university of groningen / faculty of science and engineering / mathematics and applied mathematics

McEliece variation under attack:
Cryptanalysis of a public key cryptosystem based on algebraic geometry codes

Master's Project Mathematics

August 2021

Student: A. Tuijp

First supervisor: Prof.dr. J. Top

Second supervisors: dr. P. Kılıçer, prof. dr. C. Salgado Guimarães da Silva

Abstract:
The McEliece cryptosystem is a public key cryptosystem that has (so far) not been broken by any quantum algorithms and may therefore be resistant to quantum computer attacks. McEliece makes use of error correcting codes in the process of encryption and decryption. The original McEliece cryptosystem uses classical Goppa codes, but these have the disadvantage that the public and private keys of the cryptosystem are very large matrices. Many variations with shorter keys have been proposed - and then been proven to be insecure. One example is the proposal to use algebraic geometry codes instead of classical Goppa codes. An attack on this variation, based on Schur products and error correcting pairs, is examined in detail in this project.

Contents

Introduction

Cryptology is the science concerned with sending messages across insecure channels: Alice sends Bob an encrypted message while Eve intercepts this message and tries to decrypt it. Sometimes, 'cryptography' is used as a synonym of 'cryptology', but it is also understood as the sub-area of cryptology concerned with designing cryptosystems. Cryptanalysis, on the other hand, is the sub-area of cryptology concerned with breaking the systems. Usually, some people propose a new cryptosystem (cryptography) and then others will try to break it (cryptanalysis).

Many cryptosystems that are used nowadays rely on hard problems such as factoring large integers or the (elliptic curve) discrete logarithm problem. However, thanks to Shor's algorithm (see [21]), these problems could be solved too quickly, using a quantum computer. In the year 2004, Bernstein wrote[2]:

> *Imagine that it's fifteen years from now and someone announces the successful construction of a large quantum computer. The New York Times runs a frontpage article reporting that all of the public-key algorithms used to protect the Internet have been broken. Users panic.*

About twelve years have passed since 2009. Much has happened, but - as far as we know - there are no quantum computers yet that are powerful enough to break cryptosystems. However, the interest in 'post-quantum cryptography' has not disappeared. Some examples of possible post-quantum classes of cryptographic systems can be found in [2] and are hash-based cryptography, lattice based cryptography, multivariate cryptography and, last but not least, code based cryptography.

As the name suggests, the field of code based cryptography is based on codes. To be precise, it is based on the theory of linear error correcting codes, which is concerned with storing data in such a way that a certain number of errors in the data can be found and even corrected. This finds applications for example in CDs: we still want to be able to hear the music if there is a tiny scratch on the disc. There exists many different types of error correcting codes, some of which are based on algebra and algebraic geometry.

An example of a code based cryptosystem is the McEliece cryptosystem. Originally, this cryptosystem uses a certain type of codes called Goppa codes. This is still considered as a good possibility, but there is a drawback: the size of the key is very large. In order to solve this, many people have proposed variations using other classes of codes. One example is the suggestion to use so called 'algebraic geometry codes'. In this thesis, all ingredients will be studied that are necessary in order to understand how the McEliece cryptosystem based on AG codes may be attacked, as discussed in chapter 6.

In chapter 1, the general idea behind (public key) cryptography is discussed and an example is given. Furthermore, some definitions and propositions about curves, divisors, differentials and vector spaces are given, because these concepts are important in algebraic geometry codes. Every chapter contains concrete worked out examples of the given theory, either by hand or using MAGMA. In chapter 2, some general theory about error correcting codes is given. After this, Goppa codes and algebraic geometry codes are treated in more detail.

Chapter 3 introduces an operation on vectors called the Schur product and gives examples. Not only the Schur product of two vectors is considered but also (and perhaps more importantly) the Schur product of two codes. This turns out to be a helpful tool in the process of 'decoding': finding and correcting the errors. Chapter 4 is devoted to decoding and discusses two different algorithms, one of which is based on Schur products. In chapter 5, we see how the theories of cryptology and coding theory come together in the cryptosystem that was designed by McEliece. Finally, in chapter 6, we will investigate an attack originally proposed in [4] on the McEliece variation using AG codes. In this attack, the Schur product plays a crucial role and (almost) all propositions from the entire thesis will come together in a beautiful way.

Chapter 1

Preliminaries

Before we can discuss the details of the McEliece cryptosystem and its variations, we need to understand various concepts from cryptography, algebraic geometry and coding theory. In this chapter, we give the definitions and propositions needed for the rest of the thesis. Please note that this is not an introduction to algebraic geometry or coding theory, since we only give the information that is relevant to the rest of this thesis. However, no prior knowledge on cryptography or coding theory is required and only a little bit of algebra or algebraic geometry.

1.1 Public-key Cryptography

Cryptosystems can be divided into two groups: symmetric cryptosystems and asymmetric or public key systems. The oldest cryptosystems are symmetric. In this form of cryptography, Alice and Bob share a secret key. Examples are the (not so secure) substitution cipher and AES[6]. In cryptology, we always assume that Eve knows exactly which cryptosystem is used. The security of the system depends on the secrecy of the shared key. Alice and Bob therefore have to find a way to somehow obtain the same key, while nobody else finds out about it. Unless Alice and Bob can meet in person (or have a very trustworthy mailman), this could be a tricky condition.
Public key cryptography doesn't have this problem. Alice and Bob both have their own personal secret keys, but as the name suggests, they also both have public keys, which everyone may know. If Alice wishes to send a message to Bob, she uses his public key to encrypt her message. Bob then uses his own secret key to decrypt the message. Public key cryptosystems are always based on a so called 'one-way' function, or a 'trapdoor' function: a function which is easy to use but for which it is hard to find the inverse. A well known example of a public key cryptosystem is RSA [20], which is based on the discrete logarithm problem. We will give a slightly simplified version here, as an illustration.
Key generation: Bob picks two large prime numbers, $p \neq q$. He calculates $N = pq$ and $\phi(N) = (p-1)(q-1)$. Bob chooses $e > 1$ such that $\gcd(e, \phi(N)) = 1$

and computes d such that $d \cdot e \equiv e^{-1} \mod \phi(N)$. Note that $ed = 1 + k\phi(N)$ for some integer k. Bob's secret key is d and he presents his public key (N, e).
Encrypting: Suppose Alice has a secret message for Bob, namely $m < N$. We will call m the plaintext. Alice converts the plaintext into a 'ciphertext' $c := m^e \mod N$ and sends this to Bob.
Decrypting: In order to decrypt the ciphertext c, Bob computes $m' = c^d = m^{ed} \equiv m \mod N$. This equivalence can be shown as follows: If m is a unit modulo N, this follows from Euler's theorem. If m is not a unit modulo N, then without loss of generality $m \equiv 0 \mod p$ and $m \not\equiv 0 \mod q$. We see that $m^{ed} \equiv 0 \mod p$ and $m^{ed} = m^{1+k\phi(N)} = m^{1+k(p-1)(q-1)} = m(m^{q-1})^{k(p-1)} \equiv m \cdot 1^{k(p-1)} \equiv m \mod q$. By the Chinese Remainder Theorem it follows that indeed $m^{ed} \equiv m \mod N$. We see that Bob indeed finds the same plaintext m that was sent to him by Alice.

Now suppose Eve intercepts the ciphertext. She knows which cryptosystem is used and she knows the values of m^e and e, but it is still very hard for her to find m. This is the discrete logarithm problem. Note that if Eve could find p and q from N, she could easily find d and not only find m but also decrypt any other future messages sent to Bob. Luckily, factoring integers is a hard problem as well. These problems are obviously not logically impossible to solve: if one would just try all possibilities, the right one will come up. However, if we choose our p and q to be big enough, there are many, many possibilities. If the expected time to find the solution, using the most efficient algorithms known and the strongest computers, is long enough (for example, longer than the expected lifetime of the earth), then we consider the cryptosystem secure with the given parameters.

1.2 Algebraic concepts

The concepts, properties and propositions we discuss here will be needed for the sections on algebraic geometry codes. When we talk about a curve $\mathcal{X}$, we always mean that $\mathcal{X}$ is an absolutely irreducible (that is, also irreducible over field extensions) non-singular projective curve of genus g over a finite field $\mathbb{F}$. For a formal treatment of curves, or varieties in general, we refer to [11]. AG codes involve divisors on curves. The properties of divisors and the notations that we use are given in this section. Most of them are taken from [19].

Definition 1.2.1. *Let $\mathcal{X}$ be a curve over $\mathbb{F}$.*

- *A* **divisor** *on $\mathcal{X}$ is a formal sum $D = \sum_{P \in \mathcal{X}} n_P P$, with $n_P \in \mathbb{Z}$ and $n_P = 0$ for all but a finite number of points P.*
- *The* **support** *of a divisor is the set of points with nonzero coefficient. A divisor D is called* **effective** *if all coefficients n_P are non-negative.*
- *The partial order $\geq$ on divisors is defined as follows: $D \geq E$ if and only if $D - E$ is effective.*

- *The* **degree** $\deg(D)$ *of the divisor* D *is* $\sum n_P$.
- *A divisor is said to be* **defined over** $\mathbb{F}$ *if for every term* nP *in the formal sum, with* P *defined over some extension of* $\mathbb{F}$*, also* nQ *appears in the sum for every Galois conjugate* Q *of* P.
- *If* f *is a rational function on* $\mathcal{X}$*, not identically* 0*, the* **divisor of** f *is defined as* $(f) = \sum_{P \in \mathcal{X}} v_P(f)P$*, where* $v_P(f)$ *is the order of vanishing of the function* f *at the point* P *(and a negative order of vanishing, say* $-n$*, means that* f *has a pole of order* n *at* P*).*

The divisor of a function f is in fact the 'sum' of the zeros and poles of f with their multiplicities and orders. Because f is a rational function on a projective curve, its number of zeros equals its number of poles. Therefore, the degree of (f) is 0.

The following definitions and properties concern (divisors of) differentials on curves:

Definition 1.2.2. *Let* $\mathcal{X}$ *be a curve over* $\mathbb{F}$ *and let* ω *be a differential on* $\mathcal{X}$.

- *The set of all rational differential forms on* $\mathcal{X}$ *is denoted by* $\Omega(\mathcal{X})$.
- *For every point* P *and local parameter* t_P*,* ω *can be represented in a unique way as* $\omega = f_P dt_P$*, where* f_P *is a rational function.*
- *Let* ω *be a differential on* $\mathcal{X}$*. The* **valuation** *of* ω *at* P *is defined by* $v_P(\omega) = v_P(f_P)$*. (This does not depend on the choice of* t_P*.)*
- *The* **divisor of** ω *is defined by*

$$(\omega) = \sum_{P \in \mathcal{X}} v_P(\omega)P.$$

- *If* $W = (\omega)$ *for some differential* ω*, then* W *is called a* **canonical divisor**.
- *Let* P *be a point on* $\mathcal{X}$*,* t *a local parameter at* P *and* $\omega = f\ dt$*.* f *can be written as* $\sum_i a_i t^i$*. The* **residue** $Res_P(\omega)$ *of* ω *in the point* P *is defined to be* a_{-1}.

The following two vector spaces are important:

Definition 1.2.3. *Let* D *be a divisor on* $\mathcal{X}$*, defined over* $\mathbb{F}$*. We define the* **Riemann Roch space** $L(D)$ *over* $\mathbb{F}$ *by*

$$L(D) = \{f \in \mathbb{F}(\mathcal{X})^* \mid (f) + D \geq 0\} \cup \{0\}.$$

The dimension of $L(D)$ *over* $\mathbb{F}$ *is denoted by* $l(D)$*. Furthermore, we define*

$$\Omega(D) = \{\omega \in \Omega(\mathcal{X}) \mid (\omega) - D \geq 0\}$$

and the dimension of $\Omega(D)$ *over* $\mathbb{F}$ *is denoted by* $\delta(D)$.

Proposition 1.2.4. *Let $\overline{\mathbb{F}}$ be an algebraic closure of $\mathbb{F}$, let D be a divisor on $\mathcal{X}$ and let W be a canonical divisor on $\mathcal{X}$, both defined over $\mathbb{F}$ and let ω be a differential on $\mathcal{X}$. The following statements are true:*

1. *if* $\deg(D) < 0$, *then* $l(D) = 0$.
2. $l(0) = 1$
3. $l(D) - l(W - D) = \deg(D) - g + 1$ (Riemann-Roch Theorem)
4. $\deg(W) = 2g - 2$.
5. *If* $\deg(D) > 2g - 2$, *then* $l(D) = \deg(D) - g + 1$.
6. $\sum_{P \in \mathcal{X}(\overline{\mathbb{F}})} \mathrm{Res}_P(\omega) = 0$ (Residue Theorem)
7. $\delta(D) = l(W - D)$

Proof.

1. For every rational function f, the degree of (f) is zero. Therefore, if $\deg(D) < 0$ there is no $f \in \mathbb{F}(\mathcal{X})^*$ such that $(f) + D \geq 0$ which means that $L(D) = \{0\}$, so indeed $l(D) = 0$.
2. The only rational functions on $\mathcal{X}$ with no poles are the constant functions, so $L(0) = \mathbb{F}$ and $l(0) = 1$.
3. For a proof see [7], for example.
4. This is a corollary of the Riemann-Roch theorem. First, take $D = 0$ and find $l(W) = l(0) - \deg(0) + g - 1 = 1 - 0 + g - 1 = g$. Next, choose $D = W$ and find $\deg(W) = l(W) - l(0) + g - 1 = g - 1 + g - 1 = 2g - 2$.
5. This is also a corollary of the Riemann-Roch theorem. Because $\deg(D) > 2g - 2$ and $\deg(W) = 2g - 2$, it follows that $\deg(W - D) < 0$. Statement 1 tells us that $l(W - D) = 0$. This statement now follows directly from the Riemann-Roch theorem.
6. For a proof, see [22], for example.
7. Let $W = (\omega)$, then $\phi\colon L(W - D) \to \Omega(D)$ given by $\phi(f) = f\omega$ is an isomorphism.

□

In the following example, which can also be found in [19], a Riemann-Roch space is determined. We will encounter this curve and space again in later examples.

Example 1.2.5. Consider the projective curve over $\mathbb{F}_4$ given by the equation

$$X^3 + Y^3 + Z^3 = 0$$

and the two rational functions

$$f = \frac{x}{y + z} \quad \text{and} \quad g = \frac{y}{y + z}.$$

The genus of the curve is 1, as can be calculated with the Plücker formula: $g = \frac{1}{2}(m-1)(m-2) = 1$ where $m = 3$ is the degree of the curve. The elements of the field $\mathbb{F}_4$ are $\{0, 1, \alpha, \alpha^2\}$ where $\alpha^2 = \alpha + 1$. The rational function $f = \frac{x}{y+z}$ has simple zeros at the points $P_1 = (0 : \alpha : 1)$ and $P_2 = (0 : \alpha^2 : 1)$ and a double pole at the point $Q = (0 : 1 : 1)$. The divisor of f is therefore

$$(f) = P_1 + P_2 - 2Q.$$

The rational function $g = \frac{y}{y+z}$ has single zeros at the points $P_3 = (1 : 0 : 1)$, $P_4 = (\alpha : 0 : 1)$ and $P_5 = (\alpha^2 : 0 : 1)$ and a triple pole at the point Q, so

$$(g) = P_3 + P_4 + P_5 - 3Q.$$

Now consider the divisor $D = 3Q$ and the corresponding Riemann-Roch space $L(D)$. We see that $(f) + D \geq 0$ and $(g) + D \geq 0$. Also $(1) + D = D \geq 0$. It follows that $\{1, f, g\} \subseteq L(D)$. Because these functions are linearly independent, we have $l(D) \geq 3$. Actually, because $\deg(D) = 3 > 0 = 2g - 2$, it follows from statement 5 that $l(D) = \deg(D) - g + 1 = 3 - 1 + 1 = 3$, so that $\{1, f, g\}$ is in fact a basis of $L(D)$.

The following example shows some explicit calculations involving differentials on curves.

Example 1.2.6. Again, consider the projective curve $\mathcal{X}$ over $\mathbb{F}_4$ given by the equation

$$X^3 + Y^3 + Z^3 = 0$$

and the point $Q = (0 : 1 : 1)$. The elements of the field $\mathbb{F}_4$ are $\{0, 1, \alpha, \alpha^2\}$ where $\alpha^2 = \alpha + 1$. Together with Q, all rational points on the curve are:

$$P_1 = (0 : \alpha : 1), \quad P_2 = (0 : \alpha^2 : 1), \quad P_3 = (1 : 0 : 1), \quad P_4 = (\alpha : 0 : 1),$$

$$P_5 = (\alpha^2 : 0 : 1), \quad P_6 = (1 : 1 : 0), \quad P_7 = (\alpha : 1 : 0), \quad P_8 = (\alpha^2 : 1 : 0)$$

and let $D = P_1 + \ldots P_8$. The space $\Omega(6Q - D)$ consists of all rational differential forms ω on $\mathcal{X}$ such that $(\omega) - 6Q + D \geq 0$. That means that ω has a zero of order at least 6 at Q and no poles, except for possibly poles of order one at the points P_i. According to MAGMA (see Appendix B.1), $\Omega(6Q - D)$ has dimension two and is generated by the two differential forms η and ζ:

$$\eta = \frac{y^2 + 1}{y^3 + y^2 + y}\, dy,$$

$$\zeta = \frac{xy + x}{y^3 + y^2 + y}\, dy.$$

For every point P and local parameter t_P, it is possible to write these differential forms as $f_P\, d(t_P)$. We can do this for example as follows:

$$\eta_1 = \frac{y^2z + z^3}{y^3 + y^2z + yz^2}\ d\left(\frac{y}{z}\right) \quad \text{when } x \neq 0 \text{ and } z \neq 0$$

$$\eta_2 = \frac{x^2y^3 + x^2yz^2}{y^2z^3 + yz^4 + z^5}\ d\left(\frac{x}{y}\right) \quad \text{when } y \neq 0 \text{ and } z \neq 0$$

$$\eta_3 = \frac{x^4y^2 + x^4z^2}{y^5z + y^4z^2 + y^3z^3}\ d\left(\frac{z}{x}\right) \quad \text{when } x \neq 0 \text{ and } y \neq 0$$

$$\zeta_1 = \frac{xyz + xz^2}{y^3 + y^2z + yz^2}\ d\left(\frac{y}{z}\right) \quad \text{when } x \neq 0 \text{ and } z \neq 0$$

$$\zeta_2 = \frac{x^3y^2 + x^3yz}{y^2z^3 + yz^4 + z^5}\ d\left(\frac{x}{y}\right) \quad \text{when } y \neq 0 \text{ and } z \neq 0$$

$$\zeta_3 = \frac{x^5y + x^5z}{y^5z + y^4z^2 + y^3z^3}\ d\left(\frac{z}{x}\right) \quad \text{when } x \neq 0 \text{ and } y \neq 0.$$

In order to see that for example $\eta_1 = \eta_2$ when $x, y, z, \neq 0$, notice that

$$d\left(\frac{y}{z}\right) = d\left(\left(\frac{z}{y}\right)^{-1}\right) = -\left(\frac{z}{y}\right)^{-2}\ d\left(\frac{z}{y}\right) = \frac{y^2}{z^2}\ d\left(\frac{z}{y}\right).$$

In order to find an expression for $d(\frac{z}{y})$, differentiate

$$\left(\frac{x}{y}\right)^3 + 1 + \left(\frac{z}{y}\right)^3 = 0$$

to get

$$\left(\frac{x}{y}\right)^2 d\left(\frac{x}{y}\right) + \left(\frac{z}{y}\right)^2 d\left(\frac{z}{y}\right) = 0,$$

which implies

$$d\left(\frac{z}{y}\right) = \frac{x^2}{z^2}\ d\left(\frac{x}{y}\right).$$

If we combine this with the expression for $d\left(\frac{y}{z}\right)$, we find

$$d\left(\frac{y}{z}\right) = \frac{y^2}{z^2}\ d\left(\frac{z}{y}\right) = \frac{y^2}{z^2}\ \frac{x^2}{z^2}\ d\left(\frac{x}{y}\right) = \frac{x^2y^2}{z^4}\ d\left(\frac{x}{y}\right)$$

and if we substitute this in the expression for η_1 we get indeed η_2. In the same way, it can be shown that η_i overlap on all intersections. There are no points on $\mathcal{X}$ where two coordinates are zero, so for every point P, we have written η (and ζ) in the form $\eta = f_P\ d(t_P)$, where t_P is a local parameter at P.

Using MAGMA, we know that η has a zero of order 8 at Q and poles of order 1 at all other P_i. ζ has a zero of order 6 at Q and poles of order 1

at $P_3, P_4, P_5, P_6, P_7, P_8$. It is also possible to verify these orders by hand. As an example, we show that η has a pole of order 8 at Q. This means that we need to rewrite η_2 in the form $\left(\frac{x}{y}\right)^8 f\ d\left(\frac{x}{y}\right)$ such that f is a rational function which has no poles or zeroes at $Q = (0:1:1)$. We compute such an f as follows:

$$\begin{aligned}
f &= \frac{y^8}{x^8} \cdot \frac{x^2y^3 + x^2yz^2}{y^2z^3 + yz^4 + z^5} \\
&= \frac{y^9}{x^6} \cdot \frac{y^2 + z^2}{y^2z^3 + yz^4 + z^5} \\
&= \frac{y^9}{z^3} \cdot \frac{1}{y^2 + yz + z^2} \cdot \frac{y^2 + z^2}{x^6} \\
&= \frac{y^9}{z^3} \cdot \frac{1}{y^2 + yz + z^2} \cdot \left(\frac{y+z}{x^3}\right)^2 \\
&= \frac{y^9}{z^3} \cdot \frac{1}{y^2 + yz + z^2} \cdot \left(\frac{y+z}{y^3 + z^3}\right)^2 \\
&= \frac{y^9}{z^3} \cdot \frac{1}{y^2 + yz + z^2} \cdot \left(\frac{1}{y^2 + yz + z^2}\right)^2 \\
&= \frac{y^9}{z^3(y^2 + yz + z^2)^3}.
\end{aligned}$$

Here we have used that the characteristic is 2 and that $x^3 + y^3 + z^3 = 0$. Similarly, we can show that η has a pole of order one at $P_3 = (1:0:1)$ by writing η_1 in the form $\left(\frac{y}{z}\right)^{-1} f\ d\left(\frac{y}{z}\right)$ in such a way that f is regular and nonzero at P_3:

$$\begin{aligned}
f &= \frac{y}{z} \cdot \frac{y^2z + z^3}{y^3 + y^2z + yz^2} \\
&= \frac{yz}{yz} \cdot \frac{y^2 + z^2}{y^2 + yz + z^2} \\
&= \frac{y^2 + z^2}{y^2 + yz + z^2}.
\end{aligned}$$

Using MAGMA, we can show that $\mathrm{Res}_{P_i}(\eta) = 1$ for all i and that for ζ we have the following residues:

$$\mathrm{Res}_{P_1}(\zeta) = 0, \quad \mathrm{Res}_{P_3}(\zeta) = 0, \quad \mathrm{Res}_{P_3}(\zeta) = 1, \quad \mathrm{Res}_{P_4}(\zeta) = \alpha,$$

$$\mathrm{Res}_{P_5}(\zeta) = \alpha^2, \quad \mathrm{Res}_{P_6}(\zeta) = 1, \quad \mathrm{Res}_{P_7}(\zeta) = \alpha, \quad \mathrm{Res}_{P_8}(\zeta) = \alpha^2.$$

Because η and ζ are elements of $\Omega(6Q - D)$, they have no poles other than those at the P_i. Therefore, by proposition 1.2.4(7), we should have $\sum_i \mathrm{Res}_{P_i}(\eta) = \sum_i \mathrm{Res}_{P_i}(\zeta) = 0$. We can see that the residues indeed add up to 0 (in characteristic 2).

It is also possible to calculate these residues by hand and we will show two examples: $\mathrm{Res}_{P_3}(\eta)$ and $\mathrm{Res}_{P_4}(\zeta)$. If we want to compute the residue of η at $P_3 = (1:0:1)$, we need to find a_i such that

$$\frac{y^2z + z^3}{y^3 + y^2z + yz^2} = \sum_i a_i \left(\frac{y}{z}\right)^i.$$

We set $t = \frac{y}{z}$. First notice that $\frac{1}{t^3+1} = \sum_{i=0}^{\infty} t^{3i}$, because

$$\begin{aligned}(t^3+1)\sum_{i=0}^{\infty} t^{3i} &= (t^3 + t^6 + t^9 + t^{12} + \ldots) + (1 + t^3 + t^6 + t^9 + t^{12}\ldots) \\ &= 1 + (t^3 + t^3) + (t^6 + t^6) + (t^9 + t^9) + (t^{12} + t^{12}) + \ldots \\ &= 1\end{aligned}$$

Now we find the appropriate expression as follows:

$$\begin{aligned}\frac{y^2z + z^3}{y^3 + y^2z + yz^2} &= \frac{\left(\frac{y}{z}\right)^2 + 1}{\left(\frac{y}{z}\right)^3 + \left(\frac{y}{z}\right)^2 + \left(\frac{y}{z}\right)} \\ &= \frac{t^2+1}{t^3+t^2+t} \\ &= \frac{1}{t} + \frac{1}{t^2+t+1} \\ &= \frac{1}{t} + \frac{t+1}{t^3+1} \\ &= \frac{1}{t} + (t+1)\frac{1}{t^3+1} \\ &= \frac{1}{t} + (t+1)\sum_{i=0}^{\infty} t^{3i} \\ &= \frac{1}{t} + \sum_{i=0}^{\infty} t^{3i} + \sum_{i=0}^{\infty} t^{3i+1} \\ &= \frac{1}{t} + 1 + t + t^3 + t^4 + t^6 + t^7 + t^9 + t^{10} + \ldots\end{aligned}$$

We find that $a_{-1} = 1$ so indeed $\mathrm{Res}_{P_3}(\eta) = 1$. Now we continue to the second (slightly more complicated) residue example. In order to find $\mathrm{Res}_{P_4}(\zeta)$, we need to find a_i such that

$$\frac{xyz + xz^2}{y^3 + y^2z + yz^2} = \sum_i a_i \left(\frac{y}{z}\right)^i.$$

We set $t = \frac{y}{z}$ and observe the following:

$$\begin{aligned}
\frac{xyz + xz^2}{y^3 + y^2z + yz^2} &= \frac{\left(\frac{y}{z}\right)\left(\frac{x}{z}\right) + \left(\frac{x}{z}\right)}{\left(\frac{y}{z}\right)^3 + \left(\frac{y}{z}\right)^2 + \left(\frac{y}{z}\right)} \\
&= \left(\frac{x}{z}\right)\frac{t+1}{t^3 + t^2 + t} \\
&= \left(\frac{x}{z}\right)\left(\frac{1}{t} + \frac{t}{t^2 + t + 1}\right) \\
&= \left(\frac{x}{z}\right)\left(\frac{1}{t} + \frac{t^2 + t}{t^3 + 1}\right) \\
&= \left(\frac{x}{z}\right)\left(\frac{1}{t} + (t^2 + t)\sum_{i=0}^{\infty} t^{3i}\right) \\
&= \left(\frac{x}{z}\right)\left(\frac{1}{t} + t + t^2 + t^4 + t^5 + t^7 + t^8 + t^{10} + \ldots\right)
\end{aligned}$$

We see that we need to write $\frac{x}{z}$ as $\sum_i a_i \left(\frac{y}{z}\right)^i$. Again, we set $t = \frac{y}{z}$. Because $\frac{x}{z}$ has no poles in the point $P_4 = (\alpha : 0 : 1)$, we know that

$$\frac{x}{z} = \sum_{i=0}^{\infty} b_i t^i = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + \ldots$$

In order to compute the residue, the only interesting value is b_0. In P_4 we have $t = 0$ so $b_0 = \frac{x}{z} = \frac{\alpha}{1} = \alpha$. This means that

$$\begin{aligned}
\frac{xyz + xz^2}{y^3 + y^2z + yz^2} &= \left(\frac{x}{z}\right)\left(\frac{1}{t} + t + t^2 + t^4 + t^5 + t^7 + t^8 + t^{10} + \ldots\right) \\
&= \left(\alpha + b_1 t + b_2 t^2 + \ldots\right)\left(\frac{1}{t} + t + t^2 + t^4 + t^5 \ldots\right) \\
&= \frac{\alpha}{t} + O(1)
\end{aligned}$$

so we can conclude that $\mathrm{Res}_{P_4}(\zeta) = \alpha$.

1.3 Products of vector spaces

We have now seen some explicit examples of Riemann Roch spaces. In later chapters, we will find that the products of two such spaces are going to be of interest. The following definition explains what is meant by the product of vector spaces.

Definition 1.3.1. *Given two vector spaces V and W over $\mathbb{F}$, their **product** is*

defined as

$$V \cdot W := \mathrm{Span}_{\mathbb{F}} \{vw \mid v \in V,\ w \in W\}.$$

The following proposition, concerning the product of two Riemann Roch spaces, is taken from [4] without a proof. A different formulation including a proof using sheaves can be found in [15] (Theorem 6).

Proposition 1.3.2. *Let E and F be two divisors on $\mathcal{X}$ such that* $\deg(E) \geq 2g+1$ *and* $\deg(F) \geq 2g$. *Then*

$$L(E) \cdot L(F) = L(E + F).$$

Example 1.3.3. Again, consider the projective curve over $\mathbb{F}_4$ given by the equation $X^3 + Y^3 + Z^3 = 0$, the point $Q = (0 : 1 : 1)$, the divisor $D = 3Q$ and the two rational functions $f = \frac{x}{y+z}$ and $g = \frac{y}{y+z}$. We have seen that $l(D) = 3$ and that $L(D)$ has $\{1, f, g\}$ as a basis.
Now we consider the space $L(D) \cdot L(D)$. This space is spanned by the products of the basis elements, that is: $\{1 \cdot 1, 1 \cdot f, 1 \cdot g, f \cdot f, f \cdot g, g \cdot g\}$. If we look at these functions in the point Q, we see that f^2 has a pole of order 4, fg has a pole of order 5 and g^2 has a pole of order 6. These functions have no other poles on $\mathcal{X}$, so they are elements of $L(6Q)$. The functions in $\{1, f, g, f^2, fg, g^2\}$ are linearly independent and because $l(6Q) = 6$ by statement 5 of Proposition 1.2.4, this is a basis of $L(6Q)$. Indeed, $L(3Q) \cdot L(3Q) = L(6Q)$.

In order to see that the equality in the proposition does not need to hold when the conditions on the degrees are not met, we consider the following. Instead of $D = 3Q$, we let $D = 2Q$ and see whether $L(2Q) \cdot L(2Q) = L(4Q)$. Firstly, let us look at the space $L(2Q)$. Because 1 and f have poles of order ≤ 2 at Q and no other poles on $\mathcal{X}$, they are elements of $L(2Q)$. Because 1 and f are linearly independent and because $l(2Q) = 2$, $\{1, f\}$ is a basis of $L(2Q)$. It follows that $L(2Q) \cdot L(2Q)$ is spanned by $\{1, f, f^2\}$.
Secondly, let us look at the space $L(4Q)$. By similar arguments as before, this space has $\{1, f, g, f^2\}$ as a basis. We see that $g \in L(4Q)$, but $g \notin L(2Q) \cdot L(2Q)$ so this is indeed an example of a case where the conditions on the degree are not met and $L(D) \cdot L(D) \neq L(D + D)$.

Chapter 2

Coding theory

Coding theory is concerned with storing and transmitting data. If we store information, of if we send information across a noisy channel, we want to make sure that the change of a few bits doesn't make the whole message unreadable, and preferably doesn't change the message at all. We will give some definitions to make this mathematically more precise. These (and much more information about this subject) can be found for example in [19].

2.1 Linear error correcting codes

Definition 2.1.1. *A* **linear code** C *is a linear subspace of* $\mathbb{F}_q^n$*, where* $\mathbb{F}_q$ *stands for the finite field with* q *elements.* n *is called the* **length** *of the code* C*. The* **dimension** *of a linear code is its dimension as a linear subspace over* $\mathbb{F}_q$*.*

We often refer to the elements of a code as '(code) words'. A code C is thus a set of words of length n using the alphabet $\mathbb{F}_q$. There are also non-linear codes, but we will not consider those in this thesis. Therefore, whenever we write 'code', we mean 'linear code'. We are now going to define a metric, in order to be able to speak about the distance between code words.

Definition 2.1.2. *For* $\boldsymbol{x} = (x_1, \ldots, x_n)$*,* $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$*, the* **Hamming distance** $d(\boldsymbol{x}, \boldsymbol{y})$ *is defined as the number of coordinates where* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *differ:*

$$d(\boldsymbol{x}, \boldsymbol{y}) = |\{i \mid x_i \neq y_i\}| .$$

It is not hard to see that this is indeed a well-defined metric, that is, the following properties hold for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}_q^n$:

1. $d(\mathbf{x}, \mathbf{y}) \geq 0$ and $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$;
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$;
3. $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.

Now that we have a way to describe the distance between words, we can define the minimum distance of a code. This is the smallest number that can occur as the distance between two different words in the code:

Definition 2.1.3. *The* **minimum distance** *of a code C is defined as*

$$d = d(C) = \min\{d(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x}, \boldsymbol{y} \in C,\ \boldsymbol{x} \neq \boldsymbol{y}\}$$

if C consists of more than one element and is by definition $n+1$ if C consists of one word.

Some more (not very deep or surprising, but nonetheless useful) concepts are the following:

Definition 2.1.4. *Let $\boldsymbol{x} \in \mathbb{F}_q^n$ and let C be a linear code.*

- *The* **support** *of $\boldsymbol{x}$, denoted by* $\mathrm{supp}(\boldsymbol{x})$*, is defined as the set of nonzero coordinate positions:* $\mathrm{supp}(\boldsymbol{x}) = \{i \mid x_i \neq 0\}$.
- *The* **weight** *of $\boldsymbol{x}$, denoted by* $\mathrm{wt}(\boldsymbol{x})$*, is is defined as the number of elements in its support.*
- *The* **minimum weight** *of C, denoted by* $\mathrm{mwt}(C)$*, is defined as the minimal value of the weights of the nonzero code words:*

  $$\mathrm{mwt}(C) = \min\{\mathrm{wt}(\boldsymbol{c}) \mid \boldsymbol{c} \in C, c \neq 0\}$$

  if there is a nonzero $\boldsymbol{c} \in C$ and otherwise $\mathrm{mwt}(C) = n+1$.

The definition of the minimum weight looks very similar to that of the minimum distance. In fact, the two definitions are equivalent:

Proposition 2.1.5. *The minimum distance of a linear code C is equal to its minimum weight.*

Proof. Let C be a linear code. Because it is a linear subspace of $\mathbb{F}_q^n$, we know that $0 \in C$ and for every $\mathbf{c}_1, \mathbf{c}_2 \in C$, $\mathbf{c}_1 - \mathbf{c}_2 \in C$. On the one hand, we notice that for every $\mathbf{c} \in C$, $\mathrm{wt}(\mathbf{c}) = d(0, \mathbf{c})$, so $d(C) \leq \mathrm{mwt}(C)$. On the other hand, we see that for every $\mathbf{c}_1, \mathbf{c}_2 \in C$, $d(\mathbf{c}_1, \mathbf{c}_2) = \mathrm{wt}(\mathbf{c}_1 - \mathbf{c}_2)$, so $\mathrm{mwt}(C) \leq d(C)$. □

Note that in Definition 2.1.2, $\mathbf{x}$ and $\mathbf{y}$ do not necessarily need to be in a linear code. Therefore, it makes sense to talk about the distance between a code word $\mathbf{c} \in C \subset \mathbb{F}_q^n$ and any other element $\mathbf{r} \in \mathbb{F}_q^n$. This is something we do a lot in the theory of error-correcting codes. If we say that 3 errors have been added to a code word $\mathbf{c} \in C$, we mean that 3 coordinates of the word $\mathbf{c}$ have been changed in order to form the 'received word' $\mathbf{r}$. This received word clearly does not need to be in the code C, but will sometimes still be referred to as a word. The received word is often written as $\mathbf{r} = \mathbf{c} + \mathbf{e}$, where $\mathbf{e}$ is the 'error vector' and $\mathrm{wt}(e)$ is the number of errors. We can now understand the idea of 'correcting errors': it is about retrieving $\mathbf{c}$ from $\mathbf{r} = \mathbf{c} + \mathbf{e}$.

A very small minimum distance is not very useful in terms of error correcting. For example, if $d = 1$, then there are two words $\mathbf{x}$ and $\mathbf{y}$ that only differ in one coordinate. This means that we may not even be able to notice that a single error occurred. On the other hand, suppose that $d \geq 3$ for some code C and that a single error has occurred to a word $\mathbf{c}$. This means that $\mathbf{c}$ has changed into some received 'word' $\mathbf{r} \in \mathbb{F}_q^n$ such that $d(\mathbf{c}, \mathbf{r}) = 1$. There is only one possible code word that could have turned into $\mathbf{r}$ by only changing one coordinate. This idea of detecting the errors and finding the original code word is discussed in more detail in the paragraph on decoding.

Instead of writing "C is a linear code over $\mathbb{F}_q$ with length n, dimension k and minimum distance d", we often use the following abbreviation: C is an $[n, k, d]$ code (or $[n, k, d]_q$ code). We call n, k and d the parameters of C. Because a linear code is a linear subspace of $\mathbb{F}_q^n$ of dimension k, there exists a basis of k code words. If we put these words in a matrix as row vectors, we obtain a generator matrix of the code:

Definition 2.1.6. *A $k \times n$ matrix G with entries in $\mathbb{F}_q$ is called a **generator matrix** of an $\mathbb{F}_q$-linear code C if the rows of G are a basis of C.*

As we see, we can describe a code C by a generator matrix. Another way to describe a code is by another matrix, which has C as its null space:

Definition 2.1.7. *An $(n-k) \times n$ matrix H of rank $n-k$ is called a **parity check matrix** of an $[n, k, d]$ code C if C is the null space of this matrix, that is, if $C = \{\boldsymbol{x} \in \mathbb{F}_q^n \mid H\boldsymbol{x}^T = 0\}$.*

If a code has a large minimum distance, this means that many errors can (in theory) be corrected. A high minimum distance could therefore be considered as a good property for a code. The following theorem, called the Singleton Bound, gives an upper bound for the minimum distance:

Theorem 2.1.8. *If C is an $[n, k, d]$ code, then*

$$d \leq n - k + 1.$$

Proof. Let C be an $[n, k, d]$ code and assume that H is a parity check matrix for the code C. This means that H consists of $n - k$ rows which are linearly independent. Because the minimum distance of C is d, there is no word $\mathbf{x} \in C$ of weight $d - 1$ such that $H\mathbf{x}^T = 0$. In other words: every $d - 1$ columns of H are linearly independent. Because the row rank of a matrix equals the column rank of a matrix, $d - 1$ cannot be larger than $n - k$. Therefore, $d - 1 \leq n - k$, which implies $d \leq n - k + 1$. □

We see that a lower dimension implies a higher upper bound for the minimum distance. However, a relatively high dimension is also a desirable property for a code. For a given code word in an $[n, k, d]$ code, only k coordinates contain

actual information. The other $n-k$ coordinates are used to create redundancy and the possibility to correct errors. If k is large, then the 'information rate' k/n is high, which means that the available storage space is used efficiently. The Singleton Bound shows that we cannot have a high minimum distance and a high dimension at the same time: there is always a trade-off between the information rate on one hand and the error correcting capability of a code on the other hand.

The inner product (dot product) of code words in $\mathbb{F}_q^n$ is defined as usual for vectors: $(x_1, \ldots, x_n) \cdot (y_1, \ldots, y_n) = x_1y_1 + \ldots + x_ny_n$. Using this, we can define the dual code of a code C:

Definition 2.1.9. *For a code C of length n, the **dual code** $C^\perp$ is defined as*

$$C^\perp = \{\boldsymbol{x} \in \mathbb{F}_q^n \mid \boldsymbol{c} \cdot \boldsymbol{x} = 0 \text{ for all } \boldsymbol{c} \in C\}.$$

It follows from these definitions that G is a generator matrix of a code C if and only if G is a parity check matrix of $C^\perp$. Similarly, H is a parity check matrix of a code C if and only if H is a generator matrix of $C^\perp$.

Another concept that we know from linear algebra and will come back in the definition of error correcting pairs (chapter 4) is the orthogonality of subspaces of $\mathbb{F}_q^n$:

Definition 2.1.10. *Two subspaces $A, B \subseteq F_q^n$ are **orthogonal**, notation: $A \perp B$, if $\boldsymbol{a} \cdot \boldsymbol{b} = 0$ for all $\boldsymbol{a} \in A$ and all $\boldsymbol{b} \in B$.*

Clearly, for a code C, it is true that $C^\perp \perp C$. Furthermore, if $A \perp C$ for some $A \subseteq F_q^n$, then $A \subseteq C^\perp$.

2.2 Hamming Codes

One of the first examples of an error correcting code was the Hamming (7,4) code, which was introduced by Richard W. Hamming in 1950 [10] and can be explained using Figure 2.1. Such a diagram is not found in the original paper by Hamming but it is used in most modern explanations of the Hamming code. This particular picture is taken from [19].

Example 2.2.1. Suppose we have a message of four bits: $\mathbf{m} = (m_1, m_2, m_3, m_4)$. We can put these four bits in their corresponding position in the diagram and determine the bits r_1, r_2 and r_3 using the following rule: the number of ones in each circle must be even. The code word $\mathbf{c} \in C$ corresponding to $\mathbf{m}$ is $\mathbf{c} = (m_1, m_2, m_3, m_4, r_1, r_2, r_3)$. The three bits r_1, r_2 and r_3 are called redundant bits or parity bits. If a received word $\mathbf{r}$ has a single error, we can look at the diagram to locate the error: if the number of ones inside a certain circle is odd, then the error must be inside this circle. Therefore, by counting the ones in each circle, we can uniquely determine

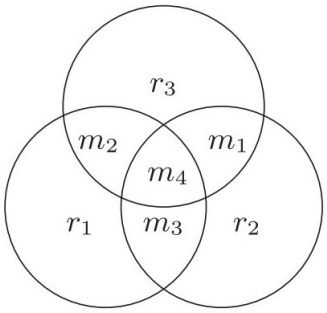

Figure 2.1: Venn diagram of the Hamming (7,4) code

the location of the error and therefore correct the error. In fact, the code C consisting of all words $\mathbf{c} \in \mathbb{F}_2^7$ that follow the rules of the diagram is a $[7, 4, 3]$ code, called the Hamming (7,4) code. A generator matrix of this code is

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

As we have seen, a single error can be detected and corrected. If there are two errors, this can be detected (because the minimum distance is 3), but it cannot be corrected: Suppose that there are errors located at the r_3 and the m_4 bits. This is detected by an odd number of ones in the left and right circle. If we try to decode this in the standard way, we would conclude that there is a single error in the m_3 bit, so we would decode incorrectly. Even if we would know that there are two errors, we cannot determine whether the errors are located at r_1 and r_2 or at r_3 and m_4.

More generally, Hamming introduced codes of length $2^m - 1$ and dimension $k = 2^m - m - 1$, for $m \in \mathbb{N}$, $m > 1$, all of which have minimum distance $d = 3$ so can correct a single error and detect a double error. Note that it is not possible to do these things at the same time: if you want to uses this code, you need to choose whether you want to correct single errors (and if more errors occurred, correct wrongly) or if you want to detect whether words contain single or double errors (not knowing how many errors there are and not correcting any errors). Hamming suggested to add an extra parity bit that makes sure the total number of ones in every word is even. In this way, it is possible to distinguish between a single and a double error: in case of a double error, the number of ones in the word is even, whereas in case of a single error, the number of ones in the word is odd. In these codes, it is still not possible to correct double errors but at least one can now correct single errors and detect double errors at the same time.

2.3 Goppa Codes

An example of an error correcting code that will turn out to be interesting for our purposes is the Goppa Code. They were introduced (in Russian) by V.D. Goppa in 1970, see [8]. Even though Goppa origininally only considered binary Goppa codes, later definitions consider $\mathbb{F}_q$-linear Goppa codes. The following definition is from [19].

Definition 2.3.1. *Let $L = (a_1, \ldots, a_n)$ be an n-tuple of n distinct elements of $\mathbb{F}_{q^m}$. A polynomial g with coefficients in $\mathbb{F}_{q^m}$ such that $g(a_j) \neq 0$ for all j is called a* **Goppa polynomial** *with respect to L. The $\mathbb{F}_q$-linear* **Goppa code** *with respect to L and g is defined by*

$$\Gamma(L, g) := \left\{ \boldsymbol{c} \in \mathbb{F}_q^n : \sum_{j=1}^{n} \frac{c_j}{X - a_j} \equiv 0 \mod g(X) \right\}.$$

Note that $(X - a_i) \bmod g$ is a unit in $\mathbb{F}_q[X]/(g)$, because $g(a_i)$ is nonzero for all i. Therefore, $\frac{1}{X-a_i} \bmod g$ is indeed well defined.
Let us now investigate a small example of a Goppa code.

Example 2.3.2. We consider the field $\mathbb{F}_8$ with the element α satisfying $\alpha^3 = \alpha + 1$ and we choose $L = (0, 1, \alpha, \alpha^2, \ldots, a^6)$. Because α has order 7 in $\mathbb{F}_8^\times$, these 7 coordinates are indeed distinct elements of $\mathbb{F}_8$. Let $g(X) = X^2+X+\alpha^3$, having no roots in $\mathbb{F}_8$, be our Goppa polynomial. The description of our code is as follows:

$$\Gamma(L, g) := \left\{ \mathbf{c} \in \mathbb{F}_2^8 : \frac{c_1}{X} + \sum_{j=2}^{8} \frac{c_j}{X + \alpha^{j-2}} \equiv 0 \mod (X^2 + X + \alpha^3) \right\}.$$

We need the inverses of X and $(X + \alpha^{j-2}) \mod g(X)$. Using that $X^2 \equiv X + \alpha^3$ and $\alpha^3 = \alpha + 1$, we find:

- $X^{-1} \equiv \alpha^4 X + \alpha^4 \mod g(X)$
- $(X + 1)^{-1} \equiv \alpha^4 X \mod g(X)$
- $(X + \alpha)^{-1} \equiv \alpha X + \alpha^4 \mod g(X)$
- $(X + \alpha^2)^{-1} \equiv X + \alpha^6 \mod g(X)$
- $(X + \alpha^3)^{-1} \equiv \alpha X + \alpha^2 \mod g(X)$
- $(X + \alpha^4)^{-1} \equiv \alpha^2 X + 1 \mod g(X)$
- $(X + \alpha^5)^{-1} \equiv \alpha^2 X + \alpha^6 \mod g(X)$
- $(X + \alpha^6)^{-1} \equiv X + \alpha^2 \mod g(X)$

If we substitute these inverses in the description of our Goppa code, we see that our words $(c_1, c_2, \ldots, c_8)$ satisfy:

$$\begin{aligned}\sum_{j=1}^{n} \frac{c_j}{X - a_j} &= c_1(\alpha^4 X + \alpha^4) + c_2(\alpha^4 X) + c_3(\alpha X + \alpha^4) + c_4(X + \alpha^6) \\ &\quad + c_5(\alpha X + \alpha^2) + c_6(\alpha^2 X + 1) + c_7(\alpha^2 X + \alpha^6) + c_8(X + \alpha^2) \\ &\equiv 0 \mod g(X).\end{aligned}$$

Because both the coefficient of X and the constant coefficient must equal zero, we are looking for the solutions of the following system:

$$\begin{pmatrix} \alpha^4 & \alpha^4 & \alpha & 1 & \alpha & \alpha^2 & \alpha^2 & 1 \\ \alpha^4 & 0 & \alpha^4 & \alpha^6 & \alpha^2 & 1 & \alpha^6 & \alpha^2 \end{pmatrix} \mathbf{c} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Using the relations in $\mathbb{F}_8$, we rewrite this as follows:

$$\begin{pmatrix} \alpha^2+\alpha & \alpha^2+\alpha & \alpha & 1 & \alpha & \alpha^2 & \alpha^2 & 1 \\ \alpha^2+\alpha & 0 & \alpha^2+\alpha & \alpha^2+1 & \alpha^2 & 1 & \alpha^2+1 & \alpha^2 \end{pmatrix} \mathbf{c} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

This is useful, because if we look at the coefficients of α^2, α and 1 for each equation, we see that we can write these two equations over $\mathbb{F}_8$ as six equations over $\mathbb{F}_2$:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \mathbf{c} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

This is equivalent to

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \mathbf{c} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

We see that the solutions are $k_1 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} + k_2 \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$, with $k_1, k_2 \in \mathbb{F}_2$. In other words, the Goppa code $\Gamma(L, g)$ is generated by the matrix

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

or, equivalently, by

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

The minimum distance of this code is easily found, since there are only four words in the code. We have already seen the three nonzero code words and they have weight 5 or 6, which means that the minimum distance of this code is 5. This code is therefore an $[8, 2, 5]$ code.

It is also possible to use MAGMA to compute (generator matrices of) codes. An example which is a bit larger than the previous one is the following. The MAGMA code can be found in Appendix B.2.

Example 2.3.3. Consider the field $\mathbb{F}_{32}$ with the element α satisfying $\alpha^5 = \alpha^2+1$ and choose $L = (0, 1, \alpha, \alpha^2, \ldots, a^{30})$. Let $g = x^5+\alpha^{13}x^4+\alpha^{12}x^2+\alpha^{26}$, having no roots in $\mathbb{F}_{32}$, be our Goppa polynomial. This gives the Goppa code with parameters $[32, 7, 11]$ and generator matrix

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

Proposition 2.3.4. *The Goppa code $\Gamma(L, g)$ is an $[n, k, d]$ code with $k \geq n-mr$ and $d \geq r+1$, where r is the degree of g, a Goppa polynomial over $\mathbb{F}_{q^m}$.*

Proof. The fact about the dimension is illustrated by the example above. If the degree of g is r, comparing the coefficients of X^0 up to X^{r-1} gives r equations over $\mathbb{F}_{q^m}$. Each of these r equations leads to m equations over $\mathbb{F}_q$. This means that there are at most rm linear independent equations, corresponding to at most rm rows in the parity check matrix of $\Gamma(L, g)$. This means that the dimension of $\Gamma(L, g)$ must be at least $n - mr$.
In order to prove the fact about the minimal distance d, assume $d \leq \deg(g)$. This means that there is a code word $\mathbf{c}$ with weight d. Let the nonzero positions of this word be $S = \{i_1, \ldots, i_d\}$. Because $\mathbf{c} \in \Gamma(L, g)$, the following equation is satisfied:

$$\frac{c_{i_1}}{X - a_{i_1}} + \ldots + \frac{c_{i_d}}{X - a_{i_d}} \equiv 0 \mod g(X).$$

If we multiply both sides of this equation by $\prod_{j \in S}(X - a_j)$, we get

$$\sum_{1 \leq k \leq d} \left(c_{i_k} \prod_{j \in S,\ j \neq i_k} (X - a_k) \right) \equiv 0 \mod g(X).$$

Because the left hand side is a polynomial of degree $d-1 < \deg(g)$, we conclude that it must be the zero polynomial. If we evaluate the left hand side at $X = a_{i_1}$, we get

$$c_{i_1} \prod_{j \in S,\ j \neq i_1} (a_{i_1} - a_k) = 0.$$

Because all a_i are distinct, this yields a contradiction. □

A subset of Goppa codes that turns out to be interesting for our purposes is the set of binary Goppa codes generated by a square free polynomial g. These codes have a higher lower bound for the minimum distance. This follows from the fact that a square free polynomial and its square generate the same Goppa code.

Lemma 2.3.5. *Let g be a square free Goppa polynomial of degree r with coefficients in $\mathbb{F}_{2^m}$. Then the binary Goppa code $\Gamma(L, g)$ is equal to $\Gamma(L, g^2)$.*

Proof. First of all, we note that if g is a Goppa polynomial with respect to L, then g^2 is also a Goppa polynomial with respect to L, so the Goppa code $\Gamma(L, g^2)$ is well defined.
($\supseteq$) This is the easier inclusion. Let $\mathbf{c} = (c_1, \ldots, c_n) \in \Gamma(L, g^2)$, which means that $\sum_{j=1}^{n} \frac{c_j}{X - a_j} \equiv 0 \mod g^2(X)$. Because g divides g^2, it is also true that $\sum_{j=1}^{n} \frac{c_j}{X - a_j} \equiv 0 \mod g(X)$, which means that $\mathbf{c} \in \Gamma(L, g)$.
($\subseteq$) Let $\mathbf{c} = (c_1, \ldots, c_n) \in \Gamma(L, g)$ and define the polynomial

$$f(X) := \prod_{j=1}^{n} (X - a_j)^{c_j}.$$

The derivative of $f(X)$ is

$$f'(X) = \left(\sum_{j=1}^{n} c_j (X - a_j)^{c_j - 1} \prod_{i \neq j} (X - a_i)^{c_i} \right).$$

Note that this expression is well defined as an element of $\mathbb{F}_{2^m}[X]/(g)$ because $g(a_j)$ is nonzero for all j. This polynomial will be useful because

$$\frac{f'(X)}{f(X)} = \sum_{j=1}^{n} \frac{c_j}{X - a_j}.$$

$f(X)$ is a polynomial of degree n, so let us write it as $f(X) = \sum_{j=0}^{n} f_j X^j$. The derivative is $f'(X) = \sum_{j=1}^{n} j f_j X^{j-1}$. We work in characteristic two, so all coefficients of the terms with odd exponents are zero, which means that $f'(X)$ only has terms with even exponents. Because we work in $\mathbb{F}_{2^m}[X]/(g)$, it is true that $(b^{(2^{m-1})})^2 = b^{(2^m)} = b$ and we see that:

$$f'(X) = \sum_{j=0}^{k} b_j x^{2j} = \sum_{j=0}^{k} \left(b_j^{(2^{m-1})} x^j \right)^2 = \left(\sum_{j=0}^{k} b_j^{(2^{m-1})} x^j \right)^2,$$

which means that $f'(X)$ is a square.
Because $\mathbf{c} \in \Gamma(L, g)$, we know that $\sum_{j=1}^{n} \frac{c_j}{X - a_j} \equiv 0 \mod g(X)$, so also $\frac{f'(X)}{f(X)} \equiv 0 \mod g(X)$. This means that $g(X)$, a square free polynomial, divides $f'(X)$. This implies that $g^2(X)$ divides $f'(X)$ as well. We conclude that

$$\sum_{j=1}^{n} \frac{c_j}{X - a_j} = \frac{f'(X)}{f(X)} \equiv 0 \mod g^2(X),$$

and because $c_i \in \mathbb{F}_2$ (by the definition of f), it is indeed true that $\mathbf{c} \in \Gamma(L, g^2)$. □

Proposition 2.3.6. *Let g be a square free Goppa polynomial of degree r with coefficients in $\mathbb{F}_{2^m}$. Then the binary Goppa code $\Gamma(L, g)$ is an $[n, k, d]$ code with $k \geq n - mr$ and $d \geq 2r + 1$.*

Proof. The statement about the dimension follows directly from Proposition 2.3.4. The statement about the minimum distance follows from the fact that the minimum distance of $\Gamma(L, g^2)$ is at least $\deg(g^2) + 1 = 2r + 1$ (Proposition 2.3.4) and the fact that $\Gamma(L, g) = \Gamma(L, g^2)$ (Lemma 2.3.5). □

The following is used by McEliece [14, p.114], and we will need it again in section 5.1 about the McEliece cryptosystem:

Proposition 2.3.7. *Corresponding to each irreducible polynomial of degree t over $\mathbb{F}_{2^m}$, there exists a binary irreducible Goppa code of length $n = 2^m$, dimension $k \geq n - tm$, capable of correcting any pattern of t or fewer errors. Moreover, there exists a fast algorithm for decoding these codes.*

Proof. Let g be an irreducible polynomial of degree t over $\mathbb{F}_{2^m}$. Since g does not have any zeroes in $\mathbb{F}_{2^m}$, we can choose L to contain exactly all elements of $\mathbb{F}_{2^m}$ and obtain a code of length $n = 2^m$. The dimension, $k \geq n - tm$, follows from Proposition 2.3.4. Because an irreducible polynomial is certainly square free, we can use Proposition 2.3.6 to conclude that the mimimum distance is $d \geq 2t + 1$. This means that, for every received word with t or less errors, there exists a unique closest code word, which means that t or fewer errors can be corrected. The fast decoding algorithm is Patterson's algorithm, to which we will come back in chapter 4. □

2.4 AG codes and geometric Goppa codes

This section considers two types of codes which are based on algebraic geometry. Just like the Goppa codes from the previous section, these codes were also introduced by Goppa, see [9]. Even for those who do not speak Russian, the two classes of codes are clearly recognizable in the article. In some sources, the algebraic geometry codes that we discuss in this section are also called Goppa codes. In order to avoid confusion, we will refer to the Goppa codes from the

previous section as 'classical Goppa codes'. We follow [19] in the names and definitions of the codes in this section. First, the 'AG code' will be introduced, and then the 'geometric Goppa code'.

From now on, let $\mathcal{X}$ be an absolutely irreducible non-singular projective curve over $\mathbb{F}_q$. Let $P_1, \ldots, P_n$ be distinct rational points on $\mathcal{X}$, let $\mathcal{P}$ be the sequence (P_i) and let D be the divisor $P_1 + \cdots + P_n$. Let G be some other divisor that has support disjoint from D. Furthermore, assume $2g - 2 < \deg(G) < n$.

Definition 2.4.1. *The* **AG code** *(or geometric RS code) $C_L(D, G)$ is the image of the linear map*

$$\mathrm{ev}_{\mathcal{P}} : L(G) \to \mathbb{F}_q^n, f \mapsto (f(P_1), \ldots, f(P_n)).$$

Because G and D have disjoint support, the fact that $f \in L(G)$ means that f can have no poles at the P_i. Therefore, the AG code is well defined. Note that the order of the P_i determines the order of the coordinates. Because the divisor D does not contain information about the order of the points, a more precise notation would perhaps have been $C_L(\mathcal{X}, \mathcal{P}, G)$ (as is used in [4]), but we will stick to the notation $C_L(D, G)$ and remember that the order of the points P_i matters. Let us look at a small example of an AG code, using a curve we have already seen in section 1.2:

Example 2.4.2. Let the curve over $\mathbb{F}_4$ be defined by $X^3 + Y^3 + Z^3 = 0$. Let $Q = (0 : 1 : 1)$, $G = 3Q$ and $n = 8$. This means that we must choose the points P_i to be all of the remaining rational points, that is:

$$P_1 = (0 : \alpha : 1), \quad P_2 = (0 : \alpha^2 : 1), \quad P_3 = (1 : 0 : 1), \quad P_4 = (\alpha : 0 : 1),$$

$$P_5 = (\alpha^2 : 0 : 1), \quad P_6 = (1 : 1 : 0), \quad P_7 = (\alpha : 1 : 0), \quad P_8 = (\alpha^2 : 1 : 0).$$

We have already seen that $L(G)$ has dimension 3 and basis $\{1, \frac{x}{y+z}, \frac{y}{y+z}\}$. If we compute the images of our basis functions, we find the following code words:

$$\mathrm{ev}_{\mathcal{P}}(1) = (1, 1, 1, 1, 1, 1, 1, 1),$$

$$\mathrm{ev}_{\mathcal{P}}\left(\frac{x}{y+z}\right) = (0, 0, 1, \alpha, \alpha^2, 1, \alpha, \alpha^2),$$

$$\mathrm{ev}_{\mathcal{P}}\left(\frac{y}{y+z}\right) = (\alpha^2, \alpha, 0, 0, 0, 1, 1, 1),$$

These words are clearly linearly independent, so we have found a generator matrix for our code $C_L(D, G)$:

$$G' = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & \alpha & \alpha^2 & 1 & \alpha & \alpha^2 \\ \alpha^2 & \alpha & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

We have seen that $n = 8$ and $k = 3$. The third row of the generator matrix is a code word of weight 5, which means that the minimum distance should be at most 5. It turns out that d is also at least 5, so that $C_L(D, G)$ is an $[8, 3, 5]$ code.

This lower bound for d follows from the following general proposition concerning the dimension and minimum distance of AG codes:

Proposition 2.4.3. *The code $C_L(D, G)$ has dimension $k = \deg(G) - g + 1$ and minimum distance $d \geq n - \deg(G)$.*

Proof. (i) First we show that $\mathrm{ev}_{\mathcal{P}} : L(G) \to \mathbb{F}_q^n$ is injective. Suppose $f \in \ker(\mathrm{ev}_{\mathcal{P}})$. That means $f(P_i) = 0$ for all P_i in the support of D, so all P_i occur in (f) with a positive coefficient. Because the P_i are distinct and $D = P_1 + \ldots + P_n$, all P_i occur in $(f) - D$ with a nonzero coefficient. We remember that $L(G) = \{f \in \mathbb{F}(\mathcal{X})^* \mid (f) + G \geq 0\} \cup \{0\}$. The fact that $f \in L(G)$ means that $(f) + G \geq 0$ and because D and G have distinct support, we conclude that $(f) + G - D \geq 0$, which means that $f \in L(G - D)$. However, we assumed that $\deg(G) < n$ and $\deg(D) = n$, so $\deg(G-D) < 0$ which implies that $l(G-D) = 0$ and therefore $f = 0$. So indeed $\mathrm{ev}_{\mathcal{P}} : L(G) \to \mathbb{F}_q^n$ is injective. Then we use the fact that $l(G) = \deg(G) - g + 1$ if $\deg(G) > 2g - 2$ and conclude that this is indeed the dimension of $C_L(D, G)$.
(ii) Let d be the minimum distance and let f be such that $\mathrm{ev}_{\mathcal{P}}(f)$ has weight d. Then there are $n - d$ points P_i for which $f(P_i) = 0$. Enumerate these points by $P_{i_1}, \ldots, P_{i_{n-d}}$ and let $E = P_{i_1} + \ldots + P_{i_{n-d}}$. By the same argument as above, $f \in L(G-E)$. So $(f)+G-E \geq 0$, which means that $\deg(G-E) \geq \deg((f)) = 0$. We use that $\deg(G-E) = \deg(G)-\deg(E)$ and that $\deg(E) = n-d$ to conclude that $\deg(G) - n + d \geq 0$, so indeed $d \geq n - \deg(G)$. □

There is another type of algebraic geometry code, also called the geometric Goppa code. We will see later that these codes are in fact the dual codes of the AG codes we introduced above.

Definition 2.4.4. *The* **geometric Goppa code** *$C_\Omega(D, G)$ of length n over $\mathbb{F}_q$ is the image of the linear map $\mathrm{Res}_{\mathcal{P}} : \Omega(G - D) \to \mathbb{F}_q^n$ defined by $\eta \mapsto (\mathrm{Res}_{P_1}(\eta), \ldots, \mathrm{Res}_{P_n}(\eta))$.*

Actually, we have already encountered an example of a geometric Goppa code: a special case of the (classical) Goppa code from section 2.3. Suppose that $\Gamma(L, g)$ is an $\mathbb{F}_q$-linear Goppa code as defined in Definition 2.3.1, such that $L = (a_1, \ldots, a_n)$ with $a_i \in \mathbb{F}_q$ and let g be the Goppa polynomial with coefficients in $\mathbb{F}_q$. We can take $\mathcal{X}$ to be the projective line, define $P_i := (a_i : 1)$, $Q := (1 : 0)$, $D := P_1 + \ldots + P_n$ and let E be the divisor given by the zeros of g on the projective line, counted with multiplicity. It is then true that $\Gamma(L, g) = C_\Omega(D, E - Q)$. For a proof, see [5] (Thm 37). They also explain that in general, a classical $\mathbb{F}_q$-linear Goppa code is defined by a polynomial g with coefficients in F_{q^m} and $a_i \in \mathbb{F}_{q^m}$. If $q \neq q^m$, then the classical Goppa code

is not exactly a geometric Goppa code, but the restriction of some geometric Goppa code, or, in other words, a 'subfield subcode' of some geometric Goppa code from the projective line: $\Gamma(L, g) = C_\Omega(D, E - Q)|_{\mathbb{F}_q}$.

Another example of a geometric Goppa code, using a familiar curve and space, is the following:

Example 2.4.5. Consider the same curve and points as in example 2.4.2. This time, let $G = 6Q$. In example 1.2.6, we already looked at the space $\Omega(6Q - D)$ and we found two differentials η and ζ that formed its basis. This means that $C_\Omega(D, 6Q)$ is generated by $\mathrm{Res}_\mathcal{P}(\eta)$ and $\mathrm{Res}_\mathcal{P}(\zeta)$. A generator matrix for $C_\Omega(D, G)$ is therefore

$$G_\Omega = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & \alpha & \alpha^2 & 1 & \alpha & \alpha^2 \end{pmatrix}.$$

We see that the dimension is 2 and as there is only one other nonzero word, we easily see that the minimum distance is 6.

Proposition 2.4.6. *The code $C_\Omega(D, G)$ has dimension $k^* = n - \deg(G) + g - 1$ and minimum distance $d^* \geq \deg(G) - 2g + 2$.*

Proof. (i) Again we have an injective map, which we show as follows. Suppose $\eta \in \ker(\mathrm{Res}_\mathcal{P})$. That means $\mathrm{Res}_{P_i}(\eta) = 0$ for all P_i in the support of D, so η has no single poles at the P_i. Because $\eta \in \Omega(G - D)$, we know that $(\eta) - G + D \geq 0$. Since $D = P_1 + \ldots + P_n$ and since G and D have distinct support, this means that if η has a pole at one of the points P_i, the order of the pole can at most be 1, so we conclude that η has no poles at the points P_i. Therefore $(\eta) - G \geq 0$, which means $\eta \in \Omega(G)$. According to proposition 1.2.4(7), the dimension of this space is $\delta(G) = l(W - G)$. We know that $\deg(W) = 2g - 2$ for a canonical divisor, and our assumption $\deg(G) > 2g - 2$ then gives that $\deg(W - G) < 0$, so $l(W - G) = 0$. This means that $\eta = 0$ and that $\mathrm{Res}_\mathcal{P}$ indeed is injective.
The dimension of $\Omega(G - D)$, which we denote by $\delta(G - D)$, is by proposition 1.2.4(7) equal to $l(W - G + D)$ for a canonical divisor W. Riemann-Roch gives us:

$$l(G - D) - l(W - G + D) = \deg(G - D) - g + 1, \text{ which leads to}$$

$$0 - l(W - G + D) = \deg(G) - n - g + 1,$$

so indeed $l(W - G + D) = n - \deg(G) + g - 1$.
(ii) Let $\eta \in \Omega(G - D)$ be a differential such that $(\mathrm{Res}_{P_1}(\eta), \ldots, \mathrm{Res}_{P_n}(\eta))$ has weight d^*. Just as before, this means that there are $n - d^*$ points for which $\mathrm{Res}_{P_i}(\eta) = 0$. Let us enumerate these points by $P_{i_1}, \ldots, P_{i_{n-d^*}}$. Let $E = P_{i_1} + \ldots + P_{i_{n-d^*}}$. We know that $(\eta) - G + D \geq 0$ and because there are no single poles at points in the support of E we get that $(\eta) - G + D - E \geq 0$. It follows that $d^* = \deg(D - E) \geq \deg(G) - \deg((\eta))$. (η) is a canonical divisor so its degree is $2g - 2$. This gives the desired result $d^* \geq \deg(G) - 2g + 2$. □

We have seen that both $C_L(D,G)$ and $C_\Omega(D,G)$ have a lower bound for their minimum distance. These lower bounds are useful because they ensure (at least in theory) a possibility to correct errors. The lower bounds are also called the designed minimum distance:

Definition 2.4.7. *The* **designed minimum distance of** $C_L(D,G)$ *is* $n - \deg(G)$ *and the* **designed minimum distance of** $C_\Omega(D,G)$ *is* $\deg(G)-2g+2$.

Proposition 2.4.8. *The codes $C_L(D,G)$ and $C_\Omega(D,G)$ are dual codes.*

Proof. First of all, we notice that the dimensions of $C_L(D,G)$ and $C_\Omega(D,G)$ add up to n: $k + k^* = (\deg(G) - g + 1) + (n - \deg(G) + g - 1) = n$. Now we will take one word from both codes and show that the dot product is zero. Let $c_1 \in C_L(D,G)$ and $c_2 \in C_\Omega(D,G)$. There are $f \in L(G)$ and $\eta \in \Omega(G-D)$ such that $c_1 = (f(P_1), \ldots, f(P_n))$ and $c_2 = (\mathrm{Res}_{P_1}(\eta), \ldots, \mathrm{Res}_{P_n}(\eta))$. Since $f \in L(G)$, $(f) + G \geq 0$ and since $\eta \in \Omega(G-D)$, $(\eta) - G + D \geq 0$. It follows that $(f\eta) + D = (f) + (\eta) + D \geq 0$. Therefore, if $f\eta$ has any poles, they are poles of order 1 located at the P_i. We use proposition 1.2.4(6) to see that

$$0 = \sum_{P \in \mathcal{X}} \mathrm{Res}_P(f\eta) = \sum_{P_i} \mathrm{Res}_{P_i}(f\eta) = \sum_{P_i} f(P_i)\mathrm{Res}_{P_i}(\eta) = c_1 \cdot c_2,$$

where the third equality follows from the fact that f has no poles at P_i. This concludes the proof. □

Example 2.4.9. In example 2.4.5, we computed the code $C_\Omega(D, 6Q)$. According to the previous proposition, this should be the dual code of $C_L(D, 6Q)$. We already found a basis for $L(6Q)$ in example 1.3.3: $\{1, f, g, f^2, fg, g^2\}$ with $f = \frac{x}{y+z}$ and $g = \frac{y}{y+z}$. A generator matrix of $C_L(D, 6Q)$ is therefore given by

$$G_L = \begin{pmatrix} ev_\mathcal{P}(1) \\ ev_\mathcal{P}(f) \\ ev_\mathcal{P}(g) \\ ev_\mathcal{P}(f^2) \\ ev_\mathcal{P}(fg) \\ ev_\mathcal{P}(g^2) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & \alpha & \alpha^2 & 1 & \alpha & \alpha^2 \\ \alpha^2 & \alpha & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & \alpha^2 & \alpha & 1 & \alpha^2 & \alpha \\ 0 & 0 & 0 & 0 & 0 & 1 & \alpha & \alpha^2 \\ \alpha & \alpha^2 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

In order to check that $C_\Omega(D, 6Q)$ en $C_L(D, 6Q)$ are indeed dual codes, we can compute the matrix multiplication $G_L(G_\Omega)^T$ and see that this gives the zero matrix. This means that $C_L(D, 6Q) \perp C_\Omega(D, 6Q)$. Because the dimensions of $C_\Omega(D, 6Q)$ en $C_L(D, 6Q)$ add up to $8 = n$, they are indeed dual codes.

Proposition 2.4.10. *Let $C_L(D,G)$ be an AG code as in Definition 2.4.1. There exists a differential form ω with simple poles at the P_i such that $Res_{P_i}(\omega) = 1$ for all i. Let W be the divisor of ω. Then*

$$(C_L(D,G))^\perp = C_L(D, W + D - G).$$

Proof. The fact that the differential form ω exists, follows from [22] (Lemma 2.2.9 on page 53). Before we prove that the codes are equal, we notice that $C_L(D, W + D - G)$ is well defined. Because ω has poles of order 1 at the P_i and $W = (\omega)$, we can write $W = W' - P_1 - \ldots - P_n = W' - D$, where W' and D have disjoint support. In de definition of $C_L(D, G)$ we assumed that D and G have disjoint support so we conclude that $W + D - G = W' - G$ also has support disjoint from D. Therefore, the code on the right hand side is indeed well defined.

Now we will show that the codes are the same. We already know that

$$(C_L(D, G))^\perp = C_\Omega(D, G).$$

By proposition 1.2.4(7), $\delta(G - D) = l(W + D - G)$, which means that the dimensions of $C_\Omega(D, G)$ and $C_L(D, W + D - G)$ are equal. In order to show that $C_L(D, W + D - G) \subseteq C_\Omega(D, G)$, take $\mathbf{c} \in C_L(D, W + D - G)$. Then

$$\begin{aligned}\mathbf{c} &= (f(P_1), \ldots, f(P_n)) \\ &= (f(P_1)\mathrm{Res}_{P_1}(\omega), \ldots, f(P_n)\mathrm{Res}_{P_n}(\omega)) \\ &= (\mathrm{Res}_{P_1}(f\omega), \ldots, \mathrm{Res}_{P_n}(f\omega)).\end{aligned}$$

In order to see that $f\omega \in \Omega(G - D)$, we use the fact that $f \in L(W + D - G)$, which means that $(f) + W + D - G \geq 0$, and see that

$$(f\omega) - (G - D) = (f) + (\omega) + D - G = (f) + W + D - G \geq 0.$$

This means that indeed $f\omega \in \Omega(G - D)$, so $\mathbf{c} \in C_\Omega(D, G)$. Together with the equality of the dimensions, this implies that

$$(C_L(D, G))^\perp = C_\Omega(D, G) = C_L(D, W + D - G).$$

□

Example 2.4.11. In example 2.4.5, we computed $C_\Omega(D, 6Q)$. In example 1.2.6, we already saw that η has simple poles and residue 1 at all points P_i and divisor $(\eta) = E = 8Q - D$. According to proposition 2.4.10, we should have

$$C_\Omega(D, 6Q) = C_L(D, E + D - 6Q) = C_L(D, 8Q - D + D - 6Q) = C_L(D, 2Q).$$

It is indeed true that $\{1, f\}$ is a basis for $L(2Q)$ and evaluating these functions in the P_i gives the same generator matrix as we saw in example 2.4.5.

The following property of AG codes will turn out to be crucial in the attack in chapter 6.

Corollary 2.4.12. *Every AG code is the dual of another AG code.*

Proof. If you take the dual code on both sides of the equation of proposition 2.4.10, you see that $C_L(D, G) = C_L(D, W + D - G)^\perp$. □

Chapter 3

The Schur product

In section 4.3 about decoding and in chapter 6 about the attack on AG codes, we will encounter an operation called the Schur product. This chapter is dedicated to this operation and its properties, in particular in relation to codes.

3.1 The Schur product of vectors

The Schur product on vectors, also called the Hadamard product, entry-wise product or element-wise product, is defined as follows:

Definition 3.1.1. *For two vectors* $\boldsymbol{a} = (a_1, \ldots, a_n)$ *and* $\boldsymbol{b} = (b_1, \ldots, b_n)$*, both in* $\mathbb{F}_q^n$*, the* **Schur product** $\boldsymbol{a} * \boldsymbol{b}$ *is defined by* $\boldsymbol{a} * \boldsymbol{b} := (a_1 b_1, \ldots, a_n b_n)$*.*

There are some properties of the Schur product that follow directly from this definition:

Proposition 3.1.2. *Let* $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_q^n$ *and* $c \in \mathbb{F}_q$*. The Schur product is symmetric and bilinear over* $\mathbb{F}_q$*, i.e., for* $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_q^n$ *and* $c \in F_q$*, the following statements are true:*

1. $\boldsymbol{a} * \boldsymbol{b} = \boldsymbol{b} * \boldsymbol{a}$
2. $(c\boldsymbol{a}) * \boldsymbol{b} = c(\boldsymbol{a} * \boldsymbol{b}) = \boldsymbol{a} * (c\boldsymbol{b})$
3. $(\boldsymbol{a} + \boldsymbol{b}) * \boldsymbol{d} = \boldsymbol{a} * \boldsymbol{d} + \boldsymbol{b} * \boldsymbol{d}$ *and* $\boldsymbol{a} * (\boldsymbol{b} + \boldsymbol{d}) = \boldsymbol{a} * \boldsymbol{d} + \boldsymbol{a} * \boldsymbol{d}$

Proof. 1. Because multiplication in $\mathbb{F}_q$ is commutative,

$$\mathbf{a} * \mathbf{b} = (a_1 b_1, \ldots, a_n b_n) = (b_1 a_1, \ldots, b_n a_n) = \mathbf{b} * \mathbf{a}.$$

2. $c\mathbf{a} = c(a_1, \ldots, a_n) = (ca_1, \ldots, ca_n)$, so

$$(c\mathbf{a}) * \mathbf{b} = (ca_1 b_1, \ldots, ca_n b_n) = c(a_1 b_1, \ldots, a_n b_n) = c(\mathbf{a} * \mathbf{b}).$$

The second equality follows in the same way.

3. Because $\mathbb{F}_q$ is distributive,

$$(\mathbf{a}+\mathbf{b})*\mathbf{d} = (a_1+b_1)d_1, \ldots, (a_n+b_n)d_n = (a_1d_1+b_1d_1, \ldots, a_nd_n+b_nd_n)$$

$$= (a_1d_1, \ldots, a_nd_n) + (b_1d_1, \ldots, b_nd_n) = \mathbf{a}*\mathbf{d} + \mathbf{b}*\mathbf{d}.$$

The second equality follows in the same way. □

Combining these properties, we find the following:

Corollary 3.1.3.

$$\sum_i a_i \boldsymbol{a}_i * \sum_j b_j \boldsymbol{b}_j = \sum_i \sum_j a_i b_j \left(\boldsymbol{a}_i * \boldsymbol{b}_j\right).$$

Proof. We can apply property 3 repeatedly to see that $\sum_i a_i\mathbf{a}_i * \sum_j b_j\mathbf{b}_j = \sum_i \left(a_i\mathbf{a}_i * \sum_j b_j\mathbf{b}_j\right) = \sum_i \sum_j \left(a_i\mathbf{a}_i * b_j\mathbf{b}_j\right)$ and then we use statement 2 to see that this equals $\sum_i \sum_j a_ib_j \left(\mathbf{a}_i * \mathbf{b}_j\right)$. □

3.2 The Schur product of codes

We now define the Schur product of two codes, that is, linear subspaces of $\mathbb{F}_q^n$:

Definition 3.2.1. *For two codes $A, B \subseteq F_q^n$, the Schur product $A*B$ is de code defined by*

$$A * B := \mathrm{Span}_{\mathbb{F}_q}\{\boldsymbol{a} * \boldsymbol{b} \mid \boldsymbol{a} \in A \text{ and } \boldsymbol{b} \in B\}.$$

In the special case $B = A$, we call $A*A$ the Schur square and we use the shorter notation $A^{(2)} := A * A$.

Example 3.2.2. The Schur square of the Hamming code
Let us look again at the Hamming code C from section 2.2: the $[7, 4, 3]$-code over $\mathbb{F}_2$ with generator matrix

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

Let the rows of this matrix be called $\mathbf{g}_1$, $\mathbf{g}_2$, $\mathbf{g}_3$ and $\mathbf{g}_4$. The Schur square $C^{(2)}$ of this code contains $\mathbf{g}_1 * \mathbf{g}_1 = \mathbf{g}_1$. Furthermore, $\mathbf{g}_1 * \mathbf{g}_2 = (0, 0, 0, 0, 0, 0, 1) \in C^{(2)}$ and also $\mathbf{g}_1 * \mathbf{g}_3 = (0, 0, 0, 0, 0, 1, 0) \in C^{(2)}$. Therefore, it follows that $\mathbf{g}_1 + (0, 0, 0, 0, 0, 0, 1) + (0, 0, 0, 0, 0, 1, 0) = (1, 0, 0, 0, 0, 0, 0) \in C^{(2)}$. In fact, if we continue in this way, we quickly find that the Schur square of the Hamming code C is equal to $\mathbb{F}_2^7$.

It is not always the case that the Schur square of a code equals the whole space. For example, look at the Goppa code C from section 2.3.

Example 3.2.3. The Schur square of a Goppa code
The code C over $\mathbb{F}_2$ has generator matrix

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

Let the rows of this matrix be $\mathbf{g}_1$ and $\mathbf{g}_2$. The only other nonzero word in C is $\mathbf{g}_3 := \mathbf{g}_1 + \mathbf{g}_2 = (1,1,1,0,1,1,1,0)$.

This means that $C^{(2)}$ is spanned by

$$\{\mathbf{g}_1 * \mathbf{g}_1,\ \ \mathbf{g}_2 * \mathbf{g}_2,\ \ \mathbf{g}_3 * \mathbf{g}_3,\ \ \mathbf{g}_1 * \mathbf{g}_2,\ \ \mathbf{g}_1 * \mathbf{g}_3,\ \ \mathbf{g}_2 * \mathbf{g}_3\}.$$

However, it follows from Prop. 3.1.2 that

$$\begin{aligned}
\mathbf{g}_1 * \mathbf{g}_3 &= \mathbf{g}_1 * (\mathbf{g}_1 + \mathbf{g}_2) = \mathbf{g}_1 * \mathbf{g}_1 + \mathbf{g}_1 * \mathbf{g}_2, \\
\mathbf{g}_2 * \mathbf{g}_3 &= \mathbf{g}_2 * (\mathbf{g}_2 + \mathbf{g}_1) = \mathbf{g}_2 * \mathbf{g}_2 + \mathbf{g}_2 * \mathbf{g}_1 = \mathbf{g}_2 * \mathbf{g}_2 + \mathbf{g}_1 * \mathbf{g}_2 \text{ and} \\
\mathbf{g}_3 * \mathbf{g}_3 &= (\mathbf{g}_1 + \mathbf{g}_2) * (\mathbf{g}_1 + \mathbf{g}_2) = (\mathbf{g}_1 + \mathbf{g}_2) * \mathbf{g}_1 + (\mathbf{g}_1 + \mathbf{g}_2) * \mathbf{g}_2 \\
&= \mathbf{g}_1 * \mathbf{g}_1 + \mathbf{g}_2 * \mathbf{g}_1 + \mathbf{g}_1 * \mathbf{g}_2 + \mathbf{g}_2 * \mathbf{g}_2 \\
&= \mathbf{g}_1 * \mathbf{g}_1 + \mathbf{g}_1 * \mathbf{g}_2 + \mathbf{g}_1 * \mathbf{g}_2 + \mathbf{g}_2 * \mathbf{g}_2,
\end{aligned}$$

which means that $C^{(2)}$ is spanned by

$$\begin{aligned}
\mathbf{g}_1 * \mathbf{g}_1 &= (1,0,1,1,0,1,0,1), \\
\mathbf{g}_2 * \mathbf{g}_2 &= (0,1,0,1,1,0,1,1), \\
\mathbf{g}_1 * \mathbf{g}_2 &= (0,0,0,1,0,0,0,1).
\end{aligned}$$

Because these are clearly linearly independent, $C^{(2)}$ is a code of dimension 3 generated by

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

Note that in codes over $\mathbb{F}_2$, it is always true that $\mathbf{c} * \mathbf{c} = \mathbf{c}$. This means that in fact $C^{(2)} = \text{Span}\{\mathbf{g}_1,\ \mathbf{g}_2,\ \mathbf{g}_1 * \mathbf{g}_2\}$.

In general, when finding the Schur product of two codes, it is only necessary to look at the Schur products of the basis words:

Proposition 3.2.4. *If $A, B \subseteq \mathbb{F}_q^n$ are two codes and if A' and B' are bases for A and B, respectively, then*

$$A * B = \text{Span}_{\mathbb{F}_q}\{\boldsymbol{a} * \boldsymbol{b} \mid \boldsymbol{a} \in A' \text{ and } \boldsymbol{b} \in B'\}.$$

Proof. The obvious inclusion is $\mathrm{Span}_{\mathbb{F}_q}\{\mathbf{a} * \mathbf{b} \mid \mathbf{a} \in A' \text{ and } \mathbf{b} \in B'\} \subseteq A * B$. In order to show the other inclusion, suppose $\mathbf{c} \in A * B = \mathrm{Span}_{\mathbb{F}_q}\{\mathbf{a} * \mathbf{b} \mid \mathbf{a} \in A \text{ and } \mathbf{b} \in B\}$. Then there are $\mathbf{a}_i \in A$ and $\mathbf{b}_i \in B$ such that $\mathbf{c} = \sum_i c_i(\mathbf{a}_i * \mathbf{b}_i)$. We can write these $\mathbf{a}_i$ and $\mathbf{b}_i$ as a combination of the elements in the basis, that is, $\mathbf{a}_i = \sum_j a_{ij}\mathbf{a}'_{ij}$ and $\mathbf{b}_i = \sum_j b_{ij}\mathbf{b}'_{ij}$. This gives

$$\mathbf{c} = \sum_i c_i \left(\sum_j a_{ij}\mathbf{a}'_{ij} * \sum_k b_{ik}\mathbf{b}'_{ik} \right).$$

Using corollary 3.1.3, we see that

$$\mathbf{c} = \sum_i c_i \left(\sum_j \sum_k a_{ij}b_{ik} \left(\mathbf{a}'_{ij} * \mathbf{b}'_{ik}\right) \right) = \sum_i \sum_j \sum_k c_i a_{ij} b_{ik} \left(\mathbf{a}'_{ij} * \mathbf{b}'_{ik}\right),$$

so indeed $\mathbf{c} \in \mathrm{Span}_{\mathbb{F}_q}\{\mathbf{a} * \mathbf{b} \mid \mathbf{a} \in A' \text{ and } \mathbf{b} \in B'\}$, which completes the proof. □

3.3 Schur squares of random codes

It is possible to give an upper bound on the dimension of $C^{(2)}$, given the dimension of a code C:

Proposition 3.3.1. *If $C \subseteq \mathbb{F}_q^n$ is a code of dimension k, the dimension of $C^{(2)}$ is at most $\binom{k}{2} + k$.*

Proof. Let C' be a basis for C. There are k elements in C' and we have

$$C^{(2)} = \mathrm{Span}\{\mathbf{c}_1 * \mathbf{c}_2 \mid \mathbf{c}_1, \mathbf{c}_2 \in C'\}.$$

$$= \mathrm{Span}\left\{\{\mathbf{c}_1 * \mathbf{c}_2 \mid \mathbf{c}_1, \mathbf{c}_2 \in C', \mathbf{c}_1 \neq \mathbf{c}_2\} \cup \{\mathbf{c} * \mathbf{c} \mid \mathbf{c} \in C'\}\right\}$$

We want to know the number of distinct $\mathbf{c}_1 * \mathbf{c}_2$. There are $\binom{k}{2}$ ways to pick $\mathbf{c}_1 \neq \mathbf{c}_2$ from C'. Because $\mathbf{c}_1 * \mathbf{c}_2 = \mathbf{c}_2 * \mathbf{c}_1$, there are at most $\binom{k}{2}$ distinct $\mathbf{c}_1 * \mathbf{c}_2$ with $\mathbf{c}_1 \neq \mathbf{c}_2$. Clearly, there are at most k distinct $\mathbf{c} * \mathbf{c}$. This means that $C^{(2)}$ is the span of a set with at most $\binom{k}{2} + k$ elements, that is, the dimension of $C^{(2)}$ is at most $\binom{k}{2} + k$. □

As we have already mentioned before, in codes over $\mathbb{F}_2$ we always have $\mathbf{c} * \mathbf{c} = \mathbf{c}$. This means that $C \subseteq C^{(2)}$ and therefore $\dim(C) \leq \dim(C^{(2)})$. Over a general finite field, $\mathbf{c} * \mathbf{c} = \mathbf{c}$ is not necessarily the case, but the fact about the dimension is true, so we also have a lower bound on the dimension of $C^{(2)}$:

Proposition 3.3.2. *If $C \subseteq \mathbb{F}_q^n$ is a code of dimension k, then the dimension of $C^{(2)}$ is at least k.*

Proof. Let G be an $k \times n$ generator matrix for C, in reduced row echelon form and let the rows of G be $\mathbf{g}_i$. That is: the leading entry in each $\mathbf{g}_i$ is a one and if $\mathbf{g}_j$ has a leading one in position m, then all $\mathbf{g}_i$ with $i \neq j$ have a zero in position m. The $\mathbf{g}_i$ form a basis for C. Now consider the Schur squares: $\mathbf{g}_i * \mathbf{g}_i$. All leading ones and all zeros stay the same, so all of the $\mathbf{g}_i * \mathbf{g}_i$ are linearly independent. Since all of these $\mathbf{g}_i * \mathbf{g}_i$ are in the basis of $C^{(2)}$, it follows that the dimension of $C^{(2)}$ is at least k. □

We have seen that the dimension of the Schur square of an $[n, k]$ is at least k and at most $\binom{k}{2} + k$. In example 3.2.2, we saw that the Schur square of an $[n, k]$ code can fill the whole space $\mathbb{F}_q^n$. In fact, this is fairly common: if C is an $[n, k]$ code with k a large number and n smaller than $\frac{k(k+1)}{2}$, the probability that this happens is very high. The bigger the difference between n and $k(k+1)/2$, the higher the probability. A proposition by Cascudo and others [3] makes this mathematically more precise. More on this can be found in Appendix 6.4.

3.4 The Schur product of AG codes

Before we look at any theorems, let us first examine the Schur square of an AG code we have already seen before:

Example 3.4.1. The Schur square of an AG code (1)
Let C be the AG code from example 2.4.2. The code over $\mathbb{F}_4$ has generator matrix

$$G' = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & \alpha & \alpha^2 & 1 & \alpha & \alpha^2 \\ \alpha^2 & \alpha & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Let the rows be $\mathbf{g}_1$, $\mathbf{g}_2$ and $\mathbf{g}_3$. The Schur square is spanned by:

$$\begin{aligned} \mathbf{g}_1 * \mathbf{g}_1 &= (1, 1, 1, 1, 1, 1, 1, 1) \\ \mathbf{g}_1 * \mathbf{g}_2 &= (0, 0, 1, \alpha, \alpha^2, 1, \alpha, \alpha^2) \\ \mathbf{g}_1 * \mathbf{g}_3 &= (\alpha^2, \alpha, 0, 0, 0, 1, 1, 1) \\ \mathbf{g}_2 * \mathbf{g}_2 &= (0, 0, 1, \alpha^2, \alpha, 1, \alpha^2, \alpha) \\ \mathbf{g}_2 * \mathbf{g}_3 &= (0, 0, 0, 0, 0, 1, \alpha, \alpha^2) \\ \mathbf{g}_3 * \mathbf{g}_3 &= (\alpha, \alpha^2, 0, 0, 0, 1, 1, 1) \end{aligned}$$

We can write the generator matrix of $C^{(2)}$ as

$$G'' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \alpha^2 & \alpha \\ 0 & 1 & 0 & 0 & 0 & 0 & \alpha^2 & \alpha \\ 0 & 0 & 1 & 0 & 0 & 0 & \alpha & \alpha^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & \alpha & \alpha^2 \end{pmatrix}$$

and see that $C^{(2)}$ is an $[8, 6, 2]$-code.

Proposition 3.4.2. *Let E and F be two divisors on a curve $\mathcal{X}$, both with support disjoint from D, such that* $\deg(E) \geq 2g+1$ *and* $\deg(F) \geq 2g$. *Then*

$$C_L(D,E) * C_L(D,F) = C_L(D,E+F).$$

Proof. This is a consequence of Proposition 1.3.2 on the product of Riemann Roch spaces:

$$\begin{aligned}
C_L(D,E) * C_L(D,F) &= \text{Span}\,\{\mathbf{a} * \mathbf{b} \mid \mathbf{a} \in C_L(D,E),\ b \in C_L(D,F)\} \\
&= \text{Span}\,\{(f(P_1),\ldots,f(P_n)) * (g(P_1),\ldots,g(P_n)) \mid f \in L(E),\ g \in L(F)\} \\
&= \text{Span}\,\{(fg(P_1),\ldots,fg(P_n)) \mid f \in L(E),\ g \in L(F)\} \\
&= \{(h(P_1),\ldots,h(P_n)) \mid h \in \text{Span}\{fg \mid f \in L(E),\ g \in L(F)\}\} \\
&= \{(h(P_1),\ldots,h(P_n)) \mid h \in L(E)\cdot L(F)\} \\
&= \{(h(P_1),\ldots,h(P_n)) \mid h \in L(E+F)\} \\
&= C_L(D,E+F).
\end{aligned}$$

□

Example 3.4.3. The Schur square of an AG code (2)
Let us look at the previous example (3.4.1) again, where we computed $C_L(D,3Q) * C_L(D,3Q)$. Instead of computing the Schur products of the code words that form the basis of $C_L(D,3Q)$, we could have used proposition 3.4.2 to see that $C_L(D,3Q) * C_L(D,3Q) = C_L(D,6Q)$. In example 2.4.9, we already computed this code, using the basis $\{1,f,g,f^2,fg,g^2\}$ for $L(6Q)$. Of course, these two ways directly give the same result, because $\text{ev}_{\mathcal{P}}(h_i) * \text{ev}_{\mathcal{P}}(h_j) = \text{ev}_{\mathcal{P}}(h_i h_j)$.
In example 1.3.3, we already saw that $L(2Q) * L(2Q) \neq L(4Q)$. It follows that also $G_L(D,2Q) * G_L(D,2Q) \neq G_L(D,4Q)$. Therefore, the conditions on the degrees of E and F in proposition 3.4.2 are really necessary.

Chapter 4

Decoding

As explained in chapter 2, the idea behind error correcting codes is that data can be stored in such a way that a certain number of errors in the data can be found and corrected. If $\mathbf{c}$ is an original code word and $\mathbf{e}$ is an error vector (containing a relatively small number of nonzero coordinates) then the process of retrieving $\mathbf{c}$ given $\mathbf{r} = \mathbf{c} + \mathbf{e}$ is what we call decoding.

4.1 General decoding

Let C be an $[n, k, d]_q$ code and let t be a positive integer such that $t < \frac{1}{2}d$. Let $\mathbf{c} \in C$ be a code word, let $\mathbf{e} \in \mathbb{F}_q^n$ be a random (error) vector with $wt(\mathbf{e}) \leq t$ and let $\mathbf{r} = \mathbf{c} + \mathbf{e}$. We know that there is only one possible code word $\mathbf{c} \in C$ that could have turned into $\mathbf{r}$ by changing at most t coordinates: suppose that $\mathbf{r} = \mathbf{c}_1 + \mathbf{e}_1 = \mathbf{c}_2 + \mathbf{e}_2$ with $\mathbf{c}_1,\ \mathbf{c}_2 \in C$ and $wt(\mathbf{e}_1),\ wt(\mathbf{e}_2) \leq t$. Then $d(\mathbf{c}_1, \mathbf{c}_2) \leq d(\mathbf{c}_1, \mathbf{r}) + d(\mathbf{r}, \mathbf{c}_2) = wt(\mathbf{e}_1) + wt(\mathbf{e}_2) \leq t + t < d$. Because the minimum distance of C is at least d, this implies that indeed $\mathbf{c}_1 = \mathbf{c}_2$.

The argument above shows that if $wt(\mathbf{e}) < \frac{1}{2}d$, there exists a unique 'closest code word' $\mathbf{c}$. When $wt(\mathbf{e}) \geq \frac{1}{2}d$, a closest code word may not exist or decoding may return the 'wrong' code word. This case is not interesting for our purposes. On the other hand, a very interesting question for us is the following: when we know that a closest code word exists, can we also find it in practice? This will prove to be the key question for the McEliece cryptosystem that we discuss later. The answer is: for some codes (for example Goppa codes), we know efficient algorithms to do this. For a random linear code however, we cannot do it efficiently.

The general decoding problem comes down to the following. Suppose you have a random linear code, a code word $\mathbf{c}$ such that $\mathbf{r} = \mathbf{c} + \mathbf{e}$ and $\mathrm{wt}(\mathbf{e}) \leq w$ for some number w. If H is the parity check matrix of the code, then $H\mathbf{r}^T = \mathbf{s} \neq 0$.

Because

$$H\mathbf{r}^T = H(\mathbf{c}+\mathbf{e})^T = H\mathbf{c}^T + H\mathbf{e}^T = H\mathbf{e}^T,$$

finding the error $\mathbf{e}$ means finding a word of weight $\leq w$ such that $H\mathbf{e}^T = \mathbf{s}$. In [1], it is shown that if you have a random triple $(H, \mathbf{s}, w)$ where H is an $m \times n$ matrix over $\mathbb{F}_2$, $\mathbf{s} \in \mathbb{F}_2^m$ and $w \in \mathbb{N}$, deciding whether there is a $\mathbf{x} \in \mathbb{F}_2^n$ such that $H\mathbf{x}^T = (s)$ and $\mathbf{x}$ with weight $\leq w$, is an NP-complete problem.

4.2 Patterson's algorithm

For some classes of codes, there are algorithms that can decode 'efficiently'. In 1975, Patterson published a decoding algorithm for Goppa codes [16]. This algorithm can decode t errors, where t is the designed minimal distance of the Goppa code, in polynomial time. We will have a look at this algorithm for decoding Goppa codes.

Let $K = \mathbb{F}_{q^m}$ and $J = \mathbb{F}_q$. Let $g(x)$ be the Goppa polynomial with respect to $L \subset K$. Let C be a codeword and R the received word, so that $R = C + E$.

Define the syndrome $S(x)$ as the polynomial of degree $< n$ such that

$$S(x) = \sum_{j=1}^{n} \frac{R_j}{x - a_j} \mod g(x)$$

and define

$$\sigma(x) = \prod_{a_j \in L \ , \ E_j \neq 0} (x - a_j)$$

and define $\eta(x)$ of degree $< n$ by

$$\eta(x) \equiv \sigma(x)S(x) \mod g(x).$$

If you substitute the expressions for $S(x)$ and $\sigma(x)$ you find that

$$\eta(a_j) = E_j \prod_{a_i \in L \ , \ E_i \neq 0 \ , \ i \neq j} (a_j - a_i) = E_j \sigma'(a_j)$$

So if we can find σ and η then we know the errors E_{a_j}. Patterson's algorithm makes use of Berlekamp-Massey to solve this equation. See [16] for more details.

4.3 Error-correcting pairs

In the year 1992, Pellikaan [17] and Kötter [13] independently introcuced a decoding algorithm using 'error-correcting pairs'. In order to discuss this decoding algorithm, we first define the error correcting pair itself:

Definition 4.3.1. *Let C be an $\mathbb{F}_q$-linear code of length n. Let A, B be subspaces of $\mathbb{F}_{q^m}^n$. Then (A, B) is called a t-error-correcting pair for C if the following conditions are satisfied:*

1. $(A * B) \perp C$,
2. $\dim(A) > t$,
3. $d(B^\perp) > t$,
4. $d(A) + d(C) > n$.

Example 4.3.2. Let C be the $[7, 3, 5]$ code over $\mathbb{F}_8$ (with α such that $\alpha^3 = \alpha + 1$) with generator matrix

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{pmatrix},$$

and parity check matrix

$$H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \\ 1 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha & \alpha^4 \\ 1 & \alpha^4 & \alpha & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{pmatrix}.$$

let $A = C$ and let B be generated by

$$G_B = \begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{pmatrix}.$$

Let the rows of G be $\mathbf{a}_1, \mathbf{a}_2$ and $\mathbf{a}_3$, let the rows of G_B be $\mathbf{b}_1$ and $\mathbf{b}_2$ and let the rows of H be $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ and $\mathbf{h}_4$. We will now verify that (A, B) is a 2-error-correcting pair for C.

1. First of al, note that $A * B$ is generated by
   - $\mathbf{a}_1 * \mathbf{b}_1 = \mathbf{h}_1$
   - $\mathbf{a}_1 * \mathbf{b}_2 = \mathbf{h}_2$
   - $\mathbf{a}_2 * \mathbf{b}_1 = \mathbf{h}_2$
   - $\mathbf{a}_2 * \mathbf{b}_2 = \mathbf{h}_3$
   - $\mathbf{a}_3 * \mathbf{b}_1 = \mathbf{h}_3$
   - $\mathbf{a}_3 * \mathbf{b}_2 = \mathbf{h}_4$

   This means that $A * B = C^\perp$ so indeed clearly $(A * B) \perp C$.
2. $\dim(A) = 3 > 2 = t$,

3. The matrix G_B is a parity check matrix for $B^\perp$. If it would be the case that $d(B^\perp) \leq t$, there would be a word $\mathbf{b} \in B^\perp$ satisfying $\text{wt}(\mathbf{b}) \leq 2$ and $G_B\mathbf{b} = \mathbf{0}$. This means that one of the columns of G_B is a multiple of one of the other columns, which is not true. Therefore, $d(B^\perp) > t$.

4. $d(A) + d(C) = 5 + 5 > 7 = n$.

Remark 4.3.3. The codes from the previous example are all from a special class of codes called Reed-Solomon codes. These codes are not discussed in this thesis but information can be found in [19]. In fact, we had $C = A = RS_3(7, 1)$ and $B = RS_2(7, 0)$. More generally, (see proposition 6.2.9 from [19]), it is true that $RS_{t+1}(n, 1)$ and $RS_t(n, n - b + 1)$ form a t-error-correcting pair for the code $RS_{n-2t}(n, b)$.

The following proposition from Pellikaan [4] will turn out to be essential in the attack in chapter 6.

Proposition 4.3.4. *Let $C_L(D, G)$ be an AG code as defined in definition 2.4.1 such that $3g - 1 < \deg(E)$ (remember that also $2g - 2 < \deg(E) < n$) and consider the dual code $C = C_L(D, G)^\perp$. The designed minimal distance of $C_L(D, G)^\perp$ is $d^* = \deg(G) + 2 - 2g$. Let $t := \left\lfloor \frac{d^*-1-g}{2} \right\rfloor$ and let F be a divisor on $\mathcal{X}$ with disjoint support from D such that $\deg(F) = t + g$. The pair defined by*

$$A = C_L(D, F) \quad \textit{and} \quad B = C_L(D, G - F)$$

is a t-error correcting pair for C.

Proof. 1. Take $\mathbf{z} \in A * B$, that is, $\mathbf{z} = \mathbf{a} * \mathbf{b}$ for some $\mathbf{a} \in A$ and $\mathbf{b} \in B$. Then there exist $f \in L(F)$ and $g \in L(G - F)$ such that

$$\mathbf{z} = (f(P_1), \ldots, f(P_n)) * (g(P_1), \ldots, g(P_n)) = (fg(P_1), \ldots, fg(P_n)).$$

Because $(f) + F \geq 0$ and $(g) + G - F \geq 0$, it follows that $(fg) + G \geq 0$, which means that $fg \in L(G)$. This implies that $\mathbf{z} \in C_L(D, G) = C^\perp$. We see that $A * B \subseteq C^\perp$ so it is indeed true that $(A * B) \perp C$.

2. By proposition 2.4.3, $\dim(A) = \deg(F) - g - 1 = t + g - g + 1 = t + 1 > t$.

3. Because

$$t = \left\lfloor \frac{d^* - 1 - g}{2} \right\rfloor = \left\lfloor \frac{\deg(G) + 2 - 2g - 1 - g}{2} \right\rfloor = \left\lfloor \frac{\deg(G) - 3g + 1}{2} \right\rfloor,$$

either $2t = \deg(G) - 3g$ or $2t = \deg(G) - 3g + 1$. Therefore, either

$\deg(G) = 2t + 3g$ or $\deg(G) = 2t + 3g - 1$. By Proposition 2.4.6,

$$\begin{aligned} d(B^\perp) &\geq \deg(G - F) + 2 - 2g \\ &= \deg(G) - \deg(F) + 2 - 2g \\ &\geq (2t + 3g - 1) - \deg(F) + 2 - 2g \\ &= 2t + 3g - 1 - (t + g) + 2 - 2g \\ &= t + 1 \\ &> t \end{aligned}$$

4. First of all,
$$d(A) \geq n - \deg(F) = n - t - g.$$
Secondly,
$$d(C) \geq \deg(G) + 2 - 2g \geq (2t + 3g - 1) + 2 - 2g = 2t + g + 1.$$
It follows that
$$d(A) + d(C) \geq n - t - g + 2t + g + 1 = n + t + 1 > n.$$

Since all four properties are satisfied, $(A,\ B)$ is indeed a t-error correcting pair for C. □

Now, we will introduce some new concepts/objects to see why the given properties of an error-correcting pair could be useful in a decoding algorithm. These definitions, lemmas and the proposition can be found in [19] (paragraph 6.2).

Definition 4.3.5. *Let A, B be linear subspaces of $\mathbb{F}_{q^m}^n$. Let $\boldsymbol{r} \in \mathbb{F}_q^n$. Define the kernel of $\boldsymbol{r}$ by*
$$K(\boldsymbol{r}) = \{\boldsymbol{a} \in A \mid (\boldsymbol{a} * \boldsymbol{b}) \cdot \boldsymbol{r} = 0 \ \forall \boldsymbol{b} \in B\}.$$

Lemma 4.3.6. *Let C be an $\mathbb{F}_q$-linear code of length n. Let $\boldsymbol{r}$ be a received word with error vector $\boldsymbol{e}$. If $(A * B) \perp C$, then $K(\boldsymbol{r}) = K(\boldsymbol{e})$.*

Proof. Because $\mathbf{r}$ is a received word with error vector $\mathbf{e}$, we can write $\mathbf{r} = \mathbf{c} + \mathbf{e}$, where $\mathbf{c}$ is a code word in the code C. Let $\mathbf{a} \in A$ and $\mathbf{b} \in B$. Because $\mathbf{a} * \mathbf{b} \in A * B$, $\mathbf{c} \in C$ and $(A * B) \perp C$, it follows that $(\mathbf{a} * \mathbf{b}) \cdot \mathbf{c} = 0$. We see that $(\mathbf{a} * \mathbf{b}) \cdot \mathbf{r} = (\mathbf{a} * \mathbf{b}) \cdot (\mathbf{c} + \mathbf{e}) = (\mathbf{a} * \mathbf{b}) \cdot \mathbf{c} + (\mathbf{a} * \mathbf{b}) \cdot \mathbf{e} = 0 + (\mathbf{a} * \mathbf{b}) \cdot \mathbf{e} = (\mathbf{a} * \mathbf{b}) \cdot \mathbf{e}$. The conclusion follows directly from the definition of the kernel: $K(\mathbf{r}) = \{\mathbf{a} \in A \mid (\mathbf{a} * \mathbf{b}) \cdot \mathbf{r} = 0 \ \forall \mathbf{b} \in B\} = \{\mathbf{a} \in A \mid (\mathbf{a} * \mathbf{b}) \cdot \mathbf{e} = 0 \ \forall \mathbf{b} \in B\} = K(\mathbf{e})$. □

Definition 4.3.7. *Let J be a subset of $\{1, \ldots, n\}$. Then*
$$A(J) := \{\boldsymbol{a} \in A \mid a_j = 0 \ \forall j \in J\}.$$

Lemma 4.3.8. *Let $(A * B) \perp C$. Let $\boldsymbol{e}$ be the error vector of the received word $\boldsymbol{r}$. If $I = \operatorname{supp}(\boldsymbol{e}) = \{i \mid e_i \neq 0\}$, then $A(I) \subseteq K(\boldsymbol{r})$. If moreover $d(B^\perp) > \operatorname{wt}(\boldsymbol{e})$, then $A(I) = K(\boldsymbol{r})$.*

Proof. ($\subseteq$) Let $\mathbf{a} \in A(I)$. Then $\mathbf{a} \in A$ and $a_i = 0$ for all $i \in I$. Because $I = \{i \mid e_i \neq 0\}$, we know that $e_i = 0$ for all i such that $i \notin I$. Now we see that for all $b \in B$, we have

$$(\mathbf{a} * \mathbf{b}) \cdot \mathbf{e} = \sum_{i=1}^{n} a_i b_i e_i = \sum_{i \in I} a_i b_i e_i + \sum_{i \notin I} a_i b_i e_i = \sum_{i \in I} 0 b_i e_i + \sum_{i \notin I} a_i b_i 0 = 0,$$

so $\mathbf{a} \in K(\mathbf{e})$. By Lemma 4.3.6 we know that $K(\mathbf{r}) = K(\mathbf{e})$ if $(A * B) \perp C$, so indeed $\mathbf{a} \in K(\mathbf{r})$ which is what we wanted to show.
($\supseteq$) Suppose $d(B^\perp) > \mathrm{wt}(\mathbf{e})$ and let $\mathbf{a} \in K(\mathbf{r})$. Then by Lemma 4.3.6, $\mathbf{a} \in K(\mathbf{e})$, so $(\mathbf{a} * \mathbf{b}) \cdot \mathbf{e} = 0$ for all $\mathbf{b} \in B$. Furthermore, $(\mathbf{a} * \mathbf{b}) \cdot \mathbf{e} = \mathbf{e} \cdot (\mathbf{a} * \mathbf{b}) = (\mathbf{e} * \mathbf{a}) \cdot \mathbf{b}$, which must then also equal 0 for all $b \in B$. In other words, $\mathbf{e} * \mathbf{a} \in B^\perp$. Since $\mathrm{wt}(\mathbf{e} * \mathbf{a}) < \mathrm{wt}(\mathbf{e}) < d(B^\perp)$, it follows that $\mathbf{e} * \mathbf{a} = 0$ which means that $a_i e_i = 0$ for all $i \in \{1, \ldots, n\}$. Therefore, a_i must be 0 for every i such that $e_i \neq 0$, that is, $a_i = 0$ for all $i \in I$. In other words: $\mathbf{a} \in A(I)$. □

Proposition 4.3.9. *Let C be an $\mathbb{F}_q$-linear code of length n. Let A, B be subspaces of $\mathbb{F}_{q^m}^n$ such that (A, B) is a t-error-correcting pair for C. Then there exists an algorithm correcting t errors for the code C with complexity $\mathcal{O}((mn)^3)$.*

Proof. Let $\mathbf{a}_1, \ldots, \mathbf{a}_l$ be a basis of A and let $\mathbf{b}_1, \ldots, \mathbf{b}_m$ be a basis of B. Let $\mathbf{r} \in \mathbb{F}_q^n$ be a received word with $\leq t$ errors, that is, $\mathbf{r} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c}$ is a code word and $\mathbf{e}$ has weight $\leq t$. The algorithm goes as follows:

- Compute $s_{ij} := (\mathbf{b}_i * \mathbf{a}_j) \cdot \mathbf{r}$ for $1 \leq i \leq m$, $1 \leq j \leq l$.
- Compute $K(\mathbf{r})$, i.e. compute the right null space of the matrix $S_r = (s_{ij})$.

  In order to see that we do indeed need the right null space of (s_{ij}), we note that $\mathbf{a} \in A$ can be written as $\mathbf{a} = a_1\mathbf{a}_1 + \ldots + a_l\mathbf{a}_l$. It follows that

$$\begin{aligned}
K(\mathbf{r}) &= \{\mathbf{a} \in A \mid (\mathbf{a} * \mathbf{b}) \cdot \mathbf{r} = 0 \ \forall \mathbf{b} \in B\} \\
&= \{\mathbf{a} \in A \mid (\mathbf{a} * \mathbf{b}_i) \cdot \mathbf{r} = 0 \ \forall i \in \{1, \ldots, m\}\} \\
&= \{\mathbf{a} \in A \mid ((a_1\mathbf{a}_1 + \ldots + a_l\mathbf{a}_l) * \mathbf{b}_i) \cdot \mathbf{r} = 0 \ \forall i \in \{1, \ldots, m\}\} \\
&= \{\mathbf{a} \in A \mid a_1(\mathbf{a}_1 * \mathbf{b}_i) \cdot \mathbf{r} + \ldots + a_l(\mathbf{a}_l * \mathbf{b}_i) \cdot \mathbf{r} = 0 \ \forall i \in \{1, \ldots, m\}\} \\
&= \left\{ \mathbf{a} \in A \mid \begin{pmatrix} a_1(\mathbf{a}_1 * \mathbf{b}_1) \cdot \mathbf{r} + \ldots + a_l(\mathbf{a}_l * \mathbf{b}_1) \cdot \mathbf{r} \\ \vdots \\ a_1(\mathbf{a}_1 * \mathbf{b}_m) \cdot \mathbf{r} + \ldots + a_l(\mathbf{a}_l * \mathbf{b}_m) \cdot \mathbf{r} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \right\} \\
&= \left\{ \mathbf{a} \in A \mid \begin{pmatrix} (\mathbf{a}_1 * \mathbf{b}_1) \cdot \mathbf{r} & \ldots & (\mathbf{a}_l * \mathbf{b}_1) \cdot \mathbf{r} \\ \vdots & & \vdots \\ (\mathbf{a}_1 * \mathbf{b}_m) \cdot \mathbf{r} & \ldots & (\mathbf{a}_l * \mathbf{b}_m) \cdot \mathbf{r} \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_l \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \right\} \\
&= \{\mathbf{a} \in A \mid (a_1, \ldots, a_l) \text{ is in the right null space of } (s_{ij})\}.
\end{aligned}$$

- Take a nonzero $\mathbf{a}$ in $K(\mathbf{r})$.

Since (A, B) is a t-error-correcting pair for C, all conditions in Lemma 4.3.8 are satisfied, which means that $K(\mathbf{r}) = A(I)$ where $I = \text{supp}(\mathbf{e})$. Therefore, our nonzero $\mathbf{a}$ is an element of $A(I)$, which means that $\mathbf{a} \in A$ and $a_j = 0\ \forall j \in \text{supp}(\mathbf{e})$.

- Compute $J := \{j \mid a_j = 0\}$.

  It follows from the previous step that $\text{supp}(\mathbf{e}) \subseteq J$. Because $\mathbf{a} \in A$, we know that $\text{wt}(\mathbf{a}) \geq d(A)$. It follows that the cardinality of J is $|J| < n - d(A)$. Since (A, B) is an error correcting pair for C, we have $d(A) + d(C) > n$, which implies $|J| < d(C)$.

- Solve $H\mathbf{x}^T = H\mathbf{r}^T$ and $x_j = 0$ for all $j \notin J$. Let the unique solution be $\mathbf{x}_0$ and compute $\mathbf{c} = \mathbf{r} - \mathbf{x}_0$.

  We now explain that the error vector $\mathbf{e}$ must satisfy these conditions. Firstly, $H\mathbf{r}^T = H(\mathbf{c} + \mathbf{e})^T = H\mathbf{c}^T + H\mathbf{e}^T = H\mathbf{e}^T$. Secondly, we have already seen that $\text{supp}(\mathbf{e}) \subseteq J$, so indeed $e_j = 0$ for all $j \notin J$. Therefore, if the solution $\mathbf{x}_0$ we found is unique, then it must be the error vector $\mathbf{e}$ and we can compute the closest code word $\mathbf{c} = \mathbf{r} - \mathbf{e} = \mathbf{r} - \mathbf{x}_0$.
  Now suppose that the solution is not unique: apart from the error vector $\mathbf{e}$, let there be another solution $\mathbf{x}_1$. This means that $H\mathbf{x}_1^T = H\mathbf{e}^T$. It follows that $H(\mathbf{x}_1 - \mathbf{e})^T = 0$, so $\mathbf{x}_1 - \mathbf{e} \in C$. Since $\text{supp}(\mathbf{x}_1) \subseteq J$ and $\text{supp}(\mathbf{e}) \subseteq J$, it is also true that $\text{supp}(\mathbf{x}_1 - \mathbf{e}) \subseteq J$. We have already seen that $|J| < d(C)$, so we end up with the fact that $\text{wt}(\mathbf{x}_1 - \mathbf{e}) \leq |J| < d(C)$ while $\mathbf{x}_1 - \mathbf{e} \in C$. This means that $\mathbf{x}_1 - \mathbf{e} = 0$, so $\mathbf{x}_1 = \mathbf{e}$ and we conclude that the solution is indeed unique.

This proves the existence of the algorithm and even gives the algorithm itself. For the complexity, see [19]. □

Example 4.3.10. Consider the same code as in example 4.3.2. Suppose that $\mathbf{r} = (0, \alpha^3, \alpha^5, \alpha, \alpha^3, \alpha^4, \alpha^2)$ is a received word with at most two errors. Let us use the error-correcting pair to decode, that is, find the closest code word $\mathbf{c} \in C$ such that $\mathbf{r} = \mathbf{c} + \mathbf{e}$ with $\text{wt}(e) \leq 2$.

- We need to compute $s_{ij} := (\mathbf{b}_i * \mathbf{a}_j) \cdot \mathbf{r}$ for $1 \leq i \leq m$, $1 \leq j \leq l$. In example 4.3.2, we have seen that $\mathbf{b}_i * \mathbf{a}_j = \mathbf{h}_{i+j-1}$, so we only need to compute $\mathbf{h}_i \cdot \mathbf{r}$:

$$\begin{aligned} s_{11} &= \mathbf{h}_1 \cdot \mathbf{r} = \alpha^4 \\ s_{12} = s_{21} &= \mathbf{h}_2 \cdot \mathbf{r} = \alpha^2 \\ s_{13} = s_{22} &= \mathbf{h}_3 \cdot \mathbf{r} = 0 \\ s_{23} &= \mathbf{h}_4 \cdot \mathbf{r} = \alpha \end{aligned}$$

- We need to find a nonzero element in the right null space of

$$(s_{ij}) = \begin{pmatrix} \alpha^4 & \alpha^2 & 0 \\ \alpha^2 & 0 & \alpha \end{pmatrix}.$$

  Take $(a_1, a_2, a_3) = (1, \alpha^2, \alpha)$.

- The nonzero element of $K(\mathbf{r})$ corresponding to this is $\mathbf{a} = \mathrm{ev}(1 + \alpha^2 x + \alpha x^2) = (\alpha^5, 1, 0, \alpha^5, 0, \alpha^4, \alpha^4)$.

- $J = \{j \mid a_j = 0\} = \{3, 5\}$, so we know that the errors are located at the positions 3 and 5.

- The equations $H\mathbf{x}^T = H\mathbf{r}^T$ and $x_j = 0$ for all $j \notin \{3, 5\}$ lead to the following system to solve:

$$\begin{pmatrix} \alpha^2 & \alpha^4 \\ \alpha^4 & \alpha \\ \alpha^6 & \alpha^5 \\ \alpha & \alpha^2 \end{pmatrix} \begin{pmatrix} x_3 \\ x_5 \end{pmatrix} = \begin{pmatrix} \alpha^4 \\ \alpha^2 \\ 0 \\ \alpha \end{pmatrix}$$

  The solution to this system is $(x_3, x_5) = (\alpha, \alpha^2)$, so the error vector is

$$\mathbf{e} = \mathbf{x} = (0, 0, \alpha, 0, \alpha^2, 0, 0).$$

- The closest code word $\mathbf{c}$ is

$$\begin{aligned} \mathbf{c} &= \mathbf{r} - \mathbf{e} \\ &= (0, \alpha^3, \alpha^5, \alpha, \alpha^3, \alpha^4, \alpha^2) - (0, 0, \alpha, 0, \alpha^2, 0, 0) \\ &= (0, \alpha^3, \alpha^6, \alpha, \alpha^5, \alpha^4, \alpha^2). \end{aligned}$$

Chapter 5

McEliece cryptosystem

In 1978, Robert McEliece proposed the public-key cryptosystem now known as the McEliece Cryptosystem[14]. He presented it as a cryptosystem which is "ideal for use in multi-user communication networks, such as those envisioned by NASA for the distribution of space-acquired data." In this chapter, this cryptosystem and its security are investigated. We will also consider the pros and cons of the McEliece cryptosystem and look at a possible variation.

5.1 Description

The McEliece cryptosystem works as follows.

Key generation: Bob picks desirable values for $n = 2^l$ and t and chooses a binary irreducible Goppa code $\mathcal{C}$ of length n and dimension $k \geq n - tl$ with an efficient decoding algorithm that can decode up to t errors. We have seen in Proposition 2.3.7 that this code does indeed exist. Let G be the $(k \times n)$ generator matrix for $\mathcal{C}$. Now Bob picks a random dense $k \times k$ nonsingular matrix S and a random $n \times n$ permutation matrix P. He computes $G' = SGP$, which will be called the public generator matrix. Bob presents his public key (G', t).
Encrypting: If Alice wants to send Bob a message, she converts the message into k-bit blocks. If m is such a block, she converts this into a word of the 'public code', that is, she computes mG'. Now she needs to add t errors, that is, a random word e of length n and weight t. She sends the ciphertext $c = mG' + e$ to Bob.
Decrypting: Bob receives the ciphertext c and computes $c' = cP^{-1} = (mG' + e)P^{-1} = mSG + eP^{-1}$. Since P is a permutation matrix, P^{-1} is as well, which means that eP^{-1} has the same weight as e, namely t. Because mSG is a word in Bob's secret Goppa code, Bob can use his efficient decoding algorithm on c' in order to find mS. Then he only needs to compute $m' = (mS)S^{-1} = m$.

5.2 Security

We have seen that the cryptosystem is correct: Alice can convert a secret message into a public message, send this to Bob and then Bob can recover the secret message from the public message. The next question would be whether Eve could also recover the secret message from the information that is available to her. If this would be the case, the system would not be secure.

Before we look at a couple of possible attacks, we investigate the public code a bit more. We know that G is the generator matrix of the secret Goppa code. Because S is an invertible $k \times k$ matrix, the matrix SG consists of k linearly independent rows, all of which are linear combinations of the rows of G. This means that SG is in fact just a different (random) generator matrix of the secret code. Because P is a permutation matrix, the columns of SGP are a permutation of the columns of SG. Therefore, the code words in the public code are exactly the permutations of the code words in the secret code: if $\mathbf{c} \in \mathcal{C}$, then $\mathbf{c}P$ is a word in the public code. This means that the public code is also a Goppa code, generated by the same g as the secret code. The L' of the public code is just a permutation of the L of the secret code.

Now suppose Eve wants to recover the secret message. First of all, she could try to decode directly in the public code (that is, find mG' from $mG' + e$). If she could do this, then she could immediately find m and the system would obviously not be secure. The general idea behind the McEliece system is that the public generator matrix looks like a generator matrix of a random code, so decoding would come down to decoding in a random code, which we know is hard. As McEliece himself already noticed (see [14]), a promising possible attack would be 'information set decoding', which comes down to trying to guess k coordinates which do not contain an error. However, for large enough parameters, this is not considered an effective attack.

Another possible attack would be to try to somehow recover the secret key, that is: find g, L and P. If Eve has access to these, she can use P^{-1} to find mSG, then use g and L in Patterson's algorithm to find mSG. Using P, she can compute $mSGP$ and then, using linear algebra and the public generator matrix, she can find m. A possible attack to do this, is trying different Goppa polynomials g of degree $\geq t$, taking an L' consisting of the elements of the field $\mathbb{F}_q$, create the corresponding Goppa code and then trying to find out whether there exists a permutation P' that transforms this Goppa code into the public code. This problem is known as the permutation code equivalence problem and it is not NP-complete but hard enough to make this attack infeasable. Note that, if Eve would indeed find a Goppa polynomial which gives an equivalent code, she computes the permutation P' and in the same way as described above, find m.

The idea behind this attempted 'attack', is that Eve was not necessarily trying

to find the original P and L from the secret key, but using the fact that the original code is a Goppa code in order to find a decoding algorithm. If she would be able to accomplish this, she could recover the secret message without finding the secret key and without decoding in a random code. She did have to solve some hard problems though, so this attack is not considered efficient enough to break the McEliece system using Goppa codes. However, as we will see in the next chapter, in some way this same tactic will be used when AG codes are used instead of Goppa codes: trying to find an efficient decoding algorithm for the public code, while not finding the secret key itself.

5.3 Pros and cons

A pro of the McEliece cryptosystem is that encoding and decoding is relatively fast [14]. Encoding is only multiplying a vector with a matrix and adding another vector. Decoding is done using an available efficient algorithm: Patterson's algorithm. Since 1978, computers have improved, so the numbers which McEliece originally proposed for n and k are not considered safe anymore. However, with larger numbers, the cryptosystem as McEliece described it is still not broken, not even by possible quantum computers.

On the other side, the cryptosystem as described here is also not very practical, because both the private key and the public key are very large: they are very large matrices. The private code can be stored by only the Goppa polynomial and the list L, which could possibly be smaller than a $k \times n$ matrix. If this is not the case, the secret generator matrix could also be written in systematic form, such that only a $k \times (n-k)$ matrix needs to be stored. The public code however, must be represented as an $k \times n$ matrix, because the structure of the code should remain secret.

5.4 McEliece based on AG codes

In order to solve the problem with the large key size, variations to the original McEliece system have been suggested. For example, people have suggested to use other codes instead of the classical Goppa code, because this results in a key that can be presented in a more compact way. We have not found a conclusive argument for why exactly the use of these codes results in shorter keys, but it could for example be along the lines that some other codes codes over a larger field require smaller parameters in order to reach the same level of security. If a code has a higher error correcting capability than a binary Goppa code, then more errors could be added and a computer would need to do more work in order to decode in a code that could have a shorter length than the binary Goppa code.

However, most of the variations that have been suggested, have already been broken. We will discuss one such suggestion in more detail. In 1966, [12] pro-

posed to use AG codes. In this case, the secret code is an AG code $\mathcal{C} = C_L(D, E)$ of length n together with an efficient decoding algorithm that can decode up to t errors. Apart from this, there are no changes to the original McEliece system.

When you use AG codes instead of Goppa codes, it remains true that the public code is just a permutation of the secret code. The 'security' of the McEliece system therefore relies on the difficulty of decoding an AG code $C_L(D, E)$ without knowing D and E. However, as we will see in the next chapter, decoding an unknown AG code turns out to be easier than decoding a random linear code.

Chapter 6

Variations and attacks

As promised, this chapter discusses an attack to the modified McEliece system using AG codes. The attack was published in [4]. In this attack, Schur products and many other different aspects from coding theory, cryptography and algebraic geometry come together in a beautiful way. Most of the definitions and propositions that we studied in earlier chapters will pop up again.

6.1 Propositions

Apart from the propositions we already looked at, there are several new propositions that will be used in the attack. These will be discussed in this section. All of the propositions and lemmas are taken from [4]. The proofs presented here are either more detailed versions of proofs in [4], or my own work.

Proposition 6.1.1. *If $2g+1 \leq \deg(E) < \frac{n}{2}$ and k_1 and k_2 are the dimensions of $\mathcal{C} = C_L(D,E)$ and $\mathcal{C}^{(2)}$, respectively, then*

$$\deg(E) = k_2 - k_1 \quad \text{and} \quad g = k_2 - 2k_1 + 1.$$

Proof. Because $\deg(E) \geq 2g+1$, if follows from proposition 3.4.2 that $\mathcal{C}^{(2)} = C_L(D,2E)$. From $2g-2 < \deg(E) < \frac{n}{2}$, it follows that $2g-2 < \deg(2E) < n$, so we can use proposition 2.4.3 to find the dimension of both $\mathcal{C}$ and $\mathcal{C}^{(2)}$:

$$\begin{aligned} k_1 &= \deg(E) - g + 1 \\ k_2 &= \deg(2E) - g + 1 = 2\deg(E) - g + 1. \end{aligned}$$

From this, we can conclude that

$$\begin{aligned} k_2 - k_1 &= (2\deg(E) - g + 1) - (\deg(E) - g + 1) = \deg(E), \\ k_2 - 2k_1 + 1 &= (2\deg(E) - g + 1) - 2(\deg(E) - g + 1) + 1 = g, \end{aligned}$$

which proves both statements of the proposition. □

Lemma 6.1.2. *Let A and B be two codes in $\mathbb{F}_q^n$. Then*

$$\{\mathbf{z} \in \mathbb{F}_q^n \mid \mathbf{z} * A \subseteq B\} = \left(A * B^\perp\right)^\perp.$$

Proof. Let $\mathbf{z} \in \mathbb{F}_q^n$. Then

$$\begin{aligned}
\mathbf{z} * A \subseteq B &\iff \mathbf{z} * \mathbf{a} \in B \quad \forall \mathbf{a} \in A \\
&\iff (\mathbf{z} * \mathbf{a}) \cdot \mathbf{c} = 0 \quad \forall \mathbf{a} \in A \text{ and } \forall \mathbf{c} \in B^\perp \\
&\iff \mathbf{z} \cdot (\mathbf{a} * \mathbf{c}) = 0 \quad \forall \mathbf{a} \in A \text{ and } \forall \mathbf{c} \in B^\perp \\
&\iff \mathbf{z} \cdot \mathbf{x} = 0 \quad \forall \mathbf{x} \in A * B^\perp \\
&\iff \mathbf{z} \in \left(A * B^\perp\right)^\perp.
\end{aligned}$$

□

Lemma 6.1.3. *Let E, F be two divisors on the curve $\mathcal{X}$, both with disjoint support with D, such that* $\deg(F) \geq 2g$ *and* $\deg(E) \leq n-3$. *Then,*

$$\begin{aligned}
C_L(D, E-F) &= \left(C_L(D,F) * C_L(D,E)^\perp\right)^\perp \\
&= \left\{\boldsymbol{z} \in \mathbb{F}_q^n \mid \boldsymbol{z} * C_L(D,F) \subseteq C_L(D,E)\right\}
\end{aligned}$$

Proof. In order to prove the first equality, let W be the divisor of the differential form ω with simple poles and residue 1 at the P_i. By Proposition 2.4.10, $C_L(D,E)^\perp = C_L(D, W+D-E)$. Because W is a canonical divisor, its degree is $2g-2$, so

$$\deg(W+D-E) = (2g-2) + n - \deg(E) \geq 2g - 2 + n - (n-3) = 2g+1,$$

which means we can use Proposition 3.4.2 to deduce that

$$\begin{aligned}
C_L(D,F) * C_L(D,E)^\perp &= C_L(D,F) * C_L(D, W+D-E) \\
&= C_L(D, W+D-(E-F)) \\
&= C_L(D, E-F)^\perp
\end{aligned}$$

By taking duals on both sides, we see that indeed

$$\left(C_L(D,F) * C_L(D,E)^\perp\right)^\perp = C_L(D, E-F).$$

The second equality follows directly from Lemma 6.1.2. □

Lemma 6.1.4. *Under the conditions of Proposition 4.3.4 and assuming that* $\deg(E) \leq n-3$ *and* $t \geq 1$, *it is true that* $A = (B * C)^\perp$, *that is:*

$$C_L(D,F) = \left(C_L(D, E-F) * C_L(D,E)^\perp\right)^\perp.$$

Proof. If $\deg(E-F) \geq 2g$, this follows directly from Lemma 6.1.3. Remember that $d^* = \deg(E) + 2 - 2g$ and $t = \lfloor (d^* - 1 - g)/2 \rfloor$, so that $\deg(E) = d^* - 2 + 2g$

and $2t \leq d^* - 1 - g$. It follows that indeed

$$\begin{aligned}\deg(E-F) &= \deg(E) - \deg(F)\\ &= (d^* - 2 + 2g) - (t+g)\\ &= d^* - 2 + 3g - 2t + t\\ &\geq d^* - 2 + 3g - (d^* - 1 - g) + t\\ &= 2g + t - 1\\ &\geq 2g,\end{aligned}$$

which completes the proof. □

Definition 6.1.5. *Given a point P in the support of D, let $D' = D - P$ and define the following codes of length $n-1$ for all $i \in \mathbb{Z}$:*

$$U_i := C_L(D', iP), \qquad V_i := C_L(D', E + iP).$$

Proposition 6.1.6. *If $i \geq 1$ and $\deg(E) - \frac{n-4}{2} \leq i \leq \deg(E) - 2g - 1$, then*[†]

$$V_{-i-1} = \left\{ \boldsymbol{z} \in V_{-i} \mid \boldsymbol{z} * V_{-i+1} \subseteq V_{-i}^{(2)} \right\}.$$

Proof. Suppose $i \geq 1$ and $\deg(E) - \frac{n-4}{2} \leq i \leq \deg(E) - 2g - 1$. First of all, we see that the following (in)equalities hold:

$$\begin{aligned}\deg(E + (-i+1)P) &= \deg(E) - i + 1 \geq 2g + 2 > 2g,\\ \deg(2E - 2iP) &= 2\deg(E) - 2i \leq n - 4 = (n-1) - 3\\ \deg(E - iP) &= \deg(E) - i \geq 2g + 1\end{aligned}$$

The first two of these are necessary to apply Lemma 6.1.3, the third one is necessary to apply Proposition 3.4.2.

$$\begin{aligned}V_{-i-1} &= C_L(D', E - (i+1)P)\\ &= C_L(D', 2E - E - 2iP + iP - P)\\ &= C_L\big(D', (2E - 2iP) - (E + (-i+1)P)\big)\\ &= \left\{ \mathbf{z} \in \mathbb{F}_q^{n-1} \mid \mathbf{z} * C_L(D', E + (-i+1)P) \subseteq C_L(D', 2E - 2iP) \right\} \quad \text{by 6.1.3}\\ &= \left\{ \mathbf{z} \in \mathbb{F}_q^{n-1} \mid \mathbf{z} * V_{-i+1} \subseteq C_L(D', 2E - 2iP) \right\}\\ &= \left\{ \mathbf{z} \in \mathbb{F}_q^{n-1} \mid \mathbf{z} * V_{-i+1} \subseteq \big(C_L(D', E - iP)\big)^{(2)} \right\} \quad \text{by 3.4.2}\\ &= \left\{ \mathbf{z} \in \mathbb{F}_q^{n-1} \mid \mathbf{z} * V_{-i+1} \subseteq V_{-i}^{(2)} \right\}\\ &= \left\{ \mathbf{z} \in V_{-i} \mid \mathbf{z} * V_{-i+1} \subseteq V_{-i}^{(2)} \right\} \quad \text{because } V_{-i-1} \subseteq V_{-i}.\end{aligned}$$

This completes the proof. □

[†]In [4], the above proposition has $\deg(E) - 2g + 1$ as the upper bound, but we believe that $\deg(E) - 2g - 1$ is necessary.

6.2 The attack

Now we will study the actual attack. We know that our public generator matrix is the generator matrix of an AG code. We also know from corollary 2.4.12 that every AG code is the dual of another AG code. Therefore, we will assume that our public generator matrix G' is the generator matrix of the AG code $C_L(D,E)^\perp$ for some D and E.

Let d^* be the designed minimal distance of $C_L(D,E)^\perp$, that is, $d^* = \deg(E) - 2g + 2$. We assume that $t \leq \lfloor (d^* - g - 1)/2 \rfloor$. This is a reasonable assumption if we assume that Bob uses a decoding algorithm based on error correcting pairs, in order to decode in his secret code. Of course, we also assume that $t \geq 1$. These assumptions on t give a lower bound for $\deg(E)$: Since $t \leq \lfloor (\deg(E) - 2g + 2 - g - 1)/2 \rfloor$, we know $2t \leq \deg(E) - 3g + 1$, so $\deg(E) - 3g - t + 1 \geq t \geq 1$, which means that $\deg(E) \geq 3g + t$.

The idea of the attack is to puncture the public code at one position. This comes down to removing one column from the generator matrix and this gives the code $C_L(D',E)^\perp$. Then the code $V_{-t-g} = C_L(D', E-(t+g)P)$ is computed. Using Lemma 6.1.4, $U_{-t-g} = C_L(D,(t+g)P)$ can be determined and according to 4.3.4, these two codes make up an error correcting pair for $C_L(D',E)^\perp$. Using this pair, one can decode in the punctured public code. After this, the only thing left to do is correcting a single erasure: find the missing coordinate of the word in the public code.

Assume that $3g + t \leq \deg(E) \leq \frac{n}{2} - 2$.[†] Let $\mathbf{r} = \mathbf{c} + \mathbf{e}$ be the received word.

- Given the generator matrix of $C_L(D,E)^\perp$, compute its parity check matrix, that is, the generator matrix of $C_L(D,E)$. This can be done by writing the generator matrix of $C_L(D,E)^\perp$ in reduced row echelon form. Perform a column permutation to get the form $G' = (I|P)$, compute $H' = (-P^T|I)$ and perform the inverse column permutation. It can easily be seen that this indeed gives the parity check matrix H of $C_L(D,E)^\perp$.
- Compute the generator matrix of the Schur square of $C_L(D,E)$, deduce the dimension of the Schur square and find g and $\deg(E)$ using proposition 6.1.1.
- Choose one coordinate j corresponding to (an unknown) $P := P_j$ in the support of D and define $D' := D - P$.
- Compute $V_0 = C_L(D',E)$. This is done by *puncturing* $C_L(D,E)$ at position j: just remove the j-th column from the generator matrix of $C_L(D,E)$ and obtain a generator matrix for V_0.

[†]In [4], the lower bound is $3g + t - 2$, but we believe that $3g + t \leq \deg(E)$ is necessary in order to use proposition 6.1.6 to compute V_{-t-g}. As we explained, this is a reasonable lower bound.

- Compute $V_{-1} = C_L(D', E - P)$. This code is obtained by taking all words from $C_L(D, E)$ which have a 0 as their j-th coordinate, and then removing the j-th coordinate from these words. In coding theory, this is called *shortening* $C_L(D, E)$ at position j: write the generator matrix of $C_L(D, E)$ in such a way that there is only one nonzero entry in the j-th column, in, say, row k. Then remove row k and column j to obtain a generator matrix for V_{-1}.

- Using Proposition 6.1.6, compute $V_{-2}, \ldots, V_{-t-g}$.

  We can indeed use proposition 6.1.6 for $1 \leq i \leq t + g - 1$, because $3g + t \leq \deg(E) \leq \frac{n-4}{2}$:

  $$i \geq 1 \geq 0 \geq \deg(E) - \frac{n-4}{2}$$
  $$i \leq t + g - 1 = (t + 3g) - 2g - 1 \leq \deg(E) - 2g - 1$$

- Using Lemma 6.1.4, compute $U_{-t-g} = C_L(D, (t + g)P)$. According to Proposition 4.3.4, $A = U_{-t-g}$ and $B = V_{-t-g}$ make up an error correcting pair for $C_L(D', E)^\perp$.

- Remove the j-th coordinate from $\mathbf{r}$ and let this word be denoted by $\mathbf{r}'$. Using the error correcting pair and the decoding algorithm described in 4.3.9, find the codeword $\mathbf{c}' \in C_L(D', E)^\perp$ such that $\mathbf{r}' = \mathbf{c}' + \mathbf{e}'$ where $\mathrm{wt}(\mathbf{e}') \leq t$. Now all coordinates of $\mathbf{c}$ are known except for the j-th.

- Find the j-th coordinate of $\mathbf{c}$ by correcting an erasure. This can be done using the parity check matrix H of $C_L(D, E)^\perp$: solve $H\mathbf{c}^T = 0$ for the missing coordinate c_j.

- Using linear algebra, solve $\mathbf{m}G = \mathbf{c}$ and find the secret message $\mathbf{m}$.

6.3 Example

In this section we will discuss a small example. It is too small to be of any practical purpose, but it will illustrate the attack nicely. The MAGMA code used for the calculations can be found in Appendix B.3.

Setting

The secret curve is the curve of genus 1 defined by $X^3 + Y^3 + Z^3 = 0$ over $\mathbb{F}_{17}$. There are 18 rational points on this curve. Take $P_1, \ldots, P_{17}$ to be the following 17 points: $(16 : 1 : 0), (16 : 0 : 1), (5 : 3 : 1), (1 : 9 : 1), (8 : 10 : 1), (6 : 13 : 1), (3 : 5 : 1), (14 : 15 : 1), (12 : 11 : 1), (0 : 16 : 1), (15 : 14 : 1), (10 : 8 : 1), (4 : 7 : 1), (7 : 4 : 1), (11 : 12 : 1), (2 : 2 : 1), (13 : 6 : 1)$ and let $D = P_1 + \ldots + P_{17}$. Let Q be

the remaining point, $(9:1:1)$, and define the divisor E to be $6Q$. The private code is the $[17, 11, 6]$ code $(C_L(D, E))^{\perp}$ with generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 15 & 16 & 0 & 6 & 12 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & 11 & 10 & 4 & 1 & 12 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & 10 & 16 & 16 & 13 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 0 & 7 & 12 & 11 & 7 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 14 & 10 & 5 & 12 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 13 & 2 & 10 & 2 & 15 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 6 & 11 & 5 & 7 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 12 & 12 & 14 & 3 & 13 & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 11 & 4 & 2 & 12 & 14 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 12 & 11 & 12 & 11 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 9 & 7 & 3 & 3 & 16 & 12 \end{pmatrix}.$$

The designed minimal distance of this code is $d^* = \deg(E) - 2g + 2 = 6$ and t is chosen to be 2.
The public key is a 11×17 matrix which is obtained by first permuting the columns of G and then multiplying from the left by an invertible 11×11 matrix. For practical purposes, we will assume that the secret key actually contained a different order of the points P_i, and that the order presented above is in fact already the permuted order. In that case, G is indeed the reduced Row Echelon form of the public generator matrix and we can perform the attack on G directly.

Attack by Eve

Eve has access to the matrix G and the number t. First of all, she computes the generator matrix of $C_L(D, E)$, which is the parity check matrix of the public code. Because G is already in the form $G = (I|P)$, this can be done by computing $H = (-P^T|I)$. This gives (in reduced form) the following:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 16 & 15 & 8 & 12 & 9 & 2 & 6 & 11 & 3 & 9 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 12 & 6 & 15 & 9 & 5 & 5 & 2 & 0 & 1 & 7 & 12 \\ 0 & 0 & 1 & 0 & 0 & 0 & 13 & 14 & 9 & 10 & 8 & 3 & 7 & 5 & 8 & 15 & 8 \\ 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 14 & 4 & 8 & 2 & 1 & 13 & 14 & 6 & 8 \\ 0 & 0 & 0 & 0 & 1 & 0 & 13 & 9 & 12 & 2 & 7 & 5 & 0 & 8 & 7 & 7 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 12 & 8 & 11 & 15 & 15 & 1 & 2 & 15 & 2 & 8 & 6 \end{pmatrix}.$$

Using this matrix, Eve can compute the degree of E and the genus of the curve. In order to do this, Eve first uses the generator matrix of $C_L(D, E)$ to compute

the Schur square of $C_L(D, E)$. The generator matrix of the Schur square is

$$H_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 9 & 15 & 9 & 16 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 6 & 10 & 7 & 2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 10 & 9 & 16 & 9 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 1 & 6 & 10 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 0 & 0 & 9 & 7 & 16 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 6 & 0 & 3 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 12 & 0 & 5 & 11 & 3 & 11 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 5 & 0 & 9 & 11 & 1 & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 8 & 0 & 16 & 9 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 11 & 8 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 12 & 0 & 8 & 13 & 4 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 14 & 7 & 8 & 6 \end{pmatrix}.$$

It can be seen that k_2, the dimension of the Schur square of $C_L(D, E)$, is 12. Because the dimension of $C_L(D, E)$ is $k_1 = 6$, it follows from proposition 6.1.1 that

- $\deg(E) = k_2 - k_1 = 12 - 6 = 6$ and
- $g = k_2 - 2k_1 + 1 = 12 - 12 + 1 = 1$.

This means that Eve needs to find $B = V_{-3}$ and $A = U_{-3}$. The next step is to create V_0, that is, to puncture $C_L(D, E)$ at one position. If Eve chooses to puncture at the first position, this comes down to removing the first column of the generator matrix H. After some row operations, this leads to the following generator matrix of the punctured code V_0:

$$G_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 16 & 9 & 0 & 11 & 12 & 6 & 13 & 3 & 13 & 12 \\ 0 & 1 & 0 & 0 & 0 & 0 & 5 & 11 & 13 & 6 & 12 & 0 & 12 & 13 & 13 & 8 \\ 0 & 0 & 1 & 0 & 0 & 0 & 11 & 4 & 6 & 1 & 8 & 2 & 12 & 6 & 16 & 8 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 14 & 5 & 5 & 14 & 10 & 15 & 12 & 5 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 5 & 6 & 4 & 8 & 6 & 11 & 4 & 14 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 9 & 5 & 8 & 15 & 11 & 6 & 14 & 8 & 0 \end{pmatrix}.$$

In order to determine V_{-1}, $C_L(D, E)$ must be shortened at the same position as before. This means that all words having a zero in the first position are kept and then the first coordinate is removed. Because the first column of H contains only one nonzero entry, which is in the first row, the generator matrix of V_{-1} is obtained by removing the first row and the first column of H:

$$G_{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 12 & 6 & 15 & 9 & 5 & 5 & 2 & 0 & 1 & 7 & 12 \\ 0 & 1 & 0 & 0 & 0 & 13 & 14 & 9 & 10 & 8 & 3 & 7 & 5 & 8 & 15 & 8 \\ 0 & 0 & 1 & 0 & 0 & 3 & 0 & 14 & 4 & 8 & 2 & 1 & 13 & 14 & 6 & 8 \\ 0 & 0 & 0 & 1 & 0 & 13 & 9 & 12 & 2 & 7 & 5 & 0 & 8 & 7 & 7 & 1 \\ 0 & 0 & 0 & 0 & 1 & 12 & 8 & 11 & 15 & 15 & 1 & 2 & 15 & 2 & 8 & 6 \end{pmatrix}.$$

Because $i = 1$ and $i = 2$ satisfy

$$\deg(E) - \frac{n-4}{2} = -\frac{1}{2} \leq i \leq 3 = \deg(E) - 2g - 1,$$

proposition 6.1.6 can be used to find V_{-2} and V_{-3}:

$$V_{-i-1} = \left\{ \mathbf{z} \in V_{-i} \mid \mathbf{z} * V_{-i+1} \subseteq V_{-i}^{(2)} \right\}.$$

Let G_0 be the generator matrix of V_0, let $\mathbf{g}_j$ be the rows of G_0, let H_{-1} be the parity check matrix of V_{-1}, let $\mathbf{h}_j$ be the rows of H_{-1}, let H_{-1}^2 be the parity check matrix of $V_{-1}^{(2)}$ and let $\mathbf{k}_j$ be the rows of H_{-1}^2. Observe the following:

$$\begin{aligned}
V_{-2} &= \left\{ \mathbf{z} \in V_{-1} \mid \mathbf{z} * V_0 \subseteq V_{-1}^{(2)} \right\} \\
&= \left\{ \mathbf{z} \in V_{-1} \mid \mathbf{z} * \mathbf{b} \in V_{-1}^{(2)} \quad \forall \mathbf{b} \in V_0 \right\} \\
&= \left\{ \mathbf{z} \in V_{-1} \mid \mathbf{z} * \mathbf{g}_j \in V_{-1}^{(2)} \quad \forall j \right\} \\
&= \left\{ \mathbf{z} \in V_{-1} \mid (\mathbf{z} * \mathbf{g}_j) \cdot \mathbf{k}_i = 0 \quad \forall j, i \right\} \\
&= \left\{ \mathbf{z} \in V_{-1} \mid \mathbf{z} \cdot (\mathbf{g}_j * \mathbf{k}_i) = 0 \quad \forall j, i \right\} \\
&= \left\{ \mathbf{z} \in \mathbb{F}_{17}^{16} \mid \mathbf{z} \cdot \mathbf{h}_l = \mathbf{z} \cdot (\mathbf{g}_j * \mathbf{k}_i) = 0 \quad \forall j, i, l \right\}.
\end{aligned}$$

Computing this comes down to computing the right null space of the matrix which has as its rows $\mathbf{h}_l$ and $\mathbf{g}_j * \mathbf{k}_i$. This gives

$$G_{-2} = \begin{pmatrix}
1 & 0 & 0 & 2 & 0 & 4 & 7 & 5 & 13 & 2 & 15 & 2 & 16 & 15 & 4 & 14 \\
0 & 1 & 0 & 6 & 0 & 6 & 0 & 13 & 5 & 16 & 16 & 7 & 2 & 16 & 6 & 14 \\
0 & 0 & 1 & 3 & 0 & 8 & 10 & 16 & 10 & 12 & 0 & 1 & 3 & 1 & 10 & 11 \\
0 & 0 & 0 & 0 & 1 & 12 & 8 & 11 & 15 & 15 & 1 & 2 & 15 & 2 & 8 & 6
\end{pmatrix}.$$

In the same way, the generator matrix of $V_{-3} = \left\{ \mathbf{z} \in V_{-2} \mid \mathbf{z} * V_{-1} \subseteq V_{-2}^{(2)} \right\}$ can be computed, which is

$$G_{-3} = \begin{pmatrix}
1 & 0 & 0 & 2 & 14 & 2 & 0 & 6 & 2 & 8 & 12 & 13 & 5 & 9 & 14 & 13 \\
0 & 1 & 0 & 6 & 16 & 11 & 9 & 2 & 7 & 1 & 15 & 5 & 4 & 14 & 15 & 8 \\
0 & 0 & 1 & 3 & 10 & 9 & 5 & 7 & 7 & 9 & 10 & 4 & 0 & 4 & 5 & 3
\end{pmatrix}.$$

This is B of the Error Correcting Pair. In order to find A, Eve uses Lemma 6.1.4 to compute $A = (B * V_0^{\perp})^{\perp}$. This gives $A = U_{-3}$, which is generated by

$$U_{-3} = \begin{pmatrix}
1 & 0 & 0 & 13 & 15 & 3 & 0 & 11 & 13 & 16 & 6 & 10 & 16 & 13 & 16 & 7 \\
0 & 1 & 0 & 13 & 13 & 14 & 5 & 5 & 1 & 12 & 11 & 10 & 2 & 8 & 13 & 1 \\
0 & 0 & 1 & 9 & 7 & 1 & 13 & 2 & 4 & 7 & 1 & 15 & 0 & 14 & 6 & 10
\end{pmatrix}.$$

So Eve has indeed succeeded in finding an error correcting pair for the punctured code $C_L(D', E)^{\perp}$.

Once Eve intercepts a message, $\mathbf{r}$, she removes the first coordinate and uses her error correcting pair to find the closest code word in the punctured code. This gives her every coordinate of the word $\mathbf{c}$ in the public code, except for the first coordinate. Then she uses the parity check matrix H to solve $H\mathbf{c}^T = 0$ for the first coordinate. Now she only needs to solve $\mathbf{m}G = \mathbf{c}$ to find the secret message $\mathbf{m}$.

Check

Because we know the secret key that was used in this system, we can check whether the algorithm indeed gave Eve the codes $V_{-i} = C_L(D', 6Q - iP)$ and $U_{-3} = C_L(D', 3P)$, with $P = P_1 = (16 : 1 : 0)$. This can be checked using MAGMA, see Appendix B.3 for the code: MAGMA can compute the AG codes $C_L(D', 6Q - iP)$ and $C_L(D', 3P)$ and we can see that Eve indeed found the same codes.

6.4 Discussion

The attack as described above, is proven to be work under the assumptions that $3g + t \leq \deg(E) \leq \frac{n}{2} - 2$. As we saw, the lower bound is reasonable, because we assume that $t \leq \lfloor (d^* - g - 1)/2 \rfloor$. The upper bound however, leaves some work to do. When $\deg(E) \geq \frac{n}{2} + 2g$, one can work with the dual code instead of the given code. For values of $\deg(E)$ that lie in between, one can sometimes work with the dual code and/or use shortening tactics. For values of $\deg(E)$ such that $\frac{n}{2} - 2 < \deg(E) < \frac{n}{2} + g$ and $\deg(E) \leq 4g - 2$, the attack is not proved to work, but the authors of [4] think that it is unlikely that there exists a large family of codes such that the attack does not work for both the public code and the dual code, for any choice of shortening and for any point $P \in \mathcal{P}$.

We said that the assumption $t \leq \lfloor (d^* - g - 1)/2 \rfloor$ makes sense if Bob uses a decoding algorithm based on error correcting pairs. However, there are other algorithms that can correct more errors, see for example [18]. Apart from the attack that we described in this chapter, [4] also contains an approach using 'error correcting arrays', which covers the case $t \leq \lfloor (d^* - 1)/2 \rfloor$.

So far, there still has not been a successful attack to the original McEliece System. [4] discusses an attack to subcodes of AG codes, using the 's-closure'. As we saw in chapter 2, classical (binary) Goppa codes are subfield subcodes of AG codes, so one might wonder why subfield subcodes are not susceptible to this attack. This is explained on page 23 of [4].

Conclusion

In this thesis, the possibly post-quantum McEliece cryptosystem has been investigated. In order to understand the cryptosystem itself, some concepts from cryptology and linear codes over the finite field $\mathbb{F}_q$ had to be studied. The original McEliece cryptosystem makes use of binary Goppa codes, which have better error correcting capacities than Goppa codes over arbitrary finite fields. An example of a binary Goppa code was worked out in detail.

Because of the large key size in the McEliece system, many suggestions for improvements or variations have been made. One example is the suggestion to use AG codes instead of Goppa codes. In order to understand these AG codes, many concepts and propositions concerning divisors, Riemann Roch spaces and differentials had to be studied. The propositions and definitions were illustrated by small examples which were treated with much detail. It was shown that the two types of algebraic geometry codes, AG codes and geometric Goppa codes, are in fact each other's duals.

An important concept that kept showing up was the Schur product: the coordinatewise product of two vectors. The Schur product of two linear spaces was defined as the space which is spanned by the Schur products of the vectors from both spaces. One application of the Schur square is decoding by Error Correcting Pairs. This efficient decoding algorithm was studied in detail and illustrated by an example. Because AG codes are obtained by evaluating functions from the Riemann Roch space in various points, the Schur square of an AG code satisfies some properties that follow from the Riemann Roch theorem. It turned out that these properties make AG codes unsuitable for the McEliece cryptosystem.

In the McEliece cryptosystem, the private key is a secret code with an efficient decoding algorithm. The public code is a permuted and camouflaged version of the same code. The security of McEliece depends on the idea that the public code should look like a random code. If that would be the case, breaking the system comes down to solving the general decoding problem, which is NP-complete. It turned out that AG codes do not behave like random codes. Under some conditions, the properties of an AG code with respect to the Schur product can be used to construct an Error Correcting Pair for the public code. An

illustrative example of this attack was worked out in detail.

There remain some questions that we have not been able to answer conclusively. For example, why exactly is it true that using AG codes results in shorter keys? And why exactly are subfield subcodes (such as classical Goppa codes) not vulnerable to this attack? And is it possible to prove that it is not possible to find a large class of codes for which the attack in [4] does not work?

Bibliography

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

[2] D.J. Bernstein. Introduction to post-quantum cryptography. In D.J. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-Quantum Cryptography*. Springer, Berlin, Heidelberg, 2009.

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

[4] A. Couvreur, I. Marquez-Corbella, and G.R. Pellikaan. Cryptanalysis of mceliece cryptosystem based on algebraic geometry codes and their subcodes. *IEEE Transactions on Information Theory*, 63(8):5404–5418, 2017.

[5] A. Couvreur and H. Randriambololona. Algebraic geometry codes and some applications. *Preprint, arXiv:2009.01281v1*, 2020.

[6] J. Daemen and V. Rijmen. *The Design of Rijndael: The Advanced Encryption Standard (AES)*. Springer, Berlin, 2nd edition, 2010.

[7] William Fulton. *Algebraic curves*. Advanced Book Classics. Addison-Wesley Publishing Company, Advanced Book Program, Redwood City, CA, 1989. An introduction to algebraic geometry, Notes written with the collaboration of Richard Weiss, Reprint of 1969 original.

[8] V.D. Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6:24–30, 1970.

[9] V.D. Goppa. Algebraico-geometric codes. *Izvestiya Akademii Nauk SSSR. Seriya Matematicheskaya*, 46:762–781, 1982.

[10] R.W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

[11] R. Hartshorne. *Algebraic Geometry*. Graduate Texts in Mathematics (52). Springer-Verlag, New York, 1977.

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

[13] R. Kötter. A unified description of an error locating procedure for linear codes. In *Proceedings of Algebraic and Combinatorial Coding Theory*, pages 113–117. Voneshta Voda, 1992.

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

[15] D. Mumford. Varieties defined by quadratic equations. In *Questions on algebraic varieties, C.I.M.E., III Ciclo, Varenna, 1969*, pages 29–100. Edizioni Cremonese, Rome, 1970.

[16] N. Patterson. The algebraic decoding of goppa codes. *IEEE Transactions on Information Theory*, 21:203–207, 1975.

[17] G.R. Pellikaan. On decoding by error location and dependent sets of error positions. *Discrete Mathematics*, 106-107(1):369–381, 1992.

[18] R. Pellikaan. On the efficient decoding of algebraic-geometric codes. In *Eurocode 92*, page 231–253. Udine, CISM Courses and Lectures 339, Springer, Wien, 1993.

[19] R. Pellikaan, X. Wu, S. Bulygin, and R. Jurrius. *Codes, cryptology and curves with computer algebra*. Cambridge University Press, Cambridge, 2018.

[20] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[21] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[22] H. Stichtenoth. *Algebraic Function Fields and Codes*. Graduate Texts in Mathematics (254). Springer-Verlag, Berlin Heidelberg, 2nd edition, 2009.

Appendix A

Schur square of random codes

For a code of dimension k and n, [3] proves that if the difference $k(k+1)/2-n$ goes to infinity as a function of k, the probability that the Schur square equals the whole space goes to one. The speed of convergence is exponential if the difference $k(k+1)/2-n$ is at least linear in k. To make this even more precise, define $\mathcal{C}(n,k)$ to be the set of all $[n,k]$ codes over $\mathbb{F}_q$ whose first k coordinates make up an information set. One of the main results of [3] is the following:

Proposition A.0.1. *Let $n:\mathbb{N}\to\mathbb{N}$ be such that $k(k+1)/2 \geq n(k) \geq k$ for all $k\in\mathbb{N}$ and define $t:\mathbb{N}\to\mathbb{N}$, $t(k) := k(k+1)/2 - n(k)$. Then there exist constants $\gamma,\delta\in\mathbb{R}_{>0}$ such that, for all large enough k,*

$$\Pr(C^{(2)} = \mathbb{F}_q^{n(k)}) \geq 1 - 2^{-\gamma k} - 2^{-\delta t(k)},$$

where C is chosen uniformly at random from $\mathcal{C}(n(k),k)$.

We will try to give an idea of the proof.

The first step is to look at codes V of length $n = k(k+1)/2$ and then show that $d((V^{(2)})^{\perp})$, the minimal distance of the dual of the Schur square of V, is probably very large. The following construction is crucial in the proof:

Lemma A.0.2. *Let $V\in\mathcal{C}(k(k+1)/2,k)$ with generator matrix*

$$G = \left(\begin{array}{ccc|c} 1 & & & \boldsymbol{g}_1 \\ & \ddots & & \vdots \\ & & 1 & \boldsymbol{g}_k \end{array}\right)$$

and let D be the dual of the code $\mathrm{Span}\{\boldsymbol{g}_i * \boldsymbol{g}_j \mid 1\leq i\leq k<j\leq k\}$. Then $d((V^{(2)})^{\perp}) \geq d(D)$.

Proof. Suppose $\mathbf{x} = (x_1, \ldots, x_{k(k+1)/2}) \in (V^{(2)})^\perp$ and $\mathrm{wt}(\mathbf{x}) = m$. If we can show that D contains a word with weight m (or less), we are done. Because $\mathbf{x} \in (V^{(2)})^\perp$, we know that $\mathbf{x} \cdot (\mathbf{c}_i * \mathbf{c}_j) = 0$ for all rows $\mathbf{c}_i$ and $\mathbf{c}_j$ of G. In particular, $\mathbf{x} \cdot (\mathbf{c}_i * \mathbf{c}_j) = 0$ for all i and j such that $i \leq k < j \neq j$. If $i \neq j$, the first k coordinates of $(\mathbf{c}_i * \mathbf{c}_j)$ are zero. Now, let $\mathbf{y} := (x_{k+1}, \ldots, x_{k(k+1)/2})$. Then $\mathbf{y}$ has weight $\leq m$ and $\mathbf{y} \cdot (\mathbf{g}_i * \mathbf{g}_j) = \mathbf{x} \cdot (\mathbf{c}_i * \mathbf{c}_j) = 0$ for all $i \leq k < j$, which means that $\mathbf{y} \in D$. □

It is then shown, using Gaussian binomial coefficients and the q-ary entropy function, that for such a code D, the sum of the probabilities that there exist code words of weight $w \leq c \cdot k(k+1)/2$ is small:

Lemma A.0.3. *There exist $c, \tilde{c} \in \mathbb{R}_{>0}$ such that, for all large enough k,*

$$\sum_{w=1}^{c \cdot k(k+1)/2} \Pr\left(\left(D^{(2)}\right)^\perp \text{contains a code word of weight } w\right) \leq 2^{-\tilde{c}k}.$$

The probability that D contains a code word of weight at most $c \cdot k(k+1)/2$ is even smaller or equal, so together with the previous lemmas, this proves the following lemma:

Lemma A.0.4. *There exist constants (depending only on q) $c, \tilde{c} \in \mathbb{R}_{>0}$ such that, for all large enough k, if V is chosen uniformly at random from $\mathcal{C}(k(k+1)/2, k)$, then*

$$\Pr\left(d\left(\left(V^{(2)}\right)^\perp\right) \leq c \cdot \frac{k(k+1)}{2}\right) \leq 2^{-\tilde{c}k}.$$

Note that we are actually interested in codes C of dimension k and length $n < k(k+1)/2$. Such a code can be obtained by taking a random code $V \in \mathcal{C}(k(k+1)/2, k)$ and then puncturing (that is, removing some coordinates) at $k(k+1)/2 - k$ random positions (but not the first k). Any code word in the dual of $C^{(2)}$ gives a word of the same weight in the dual of $V^{(2)}$ (by padding with zeros at the punctured coordinates). According to [3], this implies that

$$\Pr\left(C^{(2)} \neq \mathbb{F}_q^{n(k)}\right) \leq \Pr\left(d\left(\left(V^{(2)}\right)^\perp\right) \leq c \cdot \frac{k(k+1)}{2}\right),$$

where c and $\tilde{c}$ are the constants from the previous lemma. We have not succeeded in providing a convincing argument for this. Combined with the previous lemma, this gives the following proposition, which is in fact a weaker version of proposition A.0.1:

Proposition A.0.5. *There exist constants (depending only on q) $c, \tilde{c} \in \mathbb{R}_{>0}$ such that, if $n : \mathbb{N} \to \mathbb{N}$ satisfies $k \leq n(k) \leq c \cdot \frac{k(k+1)}{2}$ for all $k \in \mathbb{N}$, then, for all large enough k,*

$$\Pr(C^{(2)} = \mathbb{F}_q^{n(k)}) \geq 1 - 2^{-\tilde{c}k}$$

where C is chosen uniformly at random from $\mathcal{C}(n(k), k)$.

The proof of proposition A.0.1 involves the vector space of quadratic forms on $\mathbb{F}_q^k$, denoted by $\mathrm{Quad}(\mathbb{F}_q^k)$. If an $[n, k]$ code C is generated by generator matrix G, the columns of G are denoted by $\pi_1, \ldots, \pi_n \in \mathbb{F}_q^k$ and a linear map is defined as follows:

$$\begin{aligned} \mathrm{ev}_C : \quad & \mathrm{Quad}(\mathbb{F}_q^k) \to \mathbb{F}_q^n \\ & Q \mapsto (Q(\pi_1), \ldots, Q(\pi_n)) \end{aligned}$$

Because the Schur square of C is spanned by the Schur products of the rows of G, the image of this map is exactly the Schur square of C. This means that the map is surjective if and only if $C^{(2)} = \mathbb{F}_q^n$. Since the dimension of the domain equals the dimension of the image plus the dimension of the kernel, it follows that $C^{(2)} = \mathbb{F}_q^n$ if and only if the dimension of $\ker(\mathrm{ev}_C)$ equals the dimension of $\mathrm{Quad}(\mathbb{F}_q^k)$ minus n, that is, if and only if $\dim(\ker(\mathrm{ev}_C)) = k(k+1)/2 - n$. For the kernel, the expected value of the cardinality is computed when $n = k(k+1)/2$ and k goes to infinity, and this turns out to be 2. This fact is then used to finally prove proposition $A.0.1$.

Appendix B

MAGMA code

In this Appendix, the MAGMA code is given that was used in some Examples. Please note that I am no coding expert and everything could probably have been done more efficiently and elegantly.

B.1 Example 1.2.6

```
K<a>:=GF(4);
P2<X,Y,Z> := ProjectiveSpace(K,2);
C := Curve(P2,X^3 + Y^3 + Z^3);
F<x,y> := FunctionField(C);
Q := C ! [0,1,1];
P:=Places(C,1);
P:=Remove(P,9); P:=Rotate(P,2); S:=P;
S[3]:=P[6];S[6]:=P[3];S[4]:=P[7];S[7]:=P[4];S[5]:=P[8];S[8]:=P[5];
X:=[*0,0,0,0,0,0,0,0*];
for i in [1..8] do
    X[i]:=Divisor(S[i]);
end for;
X;
G:=6*Divisor(Q);
Z:= DifferentialBasis(G-X[1]-X[2]-X[3]-X[4]-X[5]-X[6]-X[7]-X[8]);
print "Basis of Omega(6Q-D):", Z;
print "Valuations and residues of eta:";
print "Q", Valuation(Z[1],Place(Q));
for i in [1..8] do
     print i, Valuation(Z[1],P[i]), Residue(Z[1],P[i]);
end for;
print "Valuations and residues of zeta:";
print "Q", Valuation(Z[2],Place(Q));
for i in [1..8] do
```

```
    print i, Valuation(Z[2],P[i]), Residue(Z[2],P[i]);
end for;
```

This returns the following:

```
[*
    Divisor 1*Place at (0 : a : 1),
    Divisor 1*Place at (0 : a^2 : 1),
    Divisor 1*Place at (1 : 0 : 1),
    Divisor 1*Place at (a : 0 : 1),
    Divisor 1*Place at (a^2 : 0 : 1),
    Divisor 1*Place at (1 : 1 : 0),
    Divisor 1*Place at (a : 1 : 0),
    Divisor 1*Place at (a^2 : 1 : 0)
*]
Basis of Omega(6Q-D): [ ((y^2 + 1)/(y^3 + y^2 + y)) d(y),
    ((y + 1)/(y^3 + y^2 + y)*x) d(y) ]
Valuations and residues of eta:
Q 8
x_ser_val, n, v, e = 0, 3, 1, 3
val_den_bound = 9
1 -1 1
x_ser_val, n, v, e = 0, 3, 1, 3
val_den_bound = 9
2 -1 1
3 -1 1
4 -1 1
5 -1 1
6 -1 1
7 -1 1
8 -1 1
Valuations and residues of zeta:
Q 6
1 0 0
2 0 0
3 -1 1
4 -1 a
5 -1 a^2
6 -1 1
7 -1 a
8 -1 a^2
```

B.2 Example 2.3.3

```
K<a> := FiniteField(2^5);
MinimalPolynomial(a);
```

```
P<x> := PolynomialRing(K);
G := x^5 + a^13*x^4+a^12*x^2+a^26;
IsIrreducible(G);
L := [0] cat [a^i : i in [0 .. 30]];
GoppaCode(L, G);
```

This returns the following:

```
$.1^5 + $.1^2 + 1
true
[32, 7, 11] "Goppa code (r = 5)" Linear Code over GF(2)
Generator matrix:
[1 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 0 1 0 0 1 1]
[0 1 0 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 0 1 1 0 1 1]
[0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1 0 0 1 0 1 1]
[0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 1 0 1 1]
[0 0 0 0 1 0 1 0 0 0 1 1 0 1 1 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 0 1]
[0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0 0]
[0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 1 0 1]
```

B.3 Section 6.3

```
F:=GF(17);
squarecode := function(C);
    k:=Dimension(C);
    n:=Length(C);
    G:=GeneratorMatrix(C);
    R:=[];
    for i in [1..k] do
        for j in [i..k] do
            for k in [1..n] do
                Append(~R,G[i,k]*G[j,k]);
            end for;
        end for;
    end for;
    MP := KMatrixSpace(F, Binomial(k+1,2), n);
    R := MP ! R;
    return LinearCode(R);
end function;
productvectors := function(A,B);
    a:=NumberOfRows(A); b:=NumberOfRows(B); c:=NumberOfColumns(A);
    R:=[];
    for i in [1..a] do
        for j in [1..b] do
            for k in [1..c] do
                Append(~R,A[i,k]*B[j,k]);
```

```
            end for;
        end for;
    end for;
    MP := KMatrixSpace(F, a*b, c);
    R := MP ! R;
    return GeneratorMatrix(LinearCode(R));
end function;

P2<X,Y,Z> := ProjectiveSpace(F,2);
CC := Curve(P2,X^3 + Y^3 + Z^3);
P:=Places(CC,1);
P;
Q := CC ! [9,1,1];
S:=Remove(P,18);
D:=6*Divisor(Q);
CL:=AlgebraicGeometricCode(S,D);
CLD:=Dual(CL);
G:=GeneratorMatrix(CLD); //this is the public code

//now the attack starts
C:=LinearCode(G);
H:=ParityCheckMatrix(C);
print "H is", H;
k1:=NumberOfRows(H);
H2:=GeneratorMatrix(squarecode(LinearCode(H)));
print "H2 is", H2;
k2:=NumberOfRows(H2);
print "deg(E) = ", k2-k1, "en g =", k2-2*k1+1;

//in the end, we need V_(-t-g)=V_-3
//First, we compute V_0
GV0:=EchelonForm(RemoveColumn(H,1));
print "The generator matrix of V_0 is", GV0;

//Now we compute V_1
GV1:=RemoveRow(RemoveColumn(H,1),1);
print "The generator matrix of V_-1 is", GV1;

//Now we compute V_-2
V1:=LinearCode(GV1);
HV1:=ParityCheckMatrix(V1);
V12:=squarecode(V1);
HV12:=ParityCheckMatrix(V12);
V0HV12:=productvectors(GV0,HV12);
V2:=LinearCode(RowNullSpace(VerticalJoin(V0HV12,HV1)));
GV2:=GeneratorMatrix(V2);
```

```
print "The generator matrix of V_-2 is", GV2;

//Now we compute V_-3 (=B)
HV2:=ParityCheckMatrix(V2);
V22:=squarecode(V2);
HV22:=ParityCheckMatrix(V22);
V1HV22:=productvectors(GV1,HV22);
V3:=LinearCode(RowNullSpace(VerticalJoin(V1HV22,HV2)));
print "B is"; V3;

//Now we need U_-3 = A of the error correcting pair
BB:=GeneratorMatrix(V3);
CC:=ParityCheckMatrix(LinearCode(GV0));
BBCC:=productvectors(BB,CC);
AA:=ParityCheckMatrix(LinearCode(BBCC));
print "A is"; LinearCode(AA);

//Checking whether we found the right codes
PP:=S[1];
DD:=6*Divisor(Q)-Divisor(PP);
SS:=Remove(S,1);
VV0:=AlgebraicGeometricCode(SS,D); //test whether this equals V_0
VV1:=AlgebraicGeometricCode(SS,DD); //test whether this equals V_-1
DDD:=6*Divisor(Q)-2*Divisor(PP);
VV2:=AlgebraicGeometricCode(SS,DDD); //test whether this equals V_-2
DDDD:=6*Divisor(Q)-3*Divisor(PP);
VV3:=AlgebraicGeometricCode(SS,DDDD);//does this equal V_-3 (=B)
UU3:=AlgebraicGeometricCode(SS,3*Divisor(PP));//equals U_-3 (=A)?
print "Time to check!";
print "V_0:", VV0, "V_-1:", VV1, "V_-2:", VV2, "V_-3:", VV3, "U_-3:", UU3
```

This returns the following:

```
[
    Place at (16 : 1 : 0),
    Place at (16 : 0 : 1),
    Place at (5 : 3 : 1),
    Place at (1 : 9 : 1),
    Place at (8 : 10 : 1),
    Place at (6 : 13 : 1),
    Place at (3 : 5 : 1),
    Place at (14 : 15 : 1),
    Place at (12 : 11 : 1),
    Place at (0 : 16 : 1),
    Place at (15 : 14 : 1),
    Place at (10 : 8 : 1),
```

```
    Place at (4 : 7 : 1),
    Place at (7 : 4 : 1),
    Place at (11 : 12 : 1),
    Place at (2 : 2 : 1),
    Place at (13 : 6 : 1),
    Place at (9 : 1 : 1)
]
H is
[ 1  0  0  0  0  0 16 15  8 12  9  2  6 11  3  9  0]
[ 0  1  0  0  0  0 12  6 15  9  5  5  2  0  1  7 12]
[ 0  0  1  0  0  0 13 14  9 10  8  3  7  5  8 15  8]
[ 0  0  0  1  0  0  3  0 14  4  8  2  1 13 14  6  8]
[ 0  0  0  0  1  0 13  9 12  2  7  5  0  8  7  7  1]
[ 0  0  0  0  0  1 12  8 11 15 15  1  2 15  2  8  6]
H2 is
[ 1  0  0  0  0  0  0  0  0  0  0 10  0  9 15  9 16]
[ 0  1  0  0  0  0  0  0  0  0  0 16  0  6 10  7  2]
[ 0  0  1  0  0  0  0  0  0  0  0  7  0 10  9 16  9]
[ 0  0  0  1  0  0  0  0  0  0  0  9  0  9  1  6 10]
[ 0  0  0  0  1  0  0  0  0  0  0 13  0  0  9  7 16]
[ 0  0  0  0  0  1  0  0  0  0  0 10  0  6  0  3 12]
[ 0  0  0  0  0  0  1  0  0  0  0 12  0  5 11  3 11]
[ 0  0  0  0  0  0  0  1  0  0  0  5  0  9 11  1 13]
[ 0  0  0  0  0  0  0  0  1  0  0  8  0 16  9  5  0]
[ 0  0  0  0  0  0  0  0  0  1  0  1  0 11  8  0  5]
[ 0  0  0  0  0  0  0  0  0  0  1 12  0  8 13  4  3]
[ 0  0  0  0  0  0  0  0  0  0  0  0  1 14  7  8  6]
deg(E) =  6 en g = 1
The generator matrix of V_0 is
[ 1  0  0  0  0  0 16  9  0 11 12  6 13  3 13 12]
[ 0  1  0  0  0  0  5 11 13  6 12  0 12 13 13  8]
[ 0  0  1  0  0  0 11  4  6  1  8  2 12  6 16  8]
[ 0  0  0  1  0  0  0 14  5  5 14 10 15 12  5  1]
[ 0  0  0  0  1  0  1  5  6  4  8  6 11  4 14  6]
[ 0  0  0  0  0  1  2  9  5  8 15 11  6 14  8  0]
The generator matrix of V_-1 is
[ 1  0  0  0  0 12  6 15  9  5  5  2  0  1  7 12]
[ 0  1  0  0  0 13 14  9 10  8  3  7  5  8 15  8]
[ 0  0  1  0  0  3  0 14  4  8  2  1 13 14  6  8]
[ 0  0  0  1  0 13  9 12  2  7  5  0  8  7  7  1]
[ 0  0  0  0  1 12  8 11 15 15  1  2 15  2  8  6]
The generator matrix of V_-2 is
[ 1  0  0  2  0  4  7  5 13  2 15  2 16 15  4 14]
[ 0  1  0  6  0  6  0 13  5 16 16  7  2 16  6 14]
[ 0  0  1  3  0  8 10 16 10 12  0  1  3  1 10 11]
[ 0  0  0  0  1 12  8 11 15 15  1  2 15  2  8  6]
```

```
B is
[16, 3, 13] Linear Code over GF(17)
Generator matrix:
[ 1  0  0  2 14  2  0  6  2  8 12 13  5  9 14 13]
[ 0  1  0  6 16 11  9  2  7  1 15  5  4 14 15  8]
[ 0  0  1  3 10  9  5  7  7  9 10  4  0  4  5  3]
A is
[16, 3, 13] Linear Code over GF(17)
Generator matrix:
[ 1  0  0 13 15  3  0 11 13 16  6 10 16 13 16  7]
[ 0  1  0 13 13 14  5  5  1 12 11 10  2  8 13  1]
[ 0  0  1  9  7  1 13  2  4  7  1 15  0 14  6 10]
Time to check!
V_0: [16, 6] Linear Code over GF(17)
Generator matrix:
[ 1  0  0  0  0  0 16  9  0 11 12  6 13  3 13 12]
[ 0  1  0  0  0  0  5 11 13  6 12  0 12 13 13  8]
[ 0  0  1  0  0  0 11  4  6  1  8  2 12  6 16  8]
[ 0  0  0  1  0  0  0 14  5  5 14 10 15 12  5  1]
[ 0  0  0  0  1  0  1  5  6  4  8  6 11  4 14  6]
[ 0  0  0  0  0  1  2  9  5  8 15 11  6 14  8  0]
V_-1: [16, 5] Linear Code over GF(17)
Generator matrix:
[ 1  0  0  0  0 12  6 15  9  5  5  2  0  1  7 12]
[ 0  1  0  0  0 13 14  9 10  8  3  7  5  8 15  8]
[ 0  0  1  0  0  3  0 14  4  8  2  1 13 14  6  8]
[ 0  0  0  1  0 13  9 12  2  7  5  0  8  7  7  1]
[ 0  0  0  0  1 12  8 11 15 15  1  2 15  2  8  6]
V_-2: [16, 4] Linear Code over GF(17)
Generator matrix:
[ 1  0  0  2  0  4  7  5 13  2 15  2 16 15  4 14]
[ 0  1  0  6  0  6  0 13  5 16 16  7  2 16  6 14]
[ 0  0  1  3  0  8 10 16 10 12  0  1  3  1 10 11]
[ 0  0  0  0  1 12  8 11 15 15  1  2 15  2  8  6]
V_-3: [16, 3, 13] Linear Code over GF(17)
Generator matrix:
[ 1  0  0  2 14  2  0  6  2  8 12 13  5  9 14 13]
[ 0  1  0  6 16 11  9  2  7  1 15  5  4 14 15  8]
[ 0  0  1  3 10  9  5  7  7  9 10  4  0  4  5  3]
U_-3: [16, 3, 13] Linear Code over GF(17)
Generator matrix:
[ 1  0  0 13 15  3  0 11 13 16  6 10 16 13 16  7]
[ 0  1  0 13 13 14  5  5  1 12 11 10  2  8 13  1]
[ 0  0  1  9  7  1 13  2  4  7  1 15  0 14  6 10]
```