# CRYPTOGRAPHY BASED ON ERROR CORRECTING CODES

THÈSE N$^{O}$ 3846 (2007)

PRÉSENTÉE LE 27 JUILLET 2007

À LA FACULTÉ DES SCIENCES DE BASE

LABORATOIRE DE MATHÉMATIQUE ALGORITHMIQUE

PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Lorenz MINDER

ingénieur mathématicien diplômé EPF
de nationalité suisse et originaire de Bâle (BS)

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2007

**Abstract.** The idea to use error-correcting codes in order to construct public key cryptosystems was published in 1978 by McEliece [ME1978]. In his original construction, McEliece used Goppa codes, but various later publications suggested the use of different families of error-correcting codes. The choice of the code has a crucial impact on the security of this type of cryptosystem. Some codes have a structure that can be recovered in polynomial time, thus breaking the cryptosystem completely, while other codes have resisted attempts to cryptanalyze them for a very long time now.

In this thesis, we examine different derivatives of the McEliece cryptosystem and study their structural weaknesses. The main results are the following: In chapter 3 we devise an effective structural attack against the McEliece cryptosystem based on algebraic geometry codes defined over elliptic curves. This attack is inspired by an algorithm due to Sidelnikov and Shestakov [SS1992] which solves the corresponding problem for Reed-Solomon codes. The presented algorithm is heuristic polynomial time and thus inverts trapdoors even for very large codes.

In chapter 4, we show that the Sidelnikov cryptosystem [S1994], which is based on binary Reed-Muller codes, is insecure. The basic idea of our attack is to use the fact that minimum weight words in a Reed-Muller code have very particular properties.This attack relies on the ability to find minimum weight words in the code, a problem that is, in this specific instance, much easier than general decoding, and feasible for interesting parameters in a modest amount of time. The attack has subexponential running time if the order of the code is kept fixed, and it breaks the large keys as proposed by Sidelnikov in under an hour on a stock PC.

In the chapter 5, we finally discuss some of the problems to solve if one attempts to generalize these algorithms.

*Keywords:* Public key cryptography, McEliece cryptosystem, error-correcting codes, Reed-Muller codes, Sidelnikov cryptosystem, algebraic geometry codes, structural attack.

**Résumé.** L'idée de réaliser la cryptographie à clé publique à l'aide de codes correcteurs d'erreurs a été conçue en 1978 par McEliece [ME1978]. Ce dernier utilisait des codes de Goppa dans sa construction originale, mais différents auteurs ont par la suite proposé l'utilisation d'autres codes. Le choix du code a un impact crucial sur la sécurité de tels systèmes de cryptage. Pour quelques codes il existe des algorithmes qui reconstruisent leur structure en un temps polynomial, cassant donc complètement le système de cryptage, tandis que d'autres codes résistent déjà des décennies à la cryptanalyse.

Dans cette thèse, nous exhibons des faiblesses structurelles de différentes variantes du cryptosystème de McEliece. Les résultats principaux sont les suivants: Dans le chapitre 3 nous construisons une attaque structurelle efficace contre le système de cryptage de McEliece basé sur des codes de géométrie algébrique définis sur des courbes elliptique. Cette attaque est inspiré par un algorithme dû à Sidelnikov et Shestakov [SS1992] qui résout le problème correspondant pour les codes de Reed-Solomon. Le temps de parcours de l'algorithme est heuristiquement polynomial et peut donc être utilisé pour inverser des trapdoors provenant de très grands codes.

Dans le chapitre 4, nous montrons que le système de cryptage de Sidelnikov [S1994], qui est basé sur les codes de Reed-Muller, n'est pas sûr. L'idée de base de notre attaque est d'utiliser le fait que les mots à poids minimum dans un code de Reed-Muller ont des propriétés particulières pour déduire de l'information sur le code en question. Donc cette attaque dépend de la faisabilité de trouver des mots à poids minimaux dans le code. Ce problème est, dans ce cas particulier, beaucoup plus facile que le décodage général, et peut se faire en peu de temps. Le temps de parcours de cet algorithme est subexponentiel si l'ordre du code est fixé, et pour des tailles de clé proposées par Sidelnikov, il faut moins qu'une heure de temps sur un PC normal.

Dans le chapitre 5, nous discutons finalement quelques problèmes à résoudre si on tente de généraliser ces algorithmes.

*Mots clés:* Cryptographie à clé publique, cryptosystème de McEliece, codes correcteurs d'erreurs, codes de Reed-Muller, cryptosystème de Sidelnikov, codes de géométrie algébrique, attaque structurelle.

# Contents

## Acknowledgement

I am very grateful to Amin for supervising me, for his continuous support and his constant willingness to share his rich knowledge and savoir faire over the years I have been in the group. It was a great work and learning environment, and I am particularly thankful that I was given the opportunity to work and study in a way that would suit me well, with wide freedom in choosing the way to go and the problems to study. Thank you, Amin, for the trust and confidence you had in me!

I would also like to thank my colleagues and friends with whom I have shared offices, in particular, Andrew, Gérard and Frédérique. I lived a very friendly and motivating working athmosphere with them, and I have great memories of our cooperation, be it when preparing exercises, course notes, or also of the two weeks intensive programming session I once had with Andrew. I would like to thank Gérard for our collaboration when he was with the group, him sharing his expertise and the many interesting discussions; I have learnt many things in cryptography from Gérard. His support and encouragement has also been greatly appreciated.

I would like to thank the people who have kindly agreed to be on my thesis committee: Anne Canteaut, Robert Dalang, Arjen Lensta and Nicolas Sendrier. I am very happy and honored to have you on my committee.

Last but not least, I would like to thank my family, Memer, Beber, Ögi, Li and Dodee, who have been very supportive all along the way, and my friends, in particular my flat mates Catherine, Régis, Dön, Benjamin and Eva.

My work has been supported by a grant of the Swiss National Funds, and I would like to thank them for making this work possible.

# Chapter 1

# Introduction

We first give a brief description of public key cryptography. We then turn to public key cryptosystems based on error-correcting codes, giving a short overview of the progress made in the past thirty years in this domain. Finally, we briefly summarize the new results of this thesis.

## 1.1 Public key cryptography

In conventional cryptography, encryption and decryption were symmetric operations: Anyone who had access to the key could both encrypt and decrypt. The simplest example of such a cryptosystem is the codebook method, where each word is replaced by another one according to a given table, called the codebook. Anyone having access to the codebook can decrypt messages.

Such cryptosystems that use a single key for both encryption and decryption are called *symmetric* cryptosystems. While this type of system is and remains tremendously useful in practice, some cryptographic problems cannot be satisfactorily solved with symmetric cryptography alone.

Imagine, for example, that Alice wants to send confidential information to Bob via e-mail. They can decide to use a symmetric scheme to encrypt their messages so that no eavesdropper on the Internet can obtain useful information from the e-mail exchange other than the observation that an exchange is taking place. If this approach is used, they have to use the same key, so for example Alice decides on a key and hands it to Bob. Since Alice cannot just send the key over the Internet (given that an eavesdropper might then receive the key as well), they have to use a *secure channel* to transfer the key instead. For example, they can physically meet to exchange the keys. In fact, Alice would have to establish a secure channel with anyone she would want to exchange confidential information with. The problem is that secure channels can be prohibitively expensive.

Public key cryptography was invented in 1976 by Diffie and Hellman[DH1976] to solve this problem. A public key cryptosystem has the following components:

- *A Key generator.* The key generator computes a key pair $(K_{\text{pub}}, K_{\text{priv}})$ from a random number.
- *An encryption function.* The encryption function $\text{enc}(\cdot,\cdot)$ takes a public key $K_{\text{pub}}$ and a cleartext message $m$ and computes a ciphertext $c$, i.e.,

  $$\text{enc}(K_{\text{pub}}, m) = c.$$

- *A decryption function.* The decryption function $\text{dec}(\cdot,\cdot)$ takes a private key $K_{\text{priv}}$ and a ciphertext $c$ and computes the cleartext message $m = \text{dec}(K_{\text{priv}}, c)$ corresponding to $c$.

The message space, ciphertext space and key space can be chosen to be the set of binary vectors of certain lengths.

The cryptosystem must satisfy a few conditions. It must be correct, i.e.,

$$\text{dec}(K_{\text{priv}}, \text{enc}(K_{\text{pub}}, m)) = m$$

must hold for any generated key pair $(K_{\text{pub}}, K_{\text{priv}})$ and any message $m$ from the message space. It must be practical: Key generation, encryption, and decryption must be polynomial time operations. Last but not least, it must be secure, which means that given a public key $K_{\text{pub}}$, a ciphertext $c$, and an exact description of the system, it should be computationally infeasible to compute parts of the corresponding plaintext $\text{dec}(K_{\text{priv}}, c)$. This is not the strongest definition for the security of such systems that can be given, and it is rather vague, but it is sufficient for our needs.

## 1.2 Cryptography based on error correcting codes

Despite the fact that the idea of public key cryptography has been published more than thirty years ago now, and that public key cryptosystems are widely used in practice, astonishingly few constructions have led to systems that are likely to be safe.

To date, broadly three classes of public key cryptosystems are known: number theory based ones, lattice based constructions, and constructions based on error correcting codes.

Number theory based constructions have seen by far the widest adoption in practice, and of those, the most frequently used ones are based on the hardness of either factoring or the discrete logarithm problem.

This concentration on very few quite similar constructions is rather concerning, especially in light of the fact that despite extensive research, there is very little theoretical evidence indicating that these problems are indeed hard.

Another problem is that some of the number theoretic problems are not hard under all computational models: Specifically, Shor has shown that both factoring and the computation of discrete logarithms can be done in polynomial time on a quantum computer [S1997]. While this threat remains a theoretical one for the time being, it is not clear that this will be the case indefinitely.

It seems to be both prudent and necessary, given the scarce choice of options, to both seek new constructions of public key cryptosystems, and to carefully examine the various existing constructions. In this thesis, code based cryptosystems will be considered.

### 1.2.1 The history of code based cryptography

The idea of code based cryptosystems is almost as old as public key cryptography itself: the first such system was proposed by McEliece in 1978, [ME1978]. This original construction is not yet broken (even though the original choice of parameters has been shown to be too small, [CS1998]), and, up to the fact that its public keys are rather large, it is reasonably efficient.

Since McEliece's original publication, a significant amount of research went into analyzing and improving this system. We will give a short summary of the progress that has been made in the domain, but limiting the discussion to systems that are based on the classical decoding problem, and exclude notably the variants that use other error models, such as for example cryptosystems based on rank codes.

One line of research was concerned with improving the direct decoding attacks that McEliece had outlined in his original paper, and with choosing the parameters that would maximize resistance against these attacks. This problem was investigated, for example, by Adams and Meijer in [AM1987], Lee and Brickell [LB1988], Leon [L1988], Stern [S1989], and by Canteaut and Chabaud [CC1998].

Another line of research was concerned with modifying McEliece's construction in order to obtain a more powerful system. Niederreiter gave a derandomized variant of the McEliece system in [N1986], and also proposed to use Reed-Solomon codes instead of Goppa codes. Sidelnikov proposed a variant of the McEliece cryptosystem that would use Reed-Muller codes in [S1994]. Janwa and Moreno proposed to use algebraic geometry codes [JM1996], and recently, Gaborit proposed the use of BCH codes [G2005].

A third line was concerned with structural analysis, i.e., the study of the structure of the underlying codes in order to devise attacks against such cryptosystems. In 1992, Sidelnikov and Shestakov discovered a deterministic polynomial time structural attack against Niederreiter's proposal to use Reed-Solomon codes [SS1992]. Shortly thereafter, Sendrier discovered an effective attack against concatenated codes [S1994]. In 2001, Loidreau and Sendrier devised a structural attack against weak keys in the McEliece system [LS2001].

### 1.2.2 The current situation

I think the Achilles heel of McEliece type cryptosystems today is that their security properties are not well understood. If only the direct decoding attacks are considered, relatively simple statements about the security of such systems can be made: the succinct summary is that better codes lead to better security. When it comes to structural attacks, however, not much is known. A fundamental question that still has no satisfactory answer is the following: what properties should a code have, so that it can be considered as structurally safe to be used in a McEliece type cryptosystem?

Given the variety of constructions of error-correcting codes, it may not be possible to develop a complete theory about the structural security of error-correcting codes. Research in the domain is thus constrained, for the time being, to analyze existing constructions, and to attempt to find common patterns among the known attacks.

### 1.2.3 Contributions of this thesis

This thesis explores questions concerning the structural security of McEliece type cryptosystems. We will now give a brief overview on the topics that will be covered, and the basic ideas that will be pursued in this thesis.

Chapter 2 is an introductory chapter that serves on the one hand to recall the needed notions of coding theory, and on the other hand to motivate the later chapters. For example, the focus on algebraic constructions of codes for cryptographic use is briefly justified. A short review of the known low weight word finding algorithms will also be given: it turns out that these algorithms play a crucial role in some structural attacks.

#### Elliptic codes

In Chapter 3, we present a generalization of the Sidelnikov-Shestakov attack on Reed-Solomon codes [SS1992]: Reed-Solomon codes are the subclass of algebraic geometry codes that are defined over curves of genus 0. We will adapt this attack, so that it applies to codes defined over curves of genus 1, that is, to elliptic codes.

The motivation for looking at this type of code is twofold. First, if these codes were not structurally weak, they would be a very advantageous choice for this cryptosystem: their high minimum distance makes them resistant against direct decoding attacks for comparatively small key sizes. Second, general algebraic geometry codes lead to the only known construction of a McEliece type cryptosystem with the property that the direct decoding attack scales exponentially with the block length $n$ (See section 5.1.1). Therefore, a further generalization of the attack against elliptic codes to arbitrary algebraic geometry codes would break the best (in a sense) known construction.

We briefly illustrate the main idea of the algorithm. Let $\mathbb{F}$ be a finite field, $E$ an elliptic curve defined over $\mathbb{F}$, $\Delta$ a divisor on $E$ defined over $\mathbb{F}$, and $\{P_1, \dots, P_n\}$ a set of $n$ distinct $\mathbb{F}$-rational points on $E$, such that

$$\operatorname{supp}(\Delta) \cap \{P_1, \dots, P_n\} = \emptyset.$$

The elliptic code $\mathrm{AGC}(E, \Delta, (P_1, \dots, P_n))$ is the image of the evaluation map

$$\begin{aligned} \mathrm{ev}_{(P_1,\dots,P_n)} : \mathcal{L}(\Delta) \cap \mathbb{F}(E) &\to \mathbb{F}^n \\ f &\mapsto (f(P_1), \dots, f(P_n)), \end{aligned}$$

where $\mathcal{L}(\Delta)$ denotes the linear space associated with the divisor $\Delta$, and $\mathbb{F}(E)$ the $\mathbb{F}$-rational functions over $E$.

The trapdoor inversion problem for McEliece type cryptosystems based on elliptic codes can be stated as follows: Given a random basis of the elliptic code

$$\mathcal{C} := \mathrm{AGC}(\hat{E}, \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n)),$$

where $\hat{E}$, $\hat{\Delta}$ and $\hat{P}_1, \dots, \hat{P}_n$ are unknowns, find an elliptic curve $E$, divisor $\Delta$, and points $P_1, \dots, P_n$ on $E$ such that

$$\mathcal{C} = \mathrm{AGC}(E, \Delta, (P_1, \dots, P_n)).$$

In Chapter 3 we present an adaptation of this method to codes defined on curves of genus $g = 1$ (i.e., on elliptic curves). We now give a short sketch of the method. Let

$$\mathcal{C} = \mathrm{AGC}(E, \Delta, (P_1, \dots, P_n)).$$

be an elliptic code of parameters $[n, k, n-k]$. Using curve isomorphisms, we can assume $\Delta = k\langle\mathcal{O}\rangle$, where $\mathcal{O}$ denotes the point at infinity of $E$.

Let $v \in \mathcal{C}$ be a minimum weight codeword. Then $v$ has zeros at $k$ positions, for example at the positions $1, \dots, k$. Let $f \in \mathcal{L}(k\langle\mathcal{O}\rangle)$ correspond to $v$. Then the divisor of $f$ satisfies

$$\operatorname{div}(f) \succeq \langle P_1\rangle + \cdots + \langle P_k\rangle - k\langle\mathcal{O}\rangle. \tag{1.1}$$

But since the divisor on the right hand side is of degree 0, equation (1.1) holds with equality, and yields thus a linear equation on the representants of $\langle P_i\rangle - \langle\mathcal{O}\rangle$ in $\mathrm{Pic}^0_{\mathbb{F}}(E)$. If we collect a sufficient number of such equations we can solve for the $\mathbb{Z}$-module $\mathcal{G} \cong \mathrm{Pic}^0_{\mathbb{F}}(E)$ and also compute $z_i \in \mathcal{G}$ corresponding to the representants of $\langle P_i\rangle - \langle\mathcal{O}\rangle$ in $\mathrm{Pic}^0_{\mathbb{F}}(E)$.

The key point about the $z_i$ is that they leak geometric information about the points $P_i$ on the curve. We use the knowledge of $z_i$ to explicitly construct known rational functions on $E(\mathbb{F})$.

These rational functions, together with their known evaluations on some of the points $P_i$ can then be used to explicitly compute coordinate positions of a number of the points $P_i$. A speedy algorithm to do so has to be carefully crafted, if an effective attack is sought.

A few more steps are needed to complete the trapdoor inversion problem, but the final outcome is an algorithm that runs heuristically in $O(n^4)$ in the settings that are most interesting in practice. In the experiments that we ran, this reconstruction procedure would invert trapdoors even of very large codes ($\mathbb{F} = GF(953), n = 1008, k = 504$) in roughly two hours, and much faster for codes of more realistic size.

### Using structure in Reed-Muller codes

In 1994, Sidelnikov proposed a McEliece type cryptosystem that uses Reed-Muller codes as the underlying family of codes [S1994]. This cryptosystem is today known as the Sidelnikov cryptosystem. Reed-Muller codes are appealing for this application since there are known, efficient decoding algorithms that correct more errors than the minimum distance of the code, with high probability. For example, Sidelnikov reports in his paper [S1994] that as many as 200 errors can be corrected with very high probability when the Reed-Muller code of parameters $[1024, 176, 128]$ is used. For this reason the Sidelnikov cryptosystem has a high transmission rate and is well protected against direct decoding attacks.

In Chapter 4, we show however that the Sidelnikov cryptosystem can be structurally broken. To quickly summarize the attack, recall that a codeword in $\mathcal{R}(r, m)$, the $r$-th order Reed-Muller code of length $2^m$, is formed by evaluating a Boolean function $f$ in $m$ variables to binary vectors of length $2^m$. There is a bijective identification of $\mathcal{R}(r, m)$ with the Boolean functions in $v_1, \ldots, v_m$ of degree at most $r$, a set we denote by $\mathcal{B}(r, \{v_1, \ldots, v_m\})$. Codewords can thus be regarded as functions.

We consider the problem of recovering a private key from a given public key in the Sidelnikov cryptosystem: We are given a random basis of the code $\mathcal{R}(r, m)^\sigma$, i.e., of $\mathcal{R}(r, m)$ with its positions permuted by some unknown (secret) permutation $\sigma$. We seek a permutation $\tau$ such that

$$\mathcal{R}(r, m)^{\tau \circ \sigma} = \mathcal{R}(r, m).$$

A result due to Kasami and Tokura [KT1970] says that if $f \in \mathcal{B}(r, \{v_1, \ldots, v_m\})$ corresponds to a minimum weight codeword, then, up to an affine change of variables, we have

$$f = v_1 \cdots v_r.$$

From this result it can be shown that the code obtained from shortening $\mathcal{R}(r, m)$ on the support of a minimum weight word has a particular structure: it is a concatenated code, with the inner words lying on sets that fix the values of $v_1, \ldots, v_r$. This fact can be used to construct words in the subcode $\mathcal{R}(r-1, m)^\sigma$ of $\mathcal{R}(r, m)^\sigma$. By iterating the procedure, the complete code $\mathcal{R}(r-1, m)^\sigma$ can be recovered. Hence, the order $r$ can be iteratively reduced, until $\mathcal{R}(1,m)^\sigma$ is obtained. The problem of identifying a suitable permutation $\tau$ such that

$$\mathcal{R}(1,m)^{\tau\circ\sigma} = \mathcal{R}(1,m)$$

turns out to be easy to solve, and the same permutation $\tau$ will also satisfy $\mathcal{R}(r,m)^{\tau\circ\sigma} = \mathcal{R}(r,m)$.

This reconstruction procedure requires finding low weight words in the code, a problem that is believed to be intractable in many instances. Due to several reasons, however, it turns out to be feasible in this case: First, Reed-Muller codes are low rate codes, and the known good low weight word finding algorithms work better in this setting. Second, Reed-Muller codes contain many minimum weight words, and since any of these can be used, this reduces the search space considerably. Note also that in the example we cited before, direct decoding would require finding a word of weight $\sim 200$, while the low weight words only have weight 128 in this case, so they are much easier to find.

The result is that Reed-Muller codes of considerable size can be reconstructed with a bearable amount of computation. For example, a cryptosystem using Sidelnikov's largest suggested parameters takes roughly an hour to break on a standard PC.

### Extensions

In Chapter 5, we discuss possibilities to extend the results of the earlier chapters, and the problems that need solving if such generalizations are sought.

Section 5.1.2 is devoted to discuss possibilities to generalize the attack on elliptic codes to codes over more general curves. Our findings will be threefold: Some of the ideas can be generalized to other families rather easily, other parts lead to algorithms that, if generalized in the obvious way, will be impractical for too large genera, and a number of problems lead to interesting geometric questions.

A second question that will be discussed is the possibility to generalize the Sidelnikov-Shestakov attack to large subcodes of Reed-Solomon codes. This is in some sense a counter part to the discussion in section 3.1.3 which estimates the maximal size of the subcode if structural safety against any attack based on low weight words is sought: in section 5.2, we sketch an algorithm to break codes in the case where the subcode is too large.

# Chapter 2

# Aspects of coding theory

In this section, we present the notions of coding theory that are prerequisite for the following chapters. The intent is to merely recall the notions and to define the notations, an in-depth presentation of classical coding-theoretic notions has no space in this thesis. Instead, pointers to the relevant literature will be given.

## 2.1 Linear codes

### 2.1.1 Basic definitions

Let $\mathbb{F}$ be a finite field, called the *alphabet*, $n$ an integer (the *block length*) and $k$ another integer (the *dimension*) with $k \leq n$. An $[n, k]$ *linear code defined over* $\mathbb{F}$ is a vector space $\mathcal{C} \subset \mathbb{F}^n$ of dimension $k$. The elements in $\mathcal{C}$ are called *codewords*. The *rate* of $\mathcal{C}$ is the quantity $R = k/n$. If $\mathcal{C}$ is a code and $\mathcal{C}' \subseteq \mathcal{C}$ is also a code, then $\mathcal{C}'$ is said to be a *subcode* of $\mathcal{C}$.

Let $x_1, \ldots, x_k$ be a basis of $\mathcal{C}$. The $k \times n$ matrix with coefficients in $\mathbb{F}$ whose rows are formed of the vectors $x_1, \ldots, x_k$ is called a *generator matrix* for $\mathcal{C}$ and is generally denoted by $G$. Since the basis of a vector space is not unique, a code does not have a unique generator matrix.

The *Hamming distance* of two vectors $x, y \in \mathbb{F}^n$ is the number of positions on which $x$ and $y$ differ, i.e., the quantity

$$d(x, y) = \#\{1 \leq i \leq n \mid x_i \neq y_i\}.$$

The Hamming distance does indeed verify the axioms of a distance. In particular, it verifies the triangular inequality.

The *minimum distance* of a code $\mathcal{C}$ is the quantity

$$d := \min_{x,y \in \mathcal{C}, y \neq x} d(x, y),$$

and the *relative minimum distance* of $\mathcal{C}$ is the quantity $d/n$.

Since for linear codes, we have $x - y \in \mathcal{C}$ whenever $x, y \in \mathcal{C}$, it follows that

$$d = \min_{x \in \mathcal{C}, x \neq 0} d(x, 0).$$

The *Hamming weight* $\mathrm{wt}(x)$ of a vector $x$ is its Hamming distance to 0. In that terminology, the above statement says that the minimum distance of a code is equal to the minimum weight of a nonzero vector in $\mathcal{C}$.

An $[n, k, d]$-code is an $[n, k]$ linear code with minimum distance $d$. A *minimum weight word* in $\mathcal{C}$ is a word $x \in \mathcal{C}$ with $\mathrm{wt}(x) = d$.

A set of codes $\mathcal{F} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots\}$ where each $\mathcal{C}_i$ is defined over the same alphabet $\mathbb{F}$ is called a *family* of codes. Note that an infinite family necessarily contains members of arbitrary long block lengths.

An infinite family $\mathcal{F}$ is said to be *asymptotically good*, if positive real numbers $R_0, \delta_0$, and an infinite subfamily $\mathcal{F}' \subseteq \mathcal{F}$ exist such that each $\mathcal{C} \in \mathcal{F}'$ has rate at least $R_0$ and minimum distance at least $\delta_0 \cdot n$, where $n$ is the block length of $\mathcal{C}$.

**Example.** Fix the alphabet $\mathbb{F}$ of size $q = |\mathbb{F}|$ and $0 < R < 1$, and take the family $\mathcal{F}$ consisting of all codes of block length $n$ and dimension $\lceil nR \rceil$ defined over $\mathbb{F}$ for all $n \in \mathbb{N}$. We would expect $\mathcal{F}$ to be asymptotically good, since otherwise, no good family for the given rate $R$ would exist.

This is indeed the case. Fix $n$, set $k = \lceil nR \rceil$. Fix some arbitrary nonzero $x \in \mathbb{F}^n$ and let $\mathcal{C}$ be a randomly selected code of block length $n$. The probability that $x \in \mathcal{C}$ is $q^{k-n}$. Using this reasoning, the union bound tells us that the probability that any nonzero point in the Hamming ball of center 0 and radius $r$ is contained in $\mathcal{C}$ is at most

$$q^{k-n} \sum_{i=1}^{r} \binom{n}{i} (q-1)^i.$$

Choosing $r$ just large enough that this quantity is $< 1$, we deduce by the probabilistic method that a code of parameters $[n, k]$ and minimum distance at least $r + 1$ exists. Letting $n \to \infty$, we can deduce that $\mathcal{F}$ is an asymptotically good family for any $R$ and relative distance $\delta - \varepsilon$ for any $\varepsilon > 0$, where $\delta$ satisfies

$$H_q(\delta) \geq 1 - R, \tag{2.1}$$

where

$$H_q(x) = \begin{cases} x \log_q(q-1) - x \log_q(x) - (1-x) \log_q(1-x) & \text{if } 0 < x < 1 - q^{-1}, \\ 0 & \text{if } x = 0. \end{cases} \tag{2.2}$$

is the $q$-ary entropy function. Equation (2.1) is known as the *asymptotic Gilbert-Varshamov lower bound.*

### 2.1.2 Constructing codes from other codes

Let $\mathcal{C}$ be an $[n,k,d]$-code. Consider the set

$$\mathcal{C}^{\perp} = \{y \in \mathbb{F}^n \mid \langle y \mid x\rangle = 0 \;\forall x \in \mathcal{C}\},$$

where $\langle\cdot|\cdot\rangle$ denotes the standard scalar product of two vectors. Since $\langle\cdot|\cdot\rangle$ is bilinear, $\mathcal{C}^{\perp}$ is a vector space and thus a linear code. The dimension of $\mathcal{C}^{\perp}$ is $n-k$. Let $H$ be a generator matrix of $\mathcal{C}^{\perp}$, then any word $y \in \mathbb{F}^n$ is a codeword of $\mathcal{C}$ if and only if

$$Hy^t = 0.$$

The matrix $H$ is also called a *parity check matrix* for $\mathcal{C}$. As the name suggests, dual code of $\mathcal{C}^{\perp}$ is $\mathcal{C}$ itself, i.e., $(\mathcal{C}^{\perp})^{\perp} = \mathcal{C}$.

Let $\mathcal{C}$ be an $[n,k,d]$-code, and let $S \subset \{1,\dots,n\}$. The $[n^*,k^*,d^*]$-code obtained from removing the coordinate positions $S$ from $\mathcal{C}$ is a *punctured code*, denoted $\mathcal{C}^S$. The parameters of the punctured code satisfy $n^* = n-|S|$, $k^* \geq k-|S|$, $d^* \geq d-|S|$.

Let $\mathcal{C}'$ be the subcode of $\mathcal{C}$ obtained as follows:

$$\mathcal{C}' = \{x \in \mathcal{C} \mid x_i = 0 \;\forall i \in S\}.$$

Then the puncturing of $\mathcal{C}'$ on $S$ is the *shortened code of* $\mathcal{C}$ *on* $S$, denoted $\mathcal{C}_S$. The shortened code $\mathcal{C}_S$ has parameters $[n^*,k^*,d^*]$, with $n^* = n-|S|$, $k^* \geq k-|S|$, $d^* \geq d$.

Punctured and shortened codes are related as follows (See [HP2003, Theorem 1.5.7]): We have

$$(C_S)^{\perp} = (\mathcal{C}^{\perp})^S \text{ and } (C^S)^{\perp} = (\mathcal{C}^{\perp})_S. \tag{2.3}$$

Let $\mathcal{C}$ be an $[n,k]$-code over $\mathbb{F}$. If $\sigma$ is a permutation on $\{1,\dots,n\}$, then we define the permuted code $\mathcal{C}^{\sigma}$ as the code $\mathcal{C}$ with the coordinate-positions permuted by $\sigma$:

$$\mathcal{C}^{\sigma} := \{x \in \mathbb{F}^n \mid (x_{\sigma^{-1}(1)},\dots,x_{\sigma^{-1}(n)}) \in \mathcal{C}\}.$$

If $\mathcal{C}'$ and $\mathcal{C}$ are $[n,k]$-codes with $\mathcal{C}' = \mathcal{C}^{\sigma}$ for some permutation $\sigma$, then $\mathcal{C}'$ and $\mathcal{C}$ are said to be *equivalent.* If $\sigma$ is a permutation such that $\mathcal{C}^{\sigma} = \mathcal{C}$, then $\sigma$ belongs to the *automorphism group of* $\mathcal{C}$.

### 2.1.3 Algorithmic problems in coding theory

There are a number of algorithmic problems in coding theory whose computational hardness is important to understand.

First, the manipulations on codes given in section 2.1.2 are all easy, e.g., it is a polynomial time task to compute a parity check matrix $H$ for a given generator matrix $G$.

The *encoding problem*, i.e., constructing a map $\mathbb{F}^k \to \mathcal{C}$, where $\mathcal{C}$ is an $[n,k]$-code, is also easy; for example, encoding can be realized as a linear map:

$$\mathrm{enc} : x \in \mathbb{F}^k \mapsto xG,$$

where $G$ is a generator matrix for $\mathcal{C}$.

The *decoding problem*, however, is notoriously hard in general. Let $\mathcal{C}$ be an $[n,k]$-code and $x \in \mathbb{F}^n$ a word, then any vector $y \in \mathcal{C}$ such that

$$d(x,y) = \min_{z \in \mathcal{C}} d(x,z)$$

is called a *maximum likelihood*-decoding of $x$.

The maximum likelihood problem is generally intractable, and therefore attention has to be restricted to specific families of codes, as well as to low weight error-patterns.

We will now introduce notions of families that correct a $\delta(n)$-fraction of errors and of $t$-error-correcting codes. It should be pointed out that the meaning of those notions are not agreed upon in coding theory: in some texts, a code is said to be $t$-error-correcting whenever its minimum distance $d$ satisfies $d \geq 2t+1$. For our purposes, this notion is too weak: we will define a code to be $t$-error-correcting only if a suitable corresponding decoding algorithm exists.

Let $0 \leq p < 1$ be a small fixed error probability. For a code $\mathcal{C}$ of block length $n$, we say that a map

$$\mathrm{dec}_{\mathcal{C}} : \mathbb{F}^n \to \mathcal{C}$$

that satisfies

$$\mathrm{Prob}\big(d(x, \mathrm{dec}_{\mathcal{C}}(x)) \geq \min_{y \in \mathcal{C}, y \neq \mathrm{dec}_{\mathcal{C}}(x)} d(x,y) \;\big|\; d(x,\mathcal{C}) \leq t\big) \leq p$$

for (uniform) random $x \in \mathbb{F}^n$, is a $(1-p)$-correct decoding map for distance $t$.

Let $\mathcal{F}$ be an infinite family of codes, and let $\delta : \mathbb{N} \to \mathbb{R}$. Assume $\mathcal{F}$ is such that there exists a polynomial $\ell(n)$ such that for any $[n,k]$-code $\mathcal{C} \in \mathcal{F}$, there is an algorithm for computing an $(1-p)$-correct decoding map for distance $\lceil n\delta(n) \rceil$, in time $O(\ell(n))$. Then we say that $\mathcal{F}$ *corrects a $\delta(n)$-fraction of errors* with failure probability $p$. For any $\mathcal{C} \in \mathcal{F}$, it is often convenient to say that $\mathcal{C}$ is *$t$-error-correcting* with failure probability $p$ whenever $t \leq \delta(n) \cdot n$.

If $\mathcal{F}$ is an infinite family of codes correcting a $\delta(n)$-fraction of errors for some function $\delta(n)$ verifying $\delta(n) > \varepsilon$ for any $n$ and some $\varepsilon > 0$, and the rate of each member in $\mathcal{F}$ is at least $R_0$ for some $R_0 > 0$, then we say that $\mathcal{F}$ *corrects a constant fraction of errors* with failure probability $p$.

Note that any family that corrects a constant fraction of errors with failure probability 0 is necessarily asymptotically good. The converse statement is neither proved nor disproved. However, Dumer, Micciancio and Sudan have shown in [DMS2003] that under certain complexity theoretic assumptions, distance $2/3d$-decoding is hard in general for asymptotically good families. If the factor 2/3 could be improved to $\varepsilon$ for any $\varepsilon > 0$, this would indeed prove that asymptotically good codes do not necessarily correct a constant fraction of errors.

**Example.** Fix $0 < R < 1$. Let $\mathcal{F}$ be the family of binary Goppa codes of rate $R$ (See [MS1977]). Then one can show that any $[n, k]$-code in $\mathcal{F}$ has relative minimum distance at least

$$2\delta(n) = \frac{2(1-R)}{\lceil \log_2(n) \rceil}, \tag{2.4}$$

and Patterson's algorithm [P1975] will correct $n \cdot \delta(n)$ errors with failure probability 0. It does not asymptotically correct more errors, and since no other algorithm correcting more errors for Goppa codes is known, it is not known whether these codes correct asymptotically more than a $\delta(n)$-fraction of errors with some failure probability bound away from 1.

Certain Goppa codes meet the Gilbert Varshamov bound (see [MS1977, p. 350]), so in particular there exists some $\alpha > 0$ and an infinite subfamily $\mathcal{F}' \subseteq \mathcal{F}$ such that the relative minimum distance of each member of $\mathcal{F}'$ is at least $\alpha$.

This shows that the family $\mathcal{F}$ is asymptotically good, while yet it is not known to correct a constant fraction of errors with a non-trivial error-probability, given that $\delta(n) \longrightarrow 0$ with $n \longrightarrow \infty$. Even $\mathcal{F}'$ is not known to correct a constant fraction of errors.

## 2.2 Low weight word finding algorithms

In this section, we will examine probabilistic algorithms to find low weight words in a general linear code. The problem of finding minimum weight words is closely related to the decoding problem; most of the algorithms for finding minimum weight words can be easily modified to solve a decoding problem and vice versa: Any generic minimum weight word finding algorithm can be transformed to a generic $< d/2$-distance decoding algorithm, using the following approach, due to Chabaud [C1995].

Let $\mathcal{C}$ be an $[n, k, d]$-code, and assume $x \notin \mathcal{C}$ is a word in $\mathbb{F}^n$ with $d(x, \mathcal{C}) < d/2$. Write $x = y + e$ with $y \in \mathcal{C}$ and $\mathrm{wt}(e) < d/2$. Now, note that the linear code $\mathcal{C}' := \mathcal{C} \oplus \mathbb{F}x$ (where $\oplus$ denotes the direct sum of vector spaces) contains $e$, and that $e$ is the unique minimum weight word in $\mathcal{C}'$. So, since finding the error pattern is enough to decode, decoding can be done by constructing $\mathcal{C}'$ and then finding the minimum weight word in $\mathcal{C}'$, and this will solve the decoding problem.

Interestingly, a randomized inverse conversion is also possible, although distance $d$ decoding is needed, and not just distance $(d/2)$ decoding. To do this, pick any $x \in \mathcal{C}$ and construct $\mathcal{C}'$ such that $\mathcal{C} = \mathcal{C}' \oplus \mathbb{F}x$. Then decode $x$ in $\mathcal{C}'$, i.e., find $y$ and $e$ such that

$$x = yG' + e,$$

and $e$ is of minimum weight. If there is a minimum weight word in $\mathcal{C}$ which is also in $x - \mathcal{C}'$, then $e$ is such a minimum weight word. This happens with probability at least $1 - 1/|\mathbb{F}| \geq 1/2$.

### 2.2.1 Information set decoding

We now study the low weight word finding algorithm which is known as the *information set decoding* algorithm, since it is often used as a decoder. This algorithm is the basis of the most powerful low weight word finding algorithms known to date.

The working principle is as follows: Assume we work in an $[n, k]$-code $\mathcal{C}$ with generator matrix $G$, and we seek a word of weight at most $w_0$. A random set $I$ of $k$ among $n$ positions is chosen, the *information set (IS)*. A new generator matrix $G'$ is constructed by diagonalizing $G$ on $I$ using elementary row operations. (It is possible to select $I$ in a way so that such a diagonalization can always be performed.)

Each row of $G'$ is a codeword in the code, and has weight at most $n - k + 1$. The weight of each row is computed, and if a row of weight at most $w_0$ is found, then this row is returned, otherwise the algorithm fails (that is, one has to try with a new information set).

Information set decoding can be crudely described as a variation of the algorithm that randomly samples codewords, until one of suitable weight is found. The key improvement, though, is that the sampling is biased towards low weight words. It is obvious that the sampled codewords are by construction of weight at most $n - k + 1$. We will now illustrate that, in addition, sampling is not uniform among those low weight words, but that there is a bias towards lowest weight words.

The probability that a fixed word of weight $w$ shows up as a row in the diagonalized matrix is

$$\frac{\binom{k}{1}\binom{n-k}{w-1}}{\binom{n}{w}}. \tag{2.5}$$

Writing $k = Rn$, $w = \omega n$, dropping the term $\binom{k}{1}$ and approximating the binomial coefficients, we obtain the asymptotic (in $n$) approximation of this probability, which is

$$2^{n\left[(1-R)H_2\left(\frac{\omega}{1-R}\right) - H_2(\omega)\right]}. \tag{2.6}$$

Computing the derivative of the exponent with respect to $\omega$ shows that the response decreases monotonically with increasing $\omega$.

Note that (2.6) is the probability that a *single* word of the given weight matches; if a larger number of matching words exist, then a match is much easier to find. For example, if there are $\Delta$ acceptable target words, the probability of finding any one of them is roughly $\Delta$ times (2.6), if $\Delta$ is not too large.

### 2.2.2 Example application: graph codes

While the estimate (2.6) clearly shows that the constant rate, constant relative weight setting leads to an exponential running time of the information set decoder, it is important to also study the fringe cases which are encountered, when non-random codes are used.

We will now look at one such special case, the case where the sought weight of the word is a constant independent of $n$.

LDPC codes are codes that have a sparse parity check matrix. The parity check matrix is usually random, with the row- and column-weights selected according to some so-called *degree distribution*, which is fixed independently of $n$. The weight of each such parity check is thus $O(1)$. The key point of the decoders for such codes is that low weight parity checks contain a lot of information about likelihood of errors on the positions on which they are nonzero. See, for example [S2002], for a detailed description of LDPC decoding algorithms.

Now consider the following problem: you are given a generator matrix for an LDPC code $\mathcal{C}$ of parameters $[n, k]$. Find a set of $n-k$ sparse linearly independent parity checks.

A possibility to solve this problem is to apply a low weight word finding algorithm on $\mathcal{C}^{\perp}$ to find these sparse checks.

The largest weight of a parity check of the canonical parity check matrix of an LDPC code depends on the selected degree distribution, and therefore the difficulty of finding a complete set of low weight parity checks from the same ensemble as the original code depends largely on the selected degree distribution.

For concreteness, we consider the example of a rate 1/2 code. A possible degree distribution for this setting is the one suggested by LDPC-opt [U2001], which suggests a degree distribution with the maximal weight of a check being 20. If we take the case of a block length 1000, and assuming there is just a single weight 20 word in the dual code, equation (2.5) shows that finding it would cost roughly $2^{16}$ diagonalizations of the parity check matrix with information set decoding.

Increasing the block length $n$ increases the difficulty of finding constant weight words only insofar as the involved linear system becomes larger: The equation (2.5) can be approximated by

$$R(1-R)^{w-1} \tag{2.7}$$

for the constant weight case, showing that recovering sparse parity checks in such codes is a polynomial time problem, with the degree of the polynomial depending on the degree distribution functions.

### 2.2.3 Improved versions of information set decoding

Various better versions of the information set decoding algorithms have been proposed over the last two decades. In particular, the following improvements were suggested:

- **Allow for low weight error-patterns in the information set.** In the basic information set decoding algorithm, a weight $w$ word is matched whenever it has exactly one 1 on the $k$ positions forming the information set and $w - 1$ ones on the $n - k$ remaining positions. For interesting values of $w$, however, the average weight of a weight $w$ word on a random information set is much larger than 1. The probability of a match can therefore be increased if we look for words of weight $\ell$ on the information set and $w - \ell$ on the remaining positions, for some small $\ell > 1$.

  This can be done by first diagonalizing the matrix on the information set, and then checking all possible linear combinations of at most $\ell$ rows of the diagonalized matrix.

  Doing this check is expensive, but the advantage is that fewer information sets have to be examined, and thus fewer diagonalizations are needed. In practice, $\ell = 2$ or 3.

  This is the basic idea of the algorithm of Lee and Brickell [LB1988].

- **Avoid operating on the complete generator matrix.** Since the weight of the sought vector is in general much less than the average weight of a row in the diagonalized generator matrix, and since most candidate vectors have to be rejected, an early abort strategy brings a notable speedup: The idea is to look just at a subset of the codeword positions, rather than looking at all of them, and to do the complete computations only for promising candidates. This idea appeared first in Leon's algorithm [L1988].

- **Use square-root methods in the IS error patterns.** It turns out that when a Lee-Brickell-like approach is used, it is not necessary to evaluate all the error patterns separately, but that, with small failure probability, an exhaustive evaluation only to half the acceptable weight is necessary, which reduces the cost of evaluating an information set. This improvement was first suggested by Stern [S1989].

- **Avoid to completely rebuild information sets from scratch.** Instead of performing a complete Gaussian elimination to get a new information set, it is much cheaper to just pivot on one position in the generator matrix, adding a new position into the information set, while removing another one from it. This new information set is not independent of the previous one, as it shares most of the positions with the old one, and so the matching probability of each information set will correlate with the matching probability of the previous one.

  The net effect is, however, that the pivoting technique is much faster than diagonalizing the generator matrix on a new information set for each iteration. This idea was proposed by van Tilburg in 1988, [vT1988], and proved to be an actual improvement by Canteaud and Chabaud [CC1998].

In addition to that, many of the algorithms exist in alternative formulations that operate on the parity check matrix rather than the generator matrix. It is often better to operate on the parity check matrix, for example, if $R > 1/2$.

We will not give a detailed account of these algorithms in this thesis; good summaries already exist, see e.g., [C1996] or [C1995]. It is interesting, however, to observe that none of these improved algorithms is known to have a better exponent than the basic information set decoder, i.e., it is possible that up to a polynomial factor, they all have the same running time. Indeed, only Stern's improvement in the above list could possibly result in a better exponent, if the number of information set error patterns could be chosen to be proportional to $k$.

Chabaud gives an asymptotic analysis of Stern's algorithm in [C1992]. In this analysis, only constant weight errors on the information set are considered. He also points out, however, that constant fraction error weights would lead to exponential memory use.

In practice, choosing very small error weights (at most 4) on the information set appears to lead to the best results, see e.g. the numbers given in [CC1998].

For this reason, we will generally just use the information set decoding success probability, when it comes to estimate the cost of finding low weight words with general decoding.

## 2.3 The McEliece cryptosystem

This thesis is concerned mainly with one particular application of codes in cryptography, namely the security of different variants of the McEliece cryptosystem [ME1978]. We will now study this cryptosystem.

### 2.3.1 Basic idea

The key idea of the McEliece cryptosystem is to identify encryption with encoding, and decryption with decoding, of a certain code.

In this aim, we want to disguise a decodable linear code in a way that the construction of the code can no longer be deduced. The point is that a general description of the linear code, as given by the generator matrix, makes it easy to encode, but decoding needs far better knowledge of the code. So, the disguised code could serve as public key, and the exact description as the private key.

Let $\mathcal{C}$ be an $[n, k]$ Goppa code over $\mathbb{F} = GF(2)$. Equation (2.4) asserts that these codes are $t$-error-correcting, with

$$t = \frac{n(1-R)}{\lceil \log_2(n) \rceil}.$$

Let $G$ be the $k \times n$ generator matrix of this Goppa code. Let $A$ be a $k \times k$ random invertible matrix with coefficients in $\mathbb{F}$. The matrix $AG$ is a generator matrix for the same code. Let $\sigma$ be a permutation over $\{1, 2, \ldots, n\}$, and let $P$ be the corresponding $n \times n$ permutation matrix. The matrix $G_{\text{pub}} := AGP$ is a generator matrix of $\mathcal{C}^\sigma$.

Note that we can apply any $t$-error-correcting decoder for $\mathcal{C}$ to decode $t$ errors for $\mathcal{C}^\sigma$; it is enough to undo the permutation prior to decoding.

The disguising procedure we just gave allows us to construct a public key cryptosystem as follows:

- **Public Key.** The matrix $G_{\text{pub}}$ and the integer $t$.
- **Encryption.** To encrypt the message $x = (x_1, \ldots, x_k) \in \mathbb{F}^k$ encode with $G_{\text{pub}}$, i.e., compute $y := xG_{\text{pub}}$, construct a random error vector $e$ of length $n$ and Hamming weight at most $t$. The cryptogram is then $y + e$.
- **Private Key.** The private key is the knowledge of the construction of $\mathcal{C}^\sigma$, in particular, the factorization $A$, $G$, $P$, and the $t$-error-correcting decoder for $\mathcal{C}$.
- **Decryption.** The decryption problem is the problem of decoding the linear code $\mathcal{C}^\sigma$. Since the encryptor added at most $t$ errors, the $t$-error-correcting code for $\mathcal{C}^\sigma$, deduced from the $t$-error correcting code for $\mathcal{C}$, can be used to solve this problem.

The original McEliece cryptosystem used binary Goppa codes, much in the way as we have just defined it. However, there is fundamentally nothing imposing that choice; A priori, any other family of linear codes that can correct many errors, can be used instead. We will call a cryptosystem that works just like the McEliece cryptosystem, but uses other codes, a *McEliece type cryptosystem.*

### 2.3.2 Hardness assumptions

There are two major hardness assumptions in McEliece type cryptosystems.

#### Hardness of general linear decoding

First, if a $t$-error-correcting $[n, k]$-code is used, it should be hard to correct $t$ errors in a general $[n, k]$ linear code given by some generator matrix $G_{\text{pub}}$. This problem has seen quite some research, and it appears that it is indeed very hard in general.

On the theoretical side, Berlekamp, McEliece and van Tilborg proved in 1978 that maximum likelihood decoding of linear codes is NP-hard in general [BMEvT1978]. Unfortunately, this result has little significance to assess the hardness in our setting, for a number of reasons:

1. The underlying codes in the McEliece system are not random, but instead specific $t$-error-correcting codes have been selected.

2. The maximum-likelihood decoding problem does not have to be solved in generality. Instead it is sufficient to decode words with distance up to $t$ from $\mathcal{C}^\sigma$.

3. In the cryptographic application, the average hardness of an attack is the correct measure. NP-hardness, however, is a worst-case criterion, and so it does not give the information we would need to assess the hardness of an attack.

More recent results improve on the original result by Berlekamp, McEliece and van Tilborg. For example, the Bruck and Naor showed in [BN1990] that even infinite preprocessing of the code does not render the decoding easy. This stronger result, however, still applies to the complete maximum likelihood decoding problem only, and not to bounded distance decoding. For the bounded decoding scenario, Dumer, Micciancio and Sudan have shown in [DMS2003] that in general, decoding is hard, if the maximum weight is $> 1/2d^{1+\varepsilon}$, where $d = d(n)$ is the minimum distance of the family. This does still not prove the conjecture that any constant fraction of errors is hard to correct in the fixed rate scenario.

Asides from complexity theoretic considerations, the problem can of course also be studied from a best-known-algorithm point of view.

As mentioned in section 2.2, the best known general decoding algorithms are refinements of information set decoding. So, in particular, the best known algorithms are fully exponential in the block length if both the rate, and the fraction of errors is constant. It should be noted, though, that most McEliece type cryptosystems are not in this setting. In particular, the example on page 19 shows that the original McEliece cryptosystem, based on Goppa codes does not satisfy this condition, and attacking it with direct decoding is thus subexponential.

In practice, the estimate (2.6) (or a variant thereof) is used to get an estimate of suitable code parameters, given a desired security level.

### The code reconstruction problem

The second hardness assumption of McEliece-type cryptosystems is that, given a generator matrix $G$, it is hard to reconstruct the underlying code. We call an attack that exploits the structure of the underlying code a *structural attack*.

The difficulty of this problem depends heavily on the family of codes that has been used in the construction. For some families, efficient attacks are known, while for others, the best currently known attack is still ineffective. For example, in the original McEliece cryptosystem, Goppa codes were proposed, and this choice is still not broken. On the other hand, our considerations in section 2.2.2 show that LDPC codes, for example, are not safe.

In this thesis, structural attacks against different families of algebraic codes will be discussed in detail in Chapter 3.

### 2.3.3 Variants of the McEliece cryptosystem

#### The Niederreiter variant

In 1986, Niederreiter, apparently unaware of McEliece's work, proposed a cryptosystem [N1986] that is based on the same hardness assumptions as the McEliece system. As a consequence, any structural attack on the Niederreiter system can be applied to the corresponding McEliece system, and vice versa [LDW1994]. The same holds for general decoding attacks.

It turns out, however, that Niederreiter's variant has a some advantages over the original McEliece system:

- It is not vulnerable to message resend attacks: when the same message is encrypted twice with the original McEliece cryptosystem, then this condition can be detected, since the two cryptograms $x$ and $y$ then have distance at most $2t$, and many likely error positions can be identified, namely every position in $\operatorname{supp}(x - y)$. The remaining errors are then much easier to guess. The Niederreiter-variant is not affected by this problem.

- It often suffers less from message expansion: A cryptogram in the McEliece system has length $1/R$ times the message size, where $R$ is the rate of the code, so for any fixed, nontrivial rate, there will be a fixed blow up in the message size, typically around 2. The Niederreiter system also has message expansion. In this case, the expansion factor is a function of the number of correctable errors divided by $(1 - R)n|\mathbb{F}|$, where $n$ is the block length of the code. The expansion factor is often smaller than the one seen in the corresponding McEliece cryptosystem, and we shall see that it is not inherent.

We will now quickly describe the Niederreiter cryptosystem. Let $\mathcal{C}$ be a $t$-error-correcting $[n, k]$ code, and let $H$ be an $(n-k) \times n$ parity check matrix of $\mathcal{C}$. Let $A$ be a random invertible $(n - k) \times (n - k)$ matrix, and let $\sigma$ be a permutation over $\{1, \ldots, n\}$ with $P$ the corresponding permutation matrix. Then

$$H_{\text{pub}} := AHP$$

is a parity check matrix for the $t$-error-correcting code $\mathcal{C}^\sigma$.

Let

$$\mathcal{M} := \{x \in \mathbb{F}^n \mid \operatorname{wt}(x) \le t\}$$

be the *message space*. Then the Niederreiter cryptosystem works as follows:

- **Public Key.** The public key is $H_{\text{pub}}$ and $t$.

- **Encryption.** The cryptogram of $m \in \mathcal{M}$ is $H_{\text{pub}}m^t$.
- **Private Key.** The matrices $A$, $H$, $P$, and the construction of $\mathcal{C}$. In particular, a $t$-error-decoder for $\mathcal{C}^\sigma$.
- **Decryption.** The cryptogram $s \in \mathbb{F}^{n-k}$ has been received. The following procedure decrypts $s$:
    1. Find an arbitrary solution $x \in \mathbb{F}^n$ to the linear equation system $H_{\text{pub}}x^t = s$. Note that now we have $x - m \in \mathcal{C}^\sigma$, for the yet unknown $m$. In particular, $d(x, \mathcal{C}^\sigma) \leq t$.
    2. Use the $t$-error-correcting decoder for $\mathcal{C}^\sigma$ to decode $x$ into $y \in \mathcal{C}^\sigma$.
    3. Return $x - y$.

Given the equivalence, when it comes to structural attacks of the McEliece and the Niederreiter cryptosystem, we will in the sequel limit ourselves to analyzing the McEliece system, even though the Niederreiter variant may turn out to be more efficient in practice.

We will terminate the discussion of the Niederreiter variant by showing that, asymptotically, the Niederreiter system does not suffer from message expansion if capacity-achieving codes are used. This fact is insofar surprising, as it suggests that message expansion may not be intrinsic to code-based cryptosystems.

Recall that the capacity of the binary symmetric channel with bit-flipping probability $p$ is $1 - H_2(p)$. Thus if $\mathcal{C}$ is from a capacity-achieving family of rate $1 - H_2(p)$, it can correct a $p - o(1)$-fraction of errors with the failure-probability of the decoder tending to 0 as $n \to \infty$ (the $o(\cdot)$ is also with respect to $n$). Thus we can pick $t(n) = n(p - o(1))$, giving a message space of size

$$\sum_{i=0}^{\lfloor n(p-o(1)) \rfloor} \binom{n}{i} \approx 2^{nH_2(p-o(1))},$$

holding an $nh(p - o(1))$-bit message. The messages are of size $n - k = n(1 - (1 - H_2(p))) = nH_2(p)$. Thus the expansion factor is

$$\frac{H_2(p)}{H_2(p - o(1))} \to 1,$$

as desired.

It should be noted, however, that no capacity-achieving family of codes that is usable for a secure McEliece system is known, at this point: all capacity-achieving families with known suitable decoders to date have been broken: Concatenated codes have been broken by Sendrier [S1994], LDPC codes are broken given in section 2.2.2, and so are expander codes by the same reasoning.

**The subcoding trick**

McEliece's original paper proposes two disguising techniques: changing the basis, and permuting codeword positions. However, implicitly, a third disguising technique is used, which we will refer to as the *subcoding trick.*

If $\mathcal{C}$ is a $t$-error-correcting linear code, and $\mathcal{C}' \subseteq \mathcal{C}$ is a linear subcode of $\mathcal{C}$, then the decoder for $\mathcal{C}$ can also be applied to $\mathcal{C}'$ to decode $t$ errors; the cost is a loss in the transmission rate. For example, Goppa codes are subfield subcodes of generalized Reed-Solomon codes, and while the structure of Reed-Solomon codes has been broken a long time ago [SS1992], Goppa codes have remained unaffected by this result.

It should be noted, however, that binary Goppa codes have at least twice the minimum distance than what the lower bound that follows from the subcode argument proves, and the decoder due to Patterson [P1975] can also correct accordingly more errors; so McEliece's construction is not a pure subcode construction, but makes also use of an improved decoder.

McEliece in his original paper suggests using a $[1024, 524, 101]$ binary Goppa code, a code that is a subfield subcode of a generalized Reed-Solomon code over $GF(2^{10})$ of parameters $[1024, 974, 51]$. Thus the procedure to take the subcode causes the dimension to drop by 450, which is a rather large number.

It is of course instructive to study the question of how much the dimension has to drop in order for the code to become safe. In the specific case of subcodes of Reed-Solomon codes, we will study the question in section 3.1.

## 2.4 Families of algebraic codes

In this section, we will discuss two families of algebraic codes: Reed-Muller codes and algebraic geometry codes.

### 2.4.1 Reed-Muller codes

Reed-Muller codes are a very old and well-known family of codes. We will briefly present their construction, along with the properties that we will need. For further details, see [HP2003] or [MS1977].

**Construction**

We will construct Reed-Muller codes by the means of Boolean functions. A *Boolean function in the variables* $v_1, \dots, v_m$ is a member of the ring

$$\mathcal{B}(\{v_1, \dots, v_m\}) := \mathbb{F}_2[v_1, \dots, v_m]/(v_1^2 - v_1, \dots, v_m^2 - v_m),$$

where $\mathbb{F}_2 = GF(2)$ is the binary field. The idea is that the variables $v_i$ only ever take values in $\mathbb{F}_2$, and on these positions, we have $v_i^2 = v_i$, which is why the polynomials $v_i^2 - v_i$ are modded out. The *degree* of a Boolean function

$f \in \mathcal{B}(\{v_1, \ldots, v_m\})$ is defined to be the degree of its lowest-degree representative in $\mathbb{F}_2[v_1, \ldots, v_m]$, and is denoted by $\deg(f)$. Define

$$\mathcal{B}(r, \{v_1, \ldots, v_m\}) := \{f \in \mathcal{B}(\{v_1, \ldots, v_m\}) \mid \deg(f) \leq r\}$$

the $\mathbb{F}_2$-vector space of all Boolean functions of degree $\leq r$. By counting the monomials, we see that if $r \leq m$, we have

$$\dim(\mathcal{B}(r, \{v_1, \ldots, v_m\})) = \sum_{i=0}^{r} \binom{m}{i}. \tag{2.8}$$

Reed-Muller codes can be constructed with the help of Boolean functions. Since the variables in Boolean functions take values in $\{0, 1\}$, a Boolean function in $m$ variables can be evaluated on $2^m$ different positions. So to each Boolean function we can associate a binary word of length $2^m$, obtained by evaluating the function on each possible position. The code $\mathcal{R}(r, m)$ (where $0 \leq r \leq m$) is the set of words obtained by evaluating all the Boolean functions in $\mathcal{B}(r, \{v_1, \ldots, v_m\})$ in this way.

Since $\mathcal{B}(r, \{v_1, \ldots, v_m\})$ is a vector space and the evaluation map is linear, it follows that $\mathcal{R}(r, m)$ is a linear code. By induction on $r$, it is easily shown that the evaluation map has a trivial kernel, and therefore, using (2.8), we can conclude that this code has parameters

$$n = 2^m, \qquad k = \sum_{i=0}^{r} \binom{m}{i}.$$

The one-to-one correspondence of codewords in $\mathcal{R}(r, m)$ with functions in $\mathcal{B}(r, \{v_1, \ldots, v_m\})$ proves extraordinarily useful, and we will therefore often identify codewords of $\mathcal{R}(r, m)$ and with functions in $\mathcal{B}(r, \{v_1, \ldots, v_m\})$.

Figure 2.1 shows canonical codewords of the $\mathcal{R}(\cdot, 4)$ codes: In the matrix, the first row forms a generator matrix of the $\mathcal{R}(0, 4)$-code, The first five rows form a generator matrix for $\mathcal{R}(1, 4)$, and so on.

By induction on $r$, one can show that the minimum distance of $\mathcal{R}(r, m)$ is

$$d = 2^{m-r}.$$

The fact that all functions generating words in $\mathcal{B}(r - 1, \{v_1, \ldots, v_m\})$ are also in $\mathcal{B}(r, \{v_1, \ldots, v_m\})$ implies the following observation.

**Proposition 1** *For any $m$, we have $\mathcal{R}(0, m) \subset \mathcal{R}(1, m) \subset \cdots \subset \mathcal{R}(m, m)$.*

Another useful property we need is the observation that the dual code of a Reed-Muller code is also a Reed-Muller code, more precisely

$$\mathcal{R}(r, m)^{\perp} = \mathcal{R}(m - r - 1, m). \tag{2.9}$$

| function | corresponding codeword | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $v_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $v_3$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $v_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $v_1$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $v_3v_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $v_2v_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $v_1v_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $v_2v_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| $v_1v_3$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| $v_1v_2$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $v_2v_3v_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| $v_1v_3v_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| $v_1v_2v_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $v_1v_2v_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| $v_1v_2v_3v_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Figure 2.1: Codewords in the Reed-Muller code

### Minimum weight words in Reed-Muller codes

The fact that products of $r$ linearly independent degree one functions result in minimum weight codewords of $\mathcal{R}(r, m)$ is well-known. The following proposition states the converse, namely, that minimum weight codewords in Reed-Muller codes can always be written as a (pointwise) product of suitable words in the corresponding first order code. In other words, the only functions giving rise to minimum weight codewords are products of functions in $\mathcal{B}(1, \{v_1, \ldots, v_m\})$.

**Proposition 2** *Let $f \in \mathcal{R}(r, m)$ be a word of minimum weight. Then there exist $f_1, f_2, \ldots, f_r \in \mathcal{R}(1, m)$, such that*

$$f = f_1 \cdot f_2 \cdots f_r,$$

*where $\cdot$ denotes the componentwise product. The $f_i$ are of minimum weight in $\mathcal{R}(1, m)$.*

Proposition 2 is proved in [KT1970]. The same paper also gives more precise formulas for the weight distribution, which can be used, in particular, to estimate the number of minimum weight words:

**Proposition 3** *There are at least*

$$2^{mr-r(r-1)}.$$

*minimum weight codewords in $\mathcal{R}(r, m)$.*

### 2.4.2 Generalized Reed-Solomon codes

Generalized Reed-Solomon codes can be seen as a subclass of algebraic geometry codes, defined on curves of genus 0. We start by defining them in a more conventional way.

Let $\mathbb{F}$ be a finite field, $n$ and $k$ integers such that

$$0 \le k \le n \le |\mathbb{F}|.$$

Pick distinct elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$. Let $c_1, \ldots, c_n \in \mathbb{F}^*$. The $c_i$ are not necessarily distinct. We write

$$\mathbb{F}[X]_{<k}$$

for the $\mathbb{F}$-vector space consisting of the polynomials in $\mathbb{F}[X]$ of degree smaller than $k$. This vector space has dimension $k$. We can map elements from $\mathbb{F}[X]_{<k}$ into $\mathbb{F}^n$ via the homomorphism

$$f \mapsto (c_1 f(\alpha_1), \ldots, c_n f(\alpha_n)).$$

This homomorphism is injective, and thus $f(\mathbb{F}[X]_{<k})$ is a $k$-dimensional subspace of $\mathbb{F}^n$, i.e., it is an $[n, k]$-code.

This code is the *generalized Reed-Solomon code* of parameters $\alpha_1, \ldots, \alpha_n$ and $c_1, \ldots, c_n$ and dimension $k$. We denote this code by

$$\mathrm{GRS}_{\mathbb{F}}(k, (\alpha_1, \ldots, \alpha_n), (c_1, \ldots, c_n)).$$

Generalized Reed-Solomon codes are maximum distance separable, i.e., their minimum distance is

$$d = n - k + 1,$$

which is best possible. This property follows directly from the fundamental theorem of algebra.

We will sometimes use the abbreviation

$$\mathrm{GRS}_{\mathbb{F}}(k, (\alpha_1, \ldots, \alpha_n)) = \mathrm{GRS}_{\mathbb{F}}(k, (\alpha_1, \ldots, \alpha_n), (1, \ldots, 1)).$$

For the purpose of this text it is convenient to call generalized Reed-Solomon codes with $c_i = 1$ for $1 \le i \le n$ *ordinary* Reed-Solomon codes, but it should be noted that this is not standard terminology.

### 2.4.3 Algebraic geometry codes

Reed-Solomon codes have the constraint that their block length is limited by the alphabet size; the problem is more precisely, that a polynomial in $\mathbb{F}[X]$ cannot be evaluated on more than $|\mathbb{F}| + 1$ positions (the extra position corresponding to $\infty$), and that thus for larger block lengths, larger alphabets are needed.

In order to alleviate this problem, algebraic geometry (AG) codes have been invented by Goppa in 1977, [G1977]. The basic underlying observation is that algebraic curves defined over $\mathbb{F}$ can have more than $|\mathbb{F}| + 1$ points, which makes it possible to define codes with larger block lengths. As we will see, generalized Reed Solomon codes form the natural subclass of algebraic geometry codes that is obtained by taking lines as the curves to define the codes on.

## Notions of algebraic geometry

We will now give the notations and summarize the definitions and results from algebraic geometry that are needed to define these codes. The book by Huffman and Pless [HP2003] gives a slightly more detailed treatment of the subject concentrating on the theory needed to understand the definition of algebraic geometry codes. This book also contains a list of useful references for more in-depth treatments on the subject. We will frequently also make reference to [H2003] and [S1986].

For a field $\mathbb{F}$, we denote $\overline{\mathbb{F}}$ its algebraic closure. The *affine space* $\mathbb{A}^n(\overline{\mathbb{F}})$ is simply the set $\overline{\mathbb{F}}^n$. This space is a plane if $n = 2$. We denote points in $\mathbb{A}^2(\overline{\mathbb{F}})$ in the usual vector notation $(x, y)$. Variables representing abscissa and ordinates in this space are the lowercase letters $x$ and $y$.

An *affine algebraic curve* is the zero locus of some polynomial $f \in \overline{\mathbb{F}}[x, y]$. For example, the unit circle
$$x^2 + y^2 = 1$$
is such a curve.

The *projective space* $\mathbb{P}^n(\overline{\mathbb{F}})$ is the set
$$\left(\overline{\mathbb{F}}^{n+1} \setminus \{(0, \dots, 0)\}\right) / \sim,$$
where $(X_0, X_1, \dots, X_n) \sim (X_0', X_1', \dots, X_n')$ if and only if there exists $\lambda \in \overline{\mathbb{F}}^*$ such that $(X_0, \dots, X_n) = \lambda(X_0', \dots, X_n')$. We will be mostly concerned with the projective plane $\mathbb{P}^2(\overline{\mathbb{F}})$. The equivalence class of $(X, Y, Z)$ is denoted $(X : Y : Z)$. In this text, coordinate variables in the projective plane are denoted by the uppercase letters $X$, $Y$ and $Z$.

Note that arbitrary polynomials on $\mathbb{P}^2(\overline{\mathbb{F}})$ do not take well defined values. However, the zero set of homogeneous polynomials is well defined. A *projective algebraic curve* $\mathcal{X}$ in $\mathbb{P}^2(\overline{\mathbb{F}})$ is the zero set in $\mathbb{P}^2(\overline{\mathbb{F}})$ of some homogeneous polynomial $f \in \overline{\mathbb{F}}[X, Y, Z]$. A curve is said to be *defined over* $\mathbb{F}$ if the defining polynomial can be written with coefficients in $\mathbb{F}$. For example, the unit circle in $\mathbb{P}^2(\overline{\mathbb{F}})$ is given by
$$X^2 + Y^2 = Z^2$$
and is defined over $\mathbb{F}$. An affine curve given by $f(x, y)$ has an associated projective curve. The process of replacing $x$ by $X$ and $y$ by $Y$ in $f$, and multiplying each monomial of degree $d$ by $Z^{\deg(f)-d}$, so that a homogeneous polynomial in $X, Y$ and $Z$ is obtained is called the *homogenization of $f$ with respect to $Z$*. The converse operation, called *dehomogenization with respect to $Z$*, consists essentially of replacing $Z$ by 1. See [S1986, p. 13].

The projective curve has in general more points than the affine version; namely points with $Z = 0$, such points will be called *points at infinity.* It is often convenient to think of the curve as projective, but to do actual calculations in affine curves. We will regularly do this, and introduce special symbols (such as $\mathcal{O}$) for points at infinity.

A point on a curve is said to be $\mathbb{F}$*-rational* (or simply *rational*), if its projective coordinates can be written with all entries in $\mathbb{F}$. For a given curve $\mathcal{X}$, we denote by $\mathcal{X}(\mathbb{F})$ the set of $\mathbb{F}$-rational points of $\mathcal{X}$.

A planar curve $\mathcal{X}$ given by $F(X, Y, Z) = 0$ is said to be *irreducible* if $F$ is absolutely irreducible. An irreducible curve is said to be *smooth* if the partial derivatives of $F$ do not vanish simultaneously on any point of the curve.

For the purpose of this text, we shall always assume that the curve is irreducible and smooth.

If $\mathcal{X}$ is an irreducible projective curve in $\mathbb{P}^2(\overline{\mathbb{F}})$ given by $F(X, Y, Z) = 0$, we define the *rational functions* on $\mathcal{X}$ as follows:

$$\overline{\mathbb{F}}(\mathcal{X}) = \left( \left\{ \frac{f}{g} \;\middle|\; \begin{array}{l} f, g \text{ are homogeneous, } \deg(f) = \deg(g), \\ F \text{ does not divide } g. \end{array} \right\} \Big/ \sim \right) \cup \{0\},$$

where the equivalence relation $\sim$ is given by

$$(f, g) \sim (f', g') \iff fg' = f'g.$$

The rational functions define a field.

One can define the *order* of a nonzero rational function $f \in \overline{\mathbb{F}}(\mathcal{X})$ at a point $P \in \mathcal{X}$ in a fashion so that it behaves very similar to the usual order of rational functions in $\mathbb{F}(X)$. In particular, the order is zero if $f(P) \neq 0, \infty$, positive if $f(P) = 0$ and negative if $f(P) = \infty$. We will not give an exact definition of the order here (see [H2003, p. 169] or [S1986, p. 22]), but we note that it adheres to the usual rules, in particular,

$$\begin{aligned} \operatorname{ord}_P(f \cdot g) &= \operatorname{ord}_P(f) + \operatorname{ord}_P(g) \\ \operatorname{ord}_P(f + g) &\geq \min\{\operatorname{ord}_P(f), \operatorname{ord}_P(g)\}. \end{aligned}$$

The usual rational functions have the property that the sum of the orders over all values is zero, taking also into account the order at infinity. Rational functions over smooth algebraic curves satisfy the same remarkable identity, i.e.,

$$\sum_{P \in \mathcal{X}} \operatorname{ord}_P(f) = 0 \qquad \forall f \in \overline{\mathbb{F}}(\mathcal{X}). \tag{2.10}$$

See [H2003, Theorem 6.13] for details.

Unlike the usual rational functions, there are additional restrictions on what kinds of poles and zeros can occur. The divisor class group, which we will present now, characterizes the combinations that can occur.

**Divisor class group.** Let $\mathcal{X}$ be a smooth projective curve. A *divisor* is a formal sum of points of $\mathcal{X}$ with integer multiplicities that is finite (i.e., zero on almost all points). I.e.,

$$\sum_{P\in\mathcal{X}} n_P\langle P\rangle$$

is a divisor if $n_P = 0$ for almost every $P \in \mathcal{X}$. We use the notation $\langle\cdot\rangle$ to distinguish divisors from ordinary points. The set of divisors on $\mathcal{X}$ is denoted by $\mathrm{Div}(\mathcal{X})$.

The set of divisors form the free group generated by the elements $\langle P\rangle$ for $P \in \mathcal{X}$, and it is thus an Abelian group.

The *degree* of the divisor $\Delta = \sum_{P\in\mathcal{X}} n_P\langle P\rangle$ is defined to be the quantity

$$\deg(\Delta) := \sum_{P\in\mathcal{X}} n_P.$$

The *support* of the divisor $\Delta$, $\mathrm{supp}(\Delta)$, is the set of points on which it is nonzero, i.e.,

$$\mathrm{supp}(\Delta) := \{P \in \mathcal{X} \mid n_P \neq 0\}.$$

Let $f \in \overline{\mathbb{F}}(\mathcal{X})$ be a nonzero rational function. Then $\mathrm{ord}_P(f)$ is nonzero only for a finite number of points, and so

$$\mathrm{div}(f) := \sum_{P\in\mathcal{X}} \mathrm{ord}_P(f)\langle P\rangle$$

is a divisor, called the *divisor of $f$*. Any divisor which is the divisor of some rational function is said to be *principal.* The set of principal divisors is a subgroup of all divisors.

The statement (2.10) implies that any principal divisor has degree zero. The converse is not true in general: on curves of genus[1] $g > 0$, not all divisors of degree 0 are principal. Two divisors $\Delta_1$ and $\Delta_2$ are *linearly equivalent*, if their difference is a principal divisor. The *Degree* 0 *part of the divisor class group* of $\mathcal{X}$ is the group

$$\mathrm{Pic}^0(\mathcal{X}) := \text{Divisors of degree } 0/\text{Principal divisors}.$$

It is nontrivial if $g > 0$. A divisor $\Delta$ is said to be *defined over* $\mathbb{F}$ if it does not change with the action of $\mathrm{Aut}(\overline{\mathbb{F}}/\mathbb{F})$. Note that a divisor is defined over $\mathbb{F}$ in particular, if all its points are $\mathbb{F}$-rational. (The converse is not true, however.) Often we will consider only divisors defined over $\mathbb{F}$, and thus we will frequently work in the corresponding divisor class group:

$$\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X}) := \text{Classes of } \mathrm{Pic}^0(\mathcal{X}) \text{ that have a representative defined over } \mathbb{F}.$$

[1]We will not define the genus in this text. Lines have genus 0, and elliptic curves genus 1.

**Linear spaces.** We can define a partial ordering on divisors. Let $\Delta_1 := \sum_{P\in\mathcal{X}} n_P\langle P\rangle$ and $\Delta_2 := \sum_{P\in\mathcal{X}} m_P\langle P\rangle$. We define the ordering $\succeq$ by

$$\Delta_1 \succeq \Delta_2 \iff n_P \geq m_P \; \forall P \in \mathcal{X}.$$

We define the *linear space* of a divisor $\Delta$ as follows:

$$\mathcal{L}(\Delta) := \{f \in \overline{\mathbb{F}}(\mathcal{X}) \mid \operatorname{div}(f) + \Delta \succeq 0\} \cup \{0\}.$$

The linear space $\mathcal{L}(\Delta)$ is a $\overline{\mathbb{F}}$-vector space.

If $\deg(\Delta) < 0$, then we have $\mathcal{L}(\Delta) = \{0\}$, since $\operatorname{div}(f) + \Delta \succeq 0$ implies $0 = \deg(\operatorname{div}(f)) \geq \deg(-\Delta) > 0$. For the case that $\deg(\Delta) \geq 0$, the following version of the Riemann-Roch theorem [HP2003, Theorem 13.4.2] gives a bound on the dimension of $\Delta$.

**Theorem 1 (Riemann-Roch)** *Let $\Delta$ be a divisor on a smooth projective plane curve $\mathcal{X}$ over $\mathbb{F}$ of genus $g$. Then*

$$\dim(\mathcal{L}(\Delta)) \geq \deg(\Delta) + 1 - g.$$

*Moreover, if* $\deg(\Delta) > 2g - 2$*, the above inequality holds with equality.*

Another fact about linear spaces that we need is the assertion that if $\Delta$ is defined over $\mathbb{F}$ and $\mathcal{X}$ is defined over $\mathbb{F}$, then $\mathcal{L}(\Delta)$ has a basis in $\mathbb{F}(\mathcal{X})$ (see [S1986, Proposition 5.8]).

## The construction of AG codes

Let $\mathcal{X}$ be a smooth, irreducible curve defined over $\mathbb{F}$, and let $P_1, \dots, P_n$ be distinct, rational points on $\mathcal{X}$.

We consider the evaluation map $\operatorname{ev}_{P_1,\dots,P_n}$, which evaluates rational functions on the $P_i$, i.e.,

$$\begin{aligned} \operatorname{ev}_{P_1,\dots,P_n} : \overline{\mathbb{F}}(\mathcal{X}) &\to \overline{\mathbb{F}}^n \\ f &\mapsto (f(P_1), \dots, f(P_n)). \end{aligned}$$

Note that $\operatorname{ev}_{P_1,\dots,P_n}$ is $\overline{\mathbb{F}}$-linear.

Let $\Delta$ be a divisor defined over $\mathbb{F}$ such that $\operatorname{supp}(\Delta) \cap \{P_1, \dots, P_n\} = \emptyset$. We define the algebraic geometry code of divisor $\Delta$ and rational points $P_1, \dots, P_n$ over $\mathcal{X}$ as the image of $\mathcal{L}(\Delta) \cap \mathbb{F}(\mathcal{X})$ via $\operatorname{ev}_{P_1,\dots,P_n}$, i.e.,

$$\operatorname{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n)) = \operatorname{ev}_{P_1,\dots,P_n}(\mathcal{L}(\Delta) \cap \mathbb{F}(\mathcal{X})).$$

This set is clearly a code of block length $n$. It is defined over $\mathbb{F}$, since any $\mathbb{F}$-rational function takes a value in $\mathbb{F}$ on any $\mathbb{F}$-rational point. The code is linear by the linearity of the evaluation map. What are the dimension $k$ and minimum distance $d$ of this code?

**Proposition 4** *Let $\mathcal{X}$ be a smooth irreducible curve defined over $\mathbb{F}$ of genus $g$, $\Delta$ a divisor with $\deg(\Delta) > 2g - 2$ defined over $\mathbb{F}$, $\{P_1, \dots, P_n\}$ a set of $n$ distinct rational points. The code $\mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n)$ has dimension $k = \deg(\Delta) + 1 - g$ and minimum distance at least $d \geq n - \deg(\Delta)$.*

By the Riemann-Roch theorem, $\dim(\mathcal{L}(\Delta)) = \deg(\Delta) + 1 - g$, and, since the divisor is defined over $\mathbb{F}$, the $\mathbb{F}$-vector space $\mathcal{L}(\Delta) \cap \mathbb{F}(\mathcal{X})$ has also dimension $\deg(\Delta) + 1 - g$. Thus, if we show that $\mathrm{ev}_{P_1,\dots,P_n}|_{\mathcal{L}(\Delta)}$ is injective, we conclude that $k = \deg(\Delta) + 1 - g$.

To prove this last point, we show that any nonzero $f \in \mathcal{L}(\Delta)$ creates a word of weight at least $n - \deg(\Delta)$, which establishes the triviality of the kernel (and thus the fact that $k = \deg(\Delta) + 1 - g$) and the fact that the $d \geq n - \deg(\Delta)$ simultaneously.

Let $f \in \mathcal{L}(\Delta)$ such that $f(P_1) = \cdots = f(P_{\deg(\Delta)+1}) = 0$, say. Then we have that

$$f \in \mathcal{L}(\Delta - \langle P_1 \rangle - \cdots - \langle P_{\deg(\Delta)+1} \rangle).$$

Since the degree of the above divisor is $\deg(\Delta) - (\deg(\Delta) + 1) = -1 < 0$, we conclude that $f = 0$.

## Invariance properties of AG codes

It turns out that the description of a given algebraic geometry code is not unique, but that the same code can in general be obtained with several different descriptions. In this section we will devise two invariance properties of the construction which will prove useful later on.

**Choice of the divisor.** Recall that generalized Reed-Solomon codes can be obtained from ordinary Reed-Solomon codes by multiplying the coordinate positions by nonzero constants.

In fact, any code can be modified in that fashion, and the transformation has the property that it preserves decodability.

In our case, for a given smooth irreducible algebraic curve $\mathcal{X}$, divisor $\Delta$, rational points $P_1, \dots, P_n$, and elements $c_i \in \mathbb{F}^*$, we define the code

$$\mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n), (c_1, \dots, c_n))$$

as the code consisting of all words $(c_1 y_1, \dots, c_n y_n)$ such that

$$(y_1, \dots, y_n) \in \mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n)).$$

One may be tempted to call such a code a "generalized algebraic geometry code", but it turns out that these codes do not generalize on ordinary algebraic geometry codes. To see that, note that we can create a rational function $f$ such that $f(P_i) = c_i$ for $1 \leq i \leq n$, e.g., by using an analog of the Lagrange interpolation formula. Now, given such $f$, notice that

$$\mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n), (c_1, \dots, c_n)) = \mathrm{AGC}(\mathcal{X}, \Delta - \mathrm{div}(f), (P_1, \dots, P_n)),$$

because

$$\begin{aligned} \mathcal{L}(\Delta) &\to \mathcal{L}(\Delta - \operatorname{div}(f)) \\ g &\mapsto f \cdot g \end{aligned}$$

is an isomorphism of vector spaces. So, multiplication of coordinate positions by nonzero constants does not, in our case, generalize the class of codes.

However, it shows that many divisors generate the same code up to the multiplicative constants. More precisely, if $\Delta_1$ and $\Delta_2$ are two divisors, then

$$\mathrm{AGC}(\mathcal{X}, \Delta_1, (P_1, \ldots, P_n)) \text{ and } \mathrm{AGC}(\mathcal{X}, \Delta_2, (P_1, \ldots, P_n))$$

are the same up to multiplicative constants if $\Delta_1$ and $\Delta_2$ are linearly equivalent and $\operatorname{supp}(\Delta_1 - \Delta_2) \cap \{P_1, \ldots, P_n\} = \emptyset$.

**Curve isomorphisms.** We will now see that isomorphic curves define the same codes. Let $V_1$ and $V_2$ be two irreducible projective varieties in $\mathbb{P}^n(\overline{\mathbb{F}})$. A *rational map* $\phi : V_1 \dashrightarrow V_2$ is a map, where the component functions $f_0, \ldots, f_n$ are rational functions, i.e.

$$(X_0, \ldots, X_n) \mapsto (f_0(X_0, \ldots, X_n) : \cdots : f_n(X_0, \ldots, X_n)),$$

and such that $\phi(V_1) \subseteq V_2$. The map $\phi$ is regular at $P \in V_1$ if there exists $g \in \overline{\mathbb{F}}(V_1)$ such that $gf_i(P)$ is defined for $0 \leq i \leq n$ and not all of the $gf_i(P)$ are zero. If $\phi$ is regular at every $P \in V_1$, it is a *morphism.*

If $\phi : V_1 \to V_2$ is a morphism such that there exists a morphism $\psi : V_2 \to V_1$, such that $\phi \circ \psi = \mathrm{id}_{V_2}$ and $\psi \circ \phi = \mathrm{id}_{V_1}$, then $\phi$ is an *isomorphism.* If there is a representation $(f_0 : \cdots : f_n)$ of $\phi$ such that each $f_i$ is defined over $\mathbb{F}$, then $\phi$ is an $\mathbb{F}$*-isomorphism.* Note that $\mathbb{F}$-isomorphisms preserve the rational points of the varieties.

With that terminology, we claim the following: Let $\mathcal{X}_1$ and $\mathcal{X}_2$ be two irreducible, smooth projective curves defined over $\mathbb{F}$. If $\mathcal{X}_1$ is $\mathbb{F}$-isomorphic to $\mathcal{X}_2$, i.e., there is an $\mathbb{F}$-isomorphism $\phi : \mathcal{X}_2 \to \mathcal{X}_1$, then the codes defined over $\mathcal{X}_1$ are the same as those defined over $\mathcal{X}_2$.

Indeed, let

$$\mathcal{C}_1 := \mathrm{AGC}(\mathcal{X}_1, \Delta, (P_1, \ldots, P_n))$$

be a code defined over $\mathcal{X}_1$. Let

$$\mathcal{C}_2 := \mathrm{AGC}(\mathcal{X}_2, \phi^*(\Delta), (\phi^{-1}(P_1), \ldots, \phi^{-1}(P_n))),$$

where $\phi^*$ is the $\mathbb{Z}$-linear extension of the map that maps $\langle Q \rangle$ to $\langle \phi^{-1}(Q) \rangle$ for each $Q \in \mathcal{X}_1$.[2] Then $\mathcal{C}_1$ is the same code as $\mathcal{C}_2$, but it is defined over $\mathcal{X}_2$. Indeed,

[2]This definition of $\phi^*$ is only correct if $\phi$ is an isomorphism, see [S1986, p. 33] for the details.

let $f \in \mathcal{L}(\Delta)$. Then $g = (f \circ \phi) \in \mathcal{L}(\phi^*(\Delta))$ (see [S1986, Proposition 3.6]), and the codeword corresponding to $g$ in $\mathcal{C}_2$ is

$$(g(\phi^{-1}(P_1)), \ldots, g(\phi^{-1}(P_n))) = (f(P_1), \ldots, f(P_n)) \in \mathcal{C}_1.$$

Since both codes have the same dimension, this finishes the proof.

As a particular case of the above observation, if $\mathcal{X}$ is a curve, and $\phi$ is an isomorphism $\mathcal{X} \to \mathcal{X}$ then that isomorphism can be applied in the same way without changing the code.

**Example: Reed-Solomon codes**

It turns out to be instructive to see how Reed-Solomon codes can be seen as algebraic geometry codes, and to study the implications of the properties that we studied in the previous sections.

As we will see later on, the Sidelnikov-Shestakov attack [SS1992] on Reed-Solomon codes can be described rather naturally in this setting.

We start with an ordinary Reed-Solomon code

$$\mathrm{GRS}_{\mathbb{F}}(k, (\alpha_1, \ldots, \alpha_n))$$

Let $\mathcal{X}$ be the projective line $Y = 0$ in $\mathbb{P}^2(\overline{\mathbb{F}})$. The line $\mathcal{X}$ has $|\mathbb{F}| + 1$ rational points. Define $P_i$ for $1 \leq i \leq n$ by

$$P_i = (\alpha_i : 0 : 1).$$

We want to show that

$$\mathrm{GRS}_{\mathbb{F}}(k, (\alpha_1, \ldots, \alpha_n)) = \mathrm{AGC}(\mathcal{X}, (k-1)\langle\mathcal{O}\rangle, (P_1, \ldots, P_n)),$$

where $\mathcal{O} = (1 : 0 : 0)$ is the point at infinity on $\mathcal{X}$. First, note that the Riemann-Roch theorem implies that both codes have the same dimension, so it is enough to show that any codeword in the Reed-Solomon code is also a codeword in the given algebraic geometry code.

Let $f \in \mathbb{F}[X]_{<k}$, then $(f(\alpha_1), \ldots, f(\alpha_n))$ is a codeword in the Reed-Solomon code. Write $f(x) = f_0 + f_1 x + \cdots + f_\ell x^\ell$, where $\ell = \deg(f)$. Let

$$\hat{f}(X, Y, Z) = \frac{f_0 Z^\ell + f_1 X Z^{\ell-1} + \cdots + f_\ell X^\ell}{Z^\ell}$$

be the corresponding rational function. The function $\hat{f}$ has a pole of order at most $k-1$ at $\mathcal{O}$ and no other poles, thus

$$\mathrm{div}(\hat{f}) \succeq -(k-1)\langle\mathcal{O}\rangle,$$

and so $(\hat{f}(P_1), \ldots, \hat{f}(P_n))$ is a codeword of the algebraic geometry code. Now, by inserting the $P_i$, we see that this is the same word as $(f(\alpha_1), \ldots, f(\alpha_n))$.

Our reasoning on invariants of an algebraic geometry code allow us to directly conclude that generalized Reed-Solomon codes are also just a subclass of algebraic geometry codes.

**Example: Elliptic codes**

Consider the curve defined by the projective solutions to the Weierstrass equation,

$$E : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3, \tag{2.11}$$

where the $a_i \in \overline{\mathbb{F}}$ are parameters of the curve.

If the parameters $a_1, a_2, a_3, a_4$ and $a_6$ are such that the curve is smooth, then it is an *elliptic curve*. Elliptic curves have genus $g = 1$, and in fact, any planar, smooth curve of genus 1 is an elliptic curve.

Elliptic codes are a special class of algebraic geometry codes, with the underlying algebraic curve being an elliptic curve.

By Proposition 4, we get $d \geq n - k$, so, assuming this bound is tight, the minimum distance is one less than what we could get with the Singleton bound. Since Reed-Solomon codes are maximum distance separable, this obviously begs for the question why one would want to define codes over elliptic curves rather than just lines, given that the latter have better distance.

The answer is that elliptic curves can have more rational points than lines, and therefore they allow for codes of longer block lengths.

**Theorem 2 (Hasse)** *Let $E$ be an elliptic curve defined over $\mathbb{F}$, with $q = |\mathbb{F}|$. Then*

$$(q+1) - 2\sqrt{q} \leq |E(\mathbb{F})| \leq (q+1) + 2\sqrt{q}.$$

If $q$ is a prime, then the Hasse bound is tight; a formula due to Deuring [D1941] can be used to show that any integer order in the Hasse-interval is attained by at least one elliptic curve in this case, see [L1987].

This leads to an efficient construction of elliptic codes of block length at least

$$q + 2\lfloor\sqrt{q}\rfloor.$$

where the field size $q$ is a prime.

Since we will study cryptosystems based on elliptic curve codes more in detail, we will now recall the basic facts and notations of the divisor class group of elliptic curves which we will use extensively in the sequel. Any class in $\mathrm{Pic}^0(E)$ has a unique representant of the form

$$\langle P\rangle - \langle\mathcal{O}\rangle,$$

where $\mathcal{O}$ is the unique point at infinity of $E$. Therefore $E$ can be given the structure of the Abelian group $\mathrm{Pic}^0(E)$ via the bijection

$$\text{divisor class of } \langle P\rangle - \langle\mathcal{O}\rangle \mapsto P.$$

For the set of rational points, $E(\mathbb{F})$, a very similar assertion can be made. The same identification will define a bijection of $E(\mathbb{F})$ and $\mathrm{Pic}^0_{\mathbb{F}}(E)$ in this case. (See

[S1986, Exercise 2.13 and Remark 3.5.1].) The group operation on $E(\mathbb{F})$ is noted additively.

The application sum : $\mathrm{Div}(E) \to E$ is the $\mathbb{Z}$-linear extension of the map that maps $\langle P\rangle \mapsto P$ for every $P \in E$. Note that with these notations, we have in particular the following.

**Proposition 5** *Let* $\Delta \in \mathrm{Div}(\mathcal{X})$. *Then*

$$\Delta \textit{ is principal} \iff \deg(\Delta) = 0 \textit{ and } \mathrm{sum}(\Delta) = \mathcal{O}.$$

An important consequence of Proposition 5 is the following. Let $P, Q, R \in E$. Then

$$P,\ Q \text{ and } R \text{ are collinear} \iff P + Q + R = \mathcal{O}, \tag{2.12}$$

since this is equivalent to the statement that $\langle P\rangle + \langle Q\rangle + \langle R\rangle - 3\langle\mathcal{O}\rangle$ is a principal divisor, which can be the case only if there is a polynomial function passing through $P$, $Q$, $R$ and no other points of $E$. Such a polynomial function can only be a line. (This last conclusion follows from Bezout's theorem, i.e., Theorem 4 in section 5.1.2.)

Note that (2.12) gives a geometric definition of the group law over $E$.

# Chapter 3

# An attack against elliptic codes

In this chapter, we present a structural attack against the McEliece type cryptosystem based on elliptic codes, i.e., on algebraic geometry codes defined on elliptic curves. The attack is structural and recovers a private key from a given public key. In section 3.1, we set the stage for this new attack against elliptic codes by giving a modified presentation of the well-known Sidelnikov-Shestakov attack.

Section 3.2 is devoted to the attack on elliptic codes. As indicated by the theoretical analysis in section 3.2.7, the algorithm runs in heuristic polynomial time. The experimental running times discussed in section 3.2.8 show that this attack is practical even for keys that are several megabits large, which is larger than any parameter size that has ever been proposed for use in the McEliece cryptosystem.

Some of the ideas in this chapter, in particular section 3.2.3, are based on previous unpublished work by Daniel Bleichenbacher, Karin Melnick and Amin Shokrollahi.

## 3.1 The Sidelnikov-Shestakov attack revisited

In 1986, Niederreiter proposed a McEliece type cryptosystem that would use generalized Reed-Solomon codes [N1986]. A few years later, Sidelnikov and Shestakov published a paper [SS1992] presenting an effective deterministic polynomial time attack against this cryptosystem. This attack was a ground-breaking achievement in the cryptanalysis of McEliece type cryptosystems: it was indeed the first known effective structural attack against any such system based on algebraic codes.

Some of the ideas of this original method are recurrent in the attacks on other codes, and they reveal a great deal of the structural weaknesses that can be found in several McEliece type cryptosystems. We tailor our presentation to that need, focusing on the concepts that can be adapted to work on other codes, in particular elliptic codes. There are two key contributing factors. First, we interpret the birational transforms of the original paper in a way that is less ad hoc than the presentation provided by the original paper. As we will see later, the key point of the birational transforms is that they are also $\mathbb{F}$-isomorphisms, and that they integrate nicely into our observations from section 2.4.3. Second, we use minimum weight codewords in the most explicit manner.

This section will be closed by a brief analysis providing some evidence that sufficiently small subcodes cannot be tackled with this method, baring significant new ideas.

### 3.1.1 Geometric observations

It is a well known fact that smooth curves of genus 0 (and thus in particular lines) are isomorphic to $\mathbb{P}^1(\overline{\mathbb{F}})$ (See [H2003, p. 192]). Therefore, algebraic geometry codes over smooth curves of genus 0 are just Reed-Solomon codes. More to the point,

$$\mathrm{GRS}_{\mathbb{F}}(k,(\alpha_1,\ldots,\alpha_n)) = \mathrm{AGC}(\mathbb{P}^1(\mathbb{F}),(k-1)\langle\mathcal{O}\rangle,(P_1,\ldots,P_n)),$$

where $P_i = (\alpha_i : 1)$ and $\mathcal{O} = (1 : 0)$.

We start by studying the code invariance properties from section 2.4.3 in the setting of Reed-Solomon codes.

**Curve isomorphisms.** Even though we just saw that the curve is essentially unique in this case, it turns out to be useful to study curve isomorphisms of $\mathbb{P}^1(\mathbb{F})$ onto itself.

It is clear that

$$\begin{aligned} \mathbb{P}^1(\overline{\mathbb{F}}) &\to \mathbb{P}^1(\overline{\mathbb{F}}) \\ (X:1) &\mapsto (aX+b:1) \\ \mathcal{O} &\mapsto \mathcal{O}, \end{aligned}$$

where $a \in \mathbb{F}^*$, $b \in \mathbb{F}$ is an $\mathbb{F}$-isomorphism. The map

$$(X:Y) \mapsto (Y:X)$$

is also an $\mathbb{F}$-isomorphism: it is clearly bijective, and it is a birational transform that is defined everywhere, since it can be written

$$(X:Y) \mapsto \begin{cases} (Y/X:1) & \text{if } X \neq 0, \\ (1:X/Y) & \text{if } Y \neq 0. \end{cases}$$

By composition, we thus find that the map

$$\varphi : (X:Y) \mapsto (aX+bY : cX+dY)$$

with $a, b, c, d \in \mathbb{F}$ is an $\mathbb{F}$-isomorphism as long as it is bijective, i.e., as long as

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} \neq 0.$$

Since there are four parameters ($a, b, c$ and $d$), in a generic sense, the mapping is determined by fixing three images, i.e., given three distinct points $\hat{P}_1$, $\hat{P}_2$ and $\hat{P}_3 \in \mathbb{P}^1(\mathbb{F})$ and three distinct images $P_1$, $P_2$, $P_3 \in \mathbb{P}^1(\mathbb{F})$, there exists a unique transformation $\varphi$ of the above form with $\varphi(\hat{P}_i) = P_i$.

The cryptanalyst can take advantage of this fact in the following way: Let

$$\mathrm{AGC}(\mathbb{P}^1(\mathbb{F}), \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n))$$

be some arbitrary generalized Reed-Solomon code, and let $P_1, P_2, P_3 \in \mathbb{P}^1(\overline{\mathbb{F}})$ be distinct points that the cryptanalyst arbitrarily selects, then

$$\mathrm{AGC}(\mathbb{P}^1(\mathbb{F}), \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n)) = \mathrm{AGC}(\mathbb{P}^1(\mathbb{F}), \Delta, (P_1, P_2, P_3, \star, \dots, \star))$$

where $\Delta$ is a divisor, and $\star$ are points. In the following, denote those points by $P_4, P_5, \dots, P_n$.

**Divisor choice.** Since $\mathrm{Pic}^0(\mathbb{P}^1(\mathbb{F}))$ is trivial, we can arbitrarily fix the divisor, and get the same code up to multiplicative constants, i.e., if $Q$ is a point distinct from $P_1, \dots, P_n$, we know that $\mathrm{AGC}(\mathbb{P}^1(\mathbb{F}), \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n))$ is equal to

$$\mathrm{AGC}(\mathbb{P}^1(\mathbb{F}), (k-1)\langle Q \rangle, (P_1, \dots, P_n), (c_1, \dots, c_n)),$$

where the $c_i$ are unknown multiplicative constants.

### 3.1.2 Reconstructing Reed-Solomon codes

Fix three arbitrary but distinct points $P_1 = (\alpha_1 : 1), P_2 = (\alpha_2 : 1), P_3 = (\alpha_3 : 1)$ on $\mathbb{P}^1(\mathbb{F})$. The cryptanalyst is given a random basis of the code

$$\mathcal{C} := \mathrm{AGC}(\mathbb{P}^1(\mathbb{F}), (k-1)\langle Q \rangle, (P_1, P_2, P_3, P_4, \dots, P_n), (c_1, \dots, c_n)),$$

for known (that is, arbitrarily chosen) $P_1, P_2, P_3 \in \mathbb{P}^1(\mathbb{F})$. His task is to determine the unknowns $Q, P_4, \dots, P_n, c_1, \dots, c_n$.

He can proceed as follows. Determine two minimum weight words $v$ and $w$ of $\mathcal{C}$ which are nonzero in the first three coordinates, and such that they have only one zero located on a different coordinate. Since Reed-Solomon codes are maximum distance separable, this can be done by simply diagonalizing the generator matrix on some information set not containing the positions $1, 2$ and $3$. Without loss of generality, we assume the zeros of $v$ are given by

$$v_4 = \cdots = v_{k+2} = 0, \tag{3.1}$$

and those of $w$ by

$$w_4 = \cdots = w_{k+1} = w_{k+3} = 0.$$

Let $f_1 \in \mathcal{L}(-(k-1)\langle Q\rangle)$ be the function corresponding to $v$. Then, by (3.1), we have

$$\operatorname{div}(f_1) \succeq \langle P_4\rangle + \cdots + \langle P_{k+2}\rangle - (k-1)\langle Q\rangle,$$

and since the degree of the divisor on the right hand side is zero, we have equality. By an analogous argument,

$$\operatorname{div}(f_2) = \langle P_4\rangle + \cdots + \langle P_{k+1}\rangle + \langle P_{k+3}\rangle - (k-1)\langle Q\rangle,$$

and so

$$\operatorname{div}(f_1/f_2) = \langle P_{k+2}\rangle - \langle P_{k+3}\rangle.$$

Therefore

$$f_1/f_2 = \frac{aX + bZ}{cX + dZ},$$

for unknowns $a, b, c$ and $d$.

Since the words $v$ and $w$ are known, we can evaluate $f_1/f_2$ on the points $P_i$ where $w_i \neq 0$:

$$\frac{f_1}{f_2}(P_i) = \frac{c_i f_1(P_i)}{c_i f_2(P_i)} = \frac{v_i}{w_i}.$$

Thus, in particular, we can solve the homogeneous linear system of equations

$$\frac{f_1}{f_2}(P_i) = \frac{a\alpha_i + b}{c\alpha_i + d}, \qquad i = 1, 2, 3,$$

and therefore determine the unknowns $a, b, c$ and $d$.

Now let $1 \leq j \leq n$ be some position with $w_j \neq 0$ and such that $P_j$ (i.e., $\alpha_j$) is not yet known. To determine $\alpha_j$, we can then simply solve

$$\frac{f_1}{f_2}(P_j) = \frac{a\alpha_j + b}{c\alpha_j + d},$$

where the unknown is $\alpha_j$.

At this point, it is fairly easy to see how the cryptanalyst solves the complete problem: First all the $\alpha_i$ are found, for example by picking different pairs $(f_1, f_2)$, and repeating the procedure; then an arbitrary $Q$ is selected such that $Q \notin \{P_1, \ldots, P_n\}$. Then the constants $c_i$ have to be found. This problem is easy, it suffices to solve an appropriate linear system; we omit the details of this step at this point, since it is the same algorithm as the one studied in section 3.2.5.

### 3.1.3 Subcodes of Reed-Solomon codes

The above description makes it clear that the attack also works for subcodes $\mathcal{C}' \subseteq \mathcal{C}$ of Reed-Solomon codes if couples of minimum weight words of $\mathcal{C}$ can still be found in $\mathcal{C}'$. In the light of the subcoding trick as discussed in section 2.3.3, the question of whether and to what extent the attack can be generalized to subcodes of Reed-Solomon codes is obviously interesting.

We will consider random subcodes of a given Reed-Solomon code. Let

$$\mathcal{C} := \mathrm{GRS}_{\mathbb{F}}(k, (\alpha_1, \ldots, \alpha_n), (c_1, \ldots, c_n))$$

be an $[n, k, d]$-Reed Solomon code. Let $0 \leq \ell < k$ be some integer, and let $\mathcal{C}' \subseteq \mathcal{C}$ be a random $[n, k - \ell]$-subcode.

We will conservatively assume that finding words of weight close to $d$ is enough to break the system. Since couples of such words with only very few differing zeros are needed, the attack will be somewhat harder in reality.

For simplicity we will analyze the cost of finding words of weight exactly $d$, instead of just close to $d$. As we note below, an easy estimate for the other case is then easy to deduce.

Let $x \in \mathcal{C}'$ be a word obtained from diagonalizing on a random information set. Then by construction, $x$ has zeros on $k - 1 - \ell$ positions within this information set, say the positions $1, \ldots, k - 1 - \ell$.

Let $f \in \mathbb{F}[X]_{<k}$ be the polynomial corresponding to $x$. Then $f(\alpha_1) = \cdots = f(\alpha_{k-1-\ell}) = 0$, therefore

$$f(x) = (x - \alpha_1) \cdots (x - \alpha_{k-1-\ell}) g(x)$$

with $\deg(g(x)) \leq \ell$. Now $f$ will be of minimum weight if $g$ factors into a product of linear factors which all have zeros on the positions $\alpha_{k-\ell+1}, \ldots, \alpha_n$. The probability for this to happen is thus

$$\frac{\binom{n-k-\ell}{\ell}}{q^\ell} \approx \frac{1}{\ell!}(1 - R)^\ell \left(\frac{n}{q}\right)^\ell \tag{3.2}$$

for small $\ell$, where $q = |\mathbb{F}|$. This shows that already comparatively small values of $\ell$ are enough to render any attack that involves finding minimum weight words in $\mathcal{C}'$ impractical. For example, if $\ell = 20$ and $R = 1/2$, the above probability is less than $2^{-80}$.

Basically the same computation can be used to bound the probability that a given row in the diagonalized generator matrix is of weight at most $d + r$ for some $0 \leq r < \ell$. The result is that the dimension should be further reduced by $r$ for approximately the same success probability. For example, if $\ell = 24$, $R = 1/2$, then the probability that a random word considered by the information set decoder is of weight $\leq d + 4$ is at most $2^{-80}$.

Berger and Loidreau have previously studied the subcoding trick in [BL2005]. They suggest that very small values of $\ell$, such as $\ell = 4$, are sufficient to provide structural security. We believe, however, that such codes are not secure. We will provide a preliminary analysis of their proposal in section 5.2.

## 3.2 Elliptic codes

We have seen that trapdoors for algebraic geometry codes can be inverted if $g = 0$ using the Sidelnikov-Shestakov attack. In this section the next harder case will be studied, the case where $g = 1$. Elliptic codes are an interesting target for the cryptanalyst: First, elliptic codes lead to very good parameters. For example, our numbers in section 3.2.8 show that elliptic codes would lead to much better key sizes than most other constructions, including the original construction of McEliece, if they were not structurally weak. Second, it can be hoped that we will be able to generalize such attacks to larger families of algebraic geometry codes, and, eventually, to break the currently best known construction of McEliece type cryptosystems (see section 5.1.1). Second, minimum weight words are still trivial to find in such a code, since the elliptic codes are almost maximum distance separable. Hence, they provide a promising angle of attack.

The code reconstruction problem for elliptic codes can be stated as follows: we are given an arbitrary $k \times n$ generator matrix $G$ of an elliptic code $\mathcal{C}$, i.e.,

$$\mathcal{C} = \mathrm{AGC}(\hat{E}, \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n)),$$

where the elliptic curve $\hat{E}$, the divisor $\hat{\Delta}$ and the points $\hat{P}_1, \dots, \hat{P}_n$ are unknowns.

We want to find an elliptic curve $E$, a set of $n$ rational points $P_1, \dots, P_n$ on $E$, and a divisor $\Delta$ (on $E$) such that the code

$$\mathrm{AGC}(E, \Delta, (P_1, \dots, P_n), (c_1, \dots, c_n))$$

is equal the code $\mathcal{C}$ generated by $G$.

### 3.2.1 Code invariance

As we have done for Reed-Solomon codes, the first goal is to study the invariants of section 2.4.3 in our setting.

**Choice of the divisor.** We will first show that if

$$\gcd(k, |E(\mathbb{F})|) = 1, \tag{3.3}$$

then we can assume $\Delta = k\langle\mathcal{O}\rangle$. (Assuming (3.3) is not a problem. The value of $k$ can be controlled by shortening the code by a few positions, if required.)

Let $Q = \mathrm{sum}(\Delta)$. Equation (3.3) implies that there exists a point $R \in E(\mathbb{F})$ such that $[k]R = Q$. Now, since $\mathrm{sum}(\Delta) = Q = \mathrm{sum}(k\langle R\rangle)$ and $\deg(\Delta) = k = \deg(k\langle R\rangle)$, we have

$$\Delta \sim k\langle R\rangle.$$

Therefore, there exists a rational function $f$ such that $g \mapsto f \cdot g$ is an isomorphism $\mathcal{L}(\Delta) \to \mathcal{L}(k\langle R\rangle)$.

Now let

$$\begin{aligned} \tau : E &\to E \\ P &\mapsto P - R \end{aligned}$$

be the translation-by-$(-R)$ map; this map is an isomorphism which induces the isomorphism of linear spaces

$$\begin{aligned} \mathcal{L}(k\langle R\rangle) &\to \mathcal{L}(k\langle \mathcal{O}\rangle) \\ f &\mapsto f \circ \tau. \end{aligned}$$

By translating the $P_i$ accordingly, the evaluation map results again in the same codewords after translation, and we can thus set $\Delta = k\langle \mathcal{O}\rangle$.

**Isomorphisms of curves.** Let $s, t, u, r \in \overline{\mathbb{F}}$, then the mapping

$$\begin{aligned} \hat{E} &\to \mathbb{P}^2(\overline{\mathbb{F}}) \\ (X : Y : 1) &\mapsto (u^2 X + r : u^3 Y + s u^2 X + t : 1) \\ \mathcal{O} &\mapsto \mathcal{O} \end{aligned}$$

is an isomorphism of elliptic curves (See [S1986, p. 49]). We seek $\mathbb{F}$-isomorphisms, i.e., isomorphisms preserving the rational points. If $s, t, u, r \in \mathbb{F}$, then the given mapping is an $\mathbb{F}$-isomorphism.

Let $\hat{P}_1, \hat{P}_2 \in \hat{E}(\mathbb{F})$ be two distinct finite rational points of $\hat{E}$. Select two random finite rational points $P_1, P_2 \in \mathbb{P}^2(\overline{\mathbb{F}})$. We will now show that an $\mathbb{F}$-isomorphism of elliptic curves $\varphi$ such that $\varphi(\hat{P}_1) = P_1$ and $\varphi(\hat{P}_2) = P_2$ exists with probability $1/2$.

The isomorphism can be explicitly computed by inserting the coordinate values for $P_1, \hat{P}_1$ and $P_2, \hat{P}_2$ into the change of variables as given above, and solving for $u, r, s, t$.

We can first solve the two equations corresponding to the constraints in $X$ in the unknowns $u^2$ and $r$. Now, if the obtained value for $u^2$ is not a square in $\mathbb{F}$, then no such isomorphism exists. Otherwise, we can compute $u$ and $r$, and then use the two other equations to solve for $s$ and $t$.

Since $1/2$ of the values in $\mathbb{F}$ are squares, this shows that such a change of variable can be found with probability $1/2$, namely whenever the solution for $u^2$ is a square in $\mathbb{F}$.

### 3.2.2 Outline of the algorithm

We will now present an algorithm to tackle the reconstruction problem. We start by briefly summarizing the steps performed by the algorithm.

1. **Recover the group structure.** By [BSS1999, p. 36], the $\mathbb{F}$-rational points on the elliptic curve form an Abelian group,

$$E(\mathbb{F}) \cong \mathbb{Z}/d_1\mathbb{Z} \times \mathbb{Z}/d_2\mathbb{Z}, \tag{3.4}$$

   where $d_1 \mid \gcd(q-1, d_2)$ and $d_2 \mid E(\mathbb{F})$. The first step consists of recovering $d_1$ and $d_2$ and group elements $z_1, \ldots, z_n \in \mathbb{Z}/d_1\mathbb{Z} \times \mathbb{Z}/d_2\mathbb{Z}$ such that there exists an (unknown) group isomorphism

$$\varphi : E(\mathbb{F}) \to \mathbb{Z}/d_1\mathbb{Z} \times \mathbb{Z}/d_2\mathbb{Z}$$

   satisfying $\varphi(P_i) = z_i$.

2. **Find an isomorphic curve.** The fact that we know the $z_i$ then reveals a great deal of information about the geometry of the curve. For example, if $z_i + z_j + z_l = 0$, then this implies that $P_i, P_j$ and $P_l$ are collinear.

   We make use of this fact to deduce from a guess of the coordinates of three points, say $P_1$, $P_2$, $P_3$, the coordinates of a small number of additional points that must lie on $E$.

   Knowing such a small set of points, we can determine whether an elliptic curve passing through those points exists. If no such curve exists, we can conclude that the guess for the coordinates of $P_1$, $P_2$ and $P_3$ was wrong, and retry with another guess.

   If a suitable curve exists, then with very high probability, it is the desired curve, and we are done with this step.

3. **Compute the $P_i$.** Since $E$ is known at this point, we can compute all the rational points as well as the isomorphism

$$\mathbb{Z}/d_1\mathbb{Z} \times \mathbb{Z}/d_2\mathbb{Z} \to E(\mathbb{F})$$

   that maps $z_i \mapsto P_i$.

   Using this isomorphism, we can compute the remaining unknown points $P_i$ given the known $z_i$.

4. **Compute the $c_i$.** At this point, we are able to construct the same code, up to the stretching factors $c_1, \ldots, c_n$, i.e., we know a curve $E$ and points $P_1, \ldots, P_n$ such that the code

$$\mathcal{C}_0 := \mathrm{AGC}(E, k\langle\mathcal{O}\rangle, (P_1, \ldots, P_n))$$

   is equal to $\mathcal{C}$ up to stretching factors $c_1, \ldots, c_n$.

   We compute the code $\mathcal{C}_0$, and seek the unknowns $c_1, \ldots, c_n$, such that

$$(x_1, \ldots, x_n) \in \mathcal{C}_0 \iff (c_1x_1, \ldots, c_nx_n) \in \mathcal{C}.$$

This is done as follows. If $(y_1, \ldots, y_n)$ is any codeword in $\mathcal{C}^\perp$ and $(z_1, \ldots, z_n)$ is any codeword in $\mathcal{C}_0$, then we must have

$$\sum_{i=1}^{n} c_i z_i y_i = 0,$$

which gives us a linear condition on the $c_i$. This step can be repeated for different $z \in \mathcal{C}_0$ and $y \in \mathcal{C}^\perp$, and eventually enough linear conditions are collected to get a vector $(c_1, \ldots, c_n)$ that is unique up to a non-zero multiplicative factor.

We often also have to make heuristic assumptions. In an attempt to focus on the essential idea, we will generally just assume that these assumption hold with high probability, and postpone the discussion of their impact to section 3.2.6.

### 3.2.3 Step I: Recovering the group structure

The idea is to use the fact that minimum weight codewords correspond to functions whose divisor is exactly known. We assume that the minimum distance of $\mathcal{C}$ is $n-k$. Let $x := (x_1, \ldots, x_n)$ a codeword of minimum weight (i.e., weight $n-k$), and let $f$ be the rational function corresponding to $x$, i.e.,

$$x = (c_1 f(P_1), \ldots, c_n f(P_n)).$$

Now, without loss of generality, assume the zero positions of $x$ are $P_1, \ldots, P_k$. Then

$$f(P_1) = \cdots = f(P_k) = 0,$$

and so

$$\operatorname{div}(f) \succeq \langle P_1 \rangle + \cdots + \langle P_k \rangle - k\langle \mathcal{O} \rangle,$$

where $k\langle \mathcal{O} \rangle$ is the divisor of the code, by our reasoning in section 3.2.1. The divisor on the right hand side above is of degree 0, and therefore

$$\operatorname{div}(f) = \langle P_1 \rangle + \cdots + \langle P_k \rangle - k\langle \mathcal{O} \rangle. \tag{3.5}$$

Since $\operatorname{div}(f)$ is principal, we have

$$\mathcal{O} = \operatorname{sum}(\operatorname{div}(f)) = P_1 + \cdots + P_k - [k]\mathcal{O}. \tag{3.6}$$

Write

$$\mathcal{G} := \mathbb{Z}/d_1\mathbb{Z} \times \mathbb{Z}/d_2\mathbb{Z}$$

the Abelian group that is isomorphic to $E(\mathbb{F})$. Denote by $z_i$ the images of $P_i$ under some group isomorphism $E(\mathbb{F}) \to \mathcal{G}$. Then we can rewrite (3.6) as an equation in $\mathcal{G}$, and we get

$$z_1 + \cdots + z_k = 0.$$

If we knew the parameters $d_1$ and $d_2$ of $\mathcal{G}$, we would have to collect $n$ $\mathbb{Q}$-linearly independent such equations in order to be able to determine the $z_i$.

We can collect more than $n$ linear equations, and the resulting homogeneous system will still have a nontrivial solution in $\mathcal{G}$, and thus in particular also modulo $d_1$ and $d_2$, if we consider the unknowns $z_i$ as variables in $\mathbb{Z}/d_1\mathbb{Z}$, respectively $\mathbb{Z}/d_2\mathbb{Z}$.

Since in general, sufficiently overdetermined homogeneous systems of equations are very unlikely to have nontrivial solutions (see the discussion in section 3.2.6), this observation can be used to determine $d_1$ and $d_2$: the easiest way is to attempt solve the system modulo different integers, and if a nontrivial solution is found, conclude that the modulus is with high probability a divisor of $d_2$.

It is, however, important to note that in our case, nontrivial solutions exist not only mod $d_1$ and $d_2$, but also modulo $k$ and its divisors: Since each linear constraint is a sum of exactly $k$ of the $z_i$, we have the solution

$$z_1 = \cdots = z_n = 1 \mod k.$$

Fortunately these unhelpful solutions are easy to detect, since the value of $k$ is known.

Different techniques are conceivable to recover $d_1$ and $d_2$. As mentioned before, a simple solution to this problem is to just try to solve the overdetermined system for different moduli, and thus find moduli that lead to solutions. In the most frequent case, we have $d_1 = 1$, and so the sought value for $d_2$ is an integer in the interval $[(q+1) - 2\sqrt{q}, (q+1) + 2\sqrt{q}]$ by the Hasse theorem (Theorem 2 of chapter 2); otherwise there are in fact few possibilities, given the conditions $d_1 \mid \gcd(d_2, q-1)$ and $d_1 d_2 \in [(q+1) - 2\sqrt{q}, (q+1) + 2\sqrt{q}]$.

By (3.3), we assume $\gcd(k, |\mathcal{G}|) = 1$, thus we can separate the solutions modulo $k$ from the other ones. If (3.3) is not fulfilled, then this will be detected at this point, and appropriate measures can be taken (e.g., the code can be shortened).

Once the values for $d_1$ and $d_2$ have been recovered, we can solve for the $z_i$.

### 3.2.4 Step II: Finding an isomorphic curve

The knowledge of the group structure makes it possible to create minimum weight words with special properties. We will make use of this fact to find an elliptic curve $E$ that is $\mathbb{F}$-isomorphic to $\hat{E}$.

We first find two codewords $v$ and $w$ corresponding to rational functions $f_1$ and $f_2$ that only differ in a small factor. The rational function $f_1/f_2$ is then a fraction of two very low degree polynomials whose coefficients are not known.

Now assume that the coordinates of the point $P_i$ are known, for some known $1 \leq i \leq n$. Then the equality

$$\frac{v_i}{w_i} = \frac{f_1}{f_2}(P_i)$$

gives a linear condition on the coefficients of the rational function $f_1/f_2$. Thus, if sufficiently many points are known, and the degree of the constituting polynomials of $f_1/f_2$ is sufficiently small, then all the coefficients of the rational function $f_1/f_2$ can be determined.

We can then use the evaluations of $f_1/f_2$ on other points together with the explicit formula for $f_1/f_2$ to deduce conditions on the coordinates of these points.

Together with geometric information of the points (in particular, that it is possible to detect when three points are collinear), this information can be used to compute the coordinates for a larger number of points. Once a sufficient number of points have been reconstructed, we can solve for the coefficients of the Weierstrass equation, and determine the curve equation.

This procedure can only be carried out if we can start out with sufficiently many known points. We will see that by cleverly choosing the functions $f_1$ and $f_2$, we can get the procedure started by knowing the coordinates of only three points, say, $P_1$, $P_2$ and $P_3$.

Since we do not know any points, a priori, the idea is thus to just guess coordinate positions for those three points, and then to reconstruct a few points assuming that the guess was correct. For a wrong guess of $P_1, P_2$ and $P_3$, it is very unlikely that all the reconstructed other points will lie on a common elliptic curve, and thus we are able to detect wrong guesses.

If we were to guess all three points $P_1, P_2$ and $P_3$, this would make for $|\mathbb{F}|^6$ guesses, which is impractical for interesting values of $|\mathbb{F}|$. Given the curve isomorphisms as discussed in section 3.2.1, however, we can fix the coordinates of $P_1$ and $P_2$ arbitrarily and then only have to try all the possible values for $P_3$, resulting in a successful recovery in 1/2 of the cases.

We therefore only have to test roughly $|\mathbb{F}|^2$ guesses, which is practical if a single guess is inexpensive.

It is important to make the guessing fast, easy and reliable. For example, the guessing procedure should be designed in a way so that pathological cases do not occur.

One way to do this is to create a *recovery schedule* beforehand. The recovery schedule contains the information about which points can be recovered, and how this is done, independently of the initial guess.

Since the same recovery schedule can be applied regardless of the initial guess for coordinates, and since most of the complexity and probabilistic algorithms are moved to the schedule creation (which is only performed once) all we have to provide is a rapid algorithm that executes the schedule and rejects false guesses.

We will now describe the procedure step by step.

### Step IIa: Building the words $f_1$ and $f_2$

We first compute two codewords $v$ and $w$ corresponding to functions $f_1$ and $f_2$ such that the rational function $f_1/f_2$ is a fraction of two degree 1 polynomials, more precisely,

$$\frac{f_1}{f_2} = \frac{\alpha x + \beta}{x + \gamma}.$$

The key ingredient to be able to do this is again equation (3.5), that is, the observation that if $x$ is a word of minimum weight, then its divisor is known.

For this reason, the sought words $v$ and $w$ have to be minimum weight. We will therefore find $f_1$ and $f_2$, such that

$$\begin{aligned} f_1(P) &= g(P) \cdot \ell_1(P) \\ f_2(P) &= g(P) \cdot \ell_2(P), \end{aligned}$$

where $\ell_1$ and $\ell_2$ are polynomials of degree 1 that are zero on different vertical lines, i.e.,

$$\begin{aligned} \ell_1(x, y) &= \alpha x + \beta \\ \ell_2(x, y) &= x + \gamma, \end{aligned}$$

for unknown coefficients $\alpha, \beta$ and $\gamma$. The function $g$ must have $k-2$ zeros among the $P_i$ in order for the resulting codewords $v$ and $w$ to have minimum weight.

We first find two pairs of opposite points, i.e., four distinct integers $1 \leq i_1, i_2, j_1, j_2 \leq n$, such that

$$z_{i_1} = -z_{i_2} \text{ and } z_{j_1} = -z_{j_2}.$$

In order to construct a suitable set of zeros for the factor $g$, we find a set $X \subset \{1, \dots, n\} \setminus \{i_1, i_2, j_1, j_2\}$ of size $k-2$, such that

$$\sum_{i \in X} z_i = 0.$$

Then there exists a rational function $g$ such that

$$\operatorname{div}(g) = \sum_{i \in X} \langle P_i \rangle - (k-2)\langle \mathcal{O} \rangle.$$

The function $g$ is unique up to a constant multiplicative factor.

Analogously, since $\sum_{i \in X \cup \{i_1, i_2\}} z_i = 0$ and $\sum_{i \in X \cup \{j_1, j_2\}} z_i = 0$, there are rational functions $f_1$ and $f_2$ and corresponding to codewords $v$ and $w$ that have the zeros on the positions $X \cup \{i_1, i_2\}$ and $X \cup \{j_1, j_2\}$, respectively. Those codewords are unique by minimality and easy to construct by pivoting the generator matrix suitably.

Let $f_1$ and $f_2$ be the functions corresponding to $v$ and $w$. Then, again by (3.5), we know that

$$\begin{aligned} \operatorname{div}(f_1) &= \operatorname{div}(g) + \langle P_{i_1} \rangle + \langle P_{i_2} \rangle - 2\langle \mathcal{O} \rangle \\ \operatorname{div}(f_2) &= \operatorname{div}(g) + \langle P_{j_1} \rangle + \langle P_{j_2} \rangle - 2\langle \mathcal{O} \rangle. \end{aligned}$$

Hence, $f_1 = g \cdot \ell_1$ and $f_2 = g \cdot \ell_2$, where $\ell_1$ is the vertical line passing through $P_{i_1}$ and $P_{i_2}$ and $\ell_2$ is the vertical line passing through $P_{j_1}$ and $P_{j_2}$.

Since we can compute the codewords $v$ and $w$ explicitly, we can evaluate $f_1/f_2$ on the points on which $f_2$ is not zero.

**Step IIb: Creation of the recovery schedule**

We keep a list of processable points (code coordinates) $L$, which initially contains supp($w$), i.e., the positions on which we can evaluate the function $f_1/f_2$ as determined in the previous section.

The recovery schedule creation consists of two steps:

1. **Select the initial triple.** We select three coordinates in $L$, they will correspond to the points that later will be guessed. The three points have to satisfy a number of conditions.

   First, the coordinate positions should be such that they are not collinear, i.e., if we write $i, j, \ell \in L$ the distinct coordinate positions to select, we want
   $$z_i + z_j + z_\ell \neq 0.$$
   Second, we require that $z_i$, $z_j$ and $z_\ell$ are not in a proper subgroup of $\mathcal{G}$, i.e.,
   $$\langle z_i, z_j, z_\ell \rangle = \mathcal{G}.$$

   Furthermore, it is convenient to require that each of the lines passing through two of the three points in the triple pass through a third point in $L$. (This condition can also be verified by looking at the $z_i$s.)

   We add the triple to an array $S$, the recovery schedule. We also remove the points from $L$.

2. **Build the recovery schedule.** Now we start adding other points to the schedule according to the following rule: iteratively, any point can be added to the schedule if it is collinear to two points that are already in the schedule.

   Each added point is recorded in the array $S$, which in addition also stores with which points from the schedule it was aligned. The added points are removed from $L$.

A graphical illustration of a possible recovery schedule is given in figure 3.1. In the example, the three points $P_1, P_2$ and $P_3$ surrounded by dotted lines are the initial points, the coordinates of which are assumed to be known.

The recovery schedule $S$ corresponding to Figure 3.1 would look as follows in this case:

| # | Point | Aligned with |
|---|---|---|
| 1 | $P_1$ | |
| 2 | $P_2$ | |
| 3 | $P_3$ | |
| 4 | $P_4$ | $P_1, P_2$ |
| 5 | $P_5$ | $P_1, P_3$ |
| 6 | $P_6$ | $P_3, P_4$ |

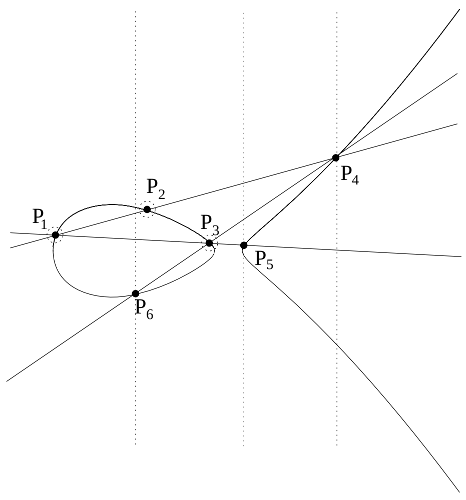

Figure 3.1: Example recovery schedule

Only a small number of points is needed in the recovery schedule: Six generic points are enough to uniquely determine the elliptic curve, so we need just a few more than six points. In practice eight to ten points are enough, but we will discuss this point further in section 3.2.5. The schedule building will be stopped as soon as a small predetermined number of points are in the schedule.

**Step IIc: The guessing procedure**

Without loss of generality assume the initial triple of the recovery schedule consists of the points $P_1, P_2$ and $P_3$. We will now show how to deduce the coordinates for all points in the recovery schedule, once the coordinates for the first three points have been fixed. Note that $f_2$ is nonzero on any point in the recovery schedule, and so the values

$$\frac{v_i}{w_i} := \frac{c_i f_1}{c_i f_2}(P_i) = \frac{\ell_1}{\ell_2}(P_i), \quad 1 \leq i \leq 3,$$

are finite and can be determined, giving the three equations

$$\frac{v_i}{w_i} = \frac{\alpha x_i + \beta}{x_i + \gamma} \quad 1 \leq i \leq 3, \tag{3.7}$$

where $x_i$ is the $x$-coordinate of $P_i$, and the unknowns are $\alpha, \beta$ and $\gamma$. We can write these equations as linear equations, and then solve for $\alpha, \beta$ and $\gamma$.

In a second step, we solve the linear equation

$$\frac{v_i}{w_i} = \frac{\ell_1}{\ell_2}(P_i) = \frac{\alpha x_i + \beta}{x_i + \gamma}$$

for every point in the recovery schedule (except the initial triple). The equations are again equations of the form (3.7), except that this time, the unknowns are the $x_i$. That way, we can determine the $x$-coordinates of all the points in the recovery schedule.

To determine the $y$-coordinates, we use the schedule itself. We proceed in the schedule order. Each point $P$ in the schedule is aligned with two points $Q$ and $R$ whose coordinates are already known. To determine the $y$-coordinate of $P$, compute the line that passes through $Q$ and $R$, and determine its intersection with the known vertical line passing through $P$: This intersection gives the coordinates of $P$. Figure 3.1 illustrates this; in the drawing, the vertical dotted lines are the lines that have been determined by evaluating $\ell_1/\ell_2$ on the $P_i$, and the straight lines are those that align three points in the schedule.

Having determined the $(x, y)$-coordinates of all the points in the schedule, it is of course easy to determine whether an elliptic curve passing through all the given points exists, and, if yes, what its parameters are: It is enough to insert the coordinate points into Weierstrass equations and solve for $a_1, a_2, a_3, a_4, a_6$. If we choose the recovery schedule large enough, i.e., such that it over-determines the curve parameters by a few equations, wrong guesses of the coordinates for the initial triple will be rejected with very high probability.

### Step IId: Computing the curve isomorphism

We now know $\mathcal{G}$ and $E$. In order to determine the points $P_i$ for all $i$, we determine the isomorphism

$$\mathbb{Z}/d_1\mathbb{Z} \times \mathbb{Z}/d_2\mathbb{Z} \to E(\mathbb{F})$$

that maps $z_i \mapsto P_i$.

To do this, we can proceed the following way. Pick two points whose coordinates have been recovered, say $P_1$ and $P_2$. Write $z_1 = (r_1, s_1), z_2 = (r_2, s_2)$, where $r_i \in \mathbb{Z}/d_1\mathbb{Z}$ and $s_i \in \mathbb{Z}/d_2\mathbb{Z}$.

We can then try to determine two integers $a, b \in \mathbb{Z}$ such that

$$a(r_1, s_1) + b(r_2, s_2) = (0, 1) \text{ in } \mathcal{G}.$$

For example, compute $a$ as a function of $b$ modulo $d_2$ using $as_1 + bs_2 = 1$, and then use this solution to determine $b$ modulo $d_1$ by solving the corresponding equation modulo $d_1$. This works since $d_1 \mid d_2$. This computation fails if $P_1$ and $P_2$ do not generate $E(\mathbb{F})$. In this case, another couple of points has to be tried. (Since we assumed that the initial triple generates $\mathcal{G}$, it should contain such a pair.)

Given $a$ and $b$, we can compute the point $S := [a]P_1 + [b]P_2$ corresponding to $(0, 1)$ in $\mathcal{G}$. In the same manner, $R$ corresponding to $(1, 0)$ in $\mathcal{G}$ can be computed.

Given the points $R$ and $S$, we have the isomorphism

$$(a, b) \in \mathcal{G} \mapsto [a]R + [b]S$$

that satisfies $z_i \mapsto P_i$.

We close this section by remarking that it is important that an isomorphism $\mathcal{G} \to E(\mathbb{F})$ is sought that maps $z_i \mapsto P_i$, and not just any isomorphism $\mathcal{G} \to E(\mathbb{F})$. This is so because group isomorphisms of $E(\mathbb{F})$ do not, in general, have corresponding curve isomorphisms.

### 3.2.5 Step III: Reconstructing the $c_i$

We can now construct the elliptic curve $E$, and the points $P_1, \dots, P_n$, such that

$$\mathrm{AGC}(E, k\langle\mathcal{O}\rangle, (P_1, \dots, P_n)) \tag{3.8}$$

is, up to a multiplicative constant in each coordinate position, the same code as

$$\mathrm{AGC}(\hat{E}, \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n)).$$

Thus, it remains to determine the constants $c_1, \dots, c_n$, such that

$$\mathrm{AGC}(E, k\langle\mathcal{O}\rangle, (P_1, \dots, P_n), (c_1, \dots, c_n)) = \mathrm{AGC}(\hat{E}, \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n)).$$

Let

$$\begin{aligned} (x_1, \dots, x_n) &\in \mathrm{AGC}(E, k\langle\mathcal{O}\rangle, (P_1, \dots, P_n), (1, \dots, 1)) \\ \text{and } (y_1, \dots, y_n) &\in \mathrm{AGC}(\hat{E}, \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n))^\perp. \end{aligned}$$

Since

$$(c_1x_1, \dots, c_nx_n) \in \mathrm{AGC}(\hat{E}, \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n)),$$

we have that $(c_1, \dots, c_n)$ satisfies

$$\sum_{i=1}^{n} c_i x_i y_i = 0,$$

giving a linear equation on the $c_i$. By choosing arbitrary codewords $x$ and $y$ in the appropriate codes, we can find $> n$ linear conditions on the $c_i$, and thus solve the homogeneous system to get a vector $(c_1, \dots, c_n)$.

We show that the solution to this system is unique with very high probability.

Assume there are two solutions $c_i$ and $\hat{c}_i$ which are not equal up to a multiplicative factor. Then we can assume $\hat{c}_i = 0$ for some $1 \le i \le n$, and $\hat{c}_j \ne 0$ for some $1 \le j \le n$. Let $x \in \mathrm{AGC}(E, k\langle\mathcal{O}\rangle, (P_1, \dots, P_n))$ be a minimum weight word such that $i, j \in \mathrm{supp}(x)$. The word

$$(\hat{c}_1x_1, \dots, \hat{c}_nx_n)$$

is nonzero and has weight at most $n - k - 1$. Since the minimum distance of $\mathrm{AGC}(\hat{E}, \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n))$ has minimum distance at least $n - k$, its dual code contains a parity check

$$y \in \mathrm{AGC}(\hat{E}, \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n))^\perp$$

that fails on $(\hat{c}_1 x_1, \ldots, \hat{c}_n x_n)$, i.e. such that

$$\sum_{\ell=1}^{n} (\hat{c}_\ell x_\ell) y_\ell \neq 0,$$

as desired.

In the "proof" above we assumed that a suitable minimum weight word $x$ exists. In fact, this assumption is the same as assuming that the punctured code

$$\mathrm{AGC}(E, k\langle \mathcal{O} \rangle, (P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_{j-1}, P_{j+1}, \ldots, P_n))$$

has minimum distance $(n-2)-k$. The arguments in section 3.2.6 justify this assumption.

### 3.2.6 Assumptions

In the above algorithm, a number of assumptions were made which do not necessarily hold in general for elliptic codes. We will now discuss the impact of these assumptions on the practicality of the attack.

#### Large evaluated point set

We implicitly assumed that the point set $\{P_1, \ldots, P_n\}$ contains many elements, and in particular, most elements of $E(\mathbb{F})$. In other words, we assumed that the block length $n$ is not much smaller than $|E(\mathbb{F})|$.

The two steps in the algorithm that are affected most by small point sets are:

- Picking two pairs of opposite points, so that $f_1$ and $f_2$ can be constructed.
- Picking an initial triple in the recovery schedule.

In order to keep the pressure off these steps, it makes sense in practice to reorder the steps of the algorithm we just presented, and to compute the recovery schedule before computing the functions $f_1$ and $f_2$. We refrained from presenting the algorithm this way, because it complicates the presentation considerably.

We now discuss the problem of finding opposite points if the point set is small. This problem can be summarized as follows: Find $1 \leq i < j \leq n$ such that $z_i + z_j = 0$.

To illustrate the problem, we start by showing how small point sets can cause the algorithm to fail. Assume $n < |E(\mathbb{F})|/2$. Then a point set $\{P_1, \ldots, P_n\}$ can be constructed such that $Q \in \{P_1, \ldots, P_n\}$ implies $(-Q) \notin \{P_1, \ldots, P_n\}$. So there is no pair $i, j$ with $P_i + P_j = \mathcal{O}$, and thus the problem that needs to be solved in step IIa does not even have a solution.

For such a point set, obviously none of the vertical lines even exist.

This construction shows that the proposed algorithm can be brought to a halt by smartly choosing a small point set. On the other hand, it also shows that whenever $(|E(\mathbb{F})| - 1)/2 + 2 < n$, then two such pairs exist, and they can be found by exhaustive search, which is only $O(n^2)$ in this case.

If the small point set is chosen randomly, then a back-of-the-envelope computation shows that large alphabets alone do not lead to efficient solutions: If we choose

$$n = O(\sqrt{|\mathbb{F}|}) = O(\sqrt{|E(\mathbb{F})|}),$$

then, since there are $O(n^2)$ pairs of points in the point set, there are still $O(1)$ pairs summing to zero, on the average. So we expect that pairs of opposite points in the point set exist with constant probability.

We now turn to study the initial triple choosing under the small point set assumption. We focus on the case where the point set is chosen randomly.

If $\mathcal{G}$ is cyclic, then for any $i$, the probability that $z_i$ generates $\mathcal{G}$ is $\varphi(|\mathcal{G}|)/|\mathcal{G}|$, where $\varphi$ is the Euler totient function. The quantity $\varphi(|\mathcal{G}|)/|\mathcal{G}|$ is not typically small. On the average, it converges to at least $3/\pi^2$, i.e., from the asymptotic formula in [AS1970, p. 826] it is trivially deduced that

$$\lim_{n\to\infty} n^{-1} \sum_{i=1}^{n} \frac{\varphi(i)}{i} \geq \frac{3}{\pi^2}.$$

Moreover, it can be bounded from below. We have, from [BS1996, Theorem 8.8.7],

$$\frac{\varphi(n)}{n} > \frac{1}{e^{\gamma} \log(\log(n)) + \frac{3}{\log(\log(n))}},$$

for $n \geq 3$. This estimate is $> 0.2$ for $100 < n < 1000$, and so the probability that three randomly selected points do not contain a generator of $\mathcal{G}$ is certainly $< 0.488$. Thus, in this case, the probability that a random triple is in a proper subgroup is certainly $< 0.512$ for those values of $n$.

We would therefore expect to only have to select roughly two triples at random to have this condition satisfied.

The condition that each pair of points is collinear with a third point in the point set, is satisfied per pair with probability

$$\frac{n-3}{|E(\mathbb{F})|} \approx \frac{n}{|E(\mathbb{F})|} =: \lambda,$$

so the probability that this condition is satisfied for the three points is roughly $\lambda^3$, and roughly $\lambda^{-3}$ triples would have to be randomly selected, until a good one is found. Since there are roughly $n^3/6$ such triples, one would expect at least one triple verifying all the conditions to exist as long as

$$\frac{1}{12}(n\lambda)^3 \gg 1.$$

So, asymptotically, as long as $n \in \Omega(|E(\mathbb{F})|^{1/2+\varepsilon})$, such a triple is expected to exist.

In summary, the cost of a sufficiently small random point set is that the block length in symbols has to be roughly halved for the same block length in bits.

Since we have seen how to break the algorithm with small point sets, and since good parameters are obtained with large point sets, we will assume that the point set is linear in the sequel, i.e.,

$$n \geq \mu|E(\mathbb{F})|$$

for some fixed $\mu$.

### Minimum distance

We have assumed that words of weight $n-k$ do exist and can be efficiently sampled. Rigorously speaking, we only know that $d \geq n-k$, so it could well be that the elliptic code is in fact maximum distance separable, i.e., has distance $d = n-k+1$.

In practice, we have never seen that happen, so a distance larger than the one provided by the bound apparently happens only in very particular cases, and is especially unlikely under the large point set assumption, as we will see now.

Consider the problem of finding words of weight $n-k$ and assume that the set of evaluated points on the elliptic curve has been selected randomly. Then, by diagonalizing the generator on a random information set, we get words of weight $n-k+1$ at most. What is the probability that the word has weight $n-k$? Without loss of generality, consider a codeword having zeros in the first $k-1$ entries. The $j$th position (with $j > k-1$) will be a zero position as well if and only if

$$P_1 + \cdots + P_{k-1} + P_j = \mathcal{O}.$$

Since the $P_i$ are random in $E(\mathbb{F})$, this probability is equal to the probability that a random set of size $n-k$ in $E(\mathbb{F})$ contains a certain fixed value. This probability is

$$\frac{n-k}{|E(\mathbb{F})|} = \mu(1-R)$$

Thus, words of weight $n-k$ are frequent in our setting, and sampling them is not a big problem, as each diagonalization of the generator matrix on an information set will yield a number proportional in $n$ of them.

The algorithm to find such words is thus probabilistic polynomial time.

### Stray solutions in linear systems

We did not prove that the overdetermined linear systems we solve do not have any undesired, systematic wrong solutions for our guesses in several places, in particular:

- In step I, when solving for the group.
- In the guessing procedure of step II, when solving for the Weierstrass equation.

The underlying heuristic argument assumes that these matrices are random matrices. Under the random matrix model, we can use the Markov inequality to estimate the probability that an overdetermined homogeneous system of equations has nontrivial solutions: Let $A$ be an $\ell \times r$ matrix with random entries. The probability that the system

$$Ax^t = 0$$

has nontrivial solutions is

$$\begin{aligned}\mathrm{Prob}(|\mathrm{rker}(A)| > 1) &= \mathrm{Prob}(|\mathrm{rker}(A)| \geq q)\\ &= \mathrm{Prob}\left(\frac{|\mathrm{rker}(A)|}{q} \geq 1\right)\\ &\leq q^{-1}\mathrm{E}(|\mathrm{rker}(A)|)\\ &= q^{-1}\left(q^r\left(\frac{1}{q}\right)^\ell\right)\\ &= q^{r-\ell-1},\end{aligned}$$

where $q = |\mathbb{F}|$ and $\mathrm{rker}(A)$ denotes the right kernel of $A$, thus each added constraint reduces the failure probability by a factor $q^{-1}$.

There is always the risk with heuristic arguments that the practical results turn out to be much worse than what we predict. In our case, the findings are the following.

For the guessing procedure, we could not detect any degenerate behavior, and, in fact, recovery schedules of size 8 seem to work just fine.

Step I, however, does not behave so nicely. It turns out that if we take $n$ of the sampled constraints, the determinant (over $\mathbb{Q}$) of the solution is often 0, causing frequent failures for large $n$. Other results on ranks of random matrices like [BKW1996] suggest that the problem is not the relative sparsity of the constraint matrix, but the fact that we do not sample minimum weight words independently: For each diagonalization we take all minimum weight words that appear to build a random matrix.

There are several solutions to this problem. The most rigorous, but also the most awkward one is to just find one minimum weight word per information set, and discard the other ones, thus arranging for independent minimum weight words. It is much better, from a computational point of view to sample, say, $4n$ minimum weight words in the usual way, and modify the way the constraint matrix is built instead: each row in the matrix can be chosen to be a random linear combination of just a few (say, 3) of the sampled constraints, causing the matrix to be more random.

It is not clear how exactly this second approach must be generalized to make it work for asymptotic $n$, but as we will see in the results for the practical running time, even for very large $n$, this slight modification caused all practical problems to disappear.

### 3.2.7 Running time

We give an analysis of the running time under the aforementioned assumptions; e.g., we assume the size of the point set is $\geq \mu \cdot |E(\mathbb{F})|$. We write $q = |\mathbb{F}|$.

As previously discussed, any word in a diagonalization has weight $n - k$ with a probability that only depends on the rate $R$ and $\mu$. In order to collect $O(n)$ relations, we thus need $O(1)$ diagonalizations, resulting in total a cost of $O(n^3)$ to collect enough relations.

Finding the group structure requires solving at most $q + 2\sqrt{q} = O(q)$ systems of equations in $n$ variables, where $q = |\mathbb{F}|$. This step costs thus $O(qn^3)$. (In the common case, where $\mathcal{G} = \mathbb{Z}/d_2\mathbb{Z}$, it is $O(\sqrt{q}n^3)$.)

The steps IIa and IIb are negligible in practice, but note that a number of subset sum problems have to be solved. The probabilistic arguments of section 3.2.6 suggest that for interesting parameters, the subset sum problems are $O(q)$. Two diagonalizations of matrices are needed, so this is $O(q + n^3)$.

The guessing step IIc takes $O(q^2)$ operations in $\mathbb{F}$. For practical values of $q$, this is best modeled as $O(q^2)$.

Thus, the total probabilistic running time of this algorithm is $O(qn^3)$.

### 3.2.8 Experimental running time

In order to see how well an attack works out in practice, it is of course customary to compute the actual obtained running times of the attack algorithm. Also, and perhaps more importantly, practical running time results allow us to verify that the heuristic assumptions from section 3.2.6 are reasonable in practice.

It should be noted that our implementation is not completely general; in order of importance, we lived with the following restrictions.

1. We assume $|E(\mathbb{F})|$ is a prime. This implies in particular also that the group $E(\mathbb{F})$ is cyclic.

2. The divisor is always of the form $k\langle P\rangle$ for some rational point not of order 2.

3. We work on a prime field, i.e., $\mathbb{F} = GF(p)$ for some prime $p$.

4. We put ourselves in the setting $\gcd(|E(\mathbb{F})|, k) = 1$, conforming to (3.3).

These assumptions were made to keep the implementation reasonably simple. For example, if $|E(\mathbb{F})|$ is not a prime, we have to do linear algebra operations on non-fields, which is tedious to implement.

**Running times for different code sizes**

We will now give a few samples of the running time of the attack against elliptic curves that we implemented in C++, and ran on a PC (Pentium IV, 2.4 GHz), though it should be noted that the implementation is not optimized.

It is of course not obvious against what instances the attack should be realistically run. As a first experiment, we picked a $[178, 89, 89]$ code over $GF(149)$ that is comparable to the Goppa code that McEliece had originally proposed:

| | Goppa code (McEliece) | Elliptic |
|---|---|---|
| $[n, k, d]$ | $[1024, 524, \geq 101]$ | $[178, 89, 89]$ |
| field | $GF(2)$ | $GF(157)$ |
| # correctable errors | 50 | 44 |
| rate | $\sim 1/2$ | $1/2$ |
| IS success probability | $2^{-53}$ | $2^{-54}$ |
| canonical key size | 537 kbits | 116 kbits |
| smallest key size | 250 kbits | 58 kbits |

In this table, IS success probability denotes the probability that a random information set is error free, and gives a measure of the security level under general decoding. The canonical key size is the size of a random generator matrix in bits, given by $nk \log_2(|\mathbb{F}|)$. The smallest key size is the size of a systematic generator matrix of the code or the dual, whichever is smaller.

In our setup, the running time over 10 runs of the algorithm to compute private keys from the given public keys, were 38.89 seconds on the average, 49 at most, and at least 34.

By today's standards, the security level of the original McEliece cryptosystem is much too weak. The following table lists attack times (again averages of 10 runs) for larger codes:

| $[n, k, d]$ | $[326, 163, 163]$ | $[500, 250, 250]$ | $[1008, 504, 504]$ |
|---|---|---|---|
| field | $GF(307)$ | $GF(467)$ | $GF(953)$ |
| # correctable errors | 81 | 124 | 251 |
| IS success probability | $2^{-100}$ | $2^{-154}$ | $2^{-313}$ |
| smallest key size | 220 kbits | 554 kbits | 2514 kbits |
| average time | 3 min 36 sec | 12 min 4 sec | 1 h 51 min |
| max time | 5 min 4 sec | 13 min 36 sec | 2 h 1 min |

Most likely, the $[1008, 504, 504]$-code over $GF(953)$ is larger than anything one would ever want to consider for a McEliece type cryptosystem. We note that for this range of parameters, the dominating computation factor appears to be $O(n^3)$.

# Chapter 4

# An attack against Sidelnikov's cryptosystem

The Sidelnikov cryptosystem was proposed 1994 by Sidelnikov [S1994]. It is a McEliece-type cryptosystem using Reed-Muller codes in combination with powerful decoding algorithms.

In this section, we present our attack against the Sidelnikov cryptosystem. This attack was published in [MS2007], and our presentation here follows the original publication, although with a number of improvements.

Reed-Muller codes are low-rate if any interesting error-correction capability is to be obtained. This makes it easy to apply algorithms such as the Canteaut-Chabaud-algorithm [CC1998] to find low weight words, and also to decode if the number of errors is less than $d/2$ (half the minimum distance). However, this disadvantage is mitigated by the fact that there are decoding algorithms for Reed-Muller codes which decode many more errors (with high probability) than even the minimum distance of the code, rendering direct decoding attacks impractical.

Low weight word finding algorithms can still be used to find minimum weight words in codes with suitable parameters, though. It turns out that because of this, the Sidelnikov-cryptosystem is not secure: The goal of this chapter is to show how to use minimum weight codewords to invert trapdoors in Reed-Muller codes. More precisely, if $G$ is the $k \times n$ generator matrix of a Reed-Muller code, $S$ is an $n \times n$ permutation matrix and $A$ is a $k \times k$ random invertible scrambler matrix, and the product $AGS$ is public knowledge, we show how to find a permutation matrix $\tilde{S}$ such that the codes generated by

$$G\tilde{S}$$

and

$$AGS$$

are the same. Since $G$ is not secret (there is only one binary Reed-Muller code for given dimensions), this means that trapdoors of this cryptosystem can be inverted.

## 4.1 Outline of the attack

We now present an algorithm which, given a permuted, scrambled Reed-Muller code $\mathcal{C}$, constructs a permutation $\tau$ such that if the positions of $\mathcal{C}$ are permuted accordingly, the resulting code is a Reed-Muller code.

The sketch of the attack is as follows. Let $\mathcal{C} = \mathcal{R}(r, m)^\sigma$ for some unknown permutation $\sigma$, given by an arbitrary generator matrix.

1. Find codewords in $\mathcal{C}$ which also belong to the subcode $\mathcal{R}(r-1, m)^\sigma$. Find enough such vectors to build a basis of $\mathcal{R}(r-1, m)^\sigma$.

2. Iterate the previous step (with decreasing $r$) until obtaining $\mathcal{R}(1, m)^\sigma$.

3. Determine a permutation $\tau$ such that $\mathcal{R}(1, m)^{\tau\circ\sigma} = \mathcal{R}(1, m)$. Then

$$\mathcal{R}(r, m)^{\tau\circ\sigma} = \mathcal{R}(r, m),$$

   and this fact can then be used to decode.

The meat of the attack lies in the first step, which is based on Proposition 2 of section 2.4.1.

## 4.2 Finding the subcode $\mathcal{R}(r-1, m)^\sigma \subseteq \mathcal{R}(r, m)^\sigma$

The basic idea of this step is to find a codeword for which we know that it is a product of other codewords, and then to split off a factor lying in the $\mathcal{R}(r-1, m)^\sigma$ subcode.

By Proposition 2, we know that a minimum weight codeword is a product of several codewords of $\mathcal{R}(1, m)^\sigma$. Hence, we do the following: We find a minimum weight codeword $x$ and split off a factor of this word.

To this end, we shorten the code on $\operatorname{supp}(x)$, and use the structure of the shortened code to find a factor of $x$ which lies in $\mathcal{R}(r-1, m)^\sigma$.

Finding enough words in $\mathcal{R}(r-1, m)^\sigma$ will result in a basis of $\mathcal{R}(r-1, m)^\sigma$.

### 4.2.1 Finding factors of minimum weight words

We drop the permutation $\sigma$ in this section, since our ideas do not depend on $\sigma$. For concreteness, we will first study the case $r = 2$ and generalize later on.

Let $x \in \mathcal{R}(2,m)$ be a minimum weight codeword. Using Proposition 2, and changing the basis, we can assume that $x = v_1v_2$. (We identify codewords with Boolean functions, as explained in section 2.4.1.)

Let $\mathcal{C}_{\text{supp}(x)}$ be the shortened code obtained from shortening $\mathcal{R}(2,m)$ on the support of $x$. In other words, $\mathcal{C}_{\text{supp}(x)}$ is the subcode of $\mathcal{R}(2,m)$ containing only the words which are zero on supp$(x)$, and with these positions punctured afterwards. Let $f \in \mathcal{B}(2,\{v_1,\dots,v_m\})$ be any codeword in this shortened code. Using the notation $\bar{v} = (v_3,\dots,v_m)$, we can then write

$$f(v_1,v_2,\bar{v}) = \lambda v_1v_2 + v_1f_1(\bar{v}) + v_2f_2(\bar{v}) + h(\bar{v}), \tag{4.1}$$

where $\lambda \in \mathbb{F}_2$, $f_1, f_2 \in \mathcal{B}(1,\{v_3,\dots,v_m\})$, and $h \in \mathcal{B}(2,\{v_3,\dots,v_m\})$. Now, the condition that $f$ is in the shortened code means that $f(1,1,\bar{v}) = 0$ for all $\bar{v}$. Rewriting this condition using (4.1) gives

$$0 = \lambda + f_1(\bar{v}) + f_2(\bar{v}) + h(\bar{v}), \tag{4.2}$$

which implies in particular that $h \in \mathcal{B}(1,\{v_3,\dots,v_m\})$.

The value of $f$ on the (disjoint) sets $\{v_1 = 0, v_2 = 0\}$, $\{v_1 = 1, v_2 = 0\}$ and $\{v_1 = 0, v_2 = 1\}$ is

| $v_1$ | $v_2$ | $f$ on that set |
|---|---|---|
| 0 | 0 | $h(\bar{v})$ |
| 1 | 0 | $f_1(\bar{v}) + h(\bar{v})$ |
| 0 | 1 | $f_2(\bar{v}) + h(\bar{v})$ |

So in particular, each of these sets contains a codeword of a $\mathcal{R}(1,m-2)$, and they are related by the condition (4.2). So, up to permutation of coordinate positions, we have

$$\mathcal{R}(2,m)_{\text{supp}(x)} \subseteq \mathcal{R}(1,m-2) \times \mathcal{R}(1,m-2) \times \mathcal{R}(1,m-2),$$

where the factors $\mathcal{R}(1,m-2)$ correspond to the positions with $\{v_1 = 0, v_2 = 0\}$, $\{v_1 = 1, v_2 = 0\}$ and $\{v_1 = 0, v_2 = 1\}$ respectively.

Codes that are subspaces of Cartesian products of nontrivial codes are called *concatenated codes*, and their block decomposition can be recovered with algorithms that will be discussed later on.

In our case, by recovering the block decomposition of the code, we get the sets $\{v_1 = 0, v_2 = 0\}$, $\{v_1 = 1, v_2 = 0\}$ and $\{v_1 = 0, v_2 = 1\}$. Now consider the words of length $2^m$ that have ones exactly on the support of $x$ and on one of the three sets. These words can be expressed as Boolean functions:

| Ones on the set | Corresponding function |
|---|---|
| $\{v_1 = v_2 = 1\} \cup \{v_1 = 1, v_2 = 0\}$ | $v_1$ |
| $\{v_1 = v_2 = 1\} \cup \{v_1 = 0, v_2 = 1\}$ | $v_2$ |
| $\{v_1 = v_2 = 1\} \cup \{v_1 = 0, v_2 = 0\}$ | $1 + v_1 + v_2$ |

Note that whatever the choice of the selected set is, the corresponding function is in $\mathcal{B}(1,\{v_1,v_2\})$, so the constructed word is a codeword of $\mathcal{R}(1,m)$; this shows how the reduction works in the case $r=2$.

We now look at the general case $r>1$, being slightly more formal this time. Let $x=v_1\cdots v_r$ be a minimum weight codeword in $\mathcal{R}(r,m)$, and $\mathcal{C}_{\mathrm{supp}(x)}$ the code $\mathcal{R}(r,m)$ shortened on the support of $x$. Write $\bar{v}=(v_{r+1},\ldots,v_m)$, and let $f$ be a codeword in $\mathcal{C}_{\mathrm{supp}(x)}$. Then we can write $f$ as

$$f(v_1,\ldots,v_r,\bar{v})=\sum_{I\in 2^{\{1,\ldots,r\}}} f_I(\bar{v})\cdot\prod_{i\in I} v_i,$$

where for each $I\subseteq\{1,\ldots,r\}$, we have $f_I\in\mathcal{B}(r-|I|,\{v_{r+1},\ldots,v_m\})$. The condition that $f$ be 0 on $\{v_1=v_2=\cdots=v_r=1\}$ implies

$$0=\sum_{I\in 2^{\{1,\ldots,r\}}} f_I(\bar{v}), \tag{4.3}$$

and shows in particular that $f_\emptyset(\bar{v})\in\mathcal{B}(r-1,\{v_{r+1},\ldots,v_m\})$. Therefore, the shortened code is in fact a concatenated code with the inner codewords being on the disjoint sets determined by the value of $(v_1,\ldots,v_r)$, that is, up to permutation, we have

$$\mathcal{C}_{\mathrm{supp}(x)}\subseteq\underbrace{\mathcal{R}(r-1,m-r)\times\cdots\times\mathcal{R}(r-1,m-r)}_{2^r-1\text{ times}}.$$

There are, however, more conditions to the concatenation than just the one implied by (4.3).

We shorten on supp$(x)$, i.e., the set $\{v_1=\cdots=v_r=1\}$, so there are $2^r-1$ sets, and each is of length $2^{m-r}$.

We apply the algorithm to recover concatenated codes of the next section (Algorithm 1) to find the sets and then construct a word $y$ of length $2^m$ that has ones exactly on the points $\{v_1=\cdots=v_r=1\}\cup S$, where $S$ is one of the determined sets, say, the set $S$ verifying

$$\{v_1=v_2=\cdots=v_\ell=0,v_{\ell+1}=v_{\ell+2}=\cdots=v_r=1\}.$$

We can write supp$(x)\cup S$ as follows:

$$\{v_1=v_2=\cdots=v_\ell=0,v_{\ell+1}=v_{\ell+2}=\cdots=v_r=1\},$$

or, even more explicitly,

$$\{v_1=v_2\wedge v_2=v_3\wedge\cdots\wedge v_{\ell-1}=v_\ell\wedge v_{\ell+1}=1\wedge\cdots\wedge v_r=1\}.$$

Translating conditions of the form "$a=b$" to the expression $(1+a+b)$, we see that this set is equal to the support of

$$y:=(1+v_1+v_2)(1+v_2+v_3)\cdots(1+v_{\ell-1}+v_\ell)v_{\ell+1}\cdots v_r.$$

Now, $\deg(y) \leq 1 \cdot (\ell - 1) + 1 \cdot (r - \ell) = r - 1$ (in fact, we have equality), which shows that $y \in \mathcal{B}(r-1, \{v_1, \dots, v_r\})$. So we have constructed a word of order $r-1$, as desired. Note that one can write $x = v_i y$ for any $1 \leq i \leq \ell$, giving factorizations of the sought form.

## 4.3 Finding inner words in the shortened code

To distinguishing the sets with different values of $(v_1, \dots, v_r)$, we use the fact that the code is a concatenated code, with an inner codeword on each of these sets.

Several approaches to recover concatenated codes are conceivable, for example in the paper [MS2007], we gave a solution that works on the code itself. In this presentation, we will make use of Sendrier's algorithm from [S1994] instead.

The idea of Sendrier's algorithm is as follows: If a code is concatenated, then any parity check on one of the inner blocks is a valid parity check for that code. Since the inner blocks are comparatively short, such parity checks have low weight. Now, assume that all the sufficiently low weight words in the dual code have their support within one inner block, and let $x = (x_1, \dots, x_n)$ be such a low weight word in the dual code. Then if $x_i = 1 = x_j$ for some indices $1 \leq i, j \leq n$, this implies that position $i$ and $j$ are in the same inner block. By collecting enough such witnesses, the concatenated structure of the code can therefore be recovered.

Of course, for this to work, we must show that the lowest weight words in the dual have their support within a single inner word. In order to apply the algorithm in our setting, we need to check the following:

**Proposition 6** *The dual code of the shortened code has minimum distance at most $2^r$, and any word of weight $2^r$ or less has support within a single inner block.*

To prove this, we first note that by equations (2.3) and (2.9), the dual code is the puncturing on the positions $\{v_1 = \cdots = v_r = 1\}$ of $\mathcal{R}(m-r-1, m)$. The code $\mathcal{R}(m-r-1, m)$ has minimum distance $2^{r+1}$. The dual of any of the inner codes is $\mathcal{R}(m-2r, m-r)$, and has minimum distance $2^r$. Since any parity check for an inner block is also a parity check for the complete code, this shows that the puncturing of $\mathcal{R}(m-r-1, m)$ has minimum distance at most $2^r$.

We now need to show that any word of weight (at most) $2^r$ has support within a single inner block. Let $f \in \mathcal{B}(m-r-1, \{v_1, \dots v_m\})$ be a nonzero codeword.

The function $f$ can be written as a polynomial in $\{v_{r+1}, \dots, v_m\}$ with each coefficient being a polynomial in $\{v_1, \dots, v_r\}$. With this notation, pick a monomial of $f$ that has maximum degree in $\{v_{r+1}, \dots, v_m\}$. Let $o$ be its degree. The coefficient of this monomial is a Boolean function $g(v_1, \dots, v_r)$ in the variables

$\{v_1, \ldots, v_r\}$. Let $\ell$ be the degree of $g$. We can assume $\ell < r$, since otherwise the factor $v_1 \cdots v_r$ appears in $g$, but this factor can be removed, since the positions where $v_1 \cdots v_r = 1$ have been punctured.

Now fix a vector $(c_1, \ldots, c_r) \in \mathbb{F}_2^r$ such that $g(c)$ is nonzero. Then the restriction $f|_{\{v_1=c_1,\ldots,v_r=c_r\}}$ is nonzero. Since $f|_{\{v_1=c_1,\ldots,v_r=c_r\}} \in \mathcal{B}(o, \{v_{r+1}, \ldots, v_m\})$, the weight of $f$ on $\{v_1 = c_1, \ldots, v_r = c_r\}$ is therefore at least $2^{m-r-o}$.

Since $g \in \mathcal{B}(\ell, \{v_1, \ldots, v_r\})$, the function $f$ is nonzero on at least $2^{r-\ell} - 1$ inner sets in the punctured code. We have $\ell + o \leq \deg(f)$, and so the following estimate for the weight of $f$ on the punctured code holds:

$$\mathrm{wt}(f) \geq 2^{m-r-o}(2^{r-\ell} - 1) \geq 2^{\ell+1}(2^{r-\ell} - 1) = 2^{r+1}(1 - 2^{\ell-r}).$$

As noted before, we have $\ell < r$. If $\ell = r - 1$, we have the guarantee that any nonzero inner word will have weight at least $2^r$, proving the result in this case. If $\ell < r - 1$, the above weight bound is $> 2^r$, proving the result in this case.

Because of Proposition 6, we can apply Sendrier's algorithm (Algorithm 1).

**Algorithm 1** Sendrier's algorithm

```
Require: $\mathcal{C}^\perp = (\mathcal{R}(r,m)^\sigma_{\mathrm{supp}(x)})^\perp$
Ensure: Block decomposition represented as connected components of $\mathcal{C}$.
  Let $V = \{1, \ldots, 2^{m-r}(2^r - 1)\}$ and $E = \emptyset$.
  while the graph $(V, E)$ has more than $2^r - 1$ connected components do
    Sample a word $x$ of weight $\leq 2^r$ in the punctured dual code.
    for each $(i, j)$ such that $x_i = 1 = x_j$ do
      Add $(i, j)$ to $E$
    end for
  end while
```

At the end of the execution, each connected component in this algorithm corresponds to an inner set.

## 4.4 The case $r = 1$

Consider the matrix $A$ formed by the rows corresponding to the codewords $v_m$, $v_{m-1}, \ldots, v_1$ of the (unpermuted) $\mathcal{R}(1, m)$. By construction, the $i$-th column of this matrix is just the number $i - 1$, if we read the vector as a binary number. Any possible binary vector of length $m$ appears exactly once among the columns of this matrix, and if we add the all-one row, we get a generator matrix for a first-order Reed-Muller code.

Now, let $f_1, f_2, \ldots, f_m, f_{m+1}$ be a random basis of $\mathcal{R}(1, m)^\sigma$. If the all-one codeword is not a linear combination of $f_1, \ldots, f_m$ only, then in the matrix $A^\sigma$ formed by the rows $f_1, \ldots, f_m$, each column-vector is distinct.

Thus, we can just reorder the columns by moving the zero vector to the first position, etc., and thus obtain the matrix $A$. The same permutation applied to the positions of $\mathcal{R}(1, m)^\sigma$ will then yield $\mathcal{R}(1, m)$.

In order to make sure that the linear combination of the all-one word has a nonzero coefficient for $f_{m+1}$, it is enough to build the matrix $A^\sigma$ as follows: First write the matrix $G$ having as rows the vectors $f_1, \dots, f_{m+1}$. Then pivot this matrix on some non-zero element in the lowest column, i.e., the one corresponding to $f_{m+1}$. Now, by removing this last row, we get the matrix $A^\sigma$.

## 4.5 Running time analysis

In the analysis, we will take the quantity $n = 2^m$ (the block length) as the input length, and we will assume $r$ to be small with respect to $m$ which leads to a low rate setting. This assumption is based on the fact that Reed-Muller codes behave very poorly when $r$ is large, and are therefore practically useless in these instances. For this reason, we will assume $r/m \to 0$ and $r < m/2$. In practice, $r$ is usually a small constant. See [DS2006] for trade-offs between $r$, $m$ and decoding thresholds.

The only computationally hard operation of the attack is the one of finding low weight words in a code, everything else is polynomial time. Thus, in order to determine the running time up to a polynomial factor, it is sufficient to verify that only a polynomial number of low weight words is needed, and then to restrict attention to the low weight word finding algorithm.

Checking that only a polynomial number of low weight words has to be found is straightforward: In order to find a single vector in $\mathcal{R}(r-1, m)^\sigma$, a minimum weight word in the original code has to be found, and then the concatenated structure of the shortened code has to be recovered. Sendrier's algorithm is typically done after any position in the dual code is being covered by the support of at least one minimum weight codeword, so $O(n \log(n))$ such minimum weight codewords have to be sampled.

Thus, finding a single vector of $\mathcal{R}(r-1, m)^\sigma$ needs the sampling of a polynomial number of low weight words. But then, since $k \leq n$, so does clearly the sampling for a complete basis of $\mathcal{R}(r-1, m)^\sigma$. And given that $r \ll n$, the reduction to $\mathcal{R}(1, m)^\sigma$ requires still only a polynomial number of samples.

Because of this, we conclude that, in the exponent, only the complexity of the low weight word finding algorithm matters asymptotically, and we restrict our attention to this algorithm. In practice, those polynomial factors do of course matter to some extent, but notice that the degree of the polynomial is not very large.

### 4.5.1 Finding very low weight codewords

As we have seen in section 2.2, the problem of finding very low weight words is generally intractable for linear codes, but can be doable in certain settings.

In our case, we have to consider two low weight word finding instances; the first one is that of finding minimum weight words in $\mathcal{R}(r,m)$, and the second one is invoked when recovering the concatenated structure, where we have to find minimum weight words in the dual of the shortened code.

We first consider the minimum weight word finding problem in $\mathcal{R}(r,m)$. We use the information set decoding algorithm. Let $G$ be a generator matrix of $\mathcal{R}(r,m)$ diagonalized on some information set $I$.

The condition for a specific word in $G$ of weight $w$ to pop up as a row in such a diagonalized matrix is that exactly one of its bits is inside the information set and the other ones are outside. To simplify, we instead compute the probability that none of its bits are in the information set, a probability which is a bit smaller. In the low rate setting, we can approximate this by noting that if $k$ is small compared to $n$, the probability that none of the positions of $I$ match with the support of the word of weight $w$ is roughly

$$\left(1-\frac{w}{n}\right)^k. \tag{4.4}$$

This probability becomes large if $k$ is very small with respect to $n$ (i.e., the rate is very low), or if $w$ is very small.

Note that (4.4) estimates the probability of finding a *single* word of the given weight given a random set $I$. If many words of the desired weight exist, the probability has to be multiplied with the number of such words.

The second problem, that of finding low weight words in the dual code, is much easier: In this case, we are in the high-rate setting and we seek words of weight $2^r$. For fixed $r$ we can invoke the arguments from section 2.2.2 to conclude that this problem is polynomial time.

### 4.5.2 Finite-length analysis

The goal of this section is to specialize (4.4) to the case of Reed-Muller codes, and to derive a crude bound which allows to estimate the feasibility of the low weight word finding problems (and thus the attack) for different values of $r$ and $m$.

We first study the hardness of the minimum weight word finding procedure for Reed-Muller codes. In this case, we have $w = 2^{m-r}$ and

$$k = \sum_{i=0}^{r} \binom{m}{i} \le \frac{m-r+1}{m-2r+1} \cdot \frac{m^r}{r!}.$$

If we plug this into (4.4), we get the hit probability of at least

$$\exp\left\{\frac{m-r+1}{m-2r+1} \cdot \frac{m^r}{r!} \cdot \ln\left(1-2^{-r}\right)\right\} \tag{4.5}$$

for a single codeword per information set. By Proposition 3, there are at least $2^{mr-r(r-1)}$ such words, and so the cost for finding any one of them can be estimated to be at most

$$2^{-\frac{m-r+1}{m-2r+1}\cdot\frac{m^r}{r!}\cdot\log_2\left(1-2^{-r}\right)-mr+r(r-1)} \tag{4.6}$$

diagonalizations of the generator matrix. This rough estimate predicts, for example, that finding a minimum weight word in $\mathcal{R}(3,11)$ would cost roughly $2^{37}$ diagonalizations, and thus finding such words is feasible in that case.

As expected and easily seen by comparing to real running times, the bound (4.6) is somewhat pessimistic, i.e., it overestimates the running time. For example, finding a minimum weight word in $\mathcal{R}(3,11)$ needs only about $2^{17}$ diagonalizations in practice. More precise estimates are of course possible, but result in uglier formulas.

The other low weight word finding problem, that of finding minimum weight words in the dual code, can be estimated with (2.7) if the value of $r$ is small, giving the running time estimate

$$2^{2^r\left(m-\log(\frac{m-r+1}{m-2r+1})-r\log(m)+\log(r!)\right)},$$

which is $O(n^{2^r})$ if regarded as a function of $n=2^m$ only.

### 4.5.3 Asymptotic analysis

Asymptotically, the running time for the algorithm is

$$O(\text{poly}(n))\cdot e^{O(\text{poly}(\log(n)))} \tag{4.7}$$

for any fixed value of $r$.

To see this, we start again with (4.5). Using the assumption that $r/m \to 0$, and writing the expression in terms of the block length $n=2^m$ instead of $m$, we see that this probability behaves like

$$\exp\left\{-\log_2(n)^r C_r(1+o(1))\right\},$$

where $C_r$ is a constant depending only on $r$. This time, we assume there is just a single minimum weight codeword, and thus a conservative estimate for the number of trials to find a minimum weight word is

$$k=\sum_{i=0}^{r}\binom{m}{i}\leq\frac{m-r+1}{m-2r+1}\cdot\frac{m^r}{r!},$$

thus we can bound (4.4) by

$$\exp\left\{\frac{m-r+1}{m-2r+1}\cdot\frac{m^r}{r!}\cdot\log\left(1-2^{-r}\right)\right\}.$$

Using the assumption that $r/m \to 0$, and writing the expression in terms of the block length $n = 2^m$ instead of $m$, we get that this probability behaves like

$$\exp\left\{-\log_2(n)^r C_r(1+o(1))\right\},$$

where $C_r$ is a constant depending only on $r$. Therefore, the number of trials to find a minimum weight word can be estimated to be at most

$$C_{\mathrm{lw}} := \exp\left\{\log_2(n)^r C_r(1+o(1))\right\}. \tag{4.8}$$

Using the fact that only a polynomial (in $n$) number of samplings is needed, we conclude that (4.7) is is indeed a bound for the running time of the algorithm.

For large $r$, the numbers become very large. That is not an artifact: If the code is not sufficiently low-rate, then finding minimum weight becomes a very hard problem; since the algorithm depends on the feasibility of this.

## 4.6 Experimental running time

To observe the behavior of the algorithm in real life, we did a number of test-runs on our implementation on a PC with a 2.4GHz processor. Our implementation uses rather simple low weight word finding algorithms, and not elaborate ones like, e.g., the ones described in [CC1998]. The following table lists obtained averages for ten runs, except for the case (*), that is ($m = 11, r = 4$), which is the data of a single run.

We only computed the values for $m > 2r$; in the other case, the attack can be carried out more efficiently on the dual code.

| | $r = 2$ | $r = 3$ | $r = 4$ |
|---|---|---|---|
| $m = 5$ ($n = 32$) | 0.0015s | | |
| $m = 6$ ($n = 64$) | 0.0023s | | |
| $m = 7$ ($n = 128$) | 0.009s | 0.03s | |
| $m = 8$ ($n = 256$) | 0.04s | 0.18s | |
| $m = 9$ ($n = 512$) | 0.24s | 1.26s | 2m 57s |
| $m = 10$ ($n = 1024$) | 1.77s | 16.15s | 22h 49m 57s |
| $m = 11$ ($n = 2048$) | 12.14s | 5m 20.8s | 10d 11h 55m (*) |

As predicted by the analysis, the performance changes a lot with $r$, and degrades quickly for larger such values. This does indeed exhibit a limit of our attack, but note that since the performance of Reed-Muller codes degrades with large $r$, choosing such values would very likely open the doors to other attacks.

# Chapter 5

# Final thoughts

## 5.1 Algebraic curves of genus $g > 1$

We have seen that algebraic codes of genus $g \leq 1$ do not lead to a viable cryptosystem, what about the case $g > 1$? The question is still open, and of considerable interest, as we will now show.

### 5.1.1 Why algebraic geometry codes?

Algebraic geometry codes have a number of stunning properties. We start by citing the result that algebraic geometry codes over a sufficiently large alphabet have an extremely good relative minimum distance for fixed rate.

**Theorem 3 (Tfasman, Vlăduţ, Zink)** *Set $q = p^{2m}$ for some prime $p$, and let $\mathbb{F} = GF(q)$. Then, asymptotically, there exist algebraic geometry codes over $\mathbb{F}$ of relative distance $\delta$ and rate $R$ such that*

$$R \geq 1 - \delta - \frac{1}{\sqrt{q} - 1} - \varepsilon$$

*for any $\varepsilon > 0$.*

In particular, if $q \geq 7^2$, these codes exceed the Gilbert-Varshamov bound (2.1) for certain non-trivial rates. For more details, see [TVZ1982].

A good minimum distance is of course not sufficient to make codes interesting for McEliece type cryptosystems: Rather, we want the codes also to correct a large number of errors. The good news is that there are polynomial time decoders that can decode algebraic geometry codes to up to half the minimum distance using, e.g., the algorithm by Vlăduţ [V1990], and beyond, using the decoder by Shokrollahi and Wasserman, [SW1999].

It follows that algebraic geometry codes lead to a McEliece type cryptosystem for which the direct decoding attack is fully exponential in the block length. In other words, they provide the best possible construction for such cryptosystems.

At the current state of affairs, algebraic codes appear to be the only family of codes which are known to have this property and that have not been broken in other respects: Some other codes, such as many graph codes and concatenated codes also correct a fixed fraction of errors, but they cannot be used in the McEliece cryptosystem due to other weaknesses. None of the other constructions of codes are currently known to correct a constant fraction of errors.

We note that while the argument we just gave is asymptotic, it is also applicable to reasonable, finite block lengths. Theorem 3 exists in non-asymptotic formulation, and Janwa and Moreno gave a number of examples [JM1996] of codes that compare favorably to most other constructions in terms of key size for a fixed security level.

### 5.1.2 Structural weaknesses of algebraic geometry codes

We will now examine whether, and to what extent, the proposed methods for elliptic codes can be generalized to the case of algebraic geometry codes. Some of the steps of the attack against elliptic codes can can be generalized in a straightforward way, but there are also a number of problems.

For the sake of concreteness, we will not try to do a general analysis, but focus on algebraic geometry codes defined over plane, smooth curves instead.

Thus, we are given an arbitrary basis of an algebraic geometry code

$$\mathcal{C} := \mathrm{AGC}(\hat{\mathcal{X}}, \hat{\Delta}, (\hat{P}_1, \dots, \hat{P}_n)),$$

where $\hat{\mathcal{X}}$ is an unknown, smooth, plane, projective curve, $\hat{\Delta}$ is an unknown divisor, and $\hat{P}_1, \dots, \hat{P}_n$ are unknown points on $\hat{\mathcal{X}}$. We wish to determine a curve $\mathcal{X}$ that is $\mathbb{F}$-isomorphic to $\hat{\mathcal{X}}$, a divisor $\Delta$, points $P_1, \dots, P_n$, and constants $c_1, \dots, c_n$ such that

$$\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n), (c_1, \dots, c_n)).$$

#### Recovering $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$

Our first task is to determine the degree 0 part of the divisor class group, $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$. Note that in general, the divisor class group can not be identified with the points on the curve.

We need an assumption corresponding to (3.3) for elliptic codes. Namely, we will assume

$$\gcd(|\mathrm{Pic}^0(\mathcal{X})|, \deg(\Delta)) = 1. \tag{5.1}$$

We use (5.1) in the following way: Fix a rational point $Q \in \mathcal{X}$. Then

$$\begin{aligned} \Delta - \deg(\Delta)\langle Q\rangle &= \Delta' \\ &\sim \deg(\Delta)\Delta'', \end{aligned}$$

for some divisors $\Delta'$ and $\Delta''$ of degree 0, and the equivalence follows from the fact that multiplication by $\deg(\Delta)$ is invertible in $\mathrm{Pic}^0(\mathcal{X})$, given (5.1). Then

$$\begin{aligned}\Delta &\sim \deg(\Delta)\Delta'' + \deg(\Delta)\langle Q\rangle \\ &= \deg(\Delta)\underbrace{\left(\Delta'' + \langle Q\rangle\right)}_{=:\Delta_0},\end{aligned}$$

where $\deg(\Delta_0) = 1$. Therefore,

$$\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \deg(\Delta)\Delta_0, (P_1, \ldots, P_n), (c_1, \ldots, c_n)).$$

We seek a representation of the divisor class group. The group $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$ does not in general have a structure as simple as the one given by (3.4), and it will typically also be much larger than $|\mathcal{X}(\mathbb{F})|$. However, even on more complicated curves, $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$ is a finite Abelian group, and so it is isomorphic to a direct sum of groups of the form $\mathbb{Z}/d_i\mathbb{Z}$ with different $d_i$s. So, there is a $\mathbb{Z}$-module $\mathcal{G}$ that is isomorphic to $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$. Our task is to reconstruct $\mathcal{G}$ (i.e., determine the $d_i$), and to compute, for each $P_i$, an element $z_i \in \mathcal{G}$ corresponding to $\langle P_i\rangle - \Delta_0$ in $\mathcal{G}$.

As before, a minimum weight codeword in $\mathcal{C}$ yields a relation among the $z_i$ in the $\mathbb{Z}$-module $\mathcal{G}$.

Finding minimum weight words grows harder as the genus increases, since the minimum distance then moves away from the Singleton bound. If the genus is not too large, which is the setting of interest in practice, minimum weight words can still be collected, although the effort to do so is larger. In the elliptic curves case, we could give fairly accurate measures of the hardness of these problems. Doing the same here would need better knowledge of the group $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$. A guess can be made by invoking a random code argument, however: We consider a codeword that we obtained by diagonalizing the generator matrix on an information set. Apart from the zero positions implied by the information set, we need $g$ more zeros, so for random codes, this would lead to a success probability of

$$\binom{n-k}{g} q^{-g}(1-q^{-1})^{n-k}.$$

To get a rough estimate for small $g$, we set $n \approx q$, and find that the probability that a given row in this generator matrix is minimum weight is then roughly

$$\frac{(1-R)^g e^{R-1}}{g!},$$

which shows that for small $g$, this step is likely doable.

Since the values of $d_i$ may be very large in practice, the approach to use different trial moduli that we suggested in the case of elliptic curves does not lead to a satisfactory solution. The candidate values for $d_i$ can be found with the following strategy instead: Select $n$ $\mathbb{Q}$-linearly independent constraints (from

$n$ minimum weight codewords), and compute the determinant of the resulting $n \times n$ constraint matrix. That determinant must be zero modulo any $d_i$. A few such determinants could be collected, with each $d_i$ being a divisor of the greatest common divisor of all these determinants. Call this greatest common divisor $M$.

Factoring $M$ then yields the relevant primes appearing in the $\mathbb{Z}$-module. For each factor $p$ of $M$, try to find a solution modulo $p$ of all constraints: If no such solution exists, there is no component $\mathbb{Z}/p\mathbb{Z}$ in the group. Otherwise, we get $r$, say, linearly independent solutions. Each of these solutions corresponds to a factor $\mathbb{Z}/p^m\mathbb{Z}$ for varying powers $m$. For each such component, detect the right power $m$ by lifting the solution.

We estimate the hardness of the determinant computation as follows. Each relation has Euclidean norm $n-k-g+1$, so the determinant can be at most $(n-k-g+1)^n \approx n^n(1-R)^n$, and can thus be represented in $n\log(n(1-R))$ bits. Assuming the intermediate values in the computation are not much larger than that, one can conclude that the running time should be $O(n^4\log(n))$, with a memory usage of $O(n^3\log^3(n))$.

Since several determinants have to be computed, it is probably worthwhile just to compute the first one exactly, and all the successive ones modulo the greatest common divisor of the previous ones only. That way, the first determinant computation is hopefully by far the most expensive one.

## Recovering the curve

The rule to detect collinear points on an elliptic curve can be generalized. The following form of Bezout's theorem (see [H2003, p. 182]) is useful:

**Theorem 4 (Bezout)** *Let $C$ and $C'$ two plane, smooth, projective curves over an algebraically closed field $\overline{\mathbb{F}}$ of degrees $d$ and $d'$ respectively. Assume that $C$ is smooth, and that $C$ is not a component of $C'$. Then $C$ and $C'$ intersect in precisely $dd'$ points, counting multiplicities.*

So lines will cross the curve at $d$ points, where $d$ is the degree of $\mathcal{X}$.

For simplicity, we sidestep the recovery schedule. Instead we seek minimum weight codewords $u, v$ and $w$ corresponding to three functions $f_1, f_2$ and $f_3$, such that

$$\begin{aligned} f_1 &= g \cdot \ell_1, \\ f_2 &= g \cdot \ell_2, \\ f_3 &= g \cdot \ell_3, \end{aligned}$$

where the $\ell_i$ are polynomials of degree 1, and thus have collinear zeros.

Henceforth, we will restrict generality and assume that the curve has a unique point at infinity $\mathcal{O}$ and that $\Delta_0 = \langle \mathcal{O} \rangle$. This latter assumption is rather restrictive assumption, and we will discuss it later on.

If it holds true, we can again try to solve the appropriate subset sum problems over the $z_i$ to find the minimum weight codewords $u$, $v$ and $w$: For each $\ell_i$ find a sum of $\leq d$ variables $z_j$ summing to zero, and for $g$ find a sum of $\deg(\Delta) - d$ such $z_i$.

The function $f_1/f_3 = \ell_1/\ell_3$ is a fraction of polynomials of degree one (in two variables), and can thus be determined if its value on five points is known. The same points can be used to determine $f_2/f_3$. Now, since we know the evaluations of $\ell_1/\ell_3$ and $\ell_2/\ell_3$ on some points, we can use this to compute the exact coordinates of those points: The equation

$$\frac{\ell_1}{\ell_3}(P_i) = \frac{u_i}{w_i}$$

provides a linear relation on the coordinates $(x_i, y_i)$ of $P_i$, and so does the corresponding equation with $\ell_2/\ell_3$, resulting in an unique solution $(x_i, y_i)$, if the lines corresponding to $\ell_1$ and $\ell_2$ are not parallel.

Once enough coordinate pairs have been recovered, we can again solve for the curve equation.

### Restrictions

Once the curve is reconstructed, a few more steps are needed to reconstruct the code completely. For the sake of brevity, though, we will omit a discussion of these further steps and concentrate on the curve recovery problem.

The algorithm as we presented it above may not work in all cases: We made a number of assumptions that deserve more attention. In particular we assumed:

1. $\Delta = \deg(\Delta)\langle\mathcal{O}\rangle$,
2. The implied subset sum problems can be solved.

We will now discuss these points.

**Divisor choice, $\Delta = \deg(\Delta)\langle\mathcal{O}\rangle$.** We assumed we could pick $\Delta = \deg(\Delta)\langle\mathcal{O}\rangle$, i.e., $\Delta_0 = \langle\mathcal{O}\rangle$.

Recall that in the case of elliptic curves, this assumption can always be made because the points on the curve can be suitably translated. In general, no such translation map exists.

The linear system that yields $\mathcal{G} \cong \mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$ and finds a representative of $\langle P\rangle - \Delta_0$ for each evaluated point $P$, can be slightly modified so that $\Delta_0$ can be freely selected: It is enough to add an unknown corresponding to $\Delta_0$ to the system of equations. (We skip the technicalities to show that this works. Note that $\deg(\Delta_0) = +1 \neq 0$, so $\Delta_0$ does not trivially have a representative in $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$. But there is a surjective homomorphism $\mathrm{Pic}_{\mathbb{F}}(\mathcal{X}) \to \mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$ that fixes $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$.)

The fact that $\Delta_0$ can be freely selected can be used in the following ways: First, one may want to run through the choices of $\Delta_0$ by adding another unknown to the system where the solution for the group is sought. Without further tweaks, this is however impractical given the rapid growth of the cardinality of $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$ with $g$.

Second, if the curve in question has specific geometric features, one may be able to find the $\Delta_0$ corresponding to $\langle \mathcal{O} \rangle$ much quicker. For example, hyperelliptic curves have the property that each generic point has a corresponding opposite, and the line passing through these two points passes through no other finite point. Under the large point set assumption, it should therefore be often easy to detect the correct $\Delta_0$ by seeking the solution such that for most $z_i$, there is a $z_j$ with $z_i = -z_j$. This can be done probabilistically in $O(n)$ time.

**Subset sum problems.** We assumed that the subset sum problems that need solving can be solved. So, we assume first that those solutions exist, and, second, that they are easy to find.

In the case of the attack against elliptic curves, we used the fact that the number of choices we had exceeded the size of $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$, and that we could therefore make a choice almost randomly.

This is not true for higher genus curves: The group $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$ grows very quickly with $g$, which makes it very questionable whether such solutions even exist for large $g$.

Geometrically speaking, the problem can be restated as follows: We assumed that there are many sets of $d$ (where $d$ is the degree of the curve) collinear rational points on the curve. For elliptic curves, this is indeed true: Given two rational points on the curve, we know that the third point on the curve that is collinear with the first two is indeed rational as well.

In general, this does not hold: For an arbitrary curve, if we pick 2 rational points on $\mathcal{X}$, then there is little reason to believe that the $d-2$ remaining collinear points on $\mathcal{X}$ are all rational as well.

We will now give a rough analysis of this problem in the case of hyperelliptic curves. We have already mentioned that vertical lines have special properties in this case. We will exclude them, and note that to make the given algorithm work, we only need to find one non-vertical line passing through $d$ points. Hyperelliptic curves have degree $d = 2g + 1$. Let

$$y = \alpha x + \beta \tag{5.2}$$

be a rational line, i.e., with $\alpha, \beta \in \mathbb{F}$. If we substitute $\alpha x + \beta$ for $y$ in the curve equation, we get a polynomial in $x$ only that has its zeros on the coordinate positions where the line crosses the curve. Therefore, the line will cross the curve on rational points if and only if this polynomial factors into linear factors in $\mathbb{F}$.

To estimate the number of such lines, we make the heuristic assumption that substituting (5.2) into the curve equation yields a random polynomial of degree $d$. Therefore, we expect the probability that a random line is purely rational to be approximately equal to

$$\frac{\#\{\text{products linear factors of degree} \leq d\}}{\#\{\text{polynomials of degree} \leq d\}}.$$

The numerator is roughly equal to $q \cdot \frac{q^d}{d!}$, and the denominator is equal to $q^{d+1}$, giving a total hit probability of $1/d!$. Since there are $q^2$ lines, we would expect a random hyperelliptic curve to contain

$$\frac{q^2}{d!} \tag{5.3}$$

such rational lines; in particular, if this quantity is much larger than 1, we can hope that such lines exist.

For the case where $q = 131$, we obtain the following numbers:

| genus | degree | Eq. (5.3) | avg over 10 curves |
|---|---|---|---|
| $g = 1$ | $d = 3$ | 2860 | 3156 |
| $g = 2$ | $d = 5$ | 143 | 168 |
| $g = 3$ | $d = 7$ | 3.4 | 4.5 |
| $g = 4$ | $d = 9$ | 0.04 | 0 |

The numbers have been rounded. The experimental average above was obtained by taking random curves and exhaustively testing all the non-vertical lines.

These computations suggest that for $g \leq 3$, it is reasonable to expect that good lines exist, but for curves of higher degree, this will not usually be the case. It should be noted, though, that the evaluated point set should be very large even for moderate $g$.

Note that finding appropriate collinear points, if they exist, is not a bottleneck. Even for $g = 3$, a baby-step-giant-step like approach should solve this problem in $O(n^4)$ once the structure of $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$ has been determined.

This discussion suggests that for specific curves, such as hyperelliptic curves of small genus, the algorithm can be made to work. A definite answer can only be given after a more extensive analysis, though. Evidently, without further improvements, the algorithm will not work for larger genera or general curves.

Several methods to sidestep the problems are conceivable. For example, instead of limiting the focus to linear functions, small degree polynomials may be worth considering. Unfortunately, this modification increases the number of points that have to be guessed drastically. It may be the case, but appears unlikely, that for larger degree curves there are significantly more curve isomorphisms, which would again reduce the search space.

To summarize, more research is needed to make this algorithm work in general. My feeling is that especially the behavior of the group $\mathrm{Pic}^0_{\mathbb{F}}(\mathcal{X})$ would need more attention in order to construct an algorithm that would work in general.

## 5.2 Large subcodes of Reed-Solomon codes

In Section 3.1.3 we discussed the effect of subcode constructions in the setting of Reed-Solomon codes. The essential observation was that if the subcode is sufficiently small so that words having weight very close to minimum weight cannot efficiently be found, then attacks relying on finding minimum weight words in the supercode can no longer be practically carried out. It is however not known whether other practical attacks can be constructed.

Berger and Loidreau have previously studied the subcoding trick in [BL2005]. They conclude that reducing the dimension by 4 is sufficient to provide structural safety. We will now present a modified Sidelnikov-Shestakov attack that appears to break this case in a feasible amount of time.

It should be noted that our attack is not the first one on this scheme. After having handed this thesis in, I have been informed that the Loidreau-Berger scheme has already been attacked by Wieschebrink [W2006]. The algorithm of Wieschebrink appears to be a different one than the algorithm presented here.

### 5.2.1 An algorithm for breaking subcodes of dimension $k-4$

We now discuss the algorithm step by step. Let

$$\mathcal{C}_0 := \mathrm{GRS}_{\mathbb{F}}(k, (\alpha_1, \dots, \alpha_n), (c_1, \dots, c_n))$$

be an $[n, k, n-k+1]$ generalized Reed-Solomon code. Let $\mathcal{C}$ be a random subcode of $\mathcal{C}_0$ of dimension $k-4$.

1. **Find a suitable pair $f_1, f_2$.** Diagonalize the generator matrix on random information sets, until a generator matrix is found which contains two rows $u$ and $v$ with three common zeros off the information set.

   Assume without loss of generality that the information set was on $\{1, \dots, k-4\}$, and that the three common zeros are $k-3$, $k-2$ and $k-1$. Furthermore, $u_1 = 1$, and $v_2 = 1$, i.e., the nonzero positions of $u$ and $v$ within the information set are 1 and 2 respectively.

   Let $f_1(x)$ be the polynomial corresponding to $u$ and $f_2(x)$ the polynomial corresponding to $v$, then

   $$(x-\alpha_2)\cdots(x-\alpha_{k-4})(x-\alpha_{k-3})\cdots(x-\alpha_{k-1}) \text{ divides } f_1(x),$$

   and thus, since $\deg(f_1) < k$,

   $$f_1(x) = (x-\alpha_2)\cdots(x-\alpha_{k-4})(x-\alpha_{k-3})\cdots(x-\alpha_{k-1}) \cdot h_1(x),$$

   where $\deg(h_1) \leq 1$. Similarly,

   $$f_2(x) = (x-\alpha_1)(x-\alpha_3)\cdots(x-\alpha_{k-1}) \cdot h_2(x)$$

with $\deg(h_2) \leq 1$. Therefore,

$$\frac{f_1}{f_2} = \frac{(x-\alpha_2)h_1(x)}{(x-\alpha_1)h_2(x)}$$

is a fraction of two quadratic polynomials.

2. **Find a second pair $g_1, g_2$.** Find a pair $g_1$ and $g_2$ having the same properties as $(f_1, f_2)$. Typically $(g_1, g_2)$ will be constructed with a different information set.

3. **Pick the guessing set.** Pick a set $L$ of 5 positions on which both $f_1/f_2$ and $g_1/g_2$ can be evaluated, i.e., positions such that the words corresponding to $f_2$ and $g_2$ are nonzero.

4. **Guess and reject.** For each $i \in L$, guess an arbitrary value for $\alpha_i$. Solve for the 6 coefficients of $f_1/f_2$, i.e., determine $a, b, c, d, e, f$ such that

   $$\frac{a\alpha_i^2 + b\alpha_i + c}{d\alpha_i^2 + e\alpha_i + f} = \frac{v_i}{w_i}$$

   for all $i \in L$. (Note that this is a homogeneous linear system.) Analogously, determine the coefficients of $g_1/g_2$.

   Use the knowledge of $f_1/f_2$ to determine candidate $\alpha_i$ for positions $i$ where $\alpha_i$ is not yet known. This requires solving a quadratic equation, and typically there may be up to two candidates. Check whether doing so with $g_1/g_2$ yields a common solution. If not, reject the guess and try with a new one.

5. **Complete.** Now that a significant number of the $\alpha_i$'s are known, use similar techniques to compute the remaining values of $\alpha_i$. Note that since at this point, many of the $\alpha_i$'s are already known, pairs with more differing zeros may be used as well.

6. **Recover the $c_i$.** We can again use the same solution used for the elliptic codes recovering algorithm to do this. One has to be careful not to make use of $\mathcal{C}^\perp$, though. For example, recover the constants for the parity check matrix instead.

### 5.2.2 Running time analysis

We give a rough, sketchy running time analysis of the algorithm.

We first estimate the probability of finding a suitable pair in a given information set. Each polynomial corresponding to a row $u$ in such a suitably diagonalized generator matrix corresponds to a product of $k-5$ linear factors multiplied by some polynomial $r_u(x)$ of degree 4 at most. We seek a pair of rows $v, w$ such that

$$\begin{aligned} r_u(x) &= (x-\alpha_{i_1})(x-\alpha_{i_2})(x-\alpha_{i_3})(a_u x - b_u) \\ r_v(x) &= (x-\alpha_{i_1})(x-\alpha_{i_2})(x-\alpha_{i_3})(a_v x - b_v). \end{aligned}$$

There are roughly

$$\frac{1}{3!}(n-k-4)^3(q^2-1)(q^2-1) \approx \frac{1}{6}(n-k-4)^3 q^4$$

suitable such pairs. Since the total number of possible pairs $(r_u, r_v)$ is

$$(q^5-1)^2 \approx q^{10},$$

and the selected subcode is random, we would expect a random pair to match with probability roughly equal to

$$\frac{(n-k-4)^3}{6q^6}$$

Since this probability is small, the probability for a matching pair given an information set can then be estimated to be

$$\frac{(k-4)(k-5)(n-k-4)^3}{2 \cdot 6 \cdot q^6}$$

We assume that checking for matches is much cheaper than diagonalizing, so we will omit estimating the cost of this step. The total number of diagonalizations to find $v, w$ would thus be roughly

$$\frac{2 \cdot 6 \cdot q^6}{(k-4)(k-5)(n-k-4)^3}.$$

Of course, it is much better to just pivot the matrix on one entry at each step, instead of diagonalizing each time anew; the hit probability would then decrease slightly, but this makes one diagonalization an

$$O(n(k-4))$$

operation.

For the guessing step, note that three of the 5 values for $\alpha_i$, $i \in L$ can be arbitrarily fixed due to curve isomorphisms, so roughly $q^2$ guesses will have to be checked, each of which can be rejected probabilistically in $O(1)$ time.

The remaining steps are very similar to the corresponding steps in the Sidelnikov-Shestakov attacks, and we will thus not reiterate the discussion here.

The effectiveness of this approach is probably best seen by considering the concrete example suggested by Berger and Loidreau in [BL2005].

### 5.2.3 The example of Berger and Loidreau

The example considered in [BL2005] is the $[255, 133, 123]$ generalized Reed Solomon code over $GF(256)$. They take a random $[255, 129, 123]$ subcode of this code.

The above computations suggest that finding a pair $f_1, f_2$ in this code takes $2^{17}$ diagonalizations. For a matrix of the given size with entries in $GF(256)$ this step can be executed in a few minutes on a PC. The guessing then takes merely $2^{16}$ steps of $O(1)$ cost, and should be much cheaper than finding the pairs.

### 5.2.4 Smaller subcodes

The strategy above can of course, with modifications, in principle be applied to subcodes of dimension $< k-4$ as well. We will briefly study the question of how small the subcode can be made, until this type of attack becomes impractical.

For simplicity, it is useful to assume $n \approx q$, i.e. make the large point set assumption. We also forget small additive constants in the factors and write $k = Rn$.

A design idea of the attack above was to find a good tradeoff between the cost of finding a suitable pair of codewords and the cost of the guessing procedure: if fewer common zeros off the information set are needed, finding pairs becomes easier, but the guessing cost increases. We introduce a variable $0 \le r \le \ell$ to choose the tradeoff.

The probability that a random of rows in a diagonalized generator matrix has $\ell - r$ common zeros off the information set is roughly

$$\frac{\frac{(1-R)^{\ell-r}q^{\ell-r}}{(\ell-r)!}q^{r+1}q^{r+1}}{q^{2(\ell+1)}} = q^{r-\ell}\frac{(1-R)^{\ell-r}}{(\ell-r)!},$$

and so the probability that a hit is found in a random information set is approximately

$$q^{2+r-\ell}\frac{R^2(1-R)^{\ell-r}}{(\ell-r)!}.$$

Let $f_1, f_2$ be the polynomials corresponding to the codewords in a found pair. The rational function $f_1/f_2$ is then a fraction of two polynomials of degree $r+1$, and so, taking into account that three points can be arbitrarily fixed, we get that $2r$ points have to be guessed until the correct solution is found.

If we count a pivoting in the generator matrix as $R(1-R)q^2$ operations, this would result in a very approximate cost of

$$2q^{\ell-r}\frac{(\ell-r)!}{R(1-R)^{\ell-r-1}} + q^{2r}$$

operations of cost $O(1)$ for a break.

As an example, consider $q = 256$. If we set the infeasibility threshold to roughly $2^{72}$ operations of cost $O(1)$, then we see that we need $r \leq 4$, and the attack will be feasible as long as $\ell < 13$, if $R = 1/2$.

## 5.3 Structural weaknesses of codes

We now turn to the bigger picture. In the light of the existence of structural attacks on McEliece cryptosystem based on some codes, an obviously vitally important question is:

What makes codes weak against structural attacks?

At this point, every attack we have studied made use of low weight words in either the code itself or in its dual: First, we have seen in section 2.2.2 that for graph based codes, low weight words in the dual code render those systems weak, and Sendrier's attack on concatenated codes [S1994] uses the very same fact. Second, our study of the Sidelnikov-Shestakov [SS1992] attack in section 3.1 makes crucial use of minimum weight words and their structure. Third, the attack on elliptic codes, being modeled after the same attack, also makes use of minimum weight words. Fourth, the Sidelnikov cryptosystem is also broken using minimum weight words.

Given the state of affairs, I think it is fair to say that low weight words have proved extremely useful to devise practical attacks on code based cryptosystems. On the other hand, the way how such minimum weight words are exploited differs from code to code. So, *why* are minimum weight words dangerous in such systems?

In a nutshell, I think the answer is that they are witnesses for the non-randomness of a code, i.e., they can be used to distinguish random codes from codes that are based on a particular construction, especially if the given construction has a tight guarantee for the minimum distance of the code: The structural hardness assumption for McEliece type systems can be stated as the assumption that a random generator matrix for a code is indistinguishable from a generator matrix of a random code. Therefore, witnesses of non-randomness violate that assumption.

Minimum weight words are not the only witnesses for non-randomness. In some cases, the automorphism group of a code is another one: Random codes have generally a very small automorphism group, but for some particular Goppa codes the automorphism group is much larger. This group can often be found, and Loidreau and Sendrier have used this fact to construct attacks against certain weak keys coming from Goppa codes with large automorphism groups [LS2001].

These thoughts lead to a number of interesting research problems in the domain of McEliece type cryptosystems.

First, it would be beneficial to systematically study algebraic codes for structural weaknesses. On the one hand, it can certainly be hoped that many of the techniques that work on some families can be adapted or generalized to other families. On the other hand, example breaks may well offer a lot of new insight into structural weaknesses or strengths, information that is indispensable for the designer of such cryptosystems. As far as I know, many potentially useful families have not seen much attention from the cryptographic community.

Second, low-weight word finding algorithms can surprisingly also be usefully applied in many structural attacks. Thus, faster low weight word finding algorithms might have much more impact on the security of McEliece type systems than what was previously thought.

Third, other measures for structure in codes than the two examples we mentioned should be sought, even if it is a priori not clear how they can be exploited to cryptanalyze McEliece type systems. Given that the two witnesses that were cited above (that is, automorphism groups, and low weight words), have both led to successful attacks, it appears that results in this domain would likely lead to more discoveries, eventually.

# Bibliography

[AS1970] M. Abramowitz, I. A. Stegun (ed.) *Handbook of mathematical functions*, Dover, 1970

[AM1987] C. M. Adams, H. Meijer *Security-Related Comments Regarding McEliece's Public-Key Cryptosystem*, CRYPTO '87, LNCS 293, Springer, 1987

[BS1996] E. Bach, J. Shallit *Algorithmic Number Theory*, Volume 1, MIT Press, 1996

[BL2005] T. Berger, P. Loidreau *How to Mask the Structure of Codes for a Cryptographic Use*, Designs, Codes and Cryptography, 35, 63-79, 2005

[BMEvT1978] E. Berlekamp, R. J. McEliece, H. van Tilborg, *On the inherent intractability of certain coding problems*, IEEE Transactions on Information Theory, 24(3):384–386, 1978

[BSS1999] I. Blake, G. Seroussi, N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999

[BKW1996] J. Bloemer, R. Karp, W. Welzl, *The Rank of Sparse Random Matrices over Finite Fields*, ICSI Technical Report tr-96-004, 1996

[BN1990] J. Bruck, M. Naor, *The Hardness of Decoding Linear Codes with Preprocessing*, IEEE Transactions on Information Theory, 36(2):381–385, 1990

[C1996] A. Canteaut, *Attaques de cryptosystèmes à mots de poids faible et construction de fonctions t-résilientes*, `http://www-rocq.inria.fr/codes/Anne.Canteaut/Publications/pub.html`, PhD thesis, Université Paris 6, 1996 (in French)

[CC1998] A. Canteaut, F. Chabaut, *A new algorithm for finding minimum-weight words in a linear code: application to primitive narrow-sense BCH-codes of length 511*, 1998, IEEE Transactions on Information Theory, 44(1):367-378

[CS1998] A. Canteaut, N. Sendrier, *Cryptanalysis of the original McEliece cryptosystem*, Advances in Cryptology - ASIACRYPT'98 , LNCS 1514, pages 187-199, Springer, 1998.

[C1992] F. Chabaud *Asymptotic analysis of probabilistic algorithms for finding short codewords*, EUROCODE 1992, Springer, 1993

[C1995] F. Chabaud, *On the security of some cryptosystems based on error-correcting codes*, Advances in Cryptology: Proceedings of EUROCRYPT'94, LNCS 950, Springer, 1995

[D1941] M. Deuring, *Die Typen der Multiplikatorenringe elliptischer Funktionenkörper*, Abhandlungen aus dem mathematischen Seminar der Universität Hamburg, 14, pages 197–272, 1941

[DH1976] W. Diffie, M. E. Hellman, *New directions in cryptography.*, IEEE Transactions on Information Theory, IT-22(6).644–654, 1976

[DMS2003] I. Dumer, D. Micciancio, M. Sudan, *Hardness of approximating the minimum distance of a linear code*, IEEE Transactions on Information Theory, 49(1):22–37, 2003

[DS2006] I. Dumer, K. Shabunov, *Soft-decision decoding of Reed-Muller codes: a simplified algorithm*, 2006, IEEE Transactions on Information Theory 52(3): 954–963

[EOS2006] D. Engelbert, R. Overbeck, A. Schmidt *A Summary of McEliece-Type Cryptosystems and their Security* Cryptology e-print archive, `http://eprint.iacr.org/2006/162`, 2006

[G2005] P. Gaborit *Shorter keys for code based cryptography*, Proceedings of Workshop on Codes and Cryptography, Bergen, p. 81–90, 2005

[G1977] V. D. Goppa *Codes associated with divisors*, Problems of Information Transmission, 13: 22–26, 1977

[HP2003] W. Carey Huffman, V. Pless *Fundamentals of Error-Correcting Codes* Cambridge University Press, 2003

[H2003] K. Hulek *Elementary Algebraic Geometry* American Mathematical Society, 2003

[JM1996] H. Janwa, O. Moreno *McEliece Public Key Cryptosystems Using Algebraic-Geometric Codes*, Designs, Codes and Cryptography 8(3): 293–307, 1996

[KT1970] T. Kasami, N. Tokura, *On the Weight Structure of Reed-Muller Codes*, IEEE Transactions on Information Theory, 16(6): 752-759, 1970

[LB1988] P. J. Lee, E. F. Brickell *An Observation on the Security of McEliece's Public-Key Cryptosystem*, EUROCRYPT 1988

[L1987] H. W. Lenstra, Jr., *Factoring integers with elliptic curves*, Annals of Mathematics, 126, pages 649–673, 1987

[L1988] J. S. Leon *A probabilitstic algorithm for computing minimum weights of large error-correcting codes*, IEEE Transactions on Information Theory, IT 34(5).1354–1359, 1988

[LDW1994] Y. X. Li, R. H. Deng, X. M. Wang, *On the Equivalence of McEliece's and Niederreiter's Public-Key Cryptosystems*, IEEE Transactions on Information Theory, Vol. 40, No. 1, 1994

[LS2001] P. Loidreau, N. Sendrier *Weak Keys in the McEliece Public-Key Cryptosystem*, IEEE Transactions on Information Theory, Vol. 47, No. 3, 2001

[MS1977] F. J. MacWilliams, N. J. A. Sloane *The Theory of Error-Correcting Codes*, Elsevier, 1977

[MS2007] L. Minder, A. Shokrollahi *Cryptanalysis of the Sidelnikov cryptosystem*, To appear in Advances in Cryptology: Proceedings of EUROCRYPT 2007, LNCS

[ME1978] R. J. McEliece, *A public key cryptosystem based on algebraic coding theory*, DSN progress report, 42-44:114-116, 1978

[N1986] H. Niederreiter, *Knapsack-type cryptosystems and algebraic coding theory*, Problems of Control and Information Theory 15, 159-166, 1986

[P1975] N. J. Patterson, *The Algebraic Decoding of Goppa Codes*, IEEE Transactions on Information Theory, Vol. 21, No. 2, 1975

[S1994] N. Sendrier, *On the Structure of a randomly permuted concatenated code*, EUROCODE 94, October 1994.

[S2000] N. Sendrier, *Finding the Permutation Between Equivalent Linear Codes: The Support Splitting Algorithm*, IEEE Transactions on Information Theory, Vol. 46, No. 4, 2000

[S2002] A. Shokrollahi, *LDPC Codes: An Introduction*, `http://algo.epfl.ch/index.php?p=output_pubs-XX&db=pubs_ldpc.txt`, 2002

[SW1999] A. Shokrollahi, H. Wasserman, *List decoding of algebraic-geometric codes*, IEEE Transactions on Information Theory, Vol. 45, No. 2, 1999

[S1997] P. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM J.Sci.Statist.Comput. 26 1484, 1997

[S1994] V. M. Sidelnikov, *A public-key cryptosystem based on binary Reed-Muller codes*, Discrete Mathematics and Applications, 4 No. 3, 1994

[SS1992] V. M. Sidelnikov, S. O. Shestakov, *On insecurity of cryptosystems based on generalized Reed-Solomon codes*, Discrete Mathematics and Applications, 2, No. 4:439–444, 1992

[S1986] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer, 1986

[S1989] J. Stern, *A method for finding codewords of small weight*, Coding Theory and Applications, LNCS 388, 106–113, Springer, 1989

[TVZ1982] M. Tsfasman, S. Vlăduţ, Th. Zink, *Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound*, Mathematische Nachrichten, Vol. 109, 1982

[U2001] R. Urbanke et al, *LDPC-opt*, a degree distribution optimizer for LDPC-codes, accessible on the web: `http://lthcwww.epfl.ch/research/ldpcopt/index.php`

[vT1988] J. van Tilburg, *On the McEliece public-key cryptosystem*, Advances in Cryptology–CRYPTO' 88, pages 119—131, Springer, 1988

[V1990] S. G. Vlăduţ *On the decoding of algebraic-geometric codes over* $\mathbb{F}_q$ *for* $q \geq 16$, IEEE Transactions on Information Theory, Vol. 36, No. 6, 1990

[W2006] C. Wieschebrink *An attack on a modified Niederreiter encryption scheme*, Public Key Cryptography, LNCS 3958, pages 14–26, 2006

## Curriculum Vitae of Lorenz Minder

| | |
|---|---|
| Address | Rue Mercerie 5 |
| | CH-1003 Lausanne |
| | Switzerland |
| Phone Numbers | +41 21 693 12 04 (office) |
| | +41 76 458 27 18 (mobile) |
| E-mail | Lorenz.Minder@epfl.ch |

Born the 19th of December 1977 in Kitchener (Ontario, Canada). Swiss and Canadian citizen. Single.

### Professional

I am currently a PhD student in mathematics in the Laboratory for Algorithmic Mathematics at EPFL (Ecole Polytechnique Fédérale de Lausanne). My PhD advisor is Prof. Amin Shokrollahi. I am planning to finish my PhD in Summer 2007.

Educational background:

- Diploma of Engineering Mathematics at EPFL in 2003. (Master of Sciences.)
- Matura (science) at the Deutsches Gymnasium Biel in 1998.

### Research interests, work, talks and publications

My mathematical interests are centered around cryptography, codes and algorithmics. My research domain is the application of coding theory to cryptography. Publications, preprints, talks and unpublished work:

- *The structure of Reed-Muller codes and the Sidelnikov cryptosystem*, invited gradutation day talk at ITA workshop 2007.
- L. Minder, *Cryptanalysis of the Sidelnikov cryptosystem.* Proceedings of Eurocrypt 2007, LNCS 4515, Springer, 2007
- A. Brown, L. Minder, A. Shokrollahi, *Improved Decoding of Interleaved AG Codes*, IMA Int. Conf. 2005
- G. Maze, L. Minder, *A New Family of Almost Identities*, to appear in Elemente der Mathematik.
- A. Brown, L. Minder, A. Shokrollahi, *Probabilistic Decoding of Interleaved RS-Codes on the Q-ary symmetric channel*, ISIT 2004.
- Diploma work: *Algorithmes pour compter le nombre de points sur une courbe elliptique définie sur un corps fini*, 2003.

### Internships

In Summer 2005, I was for two months at Digital Fountain in California. Digital Fountain is a company developing state-of-the-art error-correcting codes.

I was intern in the R&D department of the company, testing new improvements to these codes.

## Teaching and other activities

I am and was teaching assistant for different courses, notably algorithmics, analysis and linear algebra courses (the latter when I was undergrad). I regularily supervise semester projects, and also supervised a master thesis.

I have semi-regularly reviewed papers for ISIT, CRYPTO, and also for other conferences and journals.

## Computer skills

- Computer programming : Excellent knowledge in C, C++ and Objective-C. (I also know, but rarely use, other languages such as Java.) Good knowledge and regular use of numerous scripting languages (sh, awk, sed, etc.). I have contributed patches to a number of open source projects, most recently cdrkit, NetBSD pkgsrc, libstdc++.
  I write software as part of my PhD studies, mostly mathematical software, for example, programs to attack the cryptosystems I analyze.
  I privately wrote other pieces of software, such as a tool to recover data from broken NTFS Volumes (which was used to recover data from a broken drive), or special-purpose backup utilities. These programs are typically of the order of a few thousand lines of C or C++ code in size.
- Operating Systems : I have in-depth knowledge of Unix-like operating systems, in particular BSD and Linux. I also use Mac OS X and occasionally Windows.

## Languages

My native language is German. I speak and write both French and English very well. (I did my undergrad studies purely in French, and have lived for eight years in a French-speaking city now.)