonde(betalist).transpose().
 ↪matrix_from_rows(list(range(0,s+1))) * matrix.diagonal(ylist)
Malt = Mhat
Kalt = Malt*Halt
if Kalt == K:
print("Attack was succesful.")
else:
print("Attack failed.")
error = false
for i in range(s+1):
    for j in range(n):
        if not Kalt[i][j] == K[i][j]:
            chyba = false
            print('Error on position', i,j)
            print(Kalt[i][j], K[i][j])
```

```
Attack was succesful.
```