A Summary of McEliece-Type Cryptosystems and their Security

(preprint)

D. Engelbert R. Overbeck * A. Schmidt †

TU-Darmstadt,
Department of Computer Science,
Cryptography and Computer Algebra Group.
{engelber, overbeck, aschmidt}@cdc.informatik.tu-darmstadt.de

May 10, 2006

Abstract

In this paper we give an overview of some of the cryptographic applications which were derived from the proposal of R.J. McEliece to use error correcting codes for cryptographic purposes. Code based cryptography is an interesting alternative to number theoretic cryptography. Many basic cryptographic functions like encryption, signing, hashing, etc. can be realized using code theoretic concepts.

In this paper we briefly show how to correct errors in transmitted data by employing Goppa codes and describe possible applications to public key cryptography.

The main focus of this paper is to provide detailed insight into the state of art of cryptanalysis of the McEliece cryptosystem and the effect on different cryptographic applications. We conclude, that for code based cryptography a public key of 88KB offers sufficient security for encryption, while we need a public key of at least 597KB for secure signing.

Keywords: McEliece cryptosystem, public key cryptography, code based cryptography, Goppa codes.

*Funded by GK E-Commerce (Graduiertenkolleg 492 of DFG)
†Funded by DFG

Contents

1 Introduction

In this paper we want to give an overview over the McEliece cryptosystem and the primitives it is based on. First, we give some introduction into coding theory and the construction principle of the cryptosystem. In the second section, we present Goppa codes, which at the moment seem to be the best choice for cryptographic applications. In the sections three to five we present known attacks on the McEliece PKC and consequences for the choice of system parameters. Afterwards we will present CCA2-secure conversions and show how to build other cryptographic protocols from the basic scheme. Finally we will discuss performance and secure choices of parameters for the McEliece PKC.

1.1 History

In 1978 R. McEliece proposed the first public key cryptosystem which is based on coding theory. McEliece's proposal to use Goppa codes for cryptographic applications is one of the oldest public key cryptosystems and remains unbroken for appropriate system parameters. In 1986, Niederreiter proposed a different scheme which uses GRS codes. This proposal is equivalent (dual) to McEliece's proposal if we substitute the GRS codes by Goppa codes [33]. Sidelnikov and Shestakov showed 1992, that Niederreiter's proposal to use GRS codes is insecure.

Several proposals were made to modify McEliece's original scheme (see e.g. [17], [16], [18], [46] and [26]). Most of them replace the Goppa codes with other codes. However, most of them turned out to be insecure or inefficient compared to McEliece's original proposal (see e.g. [38] or [28]).

The most important variants of McEliece's scheme are the ones proposed by Kobara and Imai in 2001. These variants are CCA2-secure and provably as secure as the original scheme [27].

Parallel to the efforts to build an efficient encryption scheme based on coding theory, there were several attempts to build other cryptographic protocols based on error correcting codes. Most efforts to build a signature scheme failed (compare [52], [22], [2] and [50]), until finally in 2001 Courtois, Finiasz and Sendrier made a promising proposal [11]. In addition, there exists an identification scheme by Stern [49], which is based on coding theory.

There are also attempts to build fast hash functions and random number generators using the principles of coding theory (see e.g. [3], [13]). All in all, this provides sufficient motivation to have a closer look at the McEliece cryptosystem, as an serious alternative to the established PKCs based on number theory.

1.2 Coding Theory and Problems

The security of the cryptosystems reviewed in this paper is based on the difficulty of some classical problems of coding theory. Here we give an introduction into the topic of coding theory.

Definition 1.2.1 An (n,k)-code $\mathcal{C}$ over a finite field $\mathbb{F}$ is a k-dimensional subvectorspace of the vector space $\mathbb{F}^n$. We call $\mathcal{C}$ an (n,k,d)-code if the minimum distance is $d = \min_{\mathbf{x},\mathbf{y}\in\mathcal{C}} \operatorname{dist}(\mathbf{x},\mathbf{y})$, where "dist" denotes a distance function, e.g. hamming distance. The distance of $\mathbf{x} \in \mathbb{F}^n$ to the null-vector $\operatorname{wt}(\mathbf{x}) := \operatorname{dist}(\mathbf{0},\mathbf{x})$ is called *weight* of $\mathbf{x}$.

Definition 1.2.2 The matrix $\mathsf{C} \in \mathbb{F}^{k\times n}$ is a *generator matrix* for the (n,k) code $\mathcal{C}$ over $\mathbb{F}$, if the rows of C span $\mathcal{C}$ over $\mathbb{F}$. The matrix $\mathsf{H} \in \mathbb{F}^{(n-k)\times n}$ is called *check matrix* for the code $\mathcal{C}$ if $\mathsf{H}^\top$ is the right kernel of C. The code generated by H is called *dual code* of $\mathcal{C}$ and denoted by $\mathcal{C}^\perp$.

With these definitions, we are able to define some basic problems of coding theory. Here the distance function used will be the hamming distance although, there exist other notions of distance.

Problem 1.2.3 *The general decoding problem for linear codes is defined as follows:*

- *Let $\mathcal{C}$ be an (n,k) linear code over $\mathbb{F}$ and $\mathbf{y} \in \mathbb{F}^n$.*
- *Find $\mathbf{x} \in \mathcal{C}$ where $\operatorname{dist}(\mathbf{y},\mathbf{x})$ is minimal.*

Let $\mathbf{e}$ be a vector of weight $\leq t := \left\lfloor \frac{d-1}{2} \right\rfloor$ and $\mathbf{x} \in \mathcal{C}$. Then there is a unique solution to the general decoding problem for $\mathbf{y} = \mathbf{x} + \mathbf{e}$. The code $\mathcal{C}$ is said to be an t-error correcting code.

Problem 1.2.4 *The problem of finding weights (SUBSPACE WEIGHTS) of a linear code is defined as follows:*

- *Let $\mathcal{C}$ be an (n,k) linear code over $\mathbb{F}$ and $\mathbf{w} \in \mathbb{N}$.*
- *Find $\mathbf{x} \in \mathcal{C}$ satisfying $\operatorname{dist}(\mathbf{0},\mathbf{x}) = w$.*

Our hope that we might be able to construct secure cryptosystems based on the problems above is based on the following result.

Theorem 1.2.5 *The general decoding problem and the problem of finding weights are $\mathcal{NP}$-hard.*

Proof. See [4]. ■

We present another problem based on the equivalence of codes:

Definition 1.2.6 Two (n,k) codes $\mathcal{C}$ and $\mathcal{C}'$ over a field $\mathbb{F}$ are called *permutation equivalent* if there exists a permutation π of the permutation group $\mathcal{S}_n$ over n elements, such that

$$\mathcal{C}' = \pi(\mathcal{C}) = \left\{ \left(x_{\pi^{-1}(1)}, \cdots, x_{\pi^{-1}(n)}\right) | \mathbf{x} \in \mathcal{C} \right\}.$$

The subgroup of $\mathcal{S}_n$ which keeps $\mathcal{C}$ fixed will be called $\operatorname{Aut}(\mathcal{C})$.

Given two generator matrices G and G' the problem is to decide if the codes generated by the matrices are permutation equivalent or not. In the case where $\mathbb{F} = \mathbb{F}_2$ the definition of permutation equivalency coincides with the definition of *equivalency*.

Definition 1.2.7 Two (n,k) codes $\mathcal{C}$ and $\mathcal{C}'$ over $\mathbb{F}$ are called equivalent if there exists $\pi \in \mathcal{S}_n$, a n-tupel $(a_i)_{1\leq i\leq n} \in \mathbb{F}^*$ and a field automorphism ϕ of $\mathbb{F}$ such that

$$\mathbf{x} \in \mathcal{C} \Leftrightarrow \left(\phi\left(a_{\pi^{-1}(i)}x_{\pi^{-1}(i)}\right)\right)_{1\leq i\leq n} \in \mathcal{C}'$$

In section 3.3, we will see an algorithm which solves the problem to decide whether two codes are permutation equivalent or not.

Throughout this paper, we will use the following notation. We write $\mathcal{G} = \langle \mathsf{G} \rangle$ if the linear (n,k)-code $\mathcal{G}$ over $\mathbb{F}$ has the generator matrix G. We can write $\mathbf{x} \in \mathcal{G}$ as $(x_1, \cdots, x_n) \in \mathbb{K}^n$. For any (ordered) subset $\{j_1, \cdots j_m\} = J \subseteq \{1, \cdots n\}$ we denote the vector $(x_{j_1}, \cdots, x_{j_m}) \in \mathbb{K}^m$ with $\mathbf{x}_J$. Similarly we denote by $\mathsf{M}_{\cdot J}$ the submatrix of a $k \times n$ matrix M consisting of the columns corresponding to the indices of J and $\mathsf{M}_{J'\cdot} = \left(\mathsf{M}^\top\right)_{\cdot J'}$ for any (ordered) subset J' of $\{1, \cdots, k\}$.

1.3 McEliece PKC

This cryptosystem was proposed by McEliece [37] and is the first, which uses error correcting codes as a trapdoor. It remains unbroken in its original version. Although it uses Goppa codes (see section 2) in the original description, any subclass of the class of alternant codes could be used. However, it might not reach the desired security (compare section 3.2 or e.g. [38]). The trapdoor for the McEliece Cryptosystem using Goppa codes is the knowledge of the Goppa polynomial used to generate the code.

We briefly describe the cryptosystem:

- **System Parameters:** n, $t \in \mathbb{N}$, where $t \ll n$.
- **Key Generation:** Given the parameters n, t generate the following matrices:

G' :	$k \times n$ generator matrix of a binary irreducible (n,k) Goppa code $\mathcal{G}$ which can correct up to t errors, where k is chosen maximal.
S :	$k \times k$ random binary non-singular matrix
P :	$n \times n$ random permutation matrix

  Then, compute the $k \times n$ matrix $\mathsf{G} = \mathsf{S}\mathsf{G}'P$.
- **Public Key:** (G, t)
- **Private Key:** $(\mathsf{S}, D_{\mathcal{G}}, \mathsf{P})$, where $D_{\mathcal{G}}$ is an efficient decoding algorithm for $\mathcal{G}$ (see e.g. algorithm 2.3.1).

- **Encryption:** To encrypt a plaintext $\mathbf{m} \in \{0,1\}^k$ choose a vector $\mathbf{z} \in \{0,1\}^n$ of weight t randomly and compute the ciphertext $\mathbf{c}$ as follows:

$$\mathbf{c} = \mathbf{m}\mathsf{G} \oplus \mathbf{z} \ .$$

- **Decryption:** To decrypt a ciphertext $\mathbf{c}$ calculate

$$\mathbf{c}\mathsf{P}^{-1} = (\mathbf{m}\mathsf{S})\, \mathsf{G}' \oplus \mathbf{z}\mathsf{P}^{-1}$$

first, and apply the decoding algorithm $\mathcal{D}_{\mathsf{G}}$ for $\mathcal{G}$ to it. Since $\mathbf{c}\mathsf{P}^{-1}$ has a hamming distance of t to the Goppa code we obtain the codeword

$$\mathbf{m}\mathsf{S}\mathsf{G}' = \mathcal{D}_{\mathcal{G}}\left(\mathbf{c}\mathsf{P}^{-1}\right) \ .$$

Let $J \subseteq \{1, \cdots, n\}$ be a set, such that $\mathsf{G}_{\cdot J}$ is invertible, then we can compute the plaintext $\mathbf{m} = (\mathbf{m}\mathsf{S}\mathsf{G}')_J \,(\mathsf{G}'_{\cdot J})^{-1} S^{-1}$

There are some restrictions to the choice of the McEliece system parameters given by the attacks, if we want to get optimal security. We are going to discuss them later on.

Definition 1.3.1 The *McEliece problem* is described as follows:

- Given a McEliece public key (G, t) where $\mathsf{G} \in \{0,1\}^{k \times n}$ and a ciphertext $\mathbf{c} \in \{0,1\}^n$,
- Find the (unique) message $\mathbf{m} \in \{0,1\}^k$ s.t. $\mathrm{dist}\,(\mathbf{m}\mathsf{G}, \mathbf{c}) = t$.

It is easy to see that someone who is able to solve the general decoding problem is able to solve the McEliece problem. The reverse is presumably not true, as the code $\mathcal{G} = \langle \mathsf{G} \rangle$ is not a random one, but permutation equivalent to a code of a known class (a Goppa code in our definition). We can not assume that the McEliece-Problem is $\mathcal{NP}$-hard. Solving the McEliece-Problem would only solve the General Decoding Problem in a certain class of codes and not for all codes.

In the case of McEliece's original proposal, Canteaut and Chabaud state the following: "The row scrambler S has no cryptographic function; it only assures for McEliece's system that the public matrix is not systematic otherwise most of the bits of the plain-text would be revealed" [7]. However, for some variants of McEliece's PKC, this statement is not true, as e.g. in the case of the CCA2-secure variants (which we are going to present in section 6). The importance of P is not that easy to see. We will come back to this question in section 3.

1.4 Niederreiter PKC

The Niederreiter PKC is a knapsack-type cryptosystem which uses an (n, k)-linear code which can correct up to t errors and for which an efficient decoding algorithm is known. We describe the cryptosystem briefly:

- **System Parameters:** n, $t \in \mathbb{N}$, where $t \ll n$.
- **Key Generation:** Given the parameters n, t generate the following matrices:

H:	$(n-k) \times n$ check matrix of a binary irreducible Goppa code $\mathcal{G}$ of maximal dimension k which can correct up to t errors
M:	$(n-k) \times (n-k)$ random binary non-singular matrix
P:	$n \times n$ random permutation matrix

  Then, compute the $n \times (n-k)$ matrix $\mathsf{H}' = \mathsf{MHP}$.
- **Public Key:** (H', t)
- **Private Key:** $(\mathsf{P}, \mathcal{D}_\mathcal{G}, \mathsf{M})$, where $\mathcal{D}_\mathcal{G}$ is an efficient syndrome decoding algorithm for $\mathcal{G}$ (see e.g. algorithm 2.3.1).
- **Encryption:** A message $\mathbf{m}$ is represented as a vector $\mathbf{e} \in \{0,1\}^n$ of weight t, called plaintext. To encrypt it, we compute the syndrome

  $$\mathbf{s} = \mathsf{H}'\mathbf{e}^\top .$$
- **Decryption:** To decrypt a ciphertext $\mathbf{s}$ calculate

  $$\mathsf{M}^{-1}\mathbf{s} = \mathsf{HP}\mathbf{e}^\top$$

  first, and apply the syndrome decoding algorithm $\mathcal{D}_\mathcal{G}$ for $\mathcal{G}$ to it in order to recover $\mathsf{P}\mathbf{e}^\top$. Now, we can obtain the plaintext $\mathbf{e}^\top = \mathsf{P}^{-1}\mathsf{P}\mathbf{e}^\top$

The security of the Niederreiter PKC and the McEliece PKC are equivalent. An attacker who can break one is able to break the other and vice versa [33].

2 Goppa Codes

In this paper, we consider only irreducible binary Goppa codes. The following reasons make them interesting for cryptography:

- The lower bound for the minimum distance is easy to compute.
- The knowledge of the generating polynomial[1] allows efficient error correction.
- Without the knowledge of the generating polynomial no efficient algorithms for error correction are known.

For a comprehensive introduction to Goppa codes see [36, 34, 23].

[1] See below

2.1 Definition

In this section, we will first define Goppa codes. Based on this definition, we will describe a way to construct a generator and a parity check matrix for Goppa codes.

Goppa codes were defined by V.D. Goppa in 1970 [21].

Definition 2.1.1 (Goppa polynomial, Syndrome, binary Goppa Codes) Let m and t be positive integers and let

$$g(X) = \sum_{i=0}^{t} g_i X^i \in \mathbb{F}_{2^m}[X]$$

be a monic polynomial of degree t called *Goppa polynomial* and

$$\mathbf{L} = (\gamma_0, \ldots, \gamma_{n-1}) \in \mathbb{F}_{2^m}^n$$

a tuple of n distinct elements such that

$$g(\gamma_i) \neq 0, \quad \text{for all } 0 \leq i < n.$$

For any vector $\mathbf{c} = (\mathbf{c}_0, \ldots, \mathbf{c}_{n-1}) \in \mathbb{F}_2^n$, define the *syndrome* of $\mathbf{c}$ by

$$S_{\mathbf{c}}(X) = -\sum_{i=0}^{n-1} \frac{\mathbf{c}_i}{g(\gamma_i)} \frac{g(X) - g(\gamma_i)}{X - \gamma_i} \mod g(X). \tag{1}$$

The *binary Goppa code* $\mathcal{G}(\mathbf{L}, g(X))$ over $\mathbb{F}_2$ is the set of all $\mathbf{c} = (\mathbf{c}_0, \ldots, \mathbf{c}_{n-1}) \in \mathbb{F}_2^n$ such that the identity

$$S_{\mathbf{c}}(X) = 0 \tag{2}$$

holds in the polynomial ring $\mathbb{F}_{2^m}[X]$ or equivalently if

$$S_{\mathbf{c}}(X) \equiv \sum_{i=0}^{n-1} \frac{\mathbf{c}_i}{X - \gamma_i} \equiv 0 \mod g(X). \tag{3}$$

Thus, we have

$$\begin{aligned} \mathcal{G}(\mathbf{L}, g(X)) &= \{\mathbf{c} \in \mathbb{F}_2^n \mid S_{\mathbf{c}}(X) = 0\} \\ &= \{\mathbf{c} \in \mathbb{F}_2^n \mid S_{\mathbf{c}}(X) \equiv 0 \mod g(X)\} \end{aligned}$$

If $g(X)$ is irreducible over $\mathbb{F}_{2^m}$, then $\mathcal{G}(\mathbf{L}, g(X))$ is called an *irreducible binary Goppa code.*

Remark 2.1.2 To emphasize the dependency of vector $\mathbf{c}$ on sequence $\mathbf{L}$, we sometimes write $\mathbf{c} = (\mathbf{c}_{\gamma_0}, \ldots, \mathbf{c}_{\gamma_{n-1}})$. The elements $\gamma_0, \ldots, \gamma_{n-1} \in \mathbb{F}_{2^m}$ are called *code support.*

Goppa codes are linear codes. If $g(X)$ is irreducible, we have $g(\gamma) \neq 0$ for all $\gamma \in \mathbb{F}_{2^m}$. Thus tuple $\mathbf{L}$ from the definition may contain all elements of $\mathbb{F}_{2^m}$.

Now we will show how to construct the parity check matrix of a Goppa code $\mathcal{G}(\mathbf{L}, g(X))$. Since

$$\frac{g(X) - g(\gamma_i)}{X - \gamma_i} = \sum_{j=0}^{t} g_j \frac{X^i - \gamma_i^j}{X - \gamma_i} = \sum_{s=0}^{t-1} X^s \sum_{j=s+1}^{t} g_j \gamma_i^{j-1-s}, \quad \text{for all } 0 \le i < n,$$

we see that $\mathbf{c} \in \mathcal{G}(\mathbf{L}, g(X))$, iff for all $s = 0, \dots, t-1$

$$\sum_{i=0}^{n-1} \left(\frac{1}{g(\gamma_i)} \sum_{j=s+1}^{t} g_j \gamma_i^{j-1-s} \right) \mathbf{c}_i = 0.$$

Thus, a parity check matrix of $\mathcal{G}(\mathbf{L}, g(X))$ can be written as

$$\mathsf{H} = \begin{pmatrix} g_t g(\gamma_0)^{-1} & \cdots & g_t g(\gamma_{n-1})^{-1} \\ (g_{t-1} + g_t \gamma_0) g(\gamma_0)^{-1} & \cdots & (g_{t-1} + g_t \gamma_{n-1}) g(\gamma_{n-1})^{-1} \\ \vdots & \ddots & \vdots \\ \left(\sum_{j=1}^{t} g_j \gamma_0^{j-1} \right) g(\gamma_0)^{-1} & \cdots & \left(\sum_{j=1}^{t} g_j \gamma_{n-1}^{j-1} \right) g(\gamma_{n-1})^{-1} \end{pmatrix} = \mathsf{XYZ}$$

where

$$\mathsf{X} = \begin{pmatrix} g_t & 0 & 0 & \cdots & 0 \\ g_{t-1} & g_t & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \cdots & g_t \end{pmatrix}, \quad \mathsf{Y} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \gamma_0 & \gamma_1 & \cdots & \gamma_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_0^{t-1} & \gamma_1^{t-1} & \cdots & \gamma_{n-1}^{t-1} \end{pmatrix}, \quad \text{and}$$

$$\mathsf{Z} = \begin{pmatrix} \frac{1}{g(\gamma_0)} & & & \\ & \frac{1}{g(\gamma_1)} & & \\ & & \ddots & \\ & & & \frac{1}{g(\gamma_{n-1})} \end{pmatrix}$$

and therefore we have

$$\mathbf{c} \in \mathcal{G}(\mathbf{L}, g(X)), \quad \text{iff} \quad \mathsf{H}\mathbf{c}^T = 0. \tag{4}$$

The entries of the matrix H are elements of the extension field $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$. If we interpret $\mathbb{F}_{2^m}$ as m dimensional vector space over $\mathbb{F}_2$, we can write H as a matrix over $\mathbb{F}_2$ of dimension $mt \times n$.

The rows of matrix H generate a vector space V which is a subspace of $\mathbb{F}_2^n$. From (4) it follows that the Goppa code is a vector space which is dual to V. Therefore we obtain a generator matrix G of a Goppa code by computing the basis of the vector space dual to V. The rows of G are these basis vectors.

Since H is a $mt \times n$ matrix, the matrix G has dimension $n \times k$, with $k \ge n - mt$. Thus, it defines a (n, k) Goppa code, where $k \ge n - mt$.

2.2 The Minimum Distance of Irreducible Binary Goppa Codes

In this section we will determine the minimum distance of an irreducible binary Goppa code.

Let $\mathcal{G}(\mathbf{L}, g(X))$ be an irreducible binary Goppa code with $\mathbf{L} = (\gamma_0, \dots, \gamma_{n-1})$. Let $\mathbf{c} = (\mathbf{c}_0, \dots, \mathbf{c}_{n-1}) \in \mathcal{G}(\mathbf{L}, g(X))$ be a codeword and $\mathcal{T}_{\mathbf{c}} = \{i : \mathbf{c}_i = 1\}$. Then we define

$$\sigma_{\mathbf{c}}(X) = \prod_{j \in \mathcal{T}_{\mathbf{c}}} (X - \gamma_j) \in \mathbb{F}_{2^m}[X].$$

The derivative of $\sigma_{\mathbf{c}}(X)$ is

$$\sigma'_{\mathbf{c}}(X) = \sum_{i \in \mathcal{T}_{\mathbf{c}}} \prod_{j \in \mathcal{T}_{\mathbf{c}} \setminus \{i\}} (X - \gamma_j).$$

From (3) it follows

$$\sigma_{\mathbf{c}}(X) S_{\mathbf{c}}(X) \equiv \sigma'_{\mathbf{c}}(X) \mod g(X). \tag{5}$$

Since $g(\gamma_i) \neq 0$ for all $0 \leq i < n$, we have $\gcd(\sigma_{\mathbf{c}}(X), g(X)) = 1$. Therefore, $\sigma_{\mathbf{c}}(X)$ is invertible modulo $g(X)$ and we have

$$\frac{\sigma'_{\mathbf{c}}(X)}{\sigma_{\mathbf{c}}(X)} \equiv S_{\mathbf{c}}(X) \mod g(X).$$

It follows that

$$\forall \mathbf{c} \in \mathbb{F}_2^n : \quad \mathbf{c} \in \mathcal{G}(\mathbf{L}, g(X)) \Leftrightarrow \sigma'_{\mathbf{c}}(X) \equiv 0 \mod g(X).$$

The map $\mathbb{F}_{2^m} \longrightarrow \mathbb{F}_{2^m}$, $x \mapsto x^2$ is the Frobenius automorphism on $\mathbb{F}_{2^m}$, therefore every element $y \in \mathbb{F}_{2^m}$ has a unique square root.

The Frobenius map

$$\mathbb{F}_{2^m}[X] \longrightarrow \mathbb{F}_{2^m}[X], \quad f(X) = \sum_{i=0}^{n} f_i X^i \mapsto (f(X))^2 = \sum_{i=0}^{n} f_i^2 X^{2i}$$

is a injective, but not surjective, ring homomorphism. Its image is $\mathbb{F}_{2^m}[X^2]$, a set of polynomials, which are *perfect squares* of the ring $\mathbb{F}_{2^m}[X]$.

The polynomial $\sigma'_{\mathbf{c}}(X) = \sum_{i=1}^{n} i\sigma_i X^{i-1}$ is a perfect square, because in $\mathbb{F}_{2^m}$ we have $i\sigma_i X^{i-1} = 0$ for each even i. Since $g(X)$ is irreducible, we have

$$\forall \mathbf{c} \in \mathbb{F}_2^n : \quad \mathbf{c} \in \mathcal{G}(\mathbf{L}, g(X)) \Leftrightarrow \sigma'_{\mathbf{c}}(X) \equiv 0 \mod g^2(X).$$

Thus, for any codeword $\mathbf{c} \in \mathcal{G}(\mathbf{L}, g(X)) \setminus \{0\}$ we have

$$\mathrm{wt}(\mathbf{c}) = \deg \sigma_{\mathbf{c}}(X) \geq 1 + \deg \sigma'_{\mathbf{c}}(X) \geq 2 \deg g(X) + 1.$$

2.3 Error Correction for Irreducible Binary Goppa Codes

As mentioned above, the minimum distance of a Goppa code $\mathcal{G}$ which is generated by an irreducible polynomial of degree t is at least $2t+1$. Therefore, it is always possible to correct up to t errors. We now will describe such an error correction algorithm which corrects up to t errors in the case of irreducible binary Goppa code $\mathcal{G}(\mathbf{L},g(X))$. The error correction of non-binary or non-irreducible Goppa codes is slightly different and can be found in [36, 23].

Assume $\mathbf{m} \in \mathcal{G}(\mathbf{L}, g(X))$ is a codeword, $\mathbf{e} \in \mathbb{F}_2^n$ with $wt(\mathbf{e}) \leq t$ is an error vector, and

$$\mathbf{c} = \mathbf{m} \oplus \mathbf{e}.$$

Given $\mathbf{c}$, we want to compute $\mathbf{e}$ and $\mathbf{m}$.

Note that since $\mathbf{m}$ is a codeword, we have $S_{\mathbf{m}}(X) \equiv 0 \mod g(X)$ and therefore

$$S_{\mathbf{c}}(X) \equiv S_{\mathbf{e}}(X) \mod g(X).$$

First, we define the *error locator polynomial* $\sigma_{\mathbf{e}}(X)$. For $\mathcal{T}_{\mathbf{e}} = \{i : \mathbf{e}_i = 1\}$, we set

$$\sigma_{\mathbf{e}}(X) = \prod_{j \in \mathcal{T}_{\mathbf{e}}} (X - \gamma_j) \in \mathbb{F}_{2^m}[X].$$

From (3), it follows

$$\sigma_{\mathbf{e}}(X) S_{\mathbf{e}}(X) \equiv \sigma'_{\mathbf{e}}(X) \mod g(X). \tag{6}$$

We split $\sigma_{\mathbf{e}}(X)$ in squares and non-squares. Then we have

$$\sigma_{\mathbf{e}}(X) = \alpha^2(X) + X\beta^2(X).$$

Since the characteristic of the field is 2, we have $\sigma'_{\mathbf{e}}(X) = \beta^2(X)$. Thus equation (6) can be rewritten as follows

$$\beta^2(X)(XS_{\mathbf{e}}(X) + 1) \equiv \alpha^2(X) S_{\mathbf{e}}(X) \mod g(X) \tag{7}$$

We can assume that $\mathbf{e}$ is not a codeword, thus $S_{\mathbf{e}}(X) \not\equiv 0 \mod g(X)$. Therefore, there exists an inverse of $S_{\mathbf{e}}(X)$ modulo $g(X)$. We set $T(X) = S_{\mathbf{e}}^{-1}(X)$, and multiply equation (7) by $T(X)$. Then we obtain

$$\beta^2(X)(X + T(X)) \equiv \alpha^2(X) \mod g(X) \tag{8}$$

As mentioned in the last section, each element of $\mathbb{F}_{2^{mt}}$ has a unique square root. So let $\tau(X) \in \mathbb{F}_{2^m}[X]$ be the unique square root of the polynomial $T(X) + X$, i.e. $\tau(X)\tau(X) \equiv T(X) + X \mod g(X)$. Taking the square root of equation (8) we obtain

$$\beta(X)\tau(X) \equiv \alpha(X) \mod g(X). \tag{9}$$

In order to solve the last equation for known $\tau(X)$ and $g(X)$, we have to determine $\alpha(X)$ and $\beta(X)$ of least degree. By assumption we have $\deg(\sigma_{\mathbf{e}}(X)) \leq t$. It follows that $\deg(\alpha(X)) \leq \lfloor t/2 \rfloor$ and $\deg(\beta(X)) \leq \lfloor (t-1)/2 \rfloor$. This yields a unique solution of equation (9) which can be found by applying the extended Euclidean algorithm. We recall that this algorithm may be used to compute polynomials $\alpha_k(X) + \beta_k(X)\tau_k(X) \equiv 0 \mod g(X)$ in each step with $\deg(\beta_k(X)) = \deg(g(X)) - \deg(\alpha_{k-1}(X))$. This last formula presents the relation between the degrees of α and β. After each step, the degree of β increases as the degree of α decreases. Using this, one can see that there is a unique point in the computation of the Euclidean algorithm, where the degree of both polynomials is below the respective bound. More precisely, we run the algorithm until $\deg(\alpha_k(X))$ drops below $\lfloor (t+1)/2 \rfloor$ for the first time and get

$$\deg \alpha_k(X) \leq \lfloor (t+1)/2 \rfloor - 1 \leq \lfloor t/2 \rfloor.$$

In this round of the algorithm the following holds:

$$\begin{aligned} \deg \beta_k(X) \quad &= \deg(\beta_k(X)) = \deg(g(X)) - \deg(\alpha_{k-1}(X)) \\ &\leq t - \lfloor (t+1)/2 \rfloor = \lfloor (t-1)/2 \rfloor. \end{aligned}$$

Now, we set $\alpha(X) = \alpha_k(X)$ and $\beta(X) = \beta_k(X)$ (see algorithm 2.3.1). In [36, 34, 23], it is shown in more detail that they fulfill equation (9) and are unique.

Finally, the computation of zeroes for $\sigma_{\mathbf{e}}(X) = \alpha^2(X) + X\beta^2(X)$ leads to vectors $\mathbf{e}$ and $\mathbf{m}$. We present the complete algorithm on the following page.

Now, we analyze the runtime of the presented error correction algorihm. To compute the syndrome $S_{\mathbf{c}}(X)$ employing the check matrix H, we need at most $(n-k)n$ binary operations. To compute $T(X)$, we employ the extended Euclidean algorithm. This takes $\mathcal{O}\left(t^2m^2\right)$ binary operations, as the computations are modulo $g(X)$, a polynomial of degree t and coefficients of size m. Computing the sqare root of $T(X) + X$ takes $\mathcal{O}\left(t^2m^2\right)$ operation since it is a linear mapping on $\mathbb{F}_{2^m}[X]/g(X)$. The subsequently employed variant of the extended Euclidean algorithm takes $\mathcal{O}\left(t^2m^2\right)$ binary operations, too. These steps are all comparatively easy in comparison to the last step of the algorithm, which is to find all roots of the error locator polynomial. This last step can be performed in $n(tm^2 + tm)$ binary operations, thus the whole error correction algorithm needs

$$\mathcal{O}\left(n \cdot t \cdot m^2\right)$$

binary operations, as $mt \geq (n-k)$.

3 Attacks on the Private Key

In the following sections we present several attacks on the McEliece PKC. In this section we view attacks that aim to get the private key from the public key. We will see that not every class of linear codes is a secure choice for the McEliece cryptosystem.

Algorithm 2.3.1 Error Correction of Binary Irreducible Goppa Codes

Input: A binary irreducible Goppa code $\mathcal{G}(\mathbf{L}, g(X))$, a vector $\mathbf{c} = \mathbf{m} \oplus \mathbf{e}$, where $\mathbf{m}$ is a codeword and $\mathbf{e}$ is an error vector.
Output: The message $\mathbf{m}$ and the error vector $\mathbf{e}$.

```
/* Compute the syndrome of c */
S_c(X) = sum_{i=0}^{n-1} c_i/(X - γ_i)  mod g(X) (or use the parity check matrix H)

if S_c(X) ≡ 0  mod g(X) then
  /* there is no error, c is a codeword */
  return(c, 0)
else
  /* there are errors, c is not a codeword */
  T(X) ≡ S_c^{-1}(X)  mod g(X)
  τ(X) ≡ sqrt(T(X) + X)  mod g(X)

  /* extended Euclidean algorithm */
  i = 0; r_{-1}(X) = α_{-1}(X) = g(X); r_0(X) = α_0(X) = τ(X); β_{-1}(X) = 0;
  β_0(X) = 1
  while deg(r_i(X)) ≥ ⌊(t + 1)/2⌋ do
    i = i + 1
    Determine q_i(X) and r_i(X), s.t. r_i(X) = r_{i-2}(X) − q_i(X)r_{i-1}(X)
        and deg(r_i(X)) < deg(r_{i-1}(X))
    β_i(X) = β_{i-2}(X) + q_i(X)β_{i-1}(X)
    α_i(X) = r_i(X)

  σ(X) = c^2((α_i(X))^2 + X(β_i(X))^2) with c ∈ F_{2^m}, s.t. σ(X) is monic

  /* Determination of zeroes of σ_e(X) */
  for i = 0 to n − 1 do
    if σ(γ_i) = 0 then
      e_i = 1
    else
      e_i = 0
  m = c ⊕ e
  return(m, e)
```

3.1 The importance of S, P and M

Suppose, the set $\mathbf{L}$ which was used to generate the secret Goppa code for some public key of the McEliece PKC is known. This is true for normal applications, and if P is secret, then $\mathbf{L}$ may be revealed without security problems.

Suppose that g is unknown. Let H' be the systematic dual matrix of $\mathsf{SG}' = \mathsf{G}$. Assume further, that an attacker is able to recover P and M such that $\mathsf{M}^{-1}\mathsf{H}'\mathsf{P}^{-1} = \mathsf{H}$, where $\mathsf{H} = \mathsf{XYZ}$ has the form given in section 2 (represented over $\mathbb{F}_2$). Then he can compute g in the following way: The matrix $g_t\mathsf{Z}$ is written in in the first m rows of H. The matrix Y is determined by $\mathbf{L}$. Thus the attacker can recover (X/g_t) by solving some linear equations. Since g defines the same Goppa code as (g/g_t), the attacker is now able to correct errors efficiently. This breaks Niederreiter's as well as McEliece's cryptosystem.

If the matrix P is revealed, it is easy to recover the generator polynomial from $\mathsf{H}'\mathsf{P}^{-1}$ using equation (6), as $S_{\mathbf{c}}(X) = 0$ for every binary n vector $\mathbf{c}$ with $\mathsf{H}'\mathsf{P}^{-1}\mathbf{c}^\top = 0$.

The secret matrix S indeed has no cryptographic function in hiding the secret polynomial g. Today, there is no way to recover H with the knowledge of $\mathsf{S}^{-1}G$ only.

For the security of the McEliece PKC it is absolutely crucial to keep M secret. The knowledge of $\mathsf{M}^{-1}\mathsf{H}' = \mathsf{HP}$ is sufficient to recover g. We may interpret $\mathsf{M}^{-1}\mathsf{H}'$ to be a matrix over $\mathbb{F}_{q^m}$. As we will see in the following, this allows an efficient computation of g and P.

3.2 Attack on the original Niederreiter PKC

Niederreiter proposed his cryptosystem originally using generalized Reed-Solomon (GRS) codes. In 1992 V.M. Sidelnikov and S.O. Shestakov proposed a attack on Niederreiter's cryptosystem using GRS codes [47] which reveals an alternative private key in polynomial time. We consider this attack to be worth mentionable, as Goppa codes are subfield subcodes of GRS codes. Even though, the results from [47] do not affect the security of the original McEliece PKC.

In their attack, Sidelnikov and S.O. Shestakov take advantage of the fact, that the check matrix of GRS code is of the form

$$\bar{\mathsf{H}} = \begin{pmatrix} z_1 a_1^0 & z_1 a_1^1 & \cdots & z_1 a_1^s \\ z_2 a_2^0 & z_2 a_2^1 & \cdots & z_2 a_2^s \\ \vdots & & \ddots & \vdots \\ z_n a_n^0 & z_n a_n^1 & \cdots & z_n a_n^s \end{pmatrix} \in \mathbb{F}_q^{n\times(s+1)} . \tag{10}$$

Note that the matrix $\mathsf{X}^{-1}\mathsf{H} = \mathsf{YZ}$ from section 2 is of this from, too. It follows, that the matrix $\bar{\mathsf{H}}$ is a check matrix of a Goppa code, or to say it differently, each Goppa code is a subfield subcode of a GRS code.

A public Niederreiter key is of the form $\mathsf{H}' = \mathsf{P}\bar{\mathsf{H}}\mathsf{M}$, where M is a non-singular matrix and P a permutation matrix. The permutation matrix P does not change the structure of $\bar{\mathsf{H}}$, so we don't have to worry about P. The entries of H' can be viewed as the values of polynomials $M_{\cdot i}$ (whose coefficients are represented by the i-th column of M and therefore are denoted in the same way) multiplied by z_j:

$$\mathsf{H}' = \begin{pmatrix} z_1 M_{\cdot 1}(a_1) & z_1 M_{\cdot 2}(a_1) & \cdots & z_1 M_{\cdot s}(a_1) \\ z_2 M_{\cdot 1}(a_2) & z_2 M_{\cdot 2}(a_2) & \cdots & z_2 M_{\cdot s}(a_2) \\ \vdots & & \ddots & \vdots \\ z_n M_{\cdot 1}(a_n) & z_n M_{\cdot 2}(a_n) & \cdots & z_n M_{\cdot s}(a_n) \end{pmatrix},$$

where $M_{\cdot i}(x) = \sum_{j=0}^{s} \mathsf{M}_{ji} x^j$.

Sidelnikov and Shestakov conclude, that each entry of the row $\mathsf{H}'_{i\cdot}$ can be expressed by a polynomial in a_i. From this observation one can derive a system of polynomial equations whose solution yields the private key. We will need the notation $\bar{\mathsf{H}} = \mathsf{Z} \cdot \mathsf{A}$ with $\mathsf{A} := \mathsf{Z}^{-1}\bar{\mathsf{H}}$ and the diagonal matrix $\mathsf{Z} := \operatorname{diag}[z_1, \cdots, z_n]$.

We want to assume without loss of generality that $a_1 = 1$ and $a_2 = 0$. In order to do this, we have to view the matrices $\bar{\mathsf{H}}$, M and H' as matrices over $\mathbb{F} := \mathbb{F}_q \cup \infty$ with $1/\infty = 0$, $1/0 = \infty$ and $f(\infty) = f_{\deg f}$ for every polynomial $f(x) = \sum_{j=0}^{\deg f} f_j x^j$ over $\mathbb{F}_q$. Sidelnikov and Shestakov show that for every birational transformation ($\mathbb{F}$-automorphism)

$$\phi(x) = \frac{ax+b}{cx+d} \text{ with } a, b, c, d \in \mathbb{F}_q,\ ad - bc \neq 0$$

there exist $z'_1, \cdots, z'_1$ and a matrix M' such that

$$\mathsf{H}' = \begin{pmatrix} z'_1\phi(a_1)^0 & z'_1\phi(a_1)^1 & \cdots & z'_1\phi(a_1)^s \\ z'_2\phi(a_2)^0 & z'_2\phi(a_2)^1 & \cdots & z'_2\phi(a_2)^s \\ \vdots & & \ddots & \vdots \\ z'_n\phi(a_n)^0 & z'_n\phi(a_n)^1 & \cdots & z'_n\phi(a_n)^s \end{pmatrix} \cdot (\mathsf{M}')^{-1}\,\mathsf{M}.$$

For every three numbers $a_1, a_2, a_3 \in \mathbb{F}_q$ it is possible to find a birational transformation ϕ s.t.

$$\begin{aligned} \phi(a_1) &= 1 &&= x_1 \\ \phi(a_2) &= 0 &&= x_2 \\ \phi(a_3) &= \infty &&= x_3 \\ \phi(a_j) & &&= x_j, j \notin \{1,2,3\}. \end{aligned}$$

Thus we can make the assumption mentioned above. Note that because $x_3 = \infty$ we have $x_i \neq \infty$ for all $i \neq 3$.

We can use Algorithm 3.2.1 to recover a (alternative) private Niederreiter key from the public key. The algorithm generates a system of polynomial equations based on the assumption $x_1 = 1$, $x_2 = 0$, $x_3 = \infty$ and solves it. We are going to explain the algorithm briefly. First we have to remember the identification of the entries of H' with polynomials evaluated at the a_j. Thus for $c_i \in \mathbb{F}_q^{s+1}$, $i = 1, 2$ and $j \in \{1, \cdots, n\}$, the scalar product $\frac{1}{z_j}\mathsf{H}'_{j\cdot} c_i$ is the value of a polynomial π_i at x_j, where π_i is of degree at most s. Defining $J_1 = \{1, s+2, s+3, \cdots, 2s\}$ and $J_2 = \{2, s+2, s+3, \cdots, 2s\}$ we can solve

$\mathsf{H}'_{J_i}.c_i = 0$ for $i = 1, 2$ we get two polynomials π_1, π_2 with zeroes in $x_{s+2}, \cdots, x_{2s}$ and in x_1, x_2 respectively. We know that $x_1 = 1, x_2 = 0$ thus

$$\frac{\mathsf{H}'_{j}.c_1}{\mathsf{H}'_{j}.c_2} = \frac{\pi_1(x_j)}{\pi_2(x_j)} = \frac{\pi_1(\infty)}{\pi_2(\infty)} \cdot \frac{x_j - 1}{x_j} = \frac{\pi_1(x_3)}{\pi_2(x_3)} \cdot \frac{x_j - 1}{x_j},$$

which reveals x_j for $j \notin \{1, 2, s+2, \cdots, 2s\}$. To determine the missing x_j, $j \in \{s+2, \cdots, 2s\}$ we repeat the procedure (introducing c_3, J_3, c_4 and J_4) and take into account the knowledge of the already determined x_j. Afterwards we perform another birational transformation ϕ' on the x_j s.t. $a_i = \phi'(x_j)$ are finite.

Knowing all a_i, $i \in \{1, \cdots, n\}$ we are able to recover $z_2, \cdots, z_{s+2}$ assuming that $z_1 = 1$. Defining $J_5 := \{1, 2, \cdots s+2\}$ and solving $c_5 \mathsf{H}'_{J_5 \cdot} = 0$ for $c_5 \in \mathbb{F}_q^{s+1}$ we get a polynomial s.t. $\sum_{j=1}^{s+2} c_{5j} z_j M_{i\cdot}(x_j) = 0$ for $i = 1, \cdots, s+2$. Expressing this in matrix form we get:

$$c_5(\bar{\mathsf{H}}\mathsf{M})_{J_5 \cdot} = c_5(\mathsf{ZA})_{J_5}\mathsf{M} = 0$$

and consequently we know that $c_5(\mathsf{ZA})_{J_5} = 0$, which gives us a linear system with $s+1$ unknowns and $s+1$ equations since z_1, A and c_5 are already known. Now we can determine M and in continuation the remaining z_j. Algorithm 3.2.1 has a running time of $\mathcal{O}(s^4 + sn)$. For details see [47].

Remark 3.2.1 Algorithm 3.2.1 can not be applied to McEliece/Niederreiter cryptosystems using Goppa codes. Even though for every Goppa code there is a checkmatrix H which has the same structure as the check matrix $\bar{\mathsf{H}}$ for GRS codes in equation (10) (see [36]), there is no analogous interpretation of H' for the Niederreiter cryptosystem using Goppa codes. We are able to view H as a matrix over $\mathbb{F}_2$ if we are using Goppa codes, whereas this doesn't work for GRS codes. Thus we have different matrices M: $\mathsf{M} \in \mathbb{F}_{2^m}^{(s+1)\times(s+1)}$ for the GRS case and $\mathsf{M} \in \mathbb{F}_2^{m(s+1)\times m(s+1)}$ for Goppa codes. Thus, in the latter case, H' has no obvious structure, as long as M is unknown.

3.3 Weak Keys and the Support Splitting Algorithm

P. Loidreau and N. Sendrier proposed a way to identify a subclass of Goppa codes, namely the ones with binary generator polynomial $g \in \mathbb{F}_2[X]$. If an attacker knows, that the secret generator polynomial is binary, this reduces the search space of a brute force attack on the private key [35]. Their general idea is to take advantage of the *Support Splitting Algorithm* (SSA) presented in [44]. The SSA can be used as an oracle to decide whether two codes are permutation equivalent as well as to determine the automorphism group of a code. P. Loidreau and N. Sendrier use this ability, to determine if the generator polynomial of a Goppa code is a binary (irreducible) polynomial. If this is the case, we search the space of the Goppa codes with binary generator polynomial for a code, which is equivalent to the one given by the public generator matrix.

Algorithm 3.2.1 GRSrecover [47]

Input: $\mathsf{H}' = \left(h'_{ij}\right) \in \mathbb{F}_q^{n\times(s+1)}$ and t, a Niederreiter Public key.
Output: $\bar{\mathsf{H}}$,P of the corresponding private Niederreiter Key.

$J_1 = \{1, s+2, s+3, \cdots, 2s\}$; $J_2 = \{2, s+2, s+3, \cdots, 2s\}$;
$J_3 = \{1, 3, 4, \cdots, s+1\}$; $J_4 = \{2, 3, 4, \cdots, s+1\}$; $J_5 = \{1, 2, \cdots s+2\}$;
for $i = 1$ to 4 **do**
  solve $\mathsf{H}'_{J_i\cdot}\mathbf{c}_i = 0$ with $\mathbf{c}_i \in \mathbb{F}_q^{s+1} \setminus 0$;
for $j \notin J_1 \cup J_2$ **do**
  //
  $\beta_{1j} = \mathsf{H}'_{j\cdot}\mathbf{c}_1$; $\beta_{2j} = \mathsf{H}'_{j\cdot}\mathbf{c}_2$;
  $b_j = \beta_{1j}/\beta_{2j}$;
for $j \in \{n, 2s, \cdots, s+2\}$ **do**
  $\beta_{3j} = H_{j\cdot}\mathbf{c}_3$; $\beta_{4j} = H_{j\cdot}\mathbf{c}_4$;
  $b_j = \frac{b_n\beta_{4n}}{\beta_{3n}} \cdot \frac{\beta_{3j}}{\beta_{4j}}$; *// Note, that we already know b_n.*
$x_1 = 1$; $x_2 = 0$; $x_3 = \infty$;
for $j = 4$ to n **do**
  // Determining the values of x_j.
  $x_j = b_3/\left(b_3 - b_j\right)$;
choose some $a \in \mathbb{F}_q$ differing from all x_j;
for $j = 1$ to n **do**
  // Mapping the x_j to finite elements.
  $a_j = \left(a - x_j\right)^{-1}$; $\mathsf{A}_{j\cdot} = \left(a_j^0, \cdots, a_j^s\right)$;
solve $\mathbf{c}_5\mathsf{H}'_{J_5\cdot} = 0$ with $\mathbf{c}_5 \in \mathbb{F}_q^{s+1} \setminus 0$;
$z_1 = 1$;
find $z_2, \cdots, z_{s+2} \in \mathbb{F}_q$ s.t. $\sum_{j=1}^{s+2} c_{5j} z_j \mathsf{A}_{j\cdot} = 0$;
for $i = 0$ to s **do**
  solve $\mathsf{A}_{J_5\cdot}\mathsf{M}_{\cdot i} = \left(z_j^{-1}\mathsf{H}'_{ji}\right)^{\top}_{j \in J_5}$;
$\mathsf{M} = (\mathsf{M}_{\cdot 0}, \cdots, \mathsf{M}_{\cdot s})$;
for $j = 3$ to n **do**
  $z_j = \mathsf{H}'_{j\cdot}\left(\mathsf{M}^{-1}\right)_{\cdot 0}$;
Return $a_1, \cdots, a_n, z_1, \cdots, z_n, \mathsf{M}$;

If such a code is found, the SSA can be used to recover the permutation matrix P. There is another attack by Gibson [20], which aims to recover the matrix P, but we forbear presenting it here, as its average work factor is larger than $2^{nm(1+\mathcal{O}(1))}$ binary operations [43].

The Support Splitting Algorithm was presented to solve the problem to decide whether two codes are permutation equivalent in (almost) polynomial time. We will explain it in the following. Our notation in the following presentation of the algorithm will differ slightly from that used in [44] so as not to confuse the reader of the paper with two different definitions of a signature. The main idea is to partition the index set of the code $\mathcal{C}$ into small sets, which are fixed under operation of elements of $\mathrm{Aut}\left(\mathcal{C}\right)$. We have to introduce some definitions first:

Definition 3.3.1 Let $\mathcal{L}$ be the set of all codes and let M be a arbitrary set. A function $f : \mathcal{L} \times \mathbb{N} \mapsto M$ is called *permutation invariant* if for all (n,k) codes $\mathcal{C}$ and all permutations π on $\{1,\cdots,n\}$ the equation $f\left(\mathcal{C},i\right) = f\left(\pi\left(\mathcal{C}\right),\pi\left(i\right)\right)$ holds. A permutation invariant function f is called *discriminant for* $\mathcal{C}$ if there exist $i,j \in \{1,\cdots,n\}$ s.t. $f\left(\mathcal{C},i\right) \neq f\left(\mathcal{C},j\right)$. It is further called *fully discriminant for* $\mathcal{C}$ if

$$\forall_{i,j\in\{1,\cdots,n\}} : i \neq j \Rightarrow f\left(\mathcal{C},i\right) \neq f\left(\mathcal{C},j\right)$$

If we have two permutation equivalent codes $\mathcal{C}$ and $\mathcal{C}'$ and a fully discriminant function for $\mathcal{C}$, then we are able to name the permutation π s.t. $\pi\left(\mathcal{C}\right) = \mathcal{C}'$. In order to build a discriminant function for $\mathcal{C}$, we employ the *weight enumerator* and *punctured codes*:

Definition 3.3.2 Let $\mathcal{C}$ be an (n,k) code over $\mathbb{K}$. Let J be any subset of $\{1,\cdots,n\}$. Then the code $\mathcal{C}$ punctured in J is defined by

$$\mathcal{C}_J = \{\mathbf{x} \in \mathbb{F}^n | \mathbf{x}_J = \mathbf{0} \text{ and } \exists_{\mathbf{y}\in\mathcal{C}} \forall_{j\notin J} \mathbf{x}_j = \mathbf{y}_j\} \ .$$

The weight enumerator $\mathcal{W} : \mathcal{L} \mapsto \mathbb{N}^{\mathbb{N}}$ is the function s.t. $\mathcal{W}\left(\mathcal{C}\right)_i$ is the number of words of weight i in the code $\mathcal{C}$ for all $i \in \mathbb{N}$.

Example 3.3.3 The function $\mathcal{W}' : \mathcal{L} \times \mathbb{N} \to \mathbb{N}^{\mathbb{N}}, (\mathcal{C},i) \mapsto \mathcal{W}\left(\mathcal{C}_{\{i\}}\right)$ is permutation invariant. Furthermore, $\mathcal{W}'$ is discriminant for most binary (n,k) codes $\mathcal{C}$.

We are going to use discriminant functions to partition the index set of a code. Starting with a function f discriminant for $\mathcal{C}$, we want to construct a function g *more discriminant for* $\mathcal{C}$ in the sense of

$$\left|g\left(\mathcal{C},\{1,\cdots,n\}\right)\right| \geq \left|f\left(\mathcal{C},\{1,\cdots,n\}\right)\right|$$

for the (n,k) code $\mathcal{C}$. The function g is called *strictly more discriminant for* $\mathcal{C}$ if we can replace $\geq$ with $>$ in the inequality above. We repeat this process until we get a fully discriminant function for $\mathcal{C}$. The following two definitions will enable us to do so.

Definition 3.3.4 Let f, g be two permutation invariant functions. We define the *product* of f and g as

$$\begin{aligned} f \times g : \; & \mathcal{L} \times \mathbb{N} \to M \times M, \\ & (\mathcal{C}, i) \mapsto (f(\mathcal{C}, i), g(\mathcal{C}, i)) \ , \end{aligned}$$

and the *dual* of f as

$$\begin{aligned} f^{\perp} : \; & \mathcal{L} \times \mathbb{N} \to M, \\ & (\mathcal{C}, i) \mapsto f\left(\mathcal{C}^{\perp}, i\right) \ . \end{aligned}$$

The function f is called *self-dual* if $f = f^{\perp}$.

It is easy to see, that $f \times g$ is more discriminant than f. With the definitions above we are able to describe the Support Splitting Algorithm (algorithm 3.3.1). It mainly consists in a while-loop in which definitions 3.3.4 and 3.3.2 are used to get more discriminant functions for a given code $\mathcal{C}$, until a fully discriminant function for $\mathcal{C}$ is generated. After the while-loop the index set of $\mathcal{C}$ is partitioned in a standardized way.

Algorithm 3.3.1 Support Splitting Algorithm (SSA)

Input: G generator matrix of a linear $(n \times k)$ code $\mathcal{C}$,
$S : \mathcal{L} \times \mathbb{N} \to M$ permutation invariant discriminant for $\mathcal{C}$.

Output: $\mathcal{P} = \{(\mathcal{P}_j, j)\}_{1 \le j \le n}$, $\mathcal{P}_j \subseteq \{1, \cdots, n\}$, called labeled partition.
T a permutation invariant, discriminant function for $\mathcal{C}$.

$I_n = \{1, \cdots, n\}$;
$j = 0$;
$T_0 = S$;
while (a function strictly more discriminant for $\mathcal{C}$ than T_j exists) **do**
  choose $L \subset T_i(\mathcal{C}, I_n)$ at random;
  $T_{j+1}(\mathcal{C}, i) = T_j(\mathcal{C}, i) \times S\left(\mathcal{C}_{\{i \in I_n | T(\mathcal{C}, i) \in L\}}, i\right) \times S^{\perp}\left(\mathcal{C}_{\{i \in I_n | T(\mathcal{C}, i) \in L\}}, i\right)$;
  $j = j + 1$;
$T = T_j$;
for $j = 1$ to n **do**
  if $j \in \bigcup_{1 \le i < j} \mathcal{P}_i$ **then**
    $\mathcal{P}_j = \emptyset$;
  else
    $\mathcal{P}_j = \{i \in I_n | T(\mathcal{C}, i) = T(\mathcal{C}, j)\}$;

There are two main difficulties with the algorithm. The fist one is, that it won't terminate if we are not able to generate a fully discriminant function for $\mathcal{C}$ in the while-loop. Only then we would know, that there does not exist any further refinement of T_j. However, remark 3.3.7 will give us a termination criterion for binary Goppa codes. The second difficulty is to find a good choice for the function S. According to [35] and [44] for binary codes $\mathcal{C}$ we choose

$$\begin{aligned} S : \; & \mathcal{L} \times \mathbb{N} \to \mathbb{N}^{\mathbb{N} \times \mathbb{N}} \\ & (\mathcal{C}, i) \mapsto \left(\mathcal{W}\left(\mathcal{C}_{\{i\}} \cap \left(\mathcal{C}_{\{i\}}\right)^{\perp}\right), \mathcal{W}\left(\left(\mathcal{C}^{\perp}\right)_{\{i\}} \cap \left(\left(\mathcal{C}^{\perp}\right)_{\{i\}}\right)^{\perp}\right)\right) \end{aligned} \tag{11}$$

as input for SSA, where $\mathcal{W}$ is the weight enumerator. This function is discriminant in practice. Choosing suitable criteria for exiting the while-loop, algorithm 3.3.1 runs in time

$$\mathcal{O}\left(n^3 + 2^{\dim(\mathcal{C}\cap\mathcal{C}^\perp)} n^2 \log(n)\right) \ , \tag{12}$$

see [35]. To see that the average running time of SSA is polynomial bounded we need to estimate the $\dim\left(\mathcal{C}\cap\mathcal{C}^\perp\right)$-term in equation (12) and the cost for computing the weight enumerator $\mathcal{W}$. The worst-case computation cost of $\mathcal{W}$ for a q-ary code of length n and dimension k is proportional to nq^k operations in $\mathbb{F}_q$. However, the average cost of computing the weight enumerator is proportional to $2n$ operations [44]. We continue with determining the $\dim\left(\mathcal{C}\cap\mathcal{C}^\perp\right)$-term:

Proposition 3.3.5 *Let $\mathcal{C}$ be an (n,k) code over $\mathbb{F}_q$. We call $\mathcal{C}\cap\mathcal{C}^\perp$ the hull of $\mathcal{C}$. The average dimension of the hull of $\mathcal{C}$ tends to a constant when the size of the code goes to infinity. This constant is equal to*

$$R = \sum_{i=1}^{\infty} \frac{1}{q^i+1}.$$

The proportion of (n,k) codes over $\mathbb{F}_q$ with a hull of dimension $l \geq 0$ is asymptotically equal to

$$R_l = R_{l-1} / \left(q^l - 1\right) \text{ with } R_0 = \prod_{i=0}^{\infty} \frac{1}{1+q^{-i}}.$$

Proof. See [45], [44]. ∎

As we have already mentioned SSA is unlikely to terminate in the version of algorithm 3.3.1. Thus we have to make some assumptions on its output if we choose other termination criteria for the while-loop, than the one given in the algorithm. We will see, that these assumptions lead to a suitable termination criterion, if $\mathcal{C}$ is a Goppa code.

We write $\mathcal{P} = \text{SSA}(\mathcal{C})$ if the labeled partition $\mathcal{P} = \{(P_j, j)\}_{1\leq j\leq n}$ is output of SSA on input of the generator matrix of $\mathcal{C}$. The nonempty $\mathcal{P}_s$ of the output of SSA are called the *cells* of $\mathcal{P}$. Two labeled partitions $\mathcal{P}$ and $\mathcal{P}'$ are called *equivalent* iff a permutation $\tau \in \mathcal{S}_n$ exists, s.t. for all $s \in I_n$ $|\mathcal{P}_s| = |\mathcal{P}'_{\tau(s)}|$; we write $\mathcal{P} \equiv \mathcal{P}'$. The fundamental property of SSA is that

$$\mathcal{C} = \pi\left(\mathcal{C}'\right) \Rightarrow \mathcal{P} \equiv \mathcal{P}' \ ,$$

where $\pi \in \mathcal{S}_n$. Thus the output of SSA on input of two permutation equivalent codes is identical and the orbits of the elements of the code support under the action of $\text{Aut}(\mathcal{C})$ constitute the finest obtainable partition.

Assumption 1 *If SSA on input G and G' returns $\mathcal{P}, T$ and $\mathcal{P}', T'$ respectively, then*

$$\left(T\left(\langle\mathsf{G}\rangle, \mathbb{N}\right) = T\left(\langle\mathsf{G}'\rangle, \mathbb{N}\right) \ \wedge\ \mathcal{P} \equiv \mathcal{P}'\right) \quad\Rightarrow\quad \langle\mathsf{G}\rangle = \pi\left(\langle\mathsf{G}'\rangle\right) \ ,$$

This assumption is satisfied in practice, if the number of cells is larger than a few units. From this observation the following assumption about the behavior of SSA is derived:

Assumption 2 *On input of the generator matrix of $\mathcal{C}$ the SSA returns a labeled partition whose cells are the orbits of the elements of the code support under the action of* $\mathrm{Aut}(\mathcal{C})$.

Assumption 2 seems to hold for (binary) codes of length ≥ 50 and is based on experiments by P. Loidreau and N. Sendrier [35]. Now, if we know $\mathrm{Aut}(C)$, then we can easily determine for every for $\mathcal{C}$ discriminant function T whether there exists a strictly more discriminant function for $\mathcal{C}$, or not. Fortunately we can determine $\mathrm{Aut}(\mathcal{G})$ for a Goppa code $\mathcal{G}$ in some cases:

Theorem 3.3.6 *With the notation of remark 2.1.2. Let $\mathcal{G}(\mathbf{L}, g)$ be a binary (n, k) Goppa code defined by a generator polynomial $g \in \mathbb{F}_{q^m}[X]$ with coefficients from a subfield $\mathbb{F}_{q^s}$ of $\mathbb{F}_{q^m}$. If $n = q^m$, then* $\mathrm{AUT}(\mathcal{G})$ *contains the automorphism*

$$\sigma : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m}, x \mapsto x^{2^s}.$$

Note that the elements $x \in \mathbb{F}_{q^m}$ are the code support and corespond to positions which are determined by $\mathbf{L}$.

Proof. The proof is derived from a theorem by Moreno [36], [35]. ■

Here we will only consider $s = 1$, i.e. only binary Goppa codes with binary generator polynomial. In such cases, the group generated by the Frobenius field automorphism is in general exactly $\mathrm{AUT}(\mathcal{G})$ [35]. Based on this theorem and the assumptions above, we get the following termination criterion for algorithm 3.3.1:

Remark 3.3.7 Let $\mathcal{G}$ be a binary Goppa code over $\mathbb{F}_{q^m}$ with binary generator polynomial. Assume, that the group generated by the Frobenius field automorphism over $\mathbb{F}_{q^m}$ is exactly $\mathrm{Aut}(\mathcal{G})$. Let $\mathcal{P}^{\mathrm{Aut}}$ be the set of different orbits of the code support under the action of $\mathrm{Aut}(\mathcal{G})$. Then the condition

(a function strictly more discriminant for $\mathcal{G}$ than T_j exists)

in algorithm 3.3.1 is equivalent to

$$|T_j(\mathcal{G}, \mathbb{N})| < \left|\mathcal{P}^{\mathrm{Aut}}\right|.$$

Further, the running time of algorithm 3.3.1 is given by equation (12).

Let's return to the original problem. We do know the public McEliece key (G, t) and want to reconstruct the private key. If assumptions 1 and 2 hold, we can identify a weak key (i.e. a McEliece-Instance, generated with a binary generator polynomial) by comparing the cardinalities of SSA $(\langle \mathsf{G} \rangle)$ with the cardinalities of the different orbits of the elements of the code support under the action of Aut $(\langle \mathsf{G} \rangle)$: If the SSA does not terminate or returns a function T such that

$$|T(\mathcal{C}, \mathbb{N})| \not\leq \left|\mathcal{P}^{\mathrm{Aut}}\right|,$$

then we assume, that $\langle \mathsf{G} \rangle = \mathcal{G}$ does not have a binary generator polynomial. Otherwise, we identify a "weak key ", i.e. we assume, that $\mathcal{G}$ has a binary generator polynomial.

Once a weak key is identified, we can determine the binary Goppa polynomial used to generate the public key G by brute force. We check if

$$\mathrm{SSA}(\langle \mathsf{G} \rangle) \equiv \mathrm{SSA}(\mathcal{G}(\mathbf{L}, g(X)))$$

for all (irreducible) binary polynomials g of degree t, where $\mathcal{G}(\mathbf{L}, g(X))$ denotes the Goppa code defined by the set $\mathbf{L}$ and g (compare section 2). After having identified the generator polynomial of $\mathcal{G}$, one can determine the secret permutation matrix P. In order to do so, we have to pick a $i \in \{1, \cdots, n\}$ s.t. AUT $\left(\mathcal{G}_{\{i\}}\right) = \{1\}$ and a j out of the orbit of i under Aut $(\mathcal{G})$. Then $\mathcal{G}_{\{i\}}$ and $\langle \mathsf{G} \rangle_{\{j\}}$ are equivalent and we get the permutation by applying SSA to both. This produces partitionings containing only cells of cardinality one (under assumption 2) and the matches between the cells provide the permutation. The authors of [35] claim that most i serve the last condition. The number of irreducible polynomials of degree 50 is approximately 2^{44}. Thus the average runtime of the attack on weak keys for McEliece parameters $n = 1024$, $t = 50$ is

$$\left(2^{44} + 1\right) \mathcal{O}\left(n^3 + 2^R n^2 \log(n)\right) \approx 2^{75},$$

where R is given in proposition 3.3.5. We conclude, that the choice of $n = 1024$, $t = 50$ for McEliece does not reach the desired level of security, if we want to use binary generator polynomials.

There is a possibility to speed up this attack by a factor $(\log(n))^3$ if we first check the *idempotent subcodes* against each other in the brute force part of the attack, instead of comparing the Goppa codes themselves.

Definition 3.3.8 Let $\mathcal{G}$ be a Goppa code, then a word $a \in \mathcal{G}$ is called *idempotent* if

$$a = \left(a_{\gamma_0}, \cdots, a_{\gamma_{n-1}}\right) = \left(a_{\sigma^{-1}(\gamma_0)}, \cdots, a_{\sigma^{-1}(\gamma_{n-1})}\right) .$$

The set of all idempotents of $\mathcal{G}$ is a linear subcode of $\mathcal{G}$ and is called the *idempotent subcode* $\mathcal{I}_{\mathcal{G}}$ of $\mathcal{G}$.

The subcode $\mathcal{I}_{\mathcal{G}}$ may be mapped to a linear code $\mathcal{I}$ of length equal to the number of different orbits of $\mathbb{F}_{2^m}$ under σ [35]. The code $\mathcal{I}$ has the same dimension as $\mathcal{I}_{\mathcal{G}}$ and its length is shorter by a factor close to m. We conclude that the use of the idempotent subcode provides a speedup of the attack close to the factor m^3, thus the choice of a binary generator polynomial for the secret Goppa codes does not provide sufficient security, even for parameter sets with $n > 1024$.

Remark 3.3.9 This attack may be generalized to detect Goppa codes with a generator polynomial over any subfield of $\mathbb{F}_{2^m}$ but the class detected this way is much too big to perform an exhaustive search. Further, the number of polynomials classified by this property is much too small to provide an effective attack against the McEliece cryptosystem.

4 Ciphertext Only Attacks

In this section, we will first present algrorithms for solving the general decoding problem (see Problem 1.2.3). These algorithms yield to different attacks against cryptosystems based on linear error-correcting codes. On input of a code generator matrix G (a part of the public key) and a ciper-text $\mathbf{c}$, these attacks compute the plain text corresponding to the cipher text $\mathbf{c}$. Although these attacks require exponential time, they are faster than the brute force algorithm.

At the end of the section, we will describe an attack by Brickell and Odlyzko [6] based on lattice reduction and show why this attack does not work with McEliece or Niederreiter cryptosystems based on binary Goppa codes.

4.1 Generalized Information-Set-Decoding Attack

This attack was proposed by McEliece in his original paper [37]. Lee and Brickell systematized and generalized it in [30]. It solves the general decoding problem assuming the knowledge of an upper bound for the distance to the next code word.

We will begin by presenting the idea of the attack. Assume we are given a generator matrix G of a linear error-correcting code and a cipher text $\mathbf{c} = \mathbf{m}\mathsf{G} \oplus \mathbf{e}$ where $\mathbf{e}$ is the error vector of weight t. Then, we randomly choose k columns of G and $\mathbf{c}$. If there is no error in the chosen columns of $\mathbf{c}$ and the $k \times k$ matrix builded from k columns of G is invertible, then we can easy determine $\mathbf{m}$.

Now we will give a detailed description of the attack. It proceeds as follows. Let $\mathcal{I} \subset \{0, \ldots, n-1\}$ with $|\mathcal{I}| = k = \dim \mathsf{G}$. As in section 1.2 we denote by $\mathsf{G}_{\mathcal{I}}$, $\mathbf{c}_{\mathcal{I}}$, and $\mathbf{e}_{\mathcal{I}}$ the k columns picked from G, $\mathbf{c}$, and $\mathbf{e}$, respectively. Then the following relationship is true

$$\mathbf{c}_{\mathcal{I}} = \mathbf{m}\mathsf{G}_{\mathcal{I}} \oplus \mathbf{e}_{\mathcal{I}}.$$

If $\mathsf{G}_{\mathcal{I}}$ is non-singular and $\mathbf{e}_{\mathcal{I}} = 0$, then

$$\mathbf{m} = \mathbf{c}_{\mathcal{I}}\mathsf{G}_{\mathcal{I}}^{-1}.$$

If $\mathsf{G}_{\mathcal{I}}$ is non-singular and $\mathrm{wt}(\mathbf{e}_{\mathcal{I}})$ is small, then $\mathbf{m}$ can be recovered by guessing $\mathbf{e}_{\mathcal{I}}$ and checking whether

$$\mathrm{wt}((\mathbf{c}_{\mathcal{I}} \oplus \mathbf{e}_{\mathcal{I}})\mathsf{G}_{\mathcal{I}}^{-1}\mathsf{G} \oplus \mathbf{c}) = t.$$

We will estimate the work factor of this attack (see Algorithm GISD). The number of sets $\mathcal{I}$, such that there are exactly i errors in vector $\mathbf{c}_{\mathcal{I}}$ is $\binom{t}{i}\binom{n-t}{k-i}$.

Algorithm 4.1.1 GISD

Input: $n \times k$ generator matrix G, a cipher text $\mathbf{c} = \mathbf{m}\mathsf{G} \oplus \mathbf{e}$, where $\mathbf{m}$ is the plain-text and $\mathbf{e}$ is the error vector of weight t, a positive integer $j \leq t$.
Output: The plain-text $\mathbf{m}$

```
while true do
  Choose randomly I ⊂ {0, ..., n − 1}, with |I| = k.
  Q1 = G_I^{-1}; Q2 = Q1 G
  z = c ⊕ c_I Q2
  for i = 0 to j do
    for all e_I with wt(e_I) = i do
      if wt(z ⊕ e_I Q2) = t then
        return((c_I ⊕ e_I)Q1)
```

The number of all sets $\mathcal{I}$ with $|\mathcal{I}| = k$ is $\binom{n}{k}$. Therefore, the expected number for choosing the set $\mathcal{I}$ such that there are at most j errors in vector $\mathbf{c}_{\mathcal{I}}$ is

$$T_j = \frac{\binom{n}{k}}{\sum_{i=0}^{j} \binom{t}{i} \binom{n-t}{k-i}}.$$

The number of error vectors $\mathbf{e}_{\mathcal{I}}$ with $\mathrm{wt}(\mathbf{e}_{\mathcal{I}}) \leq j$ is

$$N_j = \sum_{i=0}^{j} \binom{k}{i}.$$

Therefore the expected work factor of the attack for given j and (n, k) Goppa code with minimum distance $2t + 1$ is

$$W_j = \alpha T_j (k^3 + N_j k),$$

where α is a small constant.

In [30] the authors propose to use $j = 2$ to minimize the W_j.

4.2 Finding-Low-Weight-Codeword Attacks

In this section, we will present three algorithms which solve the problem of finding weights (see Problem 1.2.4). These algorithms can be used to break McEliece or Niederreiter cryptosystems in the following way. Assume we know a generator matrix G of a linear error-correcting code with minimum distance t and a cipher text $\mathbf{c} = \mathbf{m}\mathsf{G} \oplus \mathbf{e}$, where $\mathrm{wt}(\mathbf{e}) < t/2$. We compute the codeword with the minimum weight in a new code generated by matrix

$$\begin{pmatrix} \mathsf{G} \\ \mathbf{c} \end{pmatrix}.$$

Since this codeword is $\mathbf{e}$, this attack can be used to recover the plain text $\mathbf{m}$ from the given cipher text $\mathbf{c}$.

All three algorithms presented below are based on the same idea. Assume we have a code $\mathcal{C}$ given by a generator matrix G. The algorithms first search for codewords of small weight in a restricted code generated by $\mathsf{G}_{\mathcal{S}}$ where $\mathcal{S}$ is a random subset of $\{0, \ldots, n-1\}$. Then, they expand these codewords to codewords in $\mathcal{C}$ and check whether the codewords in $\mathcal{C}$ have the desired weight. The algorithms differ in the way of choosing for set $\mathcal{S}$ and the strategy of searching for codewords of small weight in the restricted code.

Before we describe the algorithms, we will will give some necessary notations and definitions.

Let $\mathcal{N} = \{0, \ldots, n-1\}$ be the set of all coordinates. As in the last section, we will use the set $\mathcal{I} \subset \mathcal{N}$ with $|\mathcal{I}| = k = \dim \mathsf{G}$.

By $\mathsf{G} = (\mathsf{V}, \mathsf{W})_{\mathcal{I}}$, we will denote the decomposition of G in two matrices V and W, such that $\mathsf{V} = (\mathsf{G}_i)_{i \in \mathcal{I}}$ and $\mathsf{W} = (\mathsf{G}_i)_{i \notin \mathcal{I}}$, where G_i is the i-th column of G.

Now, we will introduce the information set which allows us to reduce the computation cost in the algorithms we will present below.

Definition 4.2.1 Let $\mathcal{I} \subseteq \mathcal{N}$, such that $|\mathcal{I}| = k$. Then $\mathcal{I}$ is an *information set* for the code $\mathcal{C}$ iff there is a generator matrix G for $\mathcal{C}$ such that $\mathsf{G} = (\mathsf{Id}_k, \mathsf{Z})_{\mathcal{I}}$.

The following statement for information sets is true.

Theorem 4.2.2 *Let $\mathcal{I}$ be an information set and $\mathsf{G} = (\mathsf{Id}_k, \mathsf{Z})_{\mathcal{I}}$ the corresponding systematic generator matrix. Then $\mathcal{I}' = (\mathcal{I} \backslash \{\lambda\}) \cup \{\mu\}$ is an information set iff $\mathsf{Z}_{\lambda,\mu} = 1$*

Proof. Since $\mathsf{G} = (\mathsf{Id}_k, \mathsf{Z})_{\mathcal{I}}$, we have

$$\mathsf{G}_\mu = \mathsf{Z}_{\lambda,\mu} + \sum_{i \in \mathcal{I} \backslash \{\lambda\}} \mathsf{Z}_{i,\mu} \mathsf{G}_i.$$

Columns indexed by $\mathcal{I}$ are linearly independent, therefore G_μ and $(\mathsf{G}_i)_{i \in \mathcal{I} \backslash \{\lambda\}}$ are linearly independent iff $\mathsf{Z}_{\lambda,\mu} = 1$. ∎

Now we will describe the algorithms by Leon, Stern, and Canteaut and Chabaud.

4.2.1 Leon

In [32], J. S. Leon proposed a probabilistic algorithm for computing minimum weights of large linear error-correcting codes. This algorithm can also be adapted for computing codewords of minimum weight in a linear code.

In this paper, we will present a version of the algorithm which is slightly different from version presented by Leon in [32]. This version was presented by Chabaud in [10].

The input of the algorithm is a generator matrix G, the weight t, and two additional integers p and l which control the runtime and the success probability of the algorithm. The algorithm returns a codeword of weight t or fails. The algorithm executes the following steps.

Step 1: Randomly choose an information set $\mathcal{I}$ and apply a Gaussian elimination in order to obtain a systematic generator matrix $\mathsf{G}^* = (\mathsf{Id}_k, \mathsf{Z})_{\mathcal{I}}$.

Step 2: Randomly choose a set $\mathcal{L} \subseteq \mathcal{N} \backslash \mathcal{I}$ consisting of l elements.

Step 3: For each linear combination $\mathbf{A}$ of p or fewer rows of matrix $\mathsf{G}^*_{\mathcal{I}\cup\mathcal{L}}$ compute $\mathrm{wt}(\mathbf{A}_{\mathcal{I}\cup\mathcal{L}})$.

Step 4: If $\mathrm{wt}(\mathbf{A}_{\mathcal{I}\cup\mathcal{L}}) \leq p$ check whether the same linear combination applied to matrix G^* has weight t. If that is the case, then return the last linear combination. If there is no linear combination which fulfills the above condition, then the algorithm fails.

Next, we will analyze the algorithm. Thereby we assume that zeros and ones in the codewords are distributed almost uniformly.

At first, we will determine the success probability. It depends on favorable choices of $\mathcal{I}$ and $\mathcal{L}$. Assume we have a codeword $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$. Fix $p, l \in \mathbb{Z}$, then the following conditions lead to favorable choices of $\mathcal{I}$ and $\mathcal{L}$:

$$\mathcal{I} \subset \mathcal{N},\ |\mathcal{I}| = k,\ \mathcal{L} \in \mathcal{N}\backslash\mathcal{I},\ |\mathcal{L}| = l,\ \ \mathrm{wt}(\mathbf{e}_{\mathcal{I}\cup\mathcal{L}}) \leq p.$$

Therefore, Leon's algorithm succeeds with probability:

$$\Pr[\text{algorithm succeeds}] = \sum_{j=1}^{p} \frac{\binom{t}{j}\binom{n-t}{k+l-j}}{\binom{n}{k+l}}.$$

Next, we will estimate the expected work factor of the algorithm.

- The Gaussian elimination performed in step 1 requires on the average $\frac{k^2}{2}(n - \frac{k+1}{2})$ bit operations.
- Step 3 requires $\sum_{j=1}^{p} \binom{k}{j}(j-1)$ additions of l-bit words.
- Since in step 4, condition $\mathrm{wt}(\mathbf{A}_{\mathcal{I}\cup\mathcal{L}}) \leq p$ is true approximately $\sum_{j=1}^{p}\binom{k}{j}\frac{\sum_{i=0}^{p-j}\binom{l}{i}}{2^l}$ times. The algorithm requires $\sum_{j=1}^{p}\binom{k}{j}(j-1)\frac{\sum_{i=0}^{p-j}\binom{l}{i}}{2^l}$ additions of n-bit words.

Therefore, the expected work factor of Leon's attack against McEliece cryptosystem is

$$\binom{n}{k+l}\frac{\frac{k^2}{2}(n-\frac{k+1}{2}) + \sum_{j=1}^{p}\binom{k}{j}(j-1)(l+\frac{n}{2^l}\sum_{i=0}^{p-j}\binom{l}{i})}{\sum_{j=1}^{p}\binom{t}{j}\binom{n-t}{k+l-j}}. \tag{13}$$

To minimize the work factor, in [10] the parameters of Leon's attack are chosen to be $p = 3$ and $l \approx k + \log_2(n)$.

Algorithm 4.2.1 LEON-LWCW

Input: $k \times n$ generator matrix G, a positive integers t, p, and l.
Output: A codeword of weight t

$\mathcal{N} = \{0, \ldots, n-1\}$

while true do

/* Step 1 */
$\mathcal{I} = \emptyset$; $\mathcal{P} = \emptyset$
for $i = 1$ **to** k **do**
Randomly choose $r \in \mathcal{N} \backslash \mathcal{I}$; $\mathcal{I} = \mathcal{I} \cup \{r\}$
Randomly choose $c \in \{1, \ldots, k\} \backslash \mathcal{P}$ such that $\mathsf{G}_{r,c} = 1$; $\mathcal{P} = \mathcal{P} \cup \{c\}$
/* Eliminate all 1's in column c */
for $j = 1$ **to** k **do**
if $j \neq r$ **and** $\mathsf{G}_{j,c} = 1$ **then**
$\mathsf{G}_j = \mathsf{G}_j - \mathsf{G}_r$, where G_x is the x-th row of G

/* now we have $\mathsf{G} = (\mathsf{Id}_k, \mathsf{Z})_{\mathcal{I}}$ */

/* Step 2 */
Randomly choose $\mathcal{L} \subset \mathcal{N} \backslash \mathcal{I}$ such that $|\mathcal{L}| = l$

/* Steps 3 and 4 */
for all linear combinations $\mathbf{A}$ of p rows of $\mathsf{G}_{\mathcal{I} \cup \mathcal{L}}$ **do**
if $\mathrm{wt}(\mathbf{A}_{\mathcal{I} \cup \mathcal{L}}) \leq p$ **then**
Construct $\mathbf{c}$ from G by taking the same rows as in $\mathbf{A}$
if $\mathrm{wt}(\mathbf{c})$=t **then**
return($\mathbf{c}$)

4.2.2 Stern

In this section, we will present a slightly modified algorithm from [48]. We apply our algorithm to a generator matrix of a code instead of a parity check matrix as presented by Stern.

On input of a generator matrix G and three integers t, p and l the algorithm returns a codeword of length t or fails. The additional parameters p and l allow us to control the runtime and the success probability of the algorithm. Thus, knowing that there exist a codeword, we can repeat the algorithm until it succeeds.

The algorithm is based on the following idea. It randomly splits G into two sub-matrices which consist of rows of matrix G. In each matrix, the algorithm computes all linear combinations of p rows and checks whether certain parts of these linear combinations are equal. If they are equal, then the algorithm checks whether the weight of remaining parts is equal t. In this case the algorithm succeeds.

The algorithm performs the following five steps:

Step 1: Randomly choose an information set $\mathcal{I}$ and apply a Gaussian elimination in order to obtain a systematic generator matrix $\mathsf{G}^* = (\mathsf{Id}_k, \mathsf{Z})_{\mathcal{I}}$.

Step 2: Randomly spit $\mathcal{I}$ into two subsets $\mathcal{I}_1$ and $\mathcal{I}_2$. Each element of $\mathcal{I}$ is added either to $\mathcal{I}_1$ or to $\mathcal{I}_2$ with probability $1/2$. This causes a splitting of the rows of Z in $\mathsf{Z}_{\mathcal{I}_1\cdot}$ and $\mathsf{Z}_{\mathcal{I}_2\cdot}$.

Step 3: Randomly choose a set $\mathcal{L} \subseteq \mathcal{N} \backslash \mathcal{I}$ consisting of l elements.

Step 4: For each linear combination $\mathbf{A}$ (resp. $\mathbf{B}$) of p rows of matrix $\mathsf{Z}_{\mathcal{I}_1\cdot}$ (resp. $\mathsf{Z}_{\mathcal{I}_2\cdot}$) compute $\mathbf{A}_{\mathcal{L}}$ (resp. $\mathbf{B}_{\mathcal{L}}$).

Step 5: For each pair $(\mathbf{A}, \mathbf{B})$ with $\mathbf{A}_{\mathcal{L}} = \mathbf{B}_{\mathcal{L}}$ check whether $\mathrm{wt}(\mathbf{A}+\mathbf{B}) = t-2p$. If that is the case, then return vector $\mathbf{e}$ consisting of a linear combination of rows of G^*, where the same rows as in $\mathbf{A}+\mathbf{B}$ are taken. If there is no pair which fulfills the above conditions, then the algorithm fails.

We will analyze the algorithm. At first, we will determine the probability it succeeds. It depends on choices of $\mathcal{I}$, $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{L}$. Assume we have a codeword $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$. Fix $p, l \in \mathbb{Z}$, then we have the following conditions:

1. $|\mathcal{I}| = k$ and $\mathrm{wt}(\mathbf{e}_{\mathcal{I}}) = 2p$,
2. $\mathcal{I}_1 \subset \mathcal{I}$, $\mathrm{wt}(\mathbf{e}_{\mathcal{I}_1}) = p$, and $\mathcal{I}_2 = \mathcal{I} \backslash \mathcal{I}_1$,
3. $\mathcal{L} \in \mathcal{N} \backslash \mathcal{I}$, $|\mathcal{L}| = l$, $\mathrm{wt}(\mathbf{e}_{\mathcal{N} \backslash \mathcal{I}}) = t-2p$, and $\mathrm{wt}(\mathbf{e}_{\mathcal{L}}) = 0$.

These conditions implicate the probabilities of choosing such sets $\mathcal{I}$, $\mathcal{I}_1$, $\mathcal{I}_2$, and

$\mathcal{L}$ which yield to the given codeword $\mathbf{e}$.

$$\Pr[\text{of choosing a favorable } \mathcal{I}] = \frac{\binom{t}{2p}\binom{k-t}{k-2p}}{\binom{n}{k}}$$
$$\Pr[\text{of choosing a favorable } \mathcal{I}_1] = \frac{\binom{2p}{p}}{4^p}$$
$$\Pr[\text{of choosing a favorable } \mathcal{L}] = \frac{\binom{n-k-t+2p}{l}}{\binom{n-k}{l}}$$

The probability of success of Stern's algorithm is the product of the above probabilities. Thus, we have

$$\begin{aligned}\Pr[\text{the algorithm succeeds}] = & \Pr[\text{of choosing a favorable } \mathcal{I}]\cdot \\ & \Pr[\text{of choosing a favorable } \mathcal{I}_1]\cdot \\ & \Pr[\text{of choosing a favorable } \mathcal{L}].\end{aligned} \tag{14}$$

Next, we will estimate the expected work factor.

- The Gaussian elimination performed in step 1 requires on the average $\frac{k^2}{2}(n - \frac{k+1}{2})$ bit operations.
- Step 4 requires on the average $2lp\binom{k/2}{p}$ bit operations.
- In step 5 we assume that the distribution of values of $\mathbf{A}_{\mathcal{L}}$ (resp. $\mathbf{B}_{\mathcal{L}}$) is roughly uniform. Then, any bit vector of dimension l is hit by approximately $\binom{k/2}{p}/2^l$ elements of $\mathbf{A}$ (resp. $\mathbf{B}$). It follows, that step 5 requires approximately $2(n-k)\binom{k/2}{p}^2/2^l$ bit operations.

Thus, Stern's algorithm requires on average

$$2lpk^2(n-k)(n - \frac{k+1}{2})\binom{k/2}{p}^3/2^l \tag{15}$$

bit operations.

By combining the results of (14) and (15), we conclude that the expected work factor of Stern's attack against McEliece cryptosystem is

$$\frac{4^{p+1}lpk^2(n-k)(n-\frac{k+1}{2})\binom{k/2}{p}^3\binom{n-k}{l}\binom{n}{k}}{2^{l+1}\binom{2p}{p}\binom{t}{2p}\binom{k-t}{k-2p}\binom{n-k-t+2p}{l}} \tag{16}$$

4.2.3 Canteaut and Chabaud

As mentioned above, Stern's algorithm has to be repeated very often in order to decrypt successfully. Each repetition performs in the first step a Gaussian

Algorithm 4.2.2 STERN-LWCW

Input: $k \times n$ generator matrix G, a positive integers t, p, and l.
Output: A codeword of weight t

$\mathcal{N} = \{0, \ldots, n-1\}$

```
while true do

   /* Step 1 */
   I = ∅; P = ∅
   for i = 1 to k do
      Randomly choose r ∈ N\I; I = I ∪ {r}
      Randomly choose c ∈ {1,...,k}\P such that G_{r,c} = 1; P = P ∪ {c}
      /* Eliminate all 1's in column c */
      for j = 1 to k do
         if j ≠ r and G_{j,c} = 1 then
            G_j = G_j − G_r, where G_x is the x-th row of G

   /* now we have G = (Id_k, Z)_I */

   /* Step 2 */
   Randomly split I into I_1 and I_2

   /* Step 3 */
   Randomly choose L ⊂ N\I such that |L| = l

   /* Steps 4 and 5 */
   for all linear combinations A of p rows of Z_{I_1} do
      store (A_L, A, index of rows) in a hash table T
   for all linear combinations B of p rows of Z_{I_2} do
      if there exists (B_L, A, index of rows) ∈ T and
      wt((A + B)_{N\(I∪L)}) = t − 2p then
         Construct c from G by taking the same rows as in A + B
         return(c)
```

elimination which is very time consuming. In [8] the authors suggest another strategy for this step. Based on theorem 4.2.2, they suggest to choose a new information set not randomly but by modifying only one element in the old one. The complexity of this new step is approximately $k(n-k)/2$ binary operations instead of $k^2(n-\frac{k+1}{2})$ in Stern's algorithm.

The precise analyze of the algorithm CC-LWCW can be found in [8, 9]. Here we will present only the results. The algorithm is analyzed via modeling by a Markov chain. For this purpose we need a random variable X_i which represent the ith iteration of the algorithm and corresponds to the number of non-zero bits of cipher text $\mathbf{c}$ in $\mathcal{I}$. X_i takes one of the values of the set $\mathcal{E} = (\{1,\ldots,t\}\backslash\{2p\}) \cup \{(2p)_S,(2p)_F\}$ The set of success states is $\mathcal{S} = \{(2p)_S\}$. The set of failure states is $\mathcal{F} = \mathcal{E}\backslash\mathcal{S}$

Theorem 4.2.3 *The following results for the algorithm* CC-LWCW *are true:*

1. *The average number of elementary operations performed in each* ***while****-iteration is*

$$\Omega_{p,l} = 2pl\left(\binom{k/2}{p}\right)+2p(n-k-l)\frac{\left(\binom{k/2}{p}\right)^2}{2^l}+S\left(p\left(\binom{k/2}{p}\right)+2^l\right)+\frac{k(n-k)}{2}$$

where S is the size of a computer word ($=32$ or 64).

2. *Let $\pi_0(u) = Pr[X_0 = u]$, $P_{u,v} = Pr[X_i = v/X_{i-1} = u]$, $Q = (P_{u,v})_{u,v\in F}$, and $R = (I-Q)^{-1}$. Then the expectation of the number of* ***while****-iterations N is*

$$E(N) = \sum_{u\in F}\pi_0(u)\sum_{v\in F}R_{u,v}$$

3. *Suppose the number of codewords of weight t is A_t (Note, that $A_t = 1$ in our attack). Then the overall work factor of the algorithm is*

$$W_{p,l} = \frac{\Omega_{p,l}E(N)}{A_t} \tag{17}$$

The exact values of the entries of the matrix P and a more detailed analysis may be found e.g. in [9]. To get a approximate work factor, one can replace the $k^2\left(n-\frac{k+1}{2}\right)$-term in equation (16) by $k(n-k)/2$.

4.3 Statistical Decoding

This attack was presented by A Kh. Al Jabri in [25]. It is based on the idea that vectors from the dual space of a binary code which are not orthogonal to the ciphertext reveal some information on the error positions. This attack needs an algorithm which finds a sufficient number of vectors of the dual code of certain weight. It is not clear what the running time of such a search would be, since the problem of finding the desired set of vectors is connected to Problem

Algorithm 4.2.3 CC-LWCW

```
Input: k × n generator matrix G, a positive integers t, p, and l.
Output: A codeword of weight t

N = {0, ..., n − 1}

/* Step 1 */
I = ∅; P = ∅
for i = 1 to k do
  Randomly choose r ∈ N\I; I = I ∪ {r}
  Randomly choose c ∈ {1, ..., k}\P such that G_{r,c} = 1; P = P ∪ {c}
  /* Eliminate all 1's in column c */
  for j = 1 to k do
    if j ≠ r and G_{j,c} = 1 then
      G_j = G_j − G_r, where G_x is the x-th row of G

/* now we have G = (Id_k, Z)_I */

while true do
  /* Step 2 */
  Randomly split I into I_1 and I_2 with |I_1| = ⌊|I|/2⌋

  /* Step 3 */
  Randomly choose L ⊂ N\I such that |L| = l

  /* Steps 4 and 5 */
  for all linear combinations A of p rows of Z_{I_1} do
    store (A_L, A, index of rows) in a hash table T
  for all linear combinations B of p rows of Z_{I_2} do
    if there exists (B_L, A, index of rows) ∈ T and
    wt((A + B)_{N\(I∪L)}) = t − 2p then
      Construct c from G by taking the same rows as in A + B
      return(c)

  /* New step 1 */
  Randomly choose λ ∈ I
  Find unique r such that G_{r,λ} = 1
  Randomly choose μ ∈ N\I, such that Z_{r,μ} = 1
  I = (I\{λ}) ∪ {μ}
  /* Update Z appropriate to new I */
  for i = 1 to k do
    if r ≠ i and G_{i,μ} = 1 then
      G_i = G_i − G_r, where G_x is the x-th row of G
```

1.2.4 (SUBSPACE WEIGHTS). Further we know little about the true minimum distance of the dual code (see e.g. [12]).

Let $\mathcal{H}_w$ be a set of vectors of weight w of the dual space of the $(n, k, 2t+1)$ linear binary code $\mathcal{G}$ with generator Matrix G. Let $\mathbf{y}$ be the sum of a codeword $\mathbf{u}\mathsf{G} \in \mathcal{G}$ and an error vector $\mathbf{e}$ with weight at most t. A Jh. Al Jabri points out, that for randomly generated codes the probability that a value of 1 appears in the i-th position of $\mathbf{h} \in \mathcal{H}_w$ with $\mathbf{y}\mathbf{h}^T = 1$ depends on i being an erroneous position in the vector $\mathbf{y}$. Let p be the probability that $\mathbf{h}_i = 1$ and i is an erroneous position, and q be the probability that $\mathbf{h}_i = 1$ and i is a non-erroneous position. Then we have

$$p = \frac{\sum_{m \text{ odd}}^{m \leq t} \binom{n-t}{w-m}\binom{t-1}{m-1}}{\sum_{m \text{ odd}}^{m \leq t} \binom{t}{m}\binom{n-t}{w-m}}, \; q = \frac{\sum_{m \text{ odd}}^{m \leq t} \binom{n-t-1}{w-m-1}\binom{t}{m}}{\sum_{m \text{ odd}}^{m \leq t} \binom{t}{m}\binom{n-t}{w-m}}$$

for all $\mathbf{h}$ satisfying that $\mathbf{y}\mathbf{h}^T = 1$.

The idea for statistical decoding is quite similar to the one of iterative decoding, see [14]. It consists in estimating the probability that $\mathbf{h}_i = 1$ and $\mathbf{y}\mathbf{h}^T = 1$ for each position i considering different vectors $\mathbf{h}$. Unlike at iterative decoding we do not determine a single error position, but try to determine an information set of non-error positions. If for example $p > q$, then we assume that i is a non-error position if the relative frequency estimate is lower then a certain bound. Once we have found a non-erroneous information set by modifying the bound, we try to correct the errors.

We can recover $\mathbf{u}$ using algorithm 4.3.1 if $\mathcal{H}_w$ is properly chosen. Note that for $i \in \{1, \cdots, n\}$ an (non-)error position the value $\mathbf{v}_i / v_{\mathbf{y}}^+$ with $v_{\mathbf{y}}^+ := \sum_{\mathbf{h} \in \mathcal{H}_w} \left(\mathbf{y}\mathbf{h}^T \mod 2\right)$ is the relative frequency estimate for p (q respectively). The mean value of $\mathbf{v}_i$ is $p v_{\mathbf{y}}^+$, and its variance is $\sigma^2 = p(p-1)v_{\mathbf{y}}^+$. The sets I_1 and I_2 are introduced to cover the cases where $p < q$ or $p > q$.

Algorithm 4.3.1 StatDec

Input: $\mathcal{H}_w$, $\mathbf{y}$.
Output: $\mathbf{u}$, the information vector.

$\mathbf{v} = \sum_{\mathbf{h} \in \mathcal{H}_w} \left(\mathbf{y}\mathbf{h}^\top \mod 2\right) \mathbf{h} \in \mathbb{Z}^n$.
choose $I_1 = \{$positions of the k largest entries of $\mathbf{v}\}$ s.t. $\mathsf{G}_{\cdot I_1}$ is invertible.
choose $I_2 = \{$positions of the k smalles entries of $\mathbf{v}\}$ s.t. $\mathsf{G}_{\cdot I_2}$ is invertible.
$\mathbf{u}_1 = \mathbf{y}_{I_1} \mathsf{G}_{\cdot I_1}^{-1}$
$\mathbf{u}_2 = \mathbf{y}_{I_2} \mathsf{G}_{\cdot I_2}^{-1}$
if weight$(\mathbf{u}_1 \mathsf{G} \oplus \mathbf{y}) \leq t$ **then**
  $\mathbf{u} = \mathbf{u}_1$
else
  $\mathbf{u} = \mathbf{u}_2$

The work factor for algorithm 4.3.1 is

$$\mathcal{O}\left(n \cdot |\mathcal{H}_w| + 2k^3 + kn\right)$$

McEliece parameters $(2^m, k, d = 2t+1)$	w	$\lvert p-q \rvert$	$\lvert\mathcal{H}_w\rvert$	$\binom{n}{w}2^{-k}$	Workfactor StatDec
$(1024, 524, 101)$	137	$0.2 \cdot 10^{-7}$	2^{51}	$2^{52.5}$	2^{61}
$(1024, 524, 101)$	153	$0.21 \cdot 10^{-8}$	2^{58}	2^{94}	2^{68}
$(2048, 1278, 141)$	363	$0.41 \cdot 10^{-14}$	2^{96}	$2^{96.9}$	2^{107}
$(65536, 65392, 9)$	32000	$0.17 \cdot 10^{-13}$	2^{93}	$2^{109.7}$	2^{109}

Table 1: StatDec for example parameter sets

binary operations having computed the set$\mathcal{H}_w$ in advance. The author of [25] claims that the latter can be done e.g. by the methods of [8], which is to be doubted (compare [39] and [14]).

The difference between p and q is very small for large codes, so we need a large set $\mathcal{H}_w$ to distinguish the relative frequency estimates for p and q respectively. Al Jabri's initial analysis of the size of $\mathcal{H}_w$ needed for error correction seems to be too optimistic. A more realistic bound seems to be

$$|\mathcal{H}_w| \approx 5.4p(1-p)\frac{1}{(p-q)^2}. \tag{18}$$

from [39], which is about a factor 2^{14} larger than Al Jabri's original bound (compare as well [14]).

It is obvious, that a set $\mathcal{H}_w$ of the desired size will not even exist if w is chosen to small. Goppa codes, BCH codes and GRS codes have a weight distribution "close" to the expected weight distribution of a random codes, which is the binomial distribution [25]. Consequently, we get the following condition for $\mathcal{H}_w$:

$$|\mathcal{H}_w| \leq \binom{n}{w}2^{-k}.$$

Table 1 shows some example sizes to attack McEliece this way, where the work factor refers to the computational costs after having computed the set $\mathcal{H}_w$. One can see, that the McEliece cryptosystem resists this kind of attack for all parameter sets secure against CC-LWCW. Further, for all parameter sets proposed, StatDec has no advantage over CC-LWCW. However, so far there is no algorithm known, which performs the precomputation efficiently.

In [39], a improved version of StatDec is proposed, but the author concludes, that this improvement is not sufficient to attack the McEliece Cryptosystem by statistical decoding due to the large amount of precomputation needed. The authors of [14] conclude, that for iterative decoding a smaller set $\mathcal{H}_w$ as for the initial StatDec is sufficient as well. However, the size of $\mathcal{H}_w$ needed is still very large and in consequence it is infeasible to compute $\mathcal{H}_w$ by the existing methods.

4.4 Lattice Attacks

In [6], the authors suggest to apply the low density algorithm from [29] to break Niederreiter cryptosystem. In this section we give an idea of this attack and explain why this attack doesn't work with Niederreiter/McEliece cryptosystems based on binary Goppa codes.

The attack proceeds as follows. Given a parity check matrix $\mathsf{H} \in \mathbb{F}_q^{n\times(n-k)}$ of a Goppa code and cipher text $\mathbf{c} = \mathbf{m}\mathsf{H}$, where $\mathbf{m}$ is a message, i.e. $\mathrm{wt}(\mathbf{m}) = t$ (see section 1.4). Let L be the lattice generated by the row vectors in the matrix

$$\mathsf{Q} = \left(\begin{array}{c|c} \mathsf{Id}_{n+1} & \begin{array}{c} r\mathsf{H} \\ \hline r\mathbf{c}^T \end{array} \\ \hline 0 & qr\mathsf{Id}_{n-k} \end{array}\right)$$

where Id_s is the identity matrix of dimension s and r is an integer. The vector $\mathbf{m}^* = (\mathbf{m}_1, \ldots, \mathbf{m}_n, -1, 0, \ldots, 0)$ is a vector in the lattice and has at most $t+1$ nonzero entries. If $r \geq t$, then the authors claim that $\mathbf{m}^*$ is a shortest vector in the lattice. So by finding this vector we can determine the corresponding plain text.

Unfortunately, this is not true for fields of characteristic 2. The reason for this failure is that $\mathbf{m}^*$ isn't the shortest vector for $q = 2$. The shortest vectors are $2\mathbf{e}_1, \ldots, 2\mathbf{e}_{n+1}$, where $\mathbf{e}_i = (\underbrace{0, \ldots, 0}_{i-1}, 1, 0, \ldots, 0)$. These vectors can be obtained by taking the first (resp. second, etc.) row twice and erase the last $(n-k)$ elements in the vector by taking appreciate rows from the sub-matrix $qr\mathsf{Id}_{n-k}$. Since these vectors have nothing to do with original message $\mathbf{m}$, this attack doesn't work with the Niederreiter cryptosystem based on binary Goppa codes.

5 Attacks infeasible with CCA2 Conversions

The attacks outlined in the following aim at revealing partial information about the message sent, or the error vector used for encryption in the McEliece case. Thus they are not stand alone attacks, i.e. they cannot be used to recover the plaintext completely or to get the private keys, but they provide ways to reduce the system size and thus the complexity of consecutive attacks.

One thing all attacks dealt with in this section have in common is that they can be avoided completely by suitable conversions for the original McEliece cryptosystem [27]. Thus the attacks are mentioned here mostly for completeness' sake and to underline the importance for using one of the proposed conversions, some of which we present later.

5.1 Taking advantage of partially known plaintexts

An attacker for the McEliece cryptosystem may use known bits of a sent message to recover the whole plaintext. More precisely, the partial knowledge of the originally sent message corresponds to a reduction in the cryptosystems parameters.

Suppose an adversary knows the target plaintext bits $\mathbf{m}_{\mathcal{I}}$ for an index set $\mathcal{I} \subset \{1, 2, \ldots, k\}$. Denote with $\mathcal{J}$ the complement of $\mathcal{I}$ in $\{1, 2, \ldots, k\}$. Then the adversary may try to recover $\mathbf{m}_{\mathcal{J}}$ using the following reduction:

$$\mathbf{m}\mathsf{G} = \mathbf{m}_{\mathcal{I}}\mathsf{G}_{\cdot\mathcal{I}} \oplus \mathbf{m}_{\mathcal{J}}\mathsf{G}_{\cdot\mathcal{J}}.$$

Therefore, we have

$$\begin{aligned} \mathbf{c} \oplus \mathbf{m}_{\mathcal{I}}\mathsf{G}_{\cdot\mathcal{I}} &= \mathbf{m}_{\mathcal{J}}\mathsf{G}_{\cdot\mathcal{J}} \oplus \mathbf{z} \\ \mathbf{c}' &= \mathbf{m}_{\mathcal{J}}\mathsf{G}_{\cdot\mathcal{J}} \oplus \mathbf{z}. \end{aligned}$$

An analogous reduction can be achieved for the Niederreiter scheme. All attacks described in the previous section, that do snot use the particular structure of the code can be applied to try and solve this equation for $\mathbf{m}_{\mathcal{J}}$. In particular, this includes the Generalized Information-Set-Decoding attack and the Finding-Low-Weight-Codeword attack. (Note that their success is no longer guaranteed as we do not know wether $\mathsf{G}_{\cdot\mathcal{J}}$ contains an Information Set, which is needed in both cases.) However, the computational cost for those attacks can be critically reduced as k drops to $|\mathcal{J}|$.

5.2 Taking advantage of known relations between messages

An adversary for the McEliece scheme may use the relation between two encrypted messages to determine error bits [5]. This attack cannot be adapted to the Niederreiter cryptosystem. Let $\mathbf{m}_1, \mathbf{m}_2$ be two messages related by Λ, e.g. $\Lambda(\mathbf{m}_1, \mathbf{m}_2) = \mathbf{m}_1 \oplus \mathbf{m}_2$. Then

$$\mathbf{c}_1 \oplus \mathbf{c}_2 \oplus \Lambda(\mathbf{m}_1, \mathbf{m}_2) = \mathbf{z}_1 \oplus \mathbf{z}_2.$$

Zero bits on the left hand side of this equation imply

$$\mathbf{z}_1|_k \oplus \mathbf{z}_2|_k = 0 \Rightarrow \begin{cases} 1 = \mathbf{z}_1|_k = \mathbf{z}_2|_k \\ 0 = \mathbf{z}_1|_k = \mathbf{z}_2|_k. \end{cases}$$

Since the weight of the error vectors $\mathbf{z}_1, \mathbf{z}_2$ is small, the first case is highly unlikely:

$$Pr(1 = \mathbf{z}_1|_k = \mathbf{z}_2|_k) = \left(\frac{t}{n}\right)^2.$$

This enables an adversary to efficiently guess error bits.

A special case is the **message-resend attack** where the attacker can recover $\mathbf{z}_1 \oplus \mathbf{z}_2 = \mathbf{c}_1 \oplus \mathbf{c}_2$.

5.3 Reaction Attack

This attack is a weaker version of an adaptively chosen cipher text attack, in that it does not require any decryptions, but only depends on the observation of the receiver's reaction on potential ciphertexts [27]. This attack again aims at determining error bits and is therefore only applicable to the McEliece cryptosystem.

An adversary may intercept ciphertexts, change a few bits, and watch the reaction of the designated receiver on these modified ciphertexts. Sending modifications of an authentic ciphertext amounts to adding further error bits. If the receiver cannot decode (reaction: repeat request), the corresponding bits were not in error originally. This may enable a Generalized Information-Set-Decoding attack, for example.

The probability to need more than k rounds before hitting an error position is

$$\frac{\binom{n-k}{t}}{\binom{n}{t}}.$$

5.4 Malleability

Adding codewords, i.e. rows of G to a ciphertext yields another valid ciphertext. Therefore, the original McEliece cryptosystem does not satisfy non-malleability. Note that this is no problem in the Niederreiter case, as there is no known relation that may be used to create new decodable syndromes from old ones.

6 Conversions achieving CCA2-Security

Suppose an adversary who wants to recover a message from its ciphertext only, has access to a decryption oracle. He may not query the oracle on the target ciphertext. Apart from that, the oracle provides him with ciphertext-plaintext pairs of his choice. A cryptoystem is *secure against adaptive chosen ciphertext attacks* (CCA2 secure) if such attacker has no advantage in deciphering a given ciphertext. It is *indistinguishable in the CCA2-model* if the attacker has no advantage in determining for a given ciphertext and two plaintexts which of them was encrypted.

In [27] Kobara and Imai review two generic conversion. One was originally presented by Pointcheval [41] and the other by Fujisaki and Okamoto [15]. Both conversions were designed to achieve CCA2 security for a restricted class of public key cryptosystems. Kobara and Imai show, that these conversions can successfully be applied to the McEliece cryptosystem.
Furthermore they and propose three conversion schemes specifically tailored for the McEliece cryptosystem. To explain these conversions, we introduce the following notation:

r, r'	Random numbers
Conv	Bijective conversion of any number in $\mathbb{Z}/\mathbb{Z}\binom{n}{t}$ to the corresponding error vector of length n
H	Cryptographic hash function, outputting bit-strings of length $\log_2 \binom{n}{t}$
R	Cryptographically secure pseudo random number generator from fixed length seeds
$\mathcal{E}$	McEliece encryption function, taking as first argument the message to be encrypted and as second one the error vector: $\mathcal{E}(\mathbf{m}, \mathbf{z}) = \mathbf{c}$
$\mathcal{D}$	McEliece decryption function: $\mathcal{D}(c) = (\mathbf{m}, \mathbf{z})$
$MSB_n(\mathbf{m})$	The n rightmost bits of $\mathbf{m}$.
$LSB_n(\mathbf{m})$	The n leftmost bits of $\mathbf{m}$.

6.1 Pointcheval's Generic Conversion

A function $f : X \times Y \to Z$, $(x, y) \mapsto z$ is *partially trapdoor one-way* (PTOWF) if it is impossible to recover x or y from their image z alone, but the knowledge of secret enables a partial inversion, i.e. finding x from z. Pointcheval [41] demonstrated how any PTOWF can be converted to a public-key cryptosystem that is indistinguishable against CCA2.

The McEliece cryptosystem draws is security from the assumption that its primitive is PTOWF: The function $(\mathbf{m}, \mathbf{z}) \mapsto \mathcal{E}(\mathbf{m}, \mathbf{z})$ can be inverted to recover m iff the private key, i.e. the generator matrix of the underlying Goppa code, is known.

Algorithm 6.1.1 Pointcheval's generic conversion – Encryption

Input: Random r, r' and the (possibly padded) message m.
Output: A McEliece-based cipher c.

$\mathbf{z} = H(\mathbf{m}||r)$
$\mathbf{z} = \mathrm{Conv}(\mathbf{z})$
$\mathbf{c}_1 = \mathcal{E}(r', \mathbf{z})$
$\mathbf{c}_2 = R(r') \oplus (\mathbf{m}||r)$
$\mathbf{c} = (\mathbf{c}_1||\mathbf{c}_2)$

6.2 Fujisaki-Okamoto's Generic Conversion

Fujisaki and Okamoto propose hybrid encryption that merges a symmetric encryption scheme which is secure in the *Find-Guess* model, with an asymmetric *One-Way-Encryption* scheme which is sufficiently probabilistic, to obtain a public-key cryptosystem which is indistinguishable against CCA2. See [15] for more details. The adaptation of Kobara and Imai to the McEliece primitive uses one-time padding with random numbers for the symmetric part, and McEliece

Algorithm 6.1.2 Pointcheval's generic conversion – Decryption

Input: A cipher c and the corresponding McEliece decryption function $\mathcal{D}$
Output: The target plaintext m

$\mathbf{c}_1 = MSB_n(\mathbf{c})$
$\mathbf{c}_2 = LSB_{\mathrm{Len}(\mathbf{m})+\mathrm{Len}(r)}(\mathbf{c})$
$(r', \mathbf{z}) = \mathcal{D}(\mathbf{c}_1)$
$(\mathbf{m}||r) = \mathbf{c}_2 \oplus R(r')$
if $\mathbf{c}_1 = \mathcal{E}(r', \mathrm{Conv}(H(\mathbf{m}||r)))$ **then**
  return m
else
  reject c

encryption for the asymmetric one.

Algorithm 6.2.1 Fujisaki-Okamoto's generic conversion – Encryption

Input: Random r, and the (possibly padded) message $\mathbf{m}$.
Output: A McEliece-based cipher $\mathbf{c}$.

$\mathbf{z} = H(r||\mathbf{m})$
$\mathbf{z} = \mathrm{Conv}(\mathbf{z})$
$\mathbf{c}_1 = \mathcal{E}(r, \mathbf{z})$
$\mathbf{c}_2 = R(r) \oplus \mathbf{m}$
$\mathbf{c} = (\mathbf{c}_1||\mathbf{c}_2)$

6.3 Kobara-Imai's Specific Conversions

Kobara and Imai also present three conversions of their own. Their main concern is to decrease data overhead introduced by the previously mentioned schemes. One of the corresponding conversions is given below.

Algorithm 6.2.2 Fujisaki-Okamoto's generic conversion – Decryption

Input: A cipher $\mathbf{c}$, and the corresponding McEliece decryption function $\mathcal{D}$
Output: The target plaintext $\mathbf{m}$

$\mathbf{c}_1 = MSB_n(\mathbf{c})$
$\mathbf{c}_2 = LSB_{\mathrm{Len}(\mathbf{m})}(\mathbf{c})$
$(r, \mathbf{z}) = \mathcal{D}(\mathbf{c}_1)$
$\mathbf{m} = \mathbf{c}_2 \oplus R(r)$
if $\mathbf{c}_1 = \mathcal{E}(r, \mathrm{Conv}(H(r||\mathbf{m})))$ **then**
  return m
else
  reject c

Algorithm 6.3.1 Kobara-Imai's Specific Conversion γ – Encryption

Input: Random r, a predetermined public constant *const* and the (possibly padded) message $\mathbf{m}$.
Output: A McEliece-based cipher $\mathbf{c}$.
Note: It is assumed that the message $\mathbf{m}$ is prepared so that $\mathrm{Len}(\mathbf{m}) \geq \log_2 \lfloor \binom{n}{t} \rfloor + k - \mathrm{Len}(const) - \mathrm{Len}(r)$ where n, k and t are the parameters used for McEliece encryption.

$\mathbf{c}_1 = R(r) \oplus (\mathbf{m}||const)$
$\mathbf{c}_2 = r \oplus H(\mathbf{c}_1)$
$\mathbf{c}_3 = LSB_{\lfloor \log_2 \binom{n}{t} \rfloor + k}(\mathbf{c}_2||\mathbf{c}_1)$
$\mathbf{c}_4 = LSB_k(\mathbf{c}_3)$
$\mathbf{c}_5 = MSB_{\lfloor \binom{n}{t} \rfloor}(\mathbf{c}_3)$
$\mathbf{z} = \mathrm{Conv}(\mathbf{c}_5)$
if $\mathrm{Len}(\mathbf{c}_2||\mathbf{c}_1) - \lfloor \log_2 \binom{n}{t} \rfloor - k > 0$ **then**
  $\mathbf{c}_6 = MSB_{\mathrm{Len}(\mathbf{c}_2||\mathbf{c}_1) - \lfloor \log_2 \binom{n}{t} \rfloor - k}(\mathbf{c}_2||\mathbf{c}_1)$
  $\mathbf{c} = (\mathbf{c}_6||\mathcal{E}(\mathbf{c}_4, \mathbf{z}))$
else
  $\mathbf{c} = \mathcal{E}(\mathbf{c}_4, \mathbf{z})$

Algorithm 6.3.2 Kobara-Imai's Specific Conversion γ – Decryption

Input: A cipher $\mathbf{c}$, the bit length of the random number used in encryption $Len(r)$ and the corresponding McEliece decryption function $\mathcal{D}$
Output: The target plaintext $\mathbf{m}$

$\mathbf{c}_6 = MSB_{Len(\mathbf{c})-n}(\mathbf{c})$
(Again, $\mathbf{c}_6$ may be empty)
$(\mathbf{c}_4, \mathbf{z}) = \mathcal{D}(LSB_n(\mathbf{c}))$
$\mathbf{c}_5 = \text{Conv}^{-1}(\mathbf{z})$
$\mathbf{c}_2 = MSB_{\text{Len}(r)}(\mathbf{c}_6||\mathbf{c}_5||\mathbf{c}_4)$
$\mathbf{c}_1 = LSB_{\text{Len}(\mathbf{c})-\text{Len}(r)}(\mathbf{c}_6||\mathbf{c}_5||\mathbf{c}_4)$
$r' = \mathbf{c}_2 \oplus H(\mathbf{c}_1)$
$(\mathbf{m}||const') = (\mathbf{c}_1) \oplus R(r')$
if $const' = const$ **then**
  return m
else
  reject c

Conversions and Data Redundancy[a]

Conversion	Dataredundancy = Ciphertext size - Plaintext size			
	(n,k) t	(1024, 524) 50	(2048,1608) 40	(2048, 1278) 70
Pointch.	$\text{Len}(r) + n$	1184	2308	2308
Fujisaki Okamoto	n	1024	2048	2048
Kobara Imai	$n + \text{Len}(const\|\|r)$ $- \log_2\lfloor\binom{n}{t}\rfloor - k$	536	480	655
Original McEliece	$n - k$	500	440	770

[a]We follow the suggestion of Kobara and Imai and use $\text{Len}(r) = \text{Len}(Const) = 160$.

Kobara and Imai claim to achieve a reduction in data redundancy even below the values for the original McEliece PKCS for large parameters. We point out that this is only true if the message is prepared in such a way that

$$\text{Len}(\mathbf{m}) \geq \log_2\lfloor\binom{n}{t}\rfloor + k - \text{Len}(r) - \text{Len}(const).$$

Nonetheless, the cut in data overhead is remarkable. Their main result concerning security is the following:

Theorem 6.3.1 *Breaking indistinguishability in the CCA2 model using any of* *the conversions presented above, is as hard as breaking the original McEliece public key system.*

Furthermore, the Known-Partial-Plaintext Attack, the Related Message Attack, the Reaction Attack and the Malleability Attack, all become impossible, since relations among plaintexts do no longer result in relations among ciphertexts. Already the simple hashing of messages before encryption prevents this.

7 Other cryptographic applications

In this section we want to look into digital signature and identification schemes using error correcting codes. Up to now there has been little research concerning the development of secure and efficient digital signatures based on the McEliece Cryptosystem. In fact McEliece claimed in his original paper "the decryption algorithm [...] cannot be used to produce unforgeable 'signatures'."[37]

The first ideas to derive digital signatures from error-correcting codes have been presented by Xinmei in [52]. Xinmei's suggestion uses a McEliece-type encryption but was attacked and modified by Harn and Wang [22] and finally broken by Alabbadi and Wicker in 1992 [1].

One year later, J. Stern proposed an identification scheme based on syndrome decoding [49] but acknowledged himself that it could not be modified to an efficient signature scheme.

Alabbadi and Wicker reviewed the chances to design digital signature schemes based on error-correcting codes in [2] but did not find feasible models. Their own proposal was successfully attacked by Stern [50].

Thus all attempts to create secure and reasonably efficient digital signatures on the basis of the McEliece cryptosystem have failed until the paper of Courtois, Finiasz and Sendrier [11].

7.1 Stern's identification scheme

Stern's identification scheme is based on the Niederreiter cryptosystem.

Let H be a $(n-k) \times n$ matrix common to all users. Chosen randomly, Stern claims that H generally will provide a parity check matrix for a code with good error correcting capability. Every user receives an n bit private key s of prescribed weight p.

- **Public key** H, $Hs^t = i$, p
- **Private key** s

The security of the scheme relies on the difficulty of the syndrome decoding problem, that is on the difficulty of determining the preimage s of $i = Hs^t$. Without the secret key, an adversary has two altenatives to deceive the verifier:

Protocol 7.1.1 Stern's Identification scheme

Prover	**Verifier**
Choose random n-bit vector y and random permutation σ, to compute $c_1 = (\sigma, Hy^t)$ $c_2 = \sigma(y)$ $c_3 = \sigma(y \oplus s)$ Send commitments for (c_1, c_2, c_3)	
	Send random request $b \in \{0, 1, 2\}$
If $b = 0 \Rightarrow$ reveal y, σ If $b = 1 \Rightarrow$ reveal $y \oplus s, \sigma$ If $b = 2 \Rightarrow$ reveal $\sigma(y), \sigma(s)$	
	If $b = 0 \Rightarrow$ check c_1, c_2 If $b = 1 \Rightarrow$ check c_1, c_3 and $Hy^t = H(y^t \oplus s^t) \oplus i$ If $b = 2 \Rightarrow$ check c_2, c_3 and $\omega(\sigma(s) = p$

1. He can work with a random s' of weight p instead of the secret key. He will succeed if he is asked $b \in \{0, 2\}$ but in case $b = 1$ he will hardly be able to produce the correct c_1, c_3 since $Hs' \neq Hs = i$.

2. He can choose s' from the set of all preimages of i under H, i.e. $s \in H^{-1}(i^t)$. This time he will fail to answer the request $b = 2$ since $\omega(s') \neq p$.

Thus the attacker has chances $2/3$ to deceive the verifier in any round. The identification scheme of Stern has not been broken. Unfortunately, it can not be adapted to obtain an efficient signature scheme. The standard method to convert the identification procedure into a prodedure for signing, is to replace verifier-queries by values suitably derived from the message to be signed. This leads to a blow-up of each (hashed) plaintext bit to $2n$ signature bits and is therefore hardly applicable here.

7.2 CFS Signature Scheme

The only working signature scheme based on the McEliece, or rather on the Niederreiter encryption was presented by Courtois, Finiasz and Sendrier in [11]. Analogously to the results on the original McEliece PKCS, the security of the CFS scheme can be reduced to the Bounded Distance Decoding Problem. The Bounded Distance Decoding Problem (BD) is the Syndrome Decoding Problem for codes with known minimal distance. This extra knowledge allows the decoder to restrict his search to codewords within the given distance to the received one. Some believe this problem not to be NP-complete, as determining the minimum distance of a linear code in itself already is NP-complete, and this additional information is given in the BD case.

Let the underlying code be a (n, k)-Goppa code, with error-correcting capability t, where $n = 2^m$ and $k = n - tm$, for some integer m. Denote with G the generator matrix and with H the parity check matrix, respectively.

The idea of the CFS algorithm is to repeatedly hash the document augmented by a counter, until the ouptput is a decodable syndrome. The signer uses his secret key to determine the corresponding error-vector. Together with the current value of the counter, this error vector will then serve as signature.

The error-vector length n can be reduced considerably, taking into account that only t of its bits are nonzero. With the parameters suggested by Courtois, Finiasz and Sendrier the number of possible error-vectors is approximately given by $\binom{n}{t} = \binom{2^{16}}{9} \approx 2^{125.5}$ so that a 126-bit counter suffices to address each of them. We need the following ingredients:

h	Public hash function
I	Functions that assigns each word of weight t and length n a unique index in the set of all these words.
$\mathcal{T}$	McEliece trapdoor function, outputting the error-vector for a given decodable syndrome
H	The public parity check matrix.

Algorithm 7.2.1 CFS digital signature – Signing

Input: $h, I, \mathcal{T}, r$ and the document to be signed d
Output: A CFS-signature s.

$z = h(d)$
choose a r-bit Vector i at random
$s = h(z||i)$
while s is not decodable **do**
  choose a r-bit Vector i at random
  $s = h(z||i)$
$e = \mathcal{T}(s)$
$s = (I(e)||i)$

The average number of attempts needed to reach a decodable syndrome can be estimated by comparing the total number of syndromes $\mathcal{N}_{\text{tot}}$ to the number of correctable syndromes $\mathcal{N}_{\text{dec}}$.

$$
\begin{aligned}
\mathcal{N}_{\text{tot}} &= 2^{n-k} = 2^{mt} = n^t \\
\mathcal{N}_{\text{dec}} &= \sum_{i=0}^{t} \binom{n}{t} \approx \frac{n^t}{t!} \\
\frac{\mathcal{N}_{\text{dec}}}{\mathcal{N}_{\text{tot}}} &= \frac{1}{t!}
\end{aligned}
$$

Thus each syndrome has a probabillity of $\frac{1}{t!}$ to be decodable. The CFS scheme

Algorithm 7.2.2 CFS signature scheme – Verification

Input: A signature $s = (I(e)||i)$, the document d and the McEliece public key H
Output: Is the signature valid?

$e = I^{-1}(I(e))$
$s_1 = H(e^t)$
$s_2 = h(h(d)|i)$
if $s_1 = s_2$ **then**
  accept s
else
  reject s

Parameter Sizes and Costs

parameters	n	2^{15}	2^{16}		2^{17}		
	t	10	9	10	8	9	10
size public key in MB	$k(n-k)/(8 \cdot 1024^2)$	0.58	1.12	1.12	2.38	2.38	2.38
signature cost	$t!t^2m^3$	2^{40}	2^{37}	2^{40}	2^{34}	2^{38}	2^{41}
verification cost	t column operations^2	2^{18}	2^{19}	2^{19}	2^{20}	2^{20}	2^{20}
signature length	$\log_2(n^t)$	150	144	160	136	153	170

needs about $t!$ iterations, producing signatures of length $\log_2(r\binom{n}{t}) \approx \log_2(n^t)$. Thus, r has to be be larger than $log_2(t!)$.

Attacking the CFS signature scheme via the birthday paradoxon is the best method so far, which is infeasible (compare [11]).

8 Performance and Parameters

The main reason why McEliece received little attention in practice is because of the huge key sizes in comparison to RSA. Like RSA, its security remains unbroken in its original form. It is as old as RSA, but less well studied. In the following we review some aspects of implementation, performance and (good) choice of parameters.

As we have already mentioned, the key sizes are quite big in comparison to RSA. However, the McEliece Cryptosystem has a much faster en- and decryption. We to take a look at the running times first and analyze the key sizes afterwards.

8.1 Performance of En-/Decryption and Key Generation

The encryption of a messgage in the original McEliece scheme takes about

$$k/2 \cdot n + t$$

binary operations plus the time to generate the error vector. For decryption, the decryption algorithm gets faster if we store some matrices in advance, which only depend on the private key. We return to the notations of section 1.3 and 1.4 respectively.

Theorem 8.1.1 *The decryption of a ciphertext of a McEliece instance generated by a $(n = 2^m, k, d)$ binary irreducible Goppa code requires $\mathcal{O}\left(ntm^2\right)$ binary operations.*

Proof. Let $J \subseteq \{1, \cdots, n\}$ with $|J| = k$ and G_J invertible. We may compute $\mathbf{m}\mathsf{SG} \oplus \mathbf{z}\mathsf{P}^{-1}$ in $n \cdot m$ binary Operations and the corresponding syndrome in $n \cdot (n-k)$ more. Applying the algorithm of Patterson ([40], algorithm 2.3.1) we need $\mathcal{O}\left(n \cdot t \cdot m^2\right)$ binary operations to identify the vector $\mathbf{z}\mathsf{P}^{-1}$ and n more to get $\mathbf{m}\mathsf{SG}$. Having computed $\left(\mathsf{SG}_J\right)^{-1}$ we need only further k^2 binary operations to recover the message m. ∎

The time needed to encrypt a message with Niederreiter depends on the method of representing the message by a appropriate plaintext $\mathbf{e}$ of length n and weight t. This could be done in several ways. We just want to point out, that the distribution of the support of $\mathbf{e}$ should be (almost) uniform to avoid correct guessing of the positions of the zeros (compare [42]). For example one could use methods derived from [51] or simple enumeration of all possible error vectors. The time of decryption depends on the time to recover the plaintext and the time to reconstruct the original message from that plaintext.

Theorem 8.1.2 *Recovering the plaintext from a ciphertext of a Niederreiter instance generated by a (n, k, d) Goppa code requires $\mathcal{O}\left(ntm^2\right)$ binary operations.*

Proof. The proof is analogeous to the one of the theorem above. ∎

When generating an instance of the McEliece Cryptosystem with $n = 2^m$ we suppose that we already know a polynomial $F \in \mathbb{F}_2[X]$ s.t. $\left(\mathbb{F}_2[X]\right)/F = \mathbb{F}_{2^m}$. From [19] we know that the number of monic irreducible polynomials of degree t over $\mathbb{F}_{2^m}$ is bigger than $\left(2^{mt} - 1\right)/t$. Thus the probability of getting an irreducible polynomial by choosing a random one of degree t with leading coefficient $\neq 0$ is larger than $1/t$. To check the irreducibility requires $\mathcal{O}\left(t^2m^2 + t^3m\right)$ operations [24]. Having found an irreducible generator polynomial g we need 2^m evaluations of $\left(g(x)\right)^{-1}$ and $n(t-1)$ multiplications in $\mathbb{F}_{2^m}$ to generate the parity check matrix. For the McEliece cryptosystem we need a Gaussian elimination ($\mathcal{O}((n-k)^3)$ binary operations) at that point, to compute the generator matrix.

Next we have to generate the permutation and the scramble matrix and multiply them with the generator matrix which can be done in $\mathcal{O}\left(k^2 n + n^2\right)$ (McEliece) and $\mathcal{O}\left((n-k)^2 n + n^2\right)$ (Niederreiter) binary operations respectively. Together with the time necessary to invert SG_J and M, this leads to the following theorem:

Theorem 8.1.3 *The running time (in binary operations) to generate a key pair for the McEliece cryptsystem is* $\mathcal{O}\left(k^2 n + n^2 + t^3(n-k) + (n-k)^3\right)$ *and* $\mathcal{O}\left((n-k)^2 n + n^2 + t^3(n-k)\right)$ *for the Niederreiter cryptosystem.*

8.2 Key Sizes

The method of storing the private key offers some variants. First we would want to store the Goppa polynomial and the generator polynomial of $\mathbb{F}_{2^m}$ and additionally the check matrix H. Second it would be better to store M^{-1} or $\left(\mathsf{SG}_{\cdot J}\right)^{-1}$, to enhance the performance of decryption. The private key stored that way has the size of

$$(n-k)n + (n-k+1+2\cdot\log_2 n) + k^2 + n\cdot\log_2 n$$

bits for McEliece Cryptosystem and

$$(n-k+1+2\cdot\log_2 n) + (n-k)^2 + n\cdot\log_2 n$$

for the Niederreiter version. Alternatively, the holder of the secret key can omit storing the matrix H, as it is not needed to compute the syndrome of the received ciphertext. However, this would decrease the speed of decryption.

To store the public key requires $n\cdot k$ bits for the McEliece cryptosystem. For the CCA2-secure variants of the McEliece PKC it is possible to give the public generator matrix G in its systematic form. If we choose the first k columns of G to be the identity matrix, then we can describe the public key by only giving the last $(n-k)$ columns of G, called the *redundant part*. This requires

$$k\cdot(n-k)$$

bits. The same is true for the the Niederreiter PKC. Table 2 shows the performance of the original McEliece PKC for some example parameters.

8.3 Choice of Parameters

Unfortunately, there is no simple criterion for the choice of t with respect to n. One should try to make it as difficult as possible to attack the cryptosystem using the known attacks. For the sample parameter sets from Table 2, Table 3 shows the theoretical work factors for the McEliece cryptosystem (the CCA2-secure variants and the original one). In comparison, Table 4 gives the estimated work factors for the RSA cryptosystem.

As one can observe from the tables, today the best attack against McEliece's cryptosystem is CC-LWCW (Algorithm 4.2.3), which is Stern-LWCW with

McEliece system parameters $(n, k, d = 2t + 1)$	Size public key in bytes		Workfactor (binary operations)	
	plain	CCA2-secure	encryption	decryption
$(1024, 524, 101)$	67,072	32,750	2^{18}	2^{22}
$(2048, 1608, 81)$	411,648	88,440	$2^{20.5}$	2^{23}
$(2048, 1278, 141)$	327,168	123,008	2^{20}	2^{24}
$(2048, 1025, 187)$	262,400	131,072	2^{20}	$2^{24.5}$
$(4096, 2056, 341)$	1,052,672	524,280	2^{22}	$2^{26.5}$

Table 2: Performance of the McEliece PKC

McEliece system parameters $(n, k, d = 2t + 1)$	Workfactor (binary operations)		
	GISD $p = 2$	Leon-LWCW $p = 3$, $l = m$	CC-LWCW [3] $p = 2$, $l = 2m - 1$
$(1024, 524, 101)$	2^{70}	2^{69}	2^{64}
$(2048, 1608, 81)$	2^{110}	2^{107}	2^{98}
$(2048, 1278, 141)$	2^{120}	2^{118}	2^{110}
$(2048, 1025, 187)$	2^{115}	2^{112}	2^{106}
$(4096, 2056, 341)$	2^{195}	2^{193}	2^{184}

Table 3: Attacking the McEliece PKC

[3] Approximation without determining the exact value of the number of expected iterations. The exact evaluation uses a Markov chain and thus no closed formula is available (see [9]).

System	Size public key in bytes	Workfactor (binary operations)		
		en-cryption	de-cryption	best attack [5]
RSA 1024-bit Modulus	256	2^{30}	2^{30}	2^{79}
RSA 2048-bit Modulus	512	2^{33}	2^{33}	2^{95}
RSA 4096-bit Modulus	1024	2^{36}	2^{36}	2^{115}

Table 4: Performance of the RSA PKC

[5] this is the NFS attack for factoring the RSA modulus, see [31].

Markov chain improvement. CC-LWCW has a polynomial space complexity and its work factor may be approximated by

$$\mathcal{O}(n^3)2^{-t\log_2(1-k/n)},$$

if t is small and k/n is not too close to 1 (compare [43]). Since $n = 2^m$ and $k = n - tm$, N. Sendrier concludes, that the maximum degree of security is obtained for an information rate $k/n \approx 1 - 1/\exp(1)$. We omitted to consider the statistical decoding attack on the McEliece Cryptosystem because of serious doubts regarding the assumptions made by the author of [25], compare section 4.3.

9 Conclusion

After more than twenty years of research the McEliece PKC cryptosystem slowly comes to the fore as a practical alternative to RSA in applications where long term security is needed. There are no known classical or quantum computer attacks on McEliece's cryptosystem, which have sub-exponential running time. Despite the lack of efficient attacks on McEliece's proposal, none of the cryptographic schemes based on coding theory is proven to be as secure as some classic problem of coding theory. Nevertheless, a key size of 123KB seems to be secure until the year 2041.

The fast increasing amount of storage space on small devices like USB Tokens, PDAs and mobile phones would even allow an application of the McEliece PKC nowadays. We believe, that the McEliece PKC might be used within the next decades, even if no quantum computer is available. The advantage of code based cryptography lies in the faster en- and decryption, which helps to reduce the battery drain of cryptographic applications on mobile devices.

Another interesting property of code based cryptography is the fact, that one can build a complete infrastructure from it. Identification schemes, signature schemes and even random number generators as well as hash functions are available.

Acknowledgment

We would like to thank Prof. Dr. Johannes Buchmann for inspiring for this work and Dr. Ulrich Vollmer for his constant support and most helpful comments.

References

[1] M. Alabbadi and S.B. Wicker. Security of Xinmei digital signature scheme. *Electronics Letters*, 29(9):890–891, 1992.

[2] M. Alabbadi and S.B. Wicker. A digital signature scheme based on linear error-correcting block codes. In *ASIACRYPT '94*, volume LNCS 917, pages 238–248, Springer 1995.

[3] D. Augot, M. Finiasz, and N.Sendrier. A family of fast syndrome based cryptographic hash functions. In *Proc. of Mycrypt 2005*, volume 3715 of *LNCS*, pages 64–83, 2005.

[4] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[5] T. Berson. Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. In *Proceedings of CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 213–220. Springer Verlag, 1997.

[6] E.F. Brickell and A.M. Odlyzko. Cryptanalysis: A survey of recent results. In *Proc. of the IEEE*, volume 76(5), pages 578–593, 1988.

[7] A. Canteaut and F. Chabaud. Improvements of the attacks on cryptosystems based on error-correcting codes. *Rapport interne du Departement Mathematiques et Informatique*, LIENS-95-21, 1995.

[8] A. Canteaut and F. Chabaut. A new algorithm for finding minimum-weight words in a linear code: application to primitive narrow-sense BCH-codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.

[9] A. Canteaut and N. Sendrier. Cryptanalysis of the original McEliece cryptosystem. In *Advances in Cryptology - ASIACRYPT '98 Proceedings*, pages 187–199. Springer-Verlag, 1998.

[10] Florent Chabaud. On the security of some cryptosystems based on error-correcting codes. *Lecture Notes in Computer Science*, 950:131–139, 1995.

[11] N. Courtois, M. Finiasz, and N.Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248, pages 157–174. Springer-Verlag, 2001.

[12] F. Levy dit Vehel and S. Litsyn. Parameters of goppa codes revisited. *IEEE Transactions on Information Theory*, 43(6):1811–1819, 1997.

[13] J.-B. Fischer and J. Stern. An eficient pseudo-random generator provably as secure as syndrome decoding. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 245–255. Springer-Verlag, 1996.

[14] M. Fossorier, H. Imai, and K. Kobara. Modeling bit flipping decoding based on non orthogonal check sums and application to iterative decoding attack of McEliece crypto-system. In *Proc. of 2004 IEEE International Symposium on Information Theory and its Applications, Parma, Italy*, October 2004.

[15] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proc. of CRYPTO*, volume 547 of *LNCS*, pages 535–554. Springer Verlag, 1999.

[16] E.M. Gabidulin, A.V. Ourivski, B. Honary, and B. Ammar. Reducible rank codes and their applications to cryptography. *IEEE Transactions on Information Theory*, 49(12):3289–3293, 2003.

[17] E.M. Gabidulin, A.V. Paramonov, and O.V. Tretjakov. Ideals over a non-commutative ring and their applications to cryptography. In *Proc. Eurocrypt '91*, volume 547 of *LNCS*. Springer Verlag, 1991.

[18] P. Gaborit. Shorter keys for code based cryptography. In *Proc. of WCC 2005*, pages 81–90, 2005.

[19] S. Gao and D. Panario. Tests and constructions of irreducible polynomials over finite fields. *Foundations of Computational Mathematics*, pages 346–361, 1997.

[20] K. Gibson. Equivalent Goppa codes and trapdoors to McEliece's public key cryptosystem. In D. W. Davies, editor, *Advances in Cryptology - Eurocrypt'91*, volume 547 of *LNCS*, pages 517–521. Springer Verlag, 1991.

[21] V. D. Goppa. A new class of linear correcting codes. *Problems of Information Transmission*, 6 (3):207 – 212, 1970.

[22] L. Harn and D.-C. Wang. Cryptanalysis and modification of digital signature scheme based on error-correcting codes. *Electronics Letters*, 28(2):157–159, 1992.

[23] Heise and Quattrocchi. *Informations- und Codierungstheorie.* Springer Berlin Heidelberg, 3 edition, 1995.

[24] IEEE 1363-2000: Standard specifications for public key cryptography, 2000.

[25] A. Al Jabri. A statistical decoding algorithm for general linear block codes. In *Cryptography and Coding 2001*, volume 2260 of *LNCS*, pages 1–8. Springer Verlag, 2001.

[26] H. Janwa and O. Moreno. McEliece public key cryptosystems using algebraic-geometric codes. *Designes, Codes and Cryptography*, 8:293–307, 1996.

[27] K. Kobara and H. Imai. Semantically secure McEliece public-key cryptosystems - conversions for McEliece PKC. In *Practice and Theory in Public Key Cryptography - PKC '01 Proceedings*. Springer Verlag, 2001.

[28] K. Kobara and H. Imai. On the one-wayness against chosen-plaintext attacks of the Loidreau's modified McEliece PKC. *IEEE Transactions on Information Theory*, 49, No. 12:3160–3168, 2003.

[29] J. C. Lagarias and Andrew M. Odlyzko. Solving low-density subset sum problems. *J. ACM*, 32(1):229–246, 1985.

[30] P.J. Lee and E.F. Brickell. An observation on the security of McEliece's public key cryptosystem. In *Advances in Cryptology-EUROCRYPT'88*, volume 330 of *LNCS*, pages 275–280. Springer Verlag, 1989. http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/E88/275.PDF.

[31] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001.

[32] J.S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.

[33] Y.X. Li, R.H. Deng, and X.M. Wang. the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, Vol. 40, pp. 271-273, 1994.

[34] R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications.* Cambridge University Press, 2 edition, 1986.

[35] P. Loidreau and N. Sendrier. Weak keys in the McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, 47, No. 3:1207 –1211, March 2001.

[36] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correctiong Codes*. North-Holland Amsterdam, 7 edition, 1992.

[37] R.J. McEliece. A public key cryptosystem based on algebraic coding theory. *DSN progress report*, 42-44:114–116, 1978.

[38] R. Overbeck. A new structural attack for GPT and variants. In *Proc. of Mycrypt 2005*, volume 3715 of *LNCS*, pages 50–63. Springer Verlag, 2005.

[39] R. Overbeck. Statistical decoding revisited. In *Proc. of ACISP 2006*, volume 4058 of *LNCS*, pages 283–294. Springer Verlag, 2006. to appear.

[40] N. Patterson. Algebraic decoding of Goppa codes. *IEEE Trans. Info. Theory*, 21:203–207, 1975.

[41] D. Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In *Proc. of PKC*, volume 1751 of *LNCS*, pages 129–146. Springer Verlag, 2000.

[42] V. C. Jr. Rocha, V.C. JR Da Rocha, and D.L. Macedo. Cryptanalysis of Krouk's public-key cipher. *Electronics Letters*, 32 (14):1279 – 1280, 1996.

[43] N. Sendrier. On the security of the McEliece public-key cryptosystem. In M. Blaum, P.G. Farrell, and H. van Tilborg, editors, *Proceedings of Workshop honoring Prof. Bob McEliece on his 60th birthday*, pages 141–163. Kluwer, 2002.

[44] N. Sendrier. Finding the permutation between equivalent linear codes: the support splitting algorithm. *IEEE Transactions on Information Theory*, 46:1193–1203, Jul 2000.

[45] N. Sendrier. On the dimension of the hull. *SIAM Journal on Discrete Mathematics*, 10(2):282–293, May 1997.

[46] V.M. Sidelnikov. A public-key cryptosystem based on binary Reed-Muller codes. *Discrete Mathematics and Applications*, 4 No. 3, 1994.

[47] V.M. Sidelnikov and S.O. Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 2, No. 4:439–444, 1992.

[48] J. Stern. A method for finding codewords of small weight. *Coding Theory and Applications*, 388:106–133, 1989.

[49] J. Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology - CRYPTO'93*, volume 773 of *LNCS*. Springer Verlag, 1994.

[50] J. Stern. Can one design a signature scheme based on error-correcting codes. In *ASIACRYPT '94*, volume 917 of *LNCS*, pages 424–426, 1995.

[51] K. Tanaka und S.Uchiyama T.Okamoto. Quantum public key cryptosystems. In *Proc. Of CRYPTO 2000, LNCS*, volume 1880, pages 147–165, 2000. Springer-Verlag.

[52] W. Xinmei. Digital signature scheme based on error-correcting codes. *Electronics Letters*, 26(13):898–899, 1990.