Code Based Cryptography and Algebraic Geometry Codes

Jens Kopp

Supervisor: Prof. Dr. Joachim Rosenthal
Co-Supervisor: Michael Schaller

Master Thesis

Institute of Mathematics
University of Zurich
Switzerland
October 2024

Contents

Outline **2**

1 Introduction **3**
1.1 Basic Notions of Coding Theory 3
1.2 Curves 6
1.2.1 Curves over algebraically closed Fields 6
1.2.2 Curves over non-closed Fields 12
1.3 Algebraic Geometry Codes and the McEliece Encryption Scheme 17

2 Distinguishers and Distinguisher based Attacks **19**
2.1 Distinguishers 19
2.2 The s-Closure, Subcodes and Subfield Subcodes 21
2.2.1 The s-Closure 21
2.2.2 Subcodes and Subfield Subcodes 24

3 Error Correcting Pairs and attacking AG-Codes **25**
3.1 Error Correcting Pairs 25
3.2 Attacking AG-Codes using ECP 28
3.3 Examples based on Hermitian Curves 34
3.3.1 Construction of Generator Matrices 36

4 Secret Sharing **36**
4.1 Shamir's Secret Sharing 36
4.2 Secret Sharing based on AG-Codes 42
4.2.1 Linear Secret Sharing Schemes 42
4.2.2 An LSSS using AG-Codes 49

5 Verifiable Secret Sharing and Secure Multi Party Computation **56**
5.1 VSS and MPC 57
5.2 Secure MPC using LSSS 75

References **96**

A SageMath Codes **98**

B Magma Codes **100**

Outline

The main focus of this master thesis are algebraic geometry codes and two interesting applications in cryptography. The first one is their usage in the McEliece encryption scheme, which was proposed for use in post-quantum cryptography. However, in the case where using algebraic geometry codes, this scheme was broken by the methods seen in Couvreur, Márquez-Corbella, and Pellikaan [7] and Section 3.2 covers the presented attack in detail. The second application of algebraic geometry codes is the construction of a linear secret sharing scheme based on algebraic geometry codes. This can be seen as a generalization of Shamir's Secret Sharing Scheme, which was first described in Shamir [23]. Linear secret sharing schemes are the main topic of Section 4.2. Finally, Section 5 deviates a bit from the close usage of an algebraic geometry code and presents a more general approach to the problems of verifiable secret sharing and multi party computation, mainly following Cramer, Damgård, and Maurer [11].

The main work of this master thesis consisted of understanding, compiling and elaborating on the main sources Couvreur, Márquez-Corbella, and Pellikaan [7], Chen and Cramer [4] and Cramer, Damgård, and Maurer [11] and providing additional examples, lemmas and proofs where necessary. Section 1 contains a very brief overview of Coding Theory and Algebraic Geometry with the aim of providing the necessary background to define algebraic geometry codes. In Section 2, we cover additional background, with the only notable addition to the source material being the proof of Proposition 2.5. Section 3 contains more notable mentions. First, it provides more background about ECP following Pellikaan et al. [21], which was omitted from Couvreur, Márquez-Corbella, and Pellikaan [7] but is interesting for the topic. On that note, we complemented Algorithm 3.7 with Lemma 3.6. In Section 3.2, we elaborated on some proofs, most notably the proof of Theorem 3.15 as well as the succeeding remark. Finally, we provide some interesting examples in Section 3.3. These include Examples 8 and 10 together with the corresponding codes in Appendices A and B. The codes in Appendix B corresponding to Example 9 are also noteworthy, but the example itself was inspired by Couvreur, Márquez-Corbella, and Pellikaan [7, Example 1].
Section 4 marks a shift in topic from the attack on the McEliece encryption scheme to the area of secret sharing. Some notable mentions include Example 12 and Lemma 4.14. In Section 4.2.2, we elaborated on the LSSS described in Chen and Cramer [4] and gave a more explicit description including Example 15. Lastly, Section 5 covers the main ideas of Gennaro, Rabin, and Rabin [14] and Cramer, Damgård, and Maurer [11]. Many protocols and theorems in Section 5.2 required proper formulations and proofs, which were mostly sketched in Cramer, Damgård, and Maurer [11]. Most notably, this includes the proof of Proposition 5.8, Protocols 9, 10 and 11 as well as the corresponding Propositions 5.12, 5.13 and 5.14 and their proofs. Furthermore, we provided many examples for better understanding. In particular, we want to point out Examples 16, 17, 20, 22, 23 and 24. The remark between Examples 22 and 23 is also noteworthy.

1 Introduction

This section contains a short overview of Coding Theory and Algebraic Geometry, mainly following chapters 1, 2 and 4 in Tsfasman, Vlăduț, and Nogin [28] with some supplementary information from Pellikaan et al. [21].

1.1 Basic Notions of Coding Theory

Definition 1.1 (Hamming Distance)**.** For $a = (a_1, \dots, a_n), b = (b_1, \dots, b_n) \in \mathbb{F}_q^n$, we define the *Hamming distance* as

$$d(a, b) = |\{i \mid a_i \neq b_i\}|.$$

The *Hamming weight* of $c = (c_1, \dots, c_n) \in \mathbb{F}_q^n$, denoted by $\|c\|$ or $\mathrm{wt}(c)$, is defined as

$$\|c\| = |\{i \mid c_i \neq 0\}|.$$

Remark. One can consider various other distances and weights, different from the Hamming distance and Hamming weight. In the following, the terms distance and weight always mean the Hamming distance and Hamming weight respectively.

Definition 1.2 (Linear Code)**.** Let $\mathbb{F}_q$ be the finite field with $q = p^m$ elements, where p is some prime number. A *linear code* C of *length* n is a linear subspace of $\mathbb{F}_q^n$.
For a linear code C we denote its *dimension* by $k = \dim C$ and its *(minimum) distance* by $d = \min\{\|c\| \mid c \in C, c \neq 0\}$, where $\|\cdot\|$ is the Hamming weight.

Example 1. Let p be a prime number and consider the finite field $\mathbb{F}_p$. Let $n \in \mathbb{N}_{>0}$. Consider the one vector $(1, \dots, 1) \in \mathbb{F}_p^n$ and set $C = \mathrm{span}((1, \dots, 1))$. It is clear by definition that C is a linear code of length n and dimension $k = 1$. For $c \in C$ with $c \neq \vec{0}$, we have that every coordinate $c_i \neq 0$ and thus $\mathrm{wt}(c) = n$. This means also that the distance of C is $d = n$.

As $\dim_{\mathbb{F}_q}(C) = k$, we can identify C with $\mathbb{F}_q^k$ and by choosing a basis we obtain an embedding $\mathbb{F}_q^k \hookrightarrow \mathbb{F}_q^n$. We denote by G its corresponding matrix, which is called the *generator matrix* of the code C. From this, we obtain the following short exact sequence:

$$0 \to \mathbb{F}_q^k \xrightarrow{G} \mathbb{F}_q^n \xrightarrow{H} \mathbb{F}_q^{n-k} \to 0.$$

The generator matrix G is a $k \times n$-matrix, mapping $\mathbb{F}_q^k$ to $C \subset \mathbb{F}_q^n$, whereas the $(n-k) \times n$-matrix H, which is called the *parity check matrix* of C, has the code C as its kernel.

Remark. The matrices G and H are not unique. If we already know a generator matrix G and a parity check matrix H, we can apply elementary row operations to them. These are

- multiplication by a unit,

- swapping two rows,
- adding a multiple of a row to another row.

This does not change the code defined by them. In particular, this means that two matrices G and G' define the same code if and only if there exists a matrix $B \in GL_k(\mathbb{F}_q)$ such that $G' = BG$.

Example 2. Consider the finite field $\mathbb{F}_3$ and the linear code $C \subset \mathbb{F}_3^7$ defined as

$$\begin{aligned} C = \{&(0,0,0,0,0,0,0), (0,1,0,2,0,1,2), (0,2,0,1,0,2,1), (0,0,1,2,0,1,1), (0,1,1,1,0,2,0), \\ &(0,2,1,0,0,0,2), (0,0,2,1,0,2,2), (0,1,2,0,0,0,1), (0,2,2,2,0,1,0)\}. \end{aligned}$$

This code has as a generator matrix

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

and as a parity check matrix

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

The elements $c \in C$ can be described either through G or H in the following ways.

- $C = \{v \cdot G \mid v \in \mathbb{F}_3^2\} = \mathrm{rowsp}(G)$.
- $C = \{v \in \mathbb{F}_3^7 \mid H \cdot v^T = 0\} = \ker(H)$.

Remark. If C is a code of dimension $\dim(C) = k$ and length n, then we have $\mathrm{rank}(G) = k$ and $\mathrm{rank}(H) = n - k$. Furthermore, G and H satisfy

$$H \cdot G^T = 0 \in Mat_{(n-k)\times k}(\mathbb{F}_q)$$

and

$$G \cdot H^T = 0 \in Mat_{k\times(n-k)}(\mathbb{F}_q).$$

Proposition 1.3 ([21, Proposition 1.3.3])**.** *Let C be a code of length n and dimension k and denote by I_k the $k \times k$ identity matrix. Then the matrix $(I_k|P)$, for $P \in Mat_{k\times(n-k)}(\mathbb{F}_q)$, is a generator matrix of C if and only if the matrix $(-P^T|I_{n-k})$ is a parity check matrix of C.*

Proof. We assume that C has a generator matrix of the form $G = (I_k|P)$. Hence, we can assume that a codeword $\mathbf{m}G = c \in C$ has the form $c = (\mathbf{m}, \mathbf{r})$, where $\mathbf{r} \in \mathbb{F}_q^{n-k}$ and $\mathbf{r} = \mathbf{m}P$. Thus, it follows that

$$\begin{aligned} c \text{ is a codeword} &\Leftrightarrow -\mathbf{m}P + \mathbf{r} = 0 \\ &\Leftrightarrow -P^T\mathbf{m}^T + \mathbf{r}^T = 0 \\ &\Leftrightarrow (-P^T|I_{n-k})(\mathbf{m}, \mathbf{r})^T = 0 \\ &\Leftrightarrow (-P^T|I_{n-k})c^T = 0. \end{aligned}$$

Therefore, we have that $(-P^T|I_{n-k})$ is a parity check matrix for C. It is also clear that the converse is also true. □

Remark. A generator matrix of the form $(I_k|P)$ is said to be in *standard form*. It is not always possible to bring a generator matrix of a code C into standard form. For example in Example 2, it is clear that we cannot do so because the first column of G is 0. However, if we can bring a generator matrix G into standard form using elementary row operations, then it is easy to obtain a parity check matrix H and vice versa.

Definition 1.4 (Dual Code)**.** For a code C, its *dual code* is $C^\perp = \{x \in \mathbb{F}_q^n \mid x \cdot c = 0 \text{ for all } c \in C\}$, where $\cdot$ denotes the standard inner product. If $\dim C = k$, then it follows that $\dim C^\perp = n - k$ and a parity check matrix H of C is a generator matrix for the code $C^\perp$ and vice versa.

Example 3. Consider the finite field $\mathbb{F}_2$ and the code $C \subset \mathbb{F}_2^4$ given by the generator matrix

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

We observe that $H = G$ is a parity check matrix for this code and thus $C = C^\perp$. Such a code is called *self-dual*.

Definition 1.5 (Equivalence)**.** Two linear codes C_1 and C_2 of the same length n over the same base field $\mathbb{F}_q$ with generator matrices G_1 and G_2 are said to be *monomially equivalent* if there exists a monomial matrix P, that is an $n \times n$ - matrix which contains exactly one nonzero element in each row and each column, such that $\tilde{G_1} = G_2P$ is a generator matrix of C_1, i.e., G_1 and $\tilde{G_1}$ generate the same code C_1. In other words, this means that by swapping and rescaling the columns of a generator matrix of C_2 we obtain a generator matrix for C_1. If all the entries of the monomial matrix P are either 0 or 1, P is called a permutation matrix and the codes C_1 and C_2 are said to be *permutation equivalent*. Note that a monomial matrix does not change the dimension or the distance of a code, i.e., two codes can only be equivalent if they have the same dimension and the same distance.

Example 4. Consider the codes C_1 and C_2 over $\mathbb{F}_3$ defined by the generator matrices

$$G_1 = \begin{pmatrix} 1 & 1 & 0 & 2 & 0 \\ 1 & 0 & 2 & 1 & 0 \end{pmatrix} \quad \text{and} \quad G_2 = \begin{pmatrix} 2 & 0 & 0 & 2 & 1 \\ 2 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

We observe that the matrix

$$P = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

is a monomial matrix, but not a permutation matrix, and $G_1P = G_2$. Thus, it follows that the codes C_1 and C_2 are monomially equivalent.

Remark. In the field $\mathbb{F}_2$, the only nonzero element is 1. Therefore, it follows that over $\mathbb{F}_2$ monomially equivalent is the same as permutation equivalent.

Definition 1.6 (Error Correcting Capacity)**.** Let C be a linear code of length n and dimension k with distance d. Then we define

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor,$$

which is called the *error correcting capacity* of the code C.

To explain why this is called error correcting capacity, we consider the following problem. We want to send a message $\mathbf{m}$ of length k, i.e., a vector $\mathbf{m} \in \mathbb{F}_q^k$. While transmitting the message, there might be some noise that changes a few entries of the vector, an error occurred and the message received is different from the message that was sent. To fix this, we can encode the message $\mathbf{m}$. If G is a generator matrix of C, we simply apply it to the message $\mathbf{m}$ and obtain the codeword $\mathbf{c} = \mathbf{m}G$. This codeword is an element of $\mathbb{F}_q^n$. We assume now that some entries were changed by the noise with an error vector $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{e}$ has weight $0 < \mathrm{wt}(\mathbf{e}) \leq t$. This means that we receive $\mathbf{y} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_q^n$. However, we note that, as $0 < \mathrm{wt}(\mathbf{e}) \leq t = \left\lfloor \frac{d-1}{2} \right\rfloor < d$, $\mathbf{e} \notin C$ and thus also $\mathbf{y} \notin C$. We also note that there is a unique codeword in C that is closest to $\mathbf{y}$, namely the codeword $\mathbf{c}$ which is exactly the one we want to receive. Indeed, assume that there exist two codewords $\mathbf{c_1}, \mathbf{c_2} \in C$ such that

$$d(\mathbf{c_1}, \mathbf{y}), d(\mathbf{c_2}, \mathbf{y}) \leq \mathrm{wt}(\mathbf{e}).$$

Then it follows that

$$d(\mathbf{c_1}, \mathbf{c_2}) \leq 2\mathrm{wt}(\mathbf{e}) \leq 2t \leq d - 1 < d,$$

which is a contradiction to C having minimum distance d. Hence, it follows that $\mathbf{c}$ is the only codeword such that $d(\mathbf{c}, \mathbf{y}) \leq \mathrm{wt}(\mathbf{e})$.

Remark. Even though it is theoretically possible to correct up to t errors and uniquely decode a received message, it is much more complicated in practice and might be too costly. There are some algorithms that can do this for some classes of codes, for example the algorithm using error correcting pairs which is explained in Section 3. We will also see the McEliece encryption scheme, which adds error vectors on purpose and exploits the fact that decoding can be very hard in general.

1.2 Curves

1.2.1 Curves over algebraically closed Fields

Let $\mathbb{K}$ be an algebraically closed field. We denote by $\mathbb{A}^n$ the *affine space* of dimension n over $\mathbb{K}$, that is $\mathbb{A}^n = \{(x_1, \ldots, x_n) \mid x_i \in \mathbb{K}\}$. We denote by $\mathbb{P}^n$ the

projective space of dimension n over $\mathbb{K}$ defined as

$$\mathbb{P}^n = \{(y_0 : y_1 : \ldots : y_n) \mid y_i \in \mathbb{K} \text{ not all } 0\}_{/\sim},$$

where the equivalence relation $\sim$ is defined as

$$(y_0 : y_1 : \ldots : y_n) \sim \lambda(y_0 : y_1 : \ldots : y_n) = (\lambda y_0 : \lambda y_1 : \ldots : \lambda y_n),\ \lambda \in \mathbb{K}^*.$$

There exists a natural embedding $\mathbb{A}^n \hookrightarrow \mathbb{P}^n$ given by the map

$$(x_1, \ldots, x_n) \mapsto (1 : x_1 : \ldots : x_n).$$

Definition 1.7 (homogeneous)**.** A polynomial $f \in \mathbb{K}[T_1, \ldots, T_n]$ is called *homogeneous* of degree d if for all $\lambda \in \mathbb{K}$

$$f(\lambda x_1, \ldots, \lambda x_n) = \lambda^d f(x_1, \ldots, x_n).$$

For any $f \in \mathbb{K}[T_1, \ldots, T_n]$, we define its *homogenization* $F \in \mathbb{K}[T_0, T_1, \ldots, T_n]$ as

$$F(T_0, T_1, \ldots, T_n) = T_0^d f\left(\frac{T_1}{T_0}, \ldots, \frac{T_n}{T_0}\right),$$

where $d = \deg(f)$. On the other hand, if we are given a homogeneous polynomial $F \in \mathbb{K}[T_0, T_1, \ldots, T_n]$ we define its *dehomogenization* $f \in \mathbb{K}[T_1, \ldots, T_n]$ as

$$f(T_1, \ldots, T_n) = F(1, T_1, \ldots, T_n).$$

Remark. It is also possible to dehomogenize in a different variable than T_0. In particular in the case where we have three variables T_0, T_1, T_2, it is a convention to use the variables $T_0 = x,\ T_1 = y$ and $T_2 = z$. Then we write $F(x, y, z)$ for a homogeneous polynomial and $f(x, y) = F(x, y, 1)$ for its dehomogenization.

Definition 1.8 (Zariski Topology)**.** We say that a set $X \subset \mathbb{A}^n$ is an *(affine) closed set* if there exists a finite number of polynomials $f_1, \ldots, f_s \in \mathbb{K}[T_1, \ldots, T_n]$ such that X can be written as

$$X = \{P = (x_1, \ldots, x_n) \in \mathbb{A}^n \mid f_1(P) = \ldots = f_s(P) = 0\},$$

i.e., X is the set of common zeros of the polynomials $f_1, \ldots, f_s$.
Similarly, we say that a set $Y \subset \mathbb{P}^n$ is a *(projective) closed set* if there exists a finite number of homogeneous polynomials $F_1, \ldots, F_r \in \mathbb{K}[T_0, T_1, \ldots, T_n]$ such that Y can be written as

$$Y = \{Q = (y_0 : y_1 : \ldots : y_n) \in \mathbb{P}^n \mid F_1(Q) = \ldots = F_r(Q) = 0\},$$

i.e., Y is the set of common zeros of the homogeneous polynomials $F_1, \ldots, F_r$.
We call a set *open* if its complement is closed. This definition defines a topology on $\mathbb{A}^n$ and $\mathbb{P}^n$ respectively. This topology is called the *Zariski topology*.

Remark. The notion of projective closed set is well-defined. Indeed, consider two representatives $(y_0 : y_1 : \ldots : y_n)$ and $(\tilde{y}_0 : \tilde{y}_1 : \ldots : \tilde{y}_n)$ of the same point $P \in \mathbb{P}^n$. This means there exists some $\lambda \in \mathbb{K}^*$ such that

$$(\tilde{y}_0 : \tilde{y}_1 : \ldots : \tilde{y}_n) = \lambda(y_0 : y_1 : \ldots : y_n) = (\lambda y_0 : \lambda y_1 : \ldots : \lambda y_n).$$

Consider now any homogeneous polynomial $F \in \mathbb{K}[T_0, T_1, \ldots, T_n]$. We have

$$F((\tilde{y}_0 : \tilde{y}_1 : \ldots : \tilde{y}_n)) = F((\lambda y_0 : \lambda y_1 : \ldots : \lambda y_n)) = \lambda^d F((y_0 : y_1 : \ldots : y_n))$$

and from this it follows that $(y_0 : y_1 : \ldots : y_n)$ is a zero of F if and only if $(\tilde{y}_0 : \tilde{y}_1 : \ldots : \tilde{y}_n)$ is a zero of F. Thus, the zeros of F are independent of their representation and hence they are well-defined.

Definition 1.9 (Projective Closure)**.** Let

$$X = \{P = (x_1, \ldots, x_n) \in \mathbb{A}^n \mid f_1(P) = \ldots = f_s(P) = 0\} \subset \mathbb{A}^n$$

be a closed affine set and denote by $F_i \in \mathbb{K}[T_0, T_1, \ldots, T_n]$ the homogenization of $f_i \in \mathbb{K}[T_1, \ldots, T_n]$. To X we assign the projective closed set

$$\tilde{X} = \{P = (x_0 : x_1 : \ldots : x_n) \in \mathbb{P}^n \mid F_1(P) = \ldots = F_s(P) = 0\} \subset \mathbb{P}^n$$

containing X. $\tilde{X}$ is called the *projective closure* of X.

For an arbitrary set $S \subset \mathbb{P}^n$ and a subset $T \subset S$, we say that T is *closed in* S if there exists a projective closed set $X \subset \mathbb{P}^n$ such that $T = S \cap X$. We say that T is *open in* S if its complement $S \backslash T$ is closed in S.
An open subset U of a closed projective set X is called a *quasi-projective set*. In particular any affine closed set is quasi-projective. Indeed, consider the affine closed set $X = \{P \in \mathbb{A}^n \mid f_i(P) = 0\}$ and its projective closure $\tilde{X} = \{Q \in \mathbb{P}^n \mid F_i(Q) = 0\}$ and identify $\mathbb{A}^n$ via its embedding into $\mathbb{P}^n$. It follows that

$$\begin{aligned} \tilde{X} \backslash X &= \tilde{X} \backslash \{Q \in \mathbb{P}^n \mid T_0(Q) = 1\} \\ &= \tilde{X} \cap \{Q \in \mathbb{P}^n \mid T_0(Q) \neq 1\} \\ &= \tilde{X} \cap \{Q \in \mathbb{P}^n \mid T_0(Q) = 0\} \end{aligned}$$

is closed. This works since by properties of the projective space we can scale a coordinate to be 1 as long as it is not equal to 0. Thus, it follows that X is open in $\tilde{X}$.

Definition 1.10 (Variety)**.** A quasi-projective set $X \subset \mathbb{P}^n$ is called *reducible* if there exist proper closed subsets $X_1, X_2 \subsetneq X$ such that $X = X_1 \cup X_2$. Otherwise X is called *irreducible*. An irreducible quasi-projective set is called a *(quasi-projective) variety*. If a variety is closed in $\mathbb{P}^n$, it is called *projective variety*.

Definition 1.11 (Dimension of a Variety)**.** For a variety X, the *dimension* $d = \dim X$ is the largest integer d such that there exists a strictly descending chain of varieties

$$X = X_0 \supsetneq \ldots \supsetneq X_d \neq \emptyset$$

such that X_i is closed in X_{i-1} for all $i = 1, \ldots, d$.

Definition 1.12 (Field of Rational Functions)**.** Let $X \subset \mathbb{P}^n$ be a variety and $F, G \in \mathbb{K}[T_0, T_1, \ldots, T_n]$ two homogeneous polynomials of the same degree d such that there exists at least one point $P \in X$ where $G(P) \neq 0$. Then the fraction $\frac{F}{G} \in \mathbb{K}(T_0 : T_1 : \ldots : T_n)$ is called a *rational function* on X. Two fractions $\frac{F}{G}$ and $\frac{F'}{G'}$ define the same rational function if $(F'G - FG')(P) = 0$ at all points $P \in X$. The set of all rational functions is denoted by $\mathbb{K}(X)$ and it is a field with the usual addition and multiplication defined as

$$\frac{F}{G} + \frac{F'}{G'} = \frac{FG' + F'G}{GG'} \quad \text{and} \quad \frac{F}{G} \cdot \frac{F'}{G'} = \frac{FF'}{GG'}.$$

The field $\mathbb{K}(X)$ is called the *field of rational functions* on X.

A rational function $f \in \mathbb{K}(X)$ is called *regular* at a point $P \in X$ if f has a representation $f = \frac{F}{G}$ such that $G(P) \neq 0$. In this case, $f(P) = \frac{F(P)}{G(P)}$ is called the *value* of f at P.

Definition 1.13 (Local Ring of a Point P)**.** Let $P \in X$ be a point. We denote by $\mathcal{O}_{X,P}$ the set of all rational functions $f \in \mathbb{K}[X]$ that are regular at the point P. This set is in fact a ring and is called the *local ring of* P.

Remark. In general, a commutative ring is called *local* if it has a unique maximal ideal. The ring $\mathcal{O}_{X,P}$ is indeed a local ring and has the unique maximal ideal

$$\mathfrak{m}_{X,P} = \{f \in \mathcal{O}_{X,P} \mid f(P) = 0\}.$$

Definition 1.14 (Curve)**.** A *curve* is a variety of dimension 1. If a curve is closed in some projective space $\mathbb{P}^n$, then it is called a *projective curve*.

Definition 1.15 (smooth)**.** Consider a curve X and point $P \in X \subset \mathbb{P}^n$. Without loss of generality we may assume that $P \in X \cap \mathbb{A}^n$. Let $F_1(x), \ldots, F_N(x) \in \mathbb{K}[x_1, \ldots, x_n]$ be such that $F_1(x) = \ldots = F_N(x) = 0$ is a complete system of equations defining $X \cap \mathbb{A}^n$. Consider the Jacobi matrix

$$\frac{\partial F}{\partial x}(P) = \left(\frac{\partial F_i}{\partial x_j}(P)\right)_{i \leq N,\ j \leq n}.$$

Then a point $P \in X$ is *smooth* if $\operatorname{rank} \frac{\partial F}{\partial x}(P) = n - 1$. A curve X is called *smooth* if it is smooth at every point P. In particular in the case of a plane curve with affine equation $f(x, y) = 0$, i.e., $N = 1$ and $n = 2$, a point $P \in X$ is smooth if and only if at least one of the partial derivatives $\frac{\partial f}{\partial x}$ or $\frac{\partial f}{\partial y}$ is not zero at the point P.

Definition 1.16 (Divisor)**.** Let X be a smooth projective curve over an algebraically closed field $\mathbb{K}$. A *divisor* D on X is a finite formal sum of the form $D = \sum a_i P_i$, where P_i are the points on X and $a_i \in \mathbb{Z}$, in particular only finitely many a_i are nonzero. We denote by $\operatorname{Div}(X)$ the set of all divisors on the curve X. $\operatorname{Div}(X)$ is an abelian group with addition

$$\sum a_i P_i + \sum b_i P_i = \sum (a_i + b_i) P_i.$$

Definition 1.17 (Degree)**.** The *degree* of a divisor $D = \sum a_i P_i$ is $\deg D = \sum a_i$.

We denote by $\mathrm{Div}^0(X)$ the set of divisors of degree zero. If for a divisor $D = \sum a_i P_i$ we have $a_i \geq 0$ for all i, then we say that D is *effective* and we write $D \geq 0$. The set of effective divisors is denoted by $\mathrm{Div}^+(X)$. Moreover, we use the notation $D \geq E$ if $D - E \geq 0$.
Let $f \in \mathbb{K}(X)^*$ be a rational function. We can assign a divisor to f in the following way. We define the *order* of f at P as

$$\mathrm{ord}_P(f) = \max\{k \in \mathbb{Z} \mid f \in \mathfrak{m}_{X,P}^k\}$$

if $f \in \mathcal{O}_{X,P}$. For general $f = \frac{g}{h} \in \mathbb{K}(X)$ such that $g, h \in \mathcal{O}_{X,P}$, we define the order as

$$\mathrm{ord}_P(f) = \mathrm{ord}_P(g) - \mathrm{ord}_P(h).$$

Using this definition of order, we can define the divisor

$$(f) = \sum \mathrm{ord}_{P_i}(f) P_i.$$

This definition makes sense since each $f \neq 0$ can only have finitely many zeros and poles on X. Divisors of the form $D = (f)$ are called *principal divisors* and they form a subgroup $\mathrm{Div}_{pr}(X) \subset \mathrm{Div}^0(X)$. In particular, $\deg(f) = 0$.

Definition 1.18 (Riemann-Roch Space)**.** Let D be a divisor on a curve X. We define the space

$$L(D) = \{f \in \mathbb{K}(X) \mid f = 0 \text{ or } (f) + D \geq 0\},$$

which is called the *Riemann-Roch space* of D.

Remark. If we have two divisors E and D such that $E \leq D$, then we have $L(E) \subset L(D)$.

Theorem 1.19 ([28, Corollary 2.1.54])**.** *For an effective divisor $D \geq 0$, we have that*

$$\dim L(D) \leq \deg D + 1.$$

Remark. If $\deg D < 0$, then it follows that $\deg D + \deg(f) = \deg D + 0 < 0$. Hence, $D + (f)$ can never be effective and thus $\dim L(D) = 0$.

Let X be a smooth projective curve and $P \in X$ a point. We note that for algebraically closed $\mathbb{K}$ the local ring $\mathcal{O}_{X,P}$ is a principal ideal domain and $\mathcal{O}_{X,P}/\mathfrak{m}_{X,P} = \mathbb{K}$. Thus, it follows that

$$\mathfrak{m}_{X,P}/\mathfrak{m}_{X,P}^2 = \langle g \rangle / \langle g^2 \rangle,$$

for some element $g \in \mathcal{O}_{X,P}$, is an $\mathcal{O}_{X,P}/\mathfrak{m}_{X,P}$-module, i.e., a $\mathbb{K}$-vector space and hence $\mathfrak{m}_{X,P}/\mathfrak{m}_{X,P}^2$ is a one dimensional vector space over $\mathbb{K}$.

Definition 1.20 (Differential)**.** Consider a rational function $f \in \mathcal{O}_{X,P}$, i.e., f is regular at P. By $d_P f$, we denote the image of $f - f(P) \in \mathfrak{m}_{X,P}$ in the one-dimensional vector space $\mathfrak{m}_{X,P}/\mathfrak{m}_{X,P}^2$. The element $d_P f$ is called the *differential* of f at P.

The differential

$$d_P : \mathcal{O}_{X,P} \to \mathfrak{m}_{X,P}/\mathfrak{m}_{X,P}^2$$

is a morphism of $\mathbb{K}$-vector spaces and satisfies

$$d_P(fg) = f(P)d_P g + g(P)d_P f.$$

Let now $U \subset X$ be an open subset in X. By $\mathbb{K}[U]$, we denote the set of all rational functions that are regular on the whole set U. We denote by $\Phi[U]$ the set of all maps φ sending each point $P \in U$ to some element in $\mathfrak{m}_{X,P}/\mathfrak{m}_{X,P}^2$. Note that $\Phi[U]$ is a module over the ring $\mathbb{K}[U]$ and any rational function $f \in \mathbb{K}[U]$ defines an element $df \in \Phi[U]$ by $(df)(P) = d_P f$.

Definition 1.21 (Differential Form)**.** An element $\varphi \in \Phi[U]$ is called a *differential form regular at U* if for any point $P \in U$ there exists a neighbourhood V of P such that the restriction $\varphi|_V$ lies in the $\mathbb{K}[V]$-submodule of $\Phi[V]$ generated by the elements df for $f \in \mathbb{K}[V]$. Differential forms regular on U form a $\mathbb{K}[U]$-module denoted by $\Omega[U]$.

Definition 1.22 (Genus)**.** Consider the $\mathbb{K}$-vector space $\Omega[X]$. We define the *genus* of X as

$$g = g(X) = \dim_{\mathbb{K}} \Omega[X].$$

We are often interested in smooth plane curves. Consider a homogeneous polynomial $F(x : y : z)$ of degree m and define the projective curve X as $X = \{(x : y : z) \in \mathbb{P}^2 \mid F(x : y : z) = 0\}$. Sometimes we write this in affine coordinates, i.e., we consider $X' = X \cap \mathbb{A}^2 = \{(x, y) \in \mathbb{A}^2 \mid f(x, y) = 0\}$, where $f(x, y) = F(x : y : 1)$ is the dehomogenization of F. In this case, we still mean the projective curve X, which is the projective closure of X'. If the curve X is smooth, we have an easy formula for the genus.

Theorem 1.23 ([28, Corollary 2.2.8])**.** *Let X be a smooth plane curve of degree m. Then X has genus*

$$g(X) = \frac{(m-1)(m-2)}{2}.$$

Remark. This theorem also holds if $\mathbb{K}$ is not algebraically closed.

Theorem 1.24 (Riemann-Roch Theorem, [28, Theorem 2.2.17])**.** *Let X be a smooth projective curve. Then, for any $D \in \mathrm{Div}(X)$,*

$$\dim L(D) - \dim L(K - D) = \deg D - g + 1,$$

where K is a canonical divisor of X.

We do not want to go into details about the definition or explicit description of such a canonical divisor K of X. Its existence and the properties provided in the following corollary are enough. More details can be found in Tsfasman, Vlăduț, and Nogin [28, Chapter 2.2].

Corollary 1.25. *Let X be a smooth projective curve and K a canonical divisor. Then the following hold:*

1. $\deg K = 2g - 2$.
2. *If* $\deg D < 0$, *then* $\dim L(K - D) = g - 1 - \deg D$.
3. *If* $\deg D > 2g - 2$, *then* $\dim L(D) = \deg D - g + 1$.

1.2.2 Curves over non-closed Fields

In the following, let $\Bbbk$ be a field which is not necessarily algebraically closed but assumed to be a perfect field, i.e., a field such that its algebraic closure $\overline{\Bbbk}$ is separable over $\Bbbk$. In particular, if $\Bbbk$ is finite or has characteristic 0, it is always perfect. In the case of algebraically closed fields, we have a 1-to-1 correspondence between varieties given by a system of equations and the radical ideals of their corresponding ideal generated by those equations. However, for non-closed fields this is not true anymore as shown by the following example.

Example 5. Let $\Bbbk = \mathbb{R}$ and consider the polynomials $F(x, y) = x^2+y^2, G(x, y) = x^2 + 2y^2 \in \mathbb{R}[x, y]$. We immediately observe that both polynomials share the same zero set, namely the set $\{(0, 0)\}$ and we also note that if we consider the same polynomials over $\overline{\mathbb{R}} = \mathbb{C}$, their zero sets will be different. The polynomials F and G are irreducible over $\mathbb{R}$ and it is a known fact that irreducible polynomials in 2 variables generate prime ideals and hence they are also radical. It is also clear that they generate two different ideals in $\mathbb{R}[x, y]$.

This example shows that we do not get the desired correspondence by simply copying the definition of variety. There is some more work to do to generalize it. To do so, we need some more definitions.

Definition 1.26 (Algebraic Function Field)**.** An *(algebraic) function field* $\mathbb{K}/\Bbbk$ is any field $\mathbb{K} \supset \Bbbk$ that is a finite algebraic extension of $\Bbbk(x)$, where $x \in \mathbb{K}$ is transcendental over $\Bbbk$. The elements of $\mathbb{K}$ that are algebraic over $\Bbbk$ form a subfield of $\mathbb{K}$, which is called the *field of constants* of $\mathbb{K}/\Bbbk$ and is denoted by $\tilde{\Bbbk}$.

Definition 1.27 (Valuation Ring)**.** A *valuation ring* of the function field $\mathbb{K}/\Bbbk$ is a subring $\Bbbk \subsetneq \mathcal{O} \subsetneq \mathbb{K}$ such that for any $f \in \mathbb{K}$ at least one of the elements f or f^{-1} belongs to $\mathcal{O}$. The set $\mathcal{O}^* = \{f \in \mathbb{K} \mid f, f^{-1} \in \mathcal{O}\}$ is called the *group of units* of $\mathcal{O}$.

Remark. If $\Bbbk$ is an algebraically closed field and X is a curve over $\Bbbk$, then the field of rational functions $\Bbbk(X)$ is a function field and the local ring $\mathcal{O}_{X,P}$ is a valuation ring.

Proposition 1.28 (Properties of Valuation Rings, [28, Proposition 2.5.1])**.** *Let $\mathcal{O}$ be a valuation ring of a function field $\mathbb{K}/\Bbbk$. Then $\mathcal{O}$ is a discrete valuation ring, i.e., it satisfies the following conditions:*

1. *$\mathcal{O}$ is a local ring with maximal ideal $P = \mathcal{O}\backslash\mathcal{O}^*$.*

2. *If $f \in \mathbb{K}$ with $f \neq 0$, then $f \in P$ if and only if $f^{-1} \notin \mathcal{O}$.*

3. *$\tilde{\Bbbk} \subset \mathcal{O}$ and $\tilde{\Bbbk} \cap P = \{0\}$.*

4. *$\mathcal{O}$ is a principal ideal domain.*

5. *If we fix a generator t of P, i.e., $P = t\mathcal{O}$, then any non-zero element $f \in \mathbb{K}$ has a unique representation of the form $f = t^n u$, for some unique $n \in \mathbb{Z}$ and $u \in \mathcal{O}^*$.*

Definition 1.29 (Place)**.** A *place* P of a function field $\mathbb{K}/\Bbbk$ is the maximal ideal P of some valuation ring $\mathcal{O}$ of $\mathbb{K}/\Bbbk$. Any element $t \in P$ with $P = t\mathcal{O}$ is called a *local parameter* at P. The set of all places of a function field $\mathbb{K}/\Bbbk$ is denoted by $\mathbf{P}_{\mathbb{K}}$.
Since a valuation ring $\mathcal{O}$ is uniquely determined by its maximal ideal P by

$$\mathcal{O} = \{f \in \mathbb{K} \mid f^{-1} \notin P\},$$

we call $\mathcal{O} = \mathcal{O}_P$ the *valuation ring of the place* P.

Remark. If X is a curve over an algebraically closed field $\Bbbk$ and we consider as a function field $\mathbb{K}$ the field of rational functions $\Bbbk(X)$, then there is a bijective correspondence between the places $\mathbf{P}_{\mathbb{K}}$ and the set of points $\{P \in X\}$.

Definition 1.30 (Discrete Valuation)**.** A *discrete valuation* ν of $\mathbb{K}/\Bbbk$ is a function $\nu : \mathbb{K}^* \to \mathbb{Z}$ satisfying the properties

1. $\nu(fg) = \nu(f) + \nu(g)$ for all $f, g \in \mathbb{K}^*$,

2. $\nu(f + g) \geq \min\{\nu(f), \nu(g)\}$ for all $f, g \in \mathbb{K}^*$,

3. there exists an element f such that $\nu(f) = 1$,

4. $\nu(c) = 0$ for all $c \in \Bbbk \backslash \{0\}$.

Furthermore, we extend ν to the whole function field $\mathbb{K}$ by setting $\nu(0) = \infty$.

Let now $P \in \mathbf{P}_{\mathbb{K}}$ be a place. We know that for a choice of a local parameter t there is a unique representation $f = t^n u$ for any nonzero $f \in \mathbb{K}$. In fact, the exponent n only depends on P but not on the choice of the local parameter t. Thus, we define the function $\nu_P : \mathbb{K} \backslash \{0\} \to \mathbb{Z}$ sending an element f to its exponent n in this representation. This map ν_P is a discrete valuation and

$$\mathcal{O}_P = \{f \in \mathbb{K} \mid \nu_p(f) \geq 0\},$$

$$\mathcal{O}_P^* = \{f \in \mathbb{K} \mid \nu_p(f) = 0\},$$

$$P = \{f \in \mathbb{K} \mid \nu_p(f) > 0\}.$$

In the case where $\nu_P(f) > 0$, we say that P is a *zero* of f of order $\nu_P(f)$. Conversely, if $\nu_P(f) < 0$, we say that P is a *pole* of f of order $-\nu_P(f)$.

Definition 1.31 (Value at a Place)**.** Since P is maximal, we have that the ring $\mathcal{O}_P/P$ is in fact a field. This field is denoted by $\Bbbk_P = \mathcal{O}_P/P$ and is called the *residue class field* of P. For $f \in \mathcal{O}_P$, we define the map $\mathbb{K} \to \Bbbk_P \cup \{\infty\}$ sending $f \mapsto f(P)$ by

$$f(P) = \begin{cases} f + P \in \Bbbk_P & f \in \mathcal{O}_P \\ \infty & f \in \mathbb{K} \backslash \mathcal{O}_P \end{cases}.$$

The image $f(P)$ under this map is called the *value* of f. If $f \in \mathcal{O}_P$, it is said to be *regular*.

Definition 1.32 (Degree of a Place)**.** By proposition 1.28, we know that $\Bbbk \subset \mathcal{O}_P$ and $\Bbbk \cap P = \{0\}$. From this, we see that the canonical residue class map $\mathcal{O}_P \to \mathcal{O}_P/P$ induces a canonical embedding $\Bbbk \hookrightarrow \Bbbk_P$. Thus, we can consider $\Bbbk$ as a subfield of $\Bbbk_P$. For any place P, we define its *degree* as $\deg P = [\Bbbk_P : \Bbbk]$.

Proposition 1.33 ([28, Proposition 2.5.7])**.** *Let P be a place of a function field $\mathbb{K}/\Bbbk$ and x be a nonzero element of P. Then*

$$\deg P \leq [\mathbb{K} : \Bbbk(x)].$$

Theorem 1.34 (Weak Approximation Theorem, [26, Theorem 1.3.1])**.** *Let $\mathbb{K}/\Bbbk$ be a function field and $P_1, \ldots, P_n \in \mathbf{P}_\mathbb{K}$ pairwise distinct places. Furthermore, let $x_1, \ldots, x_n \in \mathbb{K}$ and $r_1, \ldots, r_n \in \mathbb{Z}$. Then there exists some $f \in \mathbb{K}$ such that*

$$\nu_{P_i}(f - x_i) = r_i \quad \textit{for all} \quad i = 1, \ldots, n.$$

The proof and further background to this theorem can be found in Stichtenoth [26, Theorem 1.3.1].

Definition 1.35 (Divisors)**.** Similarly as for curves over algebraically closed fields, we can define divisors on a function field, but this time we take places instead of points. Namely, the *divisor group* $\mathrm{Div}(\mathbb{K})$ of the function field $\mathbb{K}/\Bbbk$ is the free abelian group generated by the places of $\mathbb{K}/\Bbbk$. This means that its elements are finite formal sums

$$D = \sum_{P \in \mathbf{P}_\mathbb{K}} a_P P, \quad a_P \in \mathbb{Z}.$$

In particular, almost all a_P are equal to zero. A divisor of the form $D = P$ with $P \in \mathbf{P}_\mathbb{K}$ is called *prime divisor*. For a divisor $D = \sum_{P \in \mathbf{P}_\mathbb{K}} a_P P$, we define its *degree* as

$$\deg D = \sum_{P \in \mathbf{P}_\mathbb{K}} a_P \deg P.$$

Similarly as before, we want to associate a divisor to an element $f \in \mathbb{K}$. The element f has a finite set of zeros Z and a finite set of poles N. For f, we define its *divisor of zeroes*

$$(f)_0 = \sum_{P \in Z} \nu_P(f) P,$$

its *divisor of poles*

$$(f)_\infty = \sum_{P \in N} (-\nu_P(f))P$$

and the *principal divisor* $(f) = (f)_0 - (f)_\infty$.

Theorem 1.36 ([28, Theorem 2.5.20])**.** *Let (f) be a principal divisor. Then* $\deg(f) = 0$. *Moreover, if $f \in \mathbb{K} \backslash \tilde{\Bbbk}$, then*

$$\deg(f)_0 = \deg(f)_\infty = [\mathbb{K} : \Bbbk(f)].$$

Definition 1.37 (Galois Group)**.** Let $\mathbb{F}$ be a field and $\mathbb{F}'$ a separable field extension of $\mathbb{F}$. The *Galois group* $\mathrm{Gal}(\mathbb{F}'/\mathbb{F})$ is the set of all automorphisms $\varphi : \mathbb{F}' \to \mathbb{F}'$ fixing all elements of the subfield $\mathbb{F}$, i.e.,

$$\mathrm{Gal}(\mathbb{F}'/\mathbb{F}) = \{\varphi \in \mathrm{Aut}(\mathbb{F}') \mid \varphi(x) = x \text{ for all } x \in \mathbb{F}\}.$$

For a field $\Bbbk$, we use the notation $\mathrm{Gal}(\Bbbk) = \mathrm{Gal}(\overline{\Bbbk}/\Bbbk) = \mathrm{Gal}(\Bbbk^s/\Bbbk)$, where $\Bbbk^s$ denotes the seperable closure of $\Bbbk$.

Remark. If $\Bbbk = \mathbb{F}_q$ is a finite field, then we have

$$\overline{\mathbb{F}}_q = \mathbb{F}_q^s = \bigcup_{m=1}^{\infty} \mathbb{F}_{q^m}.$$

We extend the definition of a variety in the following way. A *projective variety X over the field* $\Bbbk$ is a variety $\overline{X} \subset \mathbb{P}^m(\overline{\Bbbk})$ that can be defined by a system of equations with coefficients in the field $\Bbbk$ and similarly X is an *affine variety* if $\overline{X} \subset \mathbb{A}^m(\overline{\Bbbk})$ with the same condition on the equations. Analogously, we can define quasi-projective sets and quasi-projective varieties.
Now let $X/\Bbbk$ be a quasi-projective variety and consider a field $\Bbbk'$ such that $\Bbbk \subset \Bbbk' \subset \overline{\Bbbk}$. By $X(\Bbbk')$, we denote the set of solutions of the system corresponding to X with coordinates in $\Bbbk'$ if X is an affine variety. However, in the projective space, scaling is allowed and thus for projective varieties we need to require a representative of a solution to have coordinates in $\Bbbk'$. With this definition, we have that $\overline{X} = X(\overline{\Bbbk})$. The Galois group $\mathrm{Gal}(\overline{\Bbbk}/\Bbbk')$ acts coordinate-wise on elements $X(\overline{\Bbbk})$ and

$$X(\Bbbk') = \{x \in X(\overline{\Bbbk}) \mid \sigma(x) = x \text{ for all } \sigma \in \mathrm{Gal}(\overline{\Bbbk}/\Bbbk')\}.$$

Example 6. Consider the finite field $\Bbbk = \mathbb{F}_q$ with subfield $\Bbbk' = \mathbb{F}_r$ and some projective variety $\overline{X} \subset \mathbb{P}^m(\overline{\mathbb{F}}_q)$ and let $x = (x_0 : x_1 : \ldots : x_m) \in \overline{X}$ and $\sigma \in \mathrm{Gal}(\overline{\mathbb{F}}_q/\mathbb{F}_r)$. If x has a representation $x = \lambda x'$ such that x' has coordinates in $\mathbb{F}_r$, then it is clear that

$$\sigma(x) = \sigma(\lambda x') = \sigma(\lambda)^{m+1}\sigma(x') = \sigma(\lambda)^{m+1}x' = \sigma(\lambda)^{m+1}x = x$$

by the definition of the Galois group. On the other hand, consider $x = (x_0 : x_1 : \ldots : x_m) \in \mathbb{P}^m$. We can scale one of its coordinates to be 1 and for simplicity

assume that $x_0 = 1$. It is clear that $\sigma(x_0) = \sigma(1) = 1$ for all $\sigma \in \mathrm{Gal}(\overline{\mathbb{F}}_q/\mathbb{F}_r)$ and hence $\sigma(x) = x$ implies that $\sigma(x_i) = x_i$ for all $1 \leq i \leq m$ when choosing $x_0 = 1$. Now consider the r-Frobenius automorhpism

$$\varphi_r : \overline{\mathbb{F}}_q \to \overline{\mathbb{F}}_q$$

$$a \mapsto \varphi_r(a) = a^r.$$

Then $\varphi_r(a) = a$ if and only if $a \in \mathbb{F}_r$. Therefore, all coordinates of the chosen representative $x = (1 : x_1 : \ldots : x_m)$ must be in $\mathbb{F}_r$.

We define $\dim X = \dim \overline{X}$ and say that X is *smooth* if $\overline{X}$ is smooth. Furthermore, X is called *absolutely irreducible* if $\overline{X}$ is irreducible.

Remark. In the following, we will always assume our varieties and especially our curves to be absolutely irreducible.

Following this definition, we have that a variety X over the field $\Bbbk$ is a set $X(\overline{\Bbbk})$ of its $\overline{\Bbbk}$-points equipped with an action of the Galois group $\mathrm{Gal}(\Bbbk)$ and, similarly as before in the case of algebraically closed fields, we also have the Zariski topology. Next, we want to define points of X. A *point* P of a variety X is an orbit of the Galois group acting on $X(\overline{\Bbbk})$. Namely, a point is a set

$$P = \{\sigma\overline{P} \mid \sigma \in \mathrm{Gal}(\Bbbk)\},$$

where $\overline{P} \in X(\overline{\Bbbk})$ is a $\overline{\Bbbk}$-point.

Let X and Y be two varieties over $\Bbbk$ and $f : X(\overline{\Bbbk}) \to Y(\overline{\Bbbk})$ a map. We define the action of the Galois group $\mathrm{Gal}(\Bbbk)$ as

$$(\sigma f)(x) = \sigma(f(\sigma^{-1}x)).$$

Any rational function f is a rational map $f : X \to \mathbb{P}^1$ and thus we also have the action of the Galois group on the field of rational functions $\overline{\Bbbk}(\overline{X})$. We set $\mathbb{K} = \Bbbk(X)$ to be the set stable under the action of the Galois group and we call it the *field of rational functions* of the variety X.
The points of a curve X and places of the function field are in bijective correspondence. Namely, $P \in X$ corresponds to a maximal ideal $\mathfrak{m}_{X,P}$ of the local ring $\mathcal{O}_{X,P}$. This is the ideal consisting of elements of $\mathbb{K}$ vanishing at the point P. In particular, these are the functions on $\overline{X}$ invariant under the Galois group vanishing at one element, and thus at all elements, in the orbit P. Therefore, we can transfer between curves and function fields, which allows us to define different notions such as genus, divisor, principal divisor or the order of a rational function at a point.

Remark. If we use the above relation to define the genus, it turns out that it is the same as $g(X) = g(\overline{X})$ and Theorem 1.23 still holds.

1.3 Algebraic Geometry Codes and the McEliece Encryption Scheme

Definition 1.38 (Algebraic Geometry Code, [28, Chapter 4.1.1])**.** Let X be a smooth projective curve of genus g over the finite field $\mathbb{F}_q$. Let $\mathbb{F}_q(X)$ be the function field of X and for an $\mathbb{F}_q$-divisor E denote its Riemann-Roch space by $L(E) = \{f \in \mathbb{F}_q(X) \mid f = 0 \text{ or } (f) + E \geq 0\}$. For a given $\mathcal{P} = (P_1, \ldots, P_n)$ of pairwise distinct $\mathbb{F}_q$-rational points of X, we denote by $\mathcal{D}_\mathcal{P}$ the divisor $\mathcal{D}_\mathcal{P} = P_1 + \ldots + P_n$. For a divisor E with support disjoint from $\mathcal{D}_\mathcal{P}$, we define the *algebraic geometry code (AG-code)* as

$$C = C_L(X, \mathcal{P}, E) = \{(f(P_1), \ldots, f(P_n)) \mid f \in L(E)\}.$$

Remark. This definition also works more generally for any smooth projective variety, but then it is much harder to obtain good bounds.

Theorem 1.39 ([28, Theorem 4.1.1])**.** *Let X be a curve of genus g and $0 \leq \deg E = m < n$ and C an algebraic geometry code as defined above. Then the following hold:*

1. *If any $0 \neq f \in L(E)$ has at most n_0 zeros at $\mathbb{F}_q$-points and $n > n_0$, then the evaluation map $\mathrm{Ev}_\mathcal{P} : L(E) \to \mathbb{F}_q^n,\ f \mapsto (f(P_1), \ldots, f(P_n))$ is an embedding and $k = \dim L(E)$ and $d \geq n - n_0$.*
2. *C is a linear code with $k \geq m - g + 1$ and $d \geq n - m$.*

Proof. 1. We evaluate f at n distinct points and f has at most $n_0 < n$ zeros. Thus, $\mathrm{Ev}_\mathcal{P}(f) \neq 0$ and it is nonzero in at least $n - n_0$ coordinates, implying that $d \geq n - n_0$. This also implies that $\ker(\mathrm{Ev}_\mathcal{P}) = \{0\}$ and hence it is an embedding and $k = \dim L(E)$ by construction.

2. Let us write the divisor E as $E = E_1 - E_2$, where $E_1, E_2 \geq 0$. We take a nonzero function $f \in L(E)$ and since

$$0 \leq E + (f) = E + (f)_0 - (f)_\infty = E_1 - (f)_\infty + (f)_0 - E_2,$$

we have that f has at most $m_\infty = \deg E_1$ poles and at least $m_0 = \deg E_2$ zeros lying in the support of E. As $(f) + E \geq 0$, f cannot have any poles outside $\mathrm{supp}(E)$. However, it can have zeros, namely at most

$$m_\infty - m_0 = \deg E_1 - \deg E_2 = m$$

because $\deg(f)_0 = \deg(f)_\infty$. Since we assumed $m < n$, $\mathrm{Ev}_\mathcal{P}(f) \neq 0$ for any nonzero $f \in L(E)$ and more precisely $\mathrm{Ev}_\mathcal{P}(f)$ has at least $n - m$ nonzero coordinates and thus $d \geq n - m$. For the other inequality, we note that by the same reasoning as above $\mathrm{Ev}_\mathcal{P}$ is an embedding, i.e., we have

$$C = \mathrm{Ev}_\mathcal{P}(L(E)) \cong L(E)$$

and hence $k = \dim L(E) \geq m - g + 1$ by Theorem 1.24.

□

Remark. The parameters $d_C = n-m$ and $k_C = m-g-1$ are called the *designed distance* and *designed dimension* of the code C.

Corollary 1.40. *If $2g-2 < m < n$, then*

$$k = m - g + 1.$$

Remark. For two divisors E_1 and E_2 with $E_2 - E_1 \geq 0$, the codes $C_1 = C_L(X, \mathcal{P}, E_1)$ and $C_2 = C_L(X, \mathcal{P}, E_2)$ satisfy $C_1 \subset C_2$ for any $\mathcal{P}$ with disjoint support from E_1.

Theorem 1.41 ([8, Theorem 5])**.** *Let K be a canonical divisor of X, $D_{\mathcal{P}} = P_1 + \ldots + P_n$ and define $E^{\perp} = D_{\mathcal{P}} - E + K$. Then*

$$C_L(X, \mathcal{P}, E)^{\perp} = C_L(X, \mathcal{P}, E^{\perp})$$

and $\deg(E^{\perp}) = n - \deg(E) + 2g - 2$. In particular, the code $C^{\perp}$ is still an algebraic geometry code on the same curve defined using the same points $\mathcal{P}$ and there is an explicit description for its corresponding divisor $E^{\perp}$.

The McEliece encryption scheme: Let C be a linear code with error correcting capacity t that has an efficient decoding algorithm $\mathcal{A}_C$ and let G be a non-structured generator matrix of C. Then the public key and private key are $\mathcal{K}_{pub} = (G, t)$ and $\mathcal{K}_{secret} = (\mathcal{A}_C)$. The encryption of a message $\mathbf{m}$ is $\mathbf{y} = \mathbf{m}G + \mathbf{e}$, where $\mathbf{e}$ is a random error vector of weight t. Using the secret decoding algorithm $\mathcal{A}_C$, the receiver can correct t errors and obtains the message $\mathbf{m}$.
In general, this means that one has a linear code with length n, dimension k and error correcting capacity t over some finite field $\mathbb{F}_q$, where some structured generator matrix G is private. This generator matrix G makes decoding easy and thus the generator matrix that is made public is a different one. Namely, we take $B \in GL_k(\mathbb{F}_q)$ and an $n \times n$ permutation matrix P. Then the matrix $\tilde{G} = BGP$ is part of the public key. As explained before, we note that G and BG generate the exact same code, whereas multiplying with the permutation matrix P gives an equivalent code. We will see later that this does not affect the attack provided in Couvreur, Márquez-Corbella, and Pellikaan [8].
In this case, the McEliece encryption scheme works as follows. If we want to send the message $\mathbf{m}$, we encode it using the public key $\tilde{G} = BGP$ and add an error vector $\mathbf{e}$ of weight t and then we send the encrypted message

$$\mathbf{y} = \mathbf{m}\tilde{G} + \mathbf{e} = \mathbf{m}BGP + \mathbf{e}.$$

The receiver then uses the secret key to compute

$$\mathbf{y}' = \mathbf{y}P^{-1} = \mathbf{m}BG + \mathbf{e}P^{-1}.$$

As P^{-1} does not change the weight of the error $\mathbf{e}$, the receiver can decode the message $\mathbf{m}' = \mathbf{m}B$. Now, they find $\mathbf{m} = \mathbf{m}'B^{-1}$ and they successfully recovered the message $\mathbf{m}$.

Remark. For algebraic geometry codes, there exists an efficient decoding algorithm using error correcting pairs, which is explained in Section 3.2.

In the case of algebraic geometry codes, we are in the following setting. We assume that the divisor E has degree $m > 3g$. This guarantees that $t > 0$. We consider the code $C^\perp = C_L(X, \mathcal{P}, E)^\perp$ with generator matrix G and $t = \lfloor \frac{d^*-g-1}{2} \rfloor$, where $d^* = m - 2g + 2$ is the designed distance of $C^\perp$ by Theorem 1.41.

Remark. If G is a generator matrix of the code $C^\perp$, then G is a parity check matrix of the code C. This means in particular that if a generator matrix of $C^\perp$ is public, we also know the code C.

2 Distinguishers and Distinguisher based Attacks

In this section we establish additional background that we need in Section 3. Our main source here is Couvreur, Márquez-Corbella, and Pellikaan [8], but we also use results from Couvreur et al. [6] and Pellikaan et al. [21].

2.1 Distinguishers

Definition 2.1 (Schur Product)**.** The *Schur product*, sometimes also called the star product, is the coordinate-wise product of two vectors

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

for $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$. For subsets A and B of $\mathbb{F}_q^n$, we define

$$A * B = \langle \mathbf{a} * \mathbf{b} \ : \ \mathbf{a} \in A, \mathbf{b} \in B \rangle.$$

If $B = A$, we write $A^{(2)} = A * A$. If A is a linear code, $A^{(2)}$ is called the *square code* of A.

Proposition 2.2 ([6, Proposition 4])**.** *Let A, B be two linear codes over $\mathbb{F}_q^n$. Then the following statements hold:*

1. $\dim(A * B) \leq \dim(A) \dim(B)$.
2. $\dim(A^{(2)}) \leq \binom{\dim A + 1}{2}$.

Proof. Both statements follow directly from the fact that if $\mathbf{a}_i$ and $\mathbf{b}_j$ form a basis for A, B respectively, then $\mathbf{a}_i * \mathbf{b}_j$ generates the code $A * B$. □

Lemma 2.3 ([8, Lemma 7])**.** *Let A, B be two codes over $\mathbb{F}_q^n$. Then*

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

Proof.

$$\begin{aligned}\mathbf{z} * A \subset B &\Leftrightarrow (\mathbf{z} * \mathbf{a}) \cdot \mathbf{b} = 0 \quad \text{for all} \quad \mathbf{a} \in A, \mathbf{b} \in B^{\perp} \\ &\Leftrightarrow \mathbf{z} \cdot (\mathbf{a} * \mathbf{b}) = 0 \quad \text{for all} \quad \mathbf{a} \in A, \mathbf{b} \in B^{\perp} \\ &\Leftrightarrow \mathbf{z} \in (A * B^{\perp})^{\perp}.\end{aligned}$$

□

Definition 2.4 (GRS-Code)**.** Let q be the power of a prime number and $k, n \in \mathbb{N}$ such that $1 \leq k < n \leq q$. Let $\mathbf{x} \in \mathbb{F}_q^n$ be a tuple of n distinct elements and $\mathbf{y} \in (\mathbb{F}_q \backslash \{0\})^n$ arbitrary. Then we define the *generalized Reed-Solomon code (GRS-code)* of dimension k as

$$\mathbf{GRS}_k(\mathbf{x}, \mathbf{y}) = \{(y_1 p(x_1), \ldots, y_n p(x_n)) \mid p \in \mathbb{F}_q[X], \deg(p) < k\}.$$

As the name suggests, GRS-codes generalize Reed-Solomon codes. In particular, if we choose $y_i = 1$ for all i, i.e., $\mathbf{y}$ is the one vector, then we obtain a Reed-Solomon code.

Remark. From the definition of GRS-codes, we immediately observe that

$$\{(y_1 x_1^i, \ldots, y_n x_n^i) \mid 0 \leq i \leq k-1\}$$

is a basis of the GRS-code $\mathbf{GRS}_k(\mathbf{x}, \mathbf{y})$.

Example 7 ([28, Example 4.1.5])*.* Consider a Reed-Solomon code of dimension k and length $n = q$, i.e., $\mathbf{x}$ consists of all elements in $\mathbb{F}_q$ in some order. Now, we consider the projective line $\mathbb{P}^1$ and we take as a divisor $E = (k-1)P_\infty$ and as $\mathcal{P}$ we take all the affine $\mathbb{F}_q$-points. Then we have that the code

$$C_L(\mathbb{P}^1, \mathcal{P}, (k-1)P_\infty)$$

is permutation equivalent to the code $\mathbf{GRS}_k(\mathbf{x}, 1)$. Namely, if we choose $\mathcal{P}$ in the correct order, then the codes are equal.

Remark. We note that by construction any GRS-code is monomially equivalent to some RS-code. Hence, it is also monomially equivalent to an AG-code.

Proposition 2.5 ([6, Proposition 6])**.** *For $k \leq \frac{n+1}{2}$ we have $\mathbf{GRS}_k(\mathbf{x}, \mathbf{y})^2 = \mathbf{GRS}_{2k-1}(\mathbf{x}, \mathbf{y} * \mathbf{y})$.*

Proof. The proposition is a direct consequence of the definition of GRS-codes as evaluation codes. Consider two codewords $\mathbf{c} = (y_1 p(x_1), \ldots, y_n p(x_n))$ and $\mathbf{c}' = (y_1 q(x_1), \ldots, y_n q(x_n))$ of $\mathbf{GRS}_k(\mathbf{x}, \mathbf{y})$. Then we have

$$\mathbf{c} * \mathbf{c}' = (y_1^2 p(x_1) q(x_1), \ldots, y_n^2 p(x_n) q(x_n)) = (y_1^2 r(x_1), \ldots, y_n^2 r(x_n)),$$

where $r = pq$ is a polynomial of degree at most $2k-2$. On the other hand, if we have an element $(y_1^2 r(x_1), \ldots, y_n^2 r(x_n))$ with $\deg r \leq 2k-2$, then this is a linear combination of Schur products of elements in $\mathbf{GRS}_k(\mathbf{x}, \mathbf{y})$. Indeed, write

$$r(X) = a_0 + a_1 X + \ldots + a_{2k-2} X^{2k-2}.$$

Then we can split r in the following way:

$$r_1(X) = a_0 + a_1X + \ldots + a_{k-1}X^{k-1},$$

$$r_2(X) = h_2(X) \cdot g_2(X) = (a_kX^{k-1}) \cdot X,$$
$$r_3(X) = h_3(X) \cdot g_3(X) = (a_{k+1}X^{k-1}) \cdot X^2,$$

$$\vdots$$

$$r_{k-1}(X) = h_{k-1}(X) \cdot g_{k-1}(X) = (a_{2k-1}X^{k-1}) \cdot X^{k-2},$$
$$r_k(X) = H_k(X) \cdot g_k(X) = (a_{2k-2}X^{k-1}) \cdot X^{k-1} = (a_{2k-2}+1)X^{k-1} =: h_k(X).$$

It is clear that

$$r(X) = \sum_{i=1}^{k} r_i(X).$$

We also note that for elements of the form $(y_1^2 r_i(x_1), \ldots, y_n^2 r_i(x_n))$ we have

$$(y_1^2 r_i(x_1), \ldots, y_n^2 r_i(x_n)) = (y_1 h_i(x_1), \ldots, y_n h_i(x_n)) * (y_1 g_i(x_1), \ldots, y_n g_i(x_n)),$$

where we set $g_1 = g_k = 1$. Hence, it follows that

$$(y_1^2 r(x_1), \ldots, y_n^2 r(x_n)) = \sum_{i=1}^{k} (y_1 h_i(x_1), \ldots, y_n h_i(x_n)) * (y_1 g_i(x_1), \ldots, y_n g_i(x_n))$$

and thus it is an element of $\mathbf{GRS}_k(\mathbf{x}, \mathbf{y})^2$. □

Corollary 2.6. *If $k \leq \frac{n+1}{2}$ and C is a GRS-code of dimension k, then*

$$\dim C^2 = 2k - 1.$$

This property distinguishes GRS-codes from random codes and hence it is called a *distinguisher*. Indeed, for a random linear code C' of dimension k, one expects that the dimension of its square code C'^2 is of order $\min\{\binom{k+1}{2}, n\}$ [see 6]. In practice, one can often use a distinguisher to determine to which family a certain code C belongs and then attack the McEliece encryption scheme using a known decoding algorithm for codes belonging to that family.

2.2 The s-Closure, Subcodes and Subfield Subcodes

2.2.1 The s-Closure

Definition 2.7 (Product)**.** Let $\mathbb{K}$ be a field and $\mathbb{A}$ be a commutative unitary algebra over $\mathbb{K}$. For two subspaces $V, W \subset \mathbb{A}$, we define the *product* of V and W as

$$V \cdot W = \mathrm{span}_{\mathbb{K}}(vw : v \in V, w \in W).$$

We set $L^{(1)} = L$ and we define inductively $L^{(t+1)} = L^t \cdot L$.

Theorem 2.8 ([18, Theorem 6]). *Let E, F be two divisors on the curve X with $\deg(E) \geq 2g+1$ and $\deg(F) \geq 2g$ and let t be any positive integer. Then the following hold:*

1. $L(E) \cdot L(F) = L(E+F)$.
2. $L(E)^t = L(tE)$.

Corollary 2.9 ([8, Corollary 9]). *Let E, F be as above and such that they both have disjoint support from $\mathcal{P}$ and let t be a positive integer. Then the following hold:*

1. $C_L(X, \mathcal{P}, E) * C_L(X, \mathcal{P}, F) = C_L(X, \mathcal{P}, E+F)$.
2. $C_L(X, \mathcal{P}, E)^{(t)} = C_L(X, \mathcal{P}, tE)$.

Proof. The proof follows directly from Theorem 2.8 and the fact that the evaluation map is a morphism of algebras. □

Proposition 2.10 ([8, Proposition 10]). *Let E,F be two divisors on X with disjoint support from $\mathcal{P}$ such that $\deg F \geq 2g$ and $\deg E \leq n-3$. Then*

$$\begin{aligned} C_L(X, \mathcal{P}, E-F) &= (C_L(X, \mathcal{P}, F) * C_L(X, \mathcal{P}, E)^{\perp})^{\perp} \\ &= \{\mathbf{z} \in \mathbb{F}_q^n \mid \mathbf{z} * C_L(X, \mathcal{P}, F) \subset C_L(X, \mathcal{P}, E)\}. \end{aligned}$$

Proof. We set $A = C_L(X, \mathcal{P}, F)$ and $B = C_L(X, \mathcal{P}, E)$. It follows from Theorem 1.41 that $B^{\perp} = C_L(X, \mathcal{P}, E^{\perp})$ and since $\deg E \leq n-3$, we have $\deg E^{\perp} \geq 2g+1$. Using again Theorem 1.41 and Corollary 2.9, we obtain

$$\begin{aligned} A * B^{\perp} &= C_L(X, \mathcal{P}, E^{\perp} + F) \\ &= C_L(X, \mathcal{P}, D_{\mathcal{P}} + K - E + F) \\ &= C_L(X, \mathcal{P}, (E-F)^{\perp}) \\ &= C_L(X, \mathcal{P}, E-F)^{\perp}. \end{aligned}$$

The second equality follows directly from Lemma 2.3. □

Definition 2.11 (s-Closure, [8, Definition 11]). Let $\mathbb{K}$ be a field and $\mathbb{A}$ be a commutative unitary algebra over $\mathbb{K}$. For a subspace $L \subset \mathbb{A}$ and an integer $s \geq 2$, we define the *s-closure* of L as

$$\overline{L}^s = \{f \in \mathbb{A} \mid f \cdot L^{(s-1)} \subset L^{(s)}\}.$$

If $\overline{L}^s = L$ we say that L is *s-closed*.

Theorem 2.12 (Properties of the s-Closure, [8, Proposition 13]). *Let $L \subset \mathbb{A}$ be a subspace of $\mathbb{A}$ and let $s \geq 2$ be an integer. Then the following properties hold:*

(1) $\overline{L}^s$ is a vector space over $\mathbb{K}$.

(2) $L \subset \overline{L}^2 \subset \ldots \subset \overline{L}^s \subset \overline{L}^{s+1}$ is an increasing sequence.

(3) *For $f \in \mathbb{A}$, it holds that $f \cdot \overline{L}^s \subset \overline{(f \cdot L)}^s$ and equality holds if f is invertible in $\mathbb{A}$.*

(4) *Let $\phi : \mathbb{A} \to \mathbb{K}$ be a linear map such that the symmetric bilinear form φ defined by $\varphi(a,b) = \phi(ab)$ is non degenerate, i.e., $\varphi(a,b) = 0$ for all $b \in \mathbb{A}$ if and only if $a = 0$. Then*

$$\overline{L}^s = (L^{(s-1)} \cdot L^{(s)\perp_\varphi})^{\perp_\varphi},$$

where $L^{\perp_\varphi} = \{a \in \mathbb{A} \mid \varphi(a,b) = 0 \text{ for all } b \in L\}$.

Proof. Statement (1) is clear because the product is $\mathbb{K}$-bilinear. For (2), consider $f \in \overline{L}^s$. Then

$$f \cdot L^{(s-1)} \subset L^{(s)} \quad \text{implies} \quad f \cdot L^{(s)} \subset L^{(s+1)}$$

and therefore we also have $f \in \overline{L}^{s+1}$. To prove (3), let $g \in f \cdot \overline{L}^s$. This means that $g = fa$ for some element $a \in \overline{L}^s$ and we get that

$$g \cdot (f \cdot L)^{(s-1)} = f^s \cdot a \cdot L^{(s-1)} \subset f^s \cdot L^{(s)} = (f \cdot L)^{(s)}$$

and hence $g \in \overline{f \cdot L}^s$. If now f is also invertible in $\mathbb{A}$, then one can reverse the proof to obtain the equality. To prove (4), we use the adjunction formula

$$\varphi(fg, h) = \phi(fgh) = \varphi(f, gh),$$

which holds for all $f, g, h \in \mathbb{A}$ and then we proceed as in Lemma 2.3. □

Remark. Statement (4) in Theorem 2.12 holds in particular if $\mathbb{A} = \mathbb{F}_q^n$ and φ is the standard inner product.

Lemma 2.13 ([8, Lemma 15])**.** *Let $\mathbb{A}$ be the field of rational functions on the curve X and consider $M = L(E)$ for some divisor E on X and let $s \geq 2$ be an integer. If $L \subset M$ is a subspace, $L^{(s)} = M^{(s)}$ and $\overline{M}^s = M$, then*

$$\overline{L}^s = M.$$

Proof. Assume that $L \subset L(E)$ and let $E_0 = \sum a_i P_i$ be a divisor such that $L \subset L(E_0) \subset L(E)$ and such that E_0 is minimal for this property. Then for all i there exists $g_i \in L$ such that $\nu_{P_i}(g_i) = -a_i$ by Theorem 1.34. We take any $f \in \overline{L}^s$. Then by construction we have

$$f \cdot g_i^{s-1} \in f \cdot L^{(s-1)} \subset L^{(s)} \subset L(E_0)^{(s)} \subset L(sE_0).$$

Thus, $\nu_{P_i}(f g_i^{s-1}) = \nu_{P_i}(f) - (s-1)a_i \geq -sa_i$ and equivalently $\nu_{P_i}(f) \geq -a_i$ for all i. Therefore, $(f) \geq -E_0$ and hence $f \in L(E_0) \subset L(E) = M$ and it follows that $\overline{L}^s \subset M$. For the other direction, take $f \in M$. Then we have $f \cdot L^{(s-1)} \subset f \cdot M^{(s-1)} \subset M^{(s)} = L^{(s)}$ and thus $f \in \overline{L}^s$. □

Lemma 2.14 ([8, Proposition 18])**.** *Let E be a divisor on the curve X and let $s \geq 2$ be an integer such that $2g+1 \leq \deg(E) \leq \frac{n-2}{s}$. Then the code $C_L(X, \mathcal{P}, E)$ is s-closed.*

Proof. By Theorem 2.12 we have

$$\overline{C_L(X, \mathcal{P}, E)}^s = (C_L(X, \mathcal{P}, E)^{(s-1)} * (C_L(X, \mathcal{P}, E)^{(s)})^\perp)^\perp.$$

From Corollary 2.9 and Theorem 1.41, we obtain that

$$(C_L(X, \mathcal{P}, E)^{(s)})^\perp = C_L(X, \mathcal{P}, sE)^\perp = C_L(X, \mathcal{P}, (sE)^\perp),$$

where $(sE)^\perp = D_\mathcal{P} - sE + K$ for some canonical divisor K. We assumed that $\deg E \leq \frac{n-2}{s}$ and hence

$$\deg((sE)^\perp) = n - \deg(sE) + 2g - 2 \geq 2g$$

and as $\deg E \geq 2g+1$ we can once again apply Corollary 2.9 and obtain

$$C_L(X, \mathcal{P}, (s-1)E) * C_L(X, \mathcal{P}, sE)^\perp = C_L(X, \mathcal{P}, D_\mathcal{P} - E + K) = C_L(X, \mathcal{P}, E)^\perp.$$

□

2.2.2 Subcodes and Subfield Subcodes

In this subsection we examine the behaviour of subcodes and subfield subcodes of algebraic geometry codes and their differences. To do so, we consider the following conjecture.

Conjecture 1 ([8, Conjecture 19])**.** Let C be a uniformly random subcode of $C_L(X, \mathcal{P}, E)$ of dimension l such that

$$2g+1 \leq \deg(E) \leq \frac{n-1}{2} \quad \text{and} \quad 2k+1-g \leq \binom{l+1}{2},$$

where $k = \deg(E)+1-g$ is the dimension of $C_L(X, \mathcal{P}, E)$. Then the probability that $C^{(2)}$ is different from $C_L(X, \mathcal{P}, 2E)$ tends to 0 as k tends to infinity. In other words, this means that with a high probability $C^{(2)} = C_L(X, \mathcal{P}, 2E)$.

Corollary 2.15 ([8, Corollary 20])**.** *Assume that Conjecture 1 holds. Let $2g+1 \leq \deg(E) \leq \frac{n-2}{2}$. Let $k = \deg(E)+1-g$ be the dimension of $C_L(X, \mathcal{P}, E)$ such that $2k+1-g \leq \binom{l+1}{2}$ for some l. Then the equality $\overline{C}^{(2)} = C_L(X, \mathcal{P}, E)$ holds for uniformly random subcodes C of $C_L(X, \mathcal{P}, E)$ with a probability tending to 1 as k tends to infinity.*

Proof. Recall that we assume the conjecture to hold. Let C be an l-dimensional subcode of $C_L(X, \mathcal{P}, E)$. The conjecture implies that $C^{(2)} = C_L(X, \mathcal{P}, E)^{(2)} = C_L(X, \mathcal{P}, 2E)$ with high probability. We also have that $C_L(X, \mathcal{P}, E)$ is 2-closed by Lemma 2.14. Thus, the proof follows from Lemma 2.13. □

Suppose now that we have a subcode C of the code $C_L(X, \mathcal{P}, E)^\perp$ of dimension l. We consider a non structured generator matrix $\mathbf{G}$ of C to be our public key together with the error correcting capacity t. The idea of the attack is to recover $C_L(X, \mathcal{P}, E)^\perp$ from C using Corollary 2.15 and then apply one of the attacks based on error correcting pairs (ECP) or error correcting arrays (ECA) explained in Couvreur, Márquez-Corbella, and Pellikaan [8]. More details about ECPs are covered in Section 3. In particular, let $k = \dim C_L(X, \mathcal{P}, E)^\perp$ and assume that

$$2g + 1 \leq \deg(E) \leq \frac{n-1}{2} \quad \text{and} \quad 2k - 1 + g \leq \binom{l+1}{2}.$$

Moreover, we assume Conjecture 1 to be true. This means that with a high probability we may assume that $C^{(2)} = C_L(X, \mathcal{P}, 2E^\perp)$ and hence $\overline{C}^2 = C_L(X, \mathcal{P}, E)^\perp$. Therefore, we can proceed as follows:

1. Compute $\overline{C}^2$.
2. Use one of the algorithms to find an ECP or ECA for $C_L(X, \mathcal{P}, E)^\perp$, which then yields a decoding algorithm for C.

Remark. Assume $\mathbb{F}_q = \mathbb{F}_{p^m}$ to be a non-prime field and let $\mathbb{F}$ be a proper subfield of $\mathbb{F}_q$. We consider the subfield subcode $C = C_L(X, \mathcal{P}, E) \cap \mathbb{F}^n$. The point is that $C^{(2)} \subset C_L(X, \mathcal{P}, E)^{(2)} \cap \mathbb{F}^n$ and the 2-closure of C in general differs from $C_L(X, \mathcal{P}, E)$. In particular, this means that we cannot apply the same trick as before and thus subfield subcodes resist to this attack.

3 Error Correcting Pairs and attacking AG-Codes

In this section we explain the attack from Couvreur, Márquez-Corbella, and Pellikaan [7] closely following it in Section 3.2, but we also use results from the more general version Couvreur, Márquez-Corbella, and Pellikaan [8] and use Pellikaan et al. [21] to provide additional background about ECP.

3.1 Error Correcting Pairs

Recall the definition of the Schur product:

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

for $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$. For subsets A and B of $\mathbb{F}_q^n$ we define

$$A * B = \langle \mathbf{a} * \mathbf{b} \ : \ \mathbf{a} \in A, \mathbf{b} \in B \rangle.$$

If $B = A$, we call $A^{(2)} = A * A$ the *square* of A.

Definition 3.1 (ECP, [21, Definition 6.2.7])**.** Let C be a linear code in $\mathbb{F}_q^n$. A pair of codes (A, B) over $\mathbb{F}_q$ of length n is called a *t-error correcting pair (ECP)* for C if:

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

Definition 3.2 (Kernel)**.** Let A, B be linear subspaces of $\mathbb{F}_q^n$ and $\mathbf{r} \in \mathbb{F}_q^n$. We define the *kernel* of $\mathbf{r}$ as

$$K(\mathbf{r}) = \{\mathbf{a} \in A \mid (\mathbf{a} * \mathbf{b}) \cdot \mathbf{r} = 0 \quad \text{for all} \quad \mathbf{b} \in B\}.$$

Definition 3.3 (Syndrome Matrix)**.** Let $B^{\vee}$ be the space of all linear maps $\beta : B \to \mathbb{F}_q$. $K(\mathbf{r})$ is a subspace of A and it is the kernel of the linear map $S_r : A \to B^{\vee}$ defined by $\mathbf{a} \mapsto \beta_{\mathbf{a}}$, where $\beta_{\mathbf{a}}(\mathbf{b}) = (\mathbf{a} * \mathbf{b}) \cdot \mathbf{r}$. Let $\{\mathbf{a}_i : 1 \leq i \leq l\}$ and $\{\mathbf{b}_j : 1 \leq j \leq m\}$ be bases of A, B respectively. Then the map S_r has the $m \times l$ *syndrome matrix*

$$S_{\mathbf{r}} = ((\mathbf{b}_i * \mathbf{a}_j) \cdot \mathbf{r})_{1 \leq j \leq l, 1 \leq i \leq m}$$

with respect to these bases.

Lemma 3.4 ([21, Lemma 6.2.3])**.** *Let C be an $\mathbb{F}_q$-linear code of length n and $\mathbf{r}$ a received word with error vector $\mathbf{e}$. If $(A * B) \perp C$, then we have $K(\mathbf{r}) = K(\mathbf{e})$.*

Proof. We have that $\mathbf{r} = \mathbf{c} + \mathbf{e}$ for some codeword $\mathbf{c} \in C$. By assumption, $A * B \subset C^{\perp}$ and hence $\mathbf{a} * \mathbf{b}$ is a parity check for the code C. It follows that $(\mathbf{a} * \mathbf{b}) \cdot \mathbf{c} = 0$ and furthermore $(\mathbf{a} * \mathbf{b}) \cdot \mathbf{r} = (\mathbf{a} * \mathbf{b}) \cdot (\mathbf{c} + \mathbf{e}) = (\mathbf{a} * \mathbf{b}) \cdot \mathbf{e}$ for all $\mathbf{a} \in A, \mathbf{b} \in B$. □

Lemma 3.5 ([21, Lemma 6.2.5])**.** *Let $(A * B) \perp C$ and let $\mathbf{r}$ be a received word with error vector $\mathbf{e}$. For a subset $J \subset \{1, \ldots, n\}$, we define the subspace*

$$A(J) = \{\mathbf{a} \in A \mid a_j = 0 \text{ for all } j \in J\}.$$

If $I = \operatorname{supp}(\mathbf{e}) = \{i \mid e_i \neq 0\}$, i.e., I is the set of error positions, then $A(I) \subset K(\mathbf{r})$. Moreover, if we have $d(B^{\perp}) > \operatorname{wt}(\mathbf{e})$, then $A(I) = K(\mathbf{r})$.

Proof. Let $\mathbf{a} \in A(I)$. This means that $a_i = 0$ for all i where $e_i \neq 0$ and it follows that

$$(\mathbf{a} * \mathbf{b}) \cdot \mathbf{e} = \sum_{e_i \neq 0} a_i b_i e_i = 0$$

for all $\mathbf{b} \in B$. Hence, $\mathbf{a} \in K(\mathbf{e}) = K(\mathbf{r})$ by Lemma 3.4. We conclude that $A(I) \subset K(\mathbf{r})$. Now, suppose that $d(B^{\perp}) > \operatorname{wt}(\mathbf{e})$. Let $\mathbf{a} \in K(\mathbf{r}) = K(\mathbf{e})$. It follows that

$$(\mathbf{e} * \mathbf{a}) \cdot \mathbf{b} = \mathbf{e} \cdot (\mathbf{a} * \mathbf{b}) = 0$$

for all $\mathbf{b} \in B$ and hence $\mathbf{e} * \mathbf{a} \in B^{\perp}$. Now, we have that $\operatorname{wt}(\mathbf{e} * \mathbf{a}) \leq \operatorname{wt}(\mathbf{e}) < d(B^{\perp})$. This is only possible if $\mathbf{e} * \mathbf{a} = 0$, i.e., $e_i a_i = 0$ for all i and therefore $a_i = 0$ for all i such that $e_i \neq 0$. Hence, $\mathbf{a} \in A(I)$ and it follows that $K(\mathbf{r}) \subset A(I)$ and together with the first part of the proof it follows that $K(\mathbf{r}) = A(I)$. □

Lemma 3.6. *Consider the syndrome matrix* $S_{\mathbf{r}} = ((\mathbf{b}_i * \mathbf{a}_j) \cdot \mathbf{r})$ *and let* $K = \ker S_{\mathbf{r}} \subset \mathbb{F}_q^l$ *be the right kernel of* $S_{\mathbf{r}}$ *and denote by* G_A *a generator matrix of* A *that has as rows the chosen basis vectors* $\mathbf{a}_j$*. Then*

$$K(\mathbf{r}) = K \cdot G_A = \{\mathbf{x}G_A \mid \mathbf{x} \in K\}.$$

Proof. Take $\mathbf{a} \in K(\mathbf{r})$ and let $\mathbf{m} = (m_1, \ldots, m_l) \in \mathbb{F}_q^l$ be the preimage of $\mathbf{a}$ under G_A, i.e., $\mathbf{a} = \mathbf{m}G_A$. Consider $S_{\mathbf{r}}\mathbf{m}^T$. This vector has as its entries

$$\begin{aligned}\sum_j (\mathbf{b}_i * \mathbf{a}_j) \cdot m_j\mathbf{r} = \sum_j (\mathbf{b}_i * (m_j\mathbf{a}_j)) \cdot \mathbf{r} &= \left(\mathbf{b}_i * \left(\sum_j m_j\mathbf{a}_j\right)\right) \cdot \mathbf{r} \\ &= (\mathbf{b}_i * (\mathbf{m}G_A)) \cdot \mathbf{r} = (\mathbf{b}_i * \mathbf{a}) \cdot \mathbf{r} = 0\end{aligned}$$

since $\mathbf{a} \in K(\mathbf{r})$. Thus, it follows that $\mathbf{a} \in K \cdot G_A$ and hence $K(\mathbf{r}) \subset K \cdot G_A$. For the other inclusion, take $\mathbf{x} = (x_1, \ldots, x_l) \in K$. In particular, we have $S_{\mathbf{r}}\mathbf{x}^T = 0$ and hence for all i

$$\sum_j (\mathbf{b}_i * \mathbf{a}_j) \cdot x_j\mathbf{r} = 0.$$

Similarly as before, we obtain

$$\sum_j (\mathbf{b}_i * \mathbf{a}_j) \cdot x_j\mathbf{r} = (\mathbf{b}_i * (\mathbf{x}G_A)) \cdot \mathbf{r}$$

and as this is true for all i and $\mathbf{b}_i$ form a basis for B it follows that

$$(\mathbf{b} * (\mathbf{x}G_A)) \cdot \mathbf{r} = 0$$

for any $\mathbf{b} \in B$ and thus $\mathbf{x}G_A \in K(\mathbf{r})$. □

We can combine all these lemmas and obtain the following algorithm.

Algorithm 3.7 (Basic Algorithm, [21, Algorithm 6.3]). **Input:**

- H, a parity check matrix of the $[n, k]$-code C,
- t, the number of errors to correct,
- (A, B), a t-error-correcting pair for the code C,
- $\{\mathbf{a}_j : 1 \leq j \leq l\}$, $\{\mathbf{b}_i : 1 \leq i \leq m\}$ bases of A, B respectively,
- $\mathbf{r}$, a received word.

Output: A codeword $\mathbf{c} \in C$ with $d(\mathbf{r}, \mathbf{c}) \leq t$ if it exists.

1. Compute $s_{ij} = (\mathbf{b_i} * \mathbf{a_j}) \cdot \mathbf{r}$ for all $1 \leq i \leq m, 1 \leq j \leq l$.
2. Compute $K(\mathbf{r})$ by computing the right kernel K of the $m \times l$ matrix $S_{\mathbf{r}} = (s_{ij})$.

3. Take $\mathbf{a} \in K(\mathbf{r})$ such that $\mathbf{a} \neq 0$.
4. Compute $J = \{j \mid a_j = 0\}$.
5. If $H\mathbf{x}^T = H\mathbf{r}^T$ and $x_j = 0$ for all $j \notin J$ has the unique solution $\mathbf{x_0}$ with $\mathrm{wt}(\mathbf{x_0}) \leq t$, then return $\mathbf{c} = \mathbf{r} - \mathbf{x_0}$.

Proposition 3.8 ([21, Proposition 6.2.8])**.** *Let C be an $\mathbb{F}_q$-linear code of length n and $A, B \subset \mathbb{F}_q^n$. If (A, B) is a t-error-correcting pair for C, then Algorithm 3.7 corrects t errors of C with complexity $\mathcal{O}((mn)^3)$.*

Proof. (A, B) is a t-error-correcting pair for C, thus we have $(A * B) \perp C$ and the algorithm can be applied. If $\mathbf{r}$ is a received word with at most t errors, then the error vector $\mathbf{e}$ with support I has weight at most t and $A(I)$ is not zero because I imposes at most t linear conditions on A but the dimension of A is at least $t+1$ since (A, B) is an ECP. Let $\mathbf{a} \in K(\mathbf{r})$ be nonzero and $J = \{j \mid a_j = 0\}$. By assumption, $d(B^\perp) > t$ and thus by Lemma 3.5 $K(\mathbf{r}) = A(I)$ and $I \subset J$. This means that the errors only occur where $a_j = 0$, i.e., $\mathbf{a}$ locates the error positions and hence it is called an *error locator vector*. Since $\mathrm{wt}(\mathbf{a}) \geq d(A)$, it follows from the definition of ECP that $\mathbf{a}$ has at most $n - d(A) < d(C)$ zeros and hence $|J| < d(C)$. This allows us to find the unique closest codeword $\mathbf{c}$ to $\mathbf{r}$ using the method of *erasure decoding*. In particular, as we know that no errors occurred on $J^c = \{j \mid a_j \neq 0\}$, we know that $c_i = r_i$ for all $i \in J^c$. In the next step, we consider a generator matrix G of C and we erase all the entries of $\mathbf{r}$ and the columns of G at positions $j \in J$ and we obtain a vector $\mathbf{r}' \in \mathbb{F}_q^{n-n'}$ and a $k \times (n - n')$-matrix G', where $n' = |J|$. This means, we only keep the positions where no error occurred. We solve the system

$$\mathbf{m}G' = \mathbf{r}'$$

for $\mathbf{m} \in \mathbb{F}_q^k$ and we encode it as $\mathbf{c} = \mathbf{m}G$. It is clear from construction that $c_i = r_i$ for all $i \in J^c$ and $d(\mathbf{r}, \mathbf{c}) \leq |J| < d(C)$. Hence, $\mathbf{c}$ is the closest codeword to $\mathbf{r}$. This procedure in terms of the generator matrix can be reformulated in terms of the parity check matrix in the following way. We want to find $\mathbf{x_0}$ such that $\mathrm{wt}(\mathbf{x_0}) \leq t$, $(\mathbf{x_0})_j = 0$ for all $j \notin J$ and $\mathbf{r} - \mathbf{x_0} \in C$, i.e., $H(\mathbf{r} - \mathbf{x_0})^T = 0$, or equivalently $H{\mathbf{x_0}}^T = H\mathbf{r}^T$. In other words, this means that $\mathbf{r} - \mathbf{x_0} = \mathbf{c}$ is the closest codeword to $\mathbf{r}$. □

Remark. There exist error correcting pairs for Reed-Solomon codes, generalized Reed-Solomon codes and more generally for Algebraic Geometry codes.

3.2 Attacking AG-Codes using ECP

In the following, we assume that the divisor E has degree $m > 3g$ and hence $t > 0$. We consider the code $C^\perp = C_L(X, \mathcal{P}, E)^\perp$ with generator matrix G and $t = \lfloor \frac{d^* - g - 1}{2} \rfloor$ with designed distance $d^* = m - 2g + 2$. In the following, we will describe the attack from Couvreur, Márquez-Corbella, and Pellikaan [7] to find an ECP for the code $C^\perp$. Then we can apply Algorithm 3.7 to retrieve the message.

Theorem 3.9 ([7, Theorem 4])**.** *The pair (A, B) for $A = C_L(X, \mathcal{P}, F)$ and $B = C_L(X, \mathcal{P}, E-F)$ with $m > deg(F) \geq t+g$ such that $\deg(E-F) > t+2g-2$ is a t-error correcting pair for $C^\perp = C_L(X, \mathcal{P}, E)^\perp$. Moreover, such a pair always exists whenever $m > 2g-2$.*

Proof. Using Theorem 1.39, we verify that the conditions for an ECP are satisfied:

1. We take $\mathbf{a} \in A$ and $\mathbf{b} \in B$. Then there exist $f \in L(F)$ and $g \in L(E-F)$ such that $\mathbf{a} = f(\mathcal{P})$ and $\mathbf{b} = g(\mathcal{P})$ and hence $\mathbf{a} * \mathbf{b} = fg(\mathcal{P})$. Since $(f) \geq -F$ and $(g) \geq -(E-F) = F-E$, it follows that $(fg) \geq -F + (F-E) = E$ and thus $fg \in L(E)$ and $\mathbf{a} * \mathbf{b} \in C_L(X, \mathcal{P}, E)$. In particular, $A * B \subset (C_L(X, \mathcal{P}, E)^\perp)^\perp$.

2. We have $\deg F \geq t+g$ and hence

$$k(A) = \dim A \geq \deg F - g + 1 \geq t + g - g + 1 = t + 1 > t.$$

3. $B^\perp = C_L(X, \mathcal{P}, (E-F)^\perp)$ by Theorem 1.41. We assume that $\deg(E-F) > t + 2g - 2$ and thus

$$d(B^\perp) \geq n - \deg((E-F)^\perp) \geq n - n - 2g + 2 + t + 2g - 1 = t + 1 > t.$$

4. We assume that $\deg(E-F) > 2g-2$. Hence, we obtain that

$$\begin{aligned} d(A) + d(C^\perp) &\geq (n - \deg F) + (n - \deg(E^\perp)) \\ &= n - \deg F + n - (n - \deg E + 2g - 2) \\ &= n - \deg F + \deg E - 2g + 2 \\ &= n + \deg(E-F) - 2g + 2 \\ &> n. \end{aligned}$$

Therefore, we showed that under the assumption that $\deg(E-F) > t + 2g - 2$ the pair (A, B) is an ECP for the code $C^\perp$. Moreover, such an F always exists by [20, Theorem 3.3]. More precisely, we consider $F_1 = (t+g)P_1$. Then there exists a rational function f such that $\nu_{P_1}(f) = t+g$ and $\nu_{P_i}(f) = 0$ for all $i \neq 1$ by Theorem 1.34. We choose $F = F_1 - (f)$ and note that $\deg F = \deg F_1 = t+g$. By definition of $t = \lfloor \frac{d^*-g-1}{2} \rfloor = \lfloor \frac{m-3g+1}{2} \rfloor$, it follows that $m \geq 2t + 3g - 1$ and thus

$$\deg(E-F) = m - (t+g) \geq 2t + 3g - 1 - t - g = t + 2g - 1 > t + 2g - 2.$$

□

Corollary 3.10 ([7, Corollary 5])**.** *If $\deg E \leq n-3$ and we have B as above with $\deg F = t+g$ and $A_0 = (B * C_L(X, \mathcal{P}, E)^\perp)^\perp$, then (A_0, B) is a t-error correcting pair for $C_L(X, \mathcal{P}, E)^\perp$.*

Proof. We want to show that

$$A_0 = (B * C_L(X, \mathcal{P}, E)^{\perp})^{\perp} = C_L(X, \mathcal{P}, F).$$

Then the statement follows directly from Theorem 3.9. We note that

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

Hence, we can apply Proposition 2.10 to the divisors E and $E - F$ and obtain the statement. □

Remark. From this corollary, we see that in order to find an ECP it is enough to find a code of the form $B = C_L(X, \mathcal{P}, E - F)$.

Proposition 3.11 ([16, Proposition 18])**.** *If $2g + 1 \leq m = \deg E < \frac{n}{2}$, define $C = C_L(X, \mathcal{P}, E)$ and let $k_1 = dim(C)$ and $k_2 = dim(C^{(2)})$. Then we have $deg(E) = m = k_2 - k_1$ and $g = k_2 - 2k_1 + 1$.*

Proof. By Corollary 2.9 we have that $C^{(2)} = C_L(X, \mathcal{P}, 2E)$. Furthermore, $k_1 = m - g + 1$ and $k_2 = 2m - g + 1$ by Corollary 1.40. Hence, we conclude that $m = k_2 - k_1$ and $g = k_2 - 2k_1 + 1$. □

The main idea of the attack is to find a t-error-correcting pair (A_0, B) as above, i.e., we want to find the code $B = C_L(X, \mathcal{P}, E - F)$, where $m > deg(F) \geq t + g$ as in Theorem 3.9. To do so, we consider the following sequence: Let P be a point of $\mathcal{P}$ and define

$$\mathcal{B}_i = C_L(X, \mathcal{P}, E - iP) \quad \text{for all } i \in \mathbb{N}.$$

This sequence provides a filtration of $C_L(X, \mathcal{P}, E)$, but for $i \geq 1$ the code $\mathcal{B}_i$ is degenerated and this will be addressed later. More precisely, a code is called *degenerated* if all codewords are zero in some common coordinate, or equivalently its generator matrix contains a column consisting entirely of 0. In practice, we do not want a code to be degenerated, but since for all $f \in L(E - iP)$, f must have a zero of order i at P, it is clear that $\mathcal{B}_i$ is degenerated for all $i \geq 1$. Related to this problem is the observation that $\text{supp}(E - iP) \cap \text{supp}(D_{\mathcal{P}}) = \{P\} \neq \emptyset$ for all $i \geq 1$, which we usually do not allow for an AG-code. For the moment, we do not care about this and focus on how to find all the $\mathcal{B}_i$. First we note that $\mathcal{B}_0 = C_L(X, \mathcal{P}, E)$, which can clearly be computed by an attacker knowing the code $C_L(X, \mathcal{P}, E)^{\perp}$. We also note that the code $\mathcal{B}_1$ is exactly the codewords of $C_L(X, \mathcal{P}, E)$ that are zero at position P. This means, we already know $\mathcal{B}_0$ and $\mathcal{B}_1$. We can find more elements of this filtration by using a theorem that relies on the notion of gaps.

Remark. For the attack it does not matter which point P we choose. However, for simplicity it is convenient to choose $P = P_1$.

Definition 3.12 (Gap)**.** Let G be a divisor and P be a rational point on the curve X. An integer $\gamma \geq -\deg(G)$ is called a *G-gap* at P if $L(G + \gamma P) = L(G + (\gamma - 1)P)$. Else it is called a *non-gap*.

Theorem 3.13 ([7, Theorem 9])**.** *If we have $s \geq 1$ and $\frac{n}{2} > m = \deg(E) \geq 2g + s + 1$, then $\mathcal{B}_{s+1}$ is the solution to the following problem:*

$$\begin{cases} \mathbf{z} \in \mathcal{B}_s \\ \mathbf{z} * \mathcal{B}_{s-1} \subset (\mathcal{B}_s)^{(2)} \end{cases}.$$

Proof. Every $\mathbf{z} \in \mathcal{B}_{s+1}$ satisfies the conditions in the problem. More precisely, $\mathcal{B}_{s+1} * \mathcal{B}_{s-1} = (\mathcal{B}_s)^{(2)}$ by Corollary 2.9, which we may use due to assumptions on m. Conversely, assume that there exists $\mathbf{c} \in \mathcal{B}_s \backslash \mathcal{B}_{s+1}$ solving the problem. Then there exists $f \in L(E - sP) \backslash L(E - (s+1)P)$ such that $(f(P_1), \ldots, f(P_n)) = \mathbf{c}$, i.e., $\nu_P(f) = s$. From the Riemann-Roch Theorem (Theorem 1.24), it follows that if $\deg(E) - s \geq 2g + 1$, then any integer $\gamma \geq -s$ is an E non-gap at P. Indeed, if we have $m \geq 2g + 1 + s$ and $\gamma \geq -s$, then clearly $\gamma > -m$ and

$$\deg(E + \gamma P) = m + \gamma \geq 2g + 1 + s - s = 2g + 1 > 2g - 2$$

and similarly

$$\deg(E + (\gamma - 1)P) = 2g > 2g - 2.$$

Hence, we can apply Corollary 1.25 and obtain

$$\dim(L(E + \gamma P)) = m + \gamma - g + 1 \neq m + \gamma + g = \dim(L(E + (\gamma - 1)P))$$

and hence γ must be a non-gap. This means, there exists a $g \in L(E - (s - 1)P) \backslash L(E - sP)$, i.e., g is such that $\nu_P(g) = s - 1$. Since $\mathbf{c}$ solves the problem, we have that $\mathrm{Ev}_{\mathcal{P}}(fg) \in \mathcal{B}_s^{(2)} = C_L(X, \mathcal{P}, 2E - 2sP)$. Moreover, since $m < \frac{n}{2}$, the evaluation map $\mathrm{Ev}_{\mathcal{P}} : L(2E - 2sP) \to C_L(X, \mathcal{P}, 2E - 2sP)$ is injective. This implies that $fg \in L(2E - 2sP)$, but we have $\nu_P(fg) = 2s - 1 < 2s$ which is a contradiction. □

Now we can iterate this $(t + g)$ times and in the end we obtain $\mathcal{B}_{t+g} = (C_L(X, \mathcal{P}, E - (t + g)P)$. Namely, $(t + g)P$ has degree $t + g$ and hence we can apply Theorem 3.9 and Corollary 3.10 to find an ECP. The only problem we have, is that $\mathcal{B}_{t+g}$ is a degenerated code. We obtain a suitable non-degenerated code using the following theorem.

Theorem 3.14 ([7, Theorem 13])**.** *Let $\mathbf{G}$ be a generator matrix of $\mathcal{B}_{t+g}$ of the form*

$$\mathbf{G} = \begin{pmatrix} 0 & \mathbf{c}_1 \\ (0) & \mathbf{G}_1 \end{pmatrix},$$

where $\mathbf{c}_1 \in \mathbb{F}_q^{n-1}$, $(0 \quad \mathbf{c}_1) \in \mathcal{B}_{t+g} \backslash \mathcal{B}_{t+g+1}$ *and* $((0) \quad \mathbf{G}_1)$ *is a generator matrix of* $\mathcal{B}_{t+g+1}$. *Then there exists a rational function* h *on the curve* X *such that*

$$\mathbf{G}' = \begin{pmatrix} 1 & \mathbf{c}_1 \\ (0) & \mathbf{G}_1 \end{pmatrix}$$

is a generator matrix of the code $C_L(X, \mathcal{P}, E - (t+g)P - (h))$.

Proof. Consider $P = P_1$ and let $f \in L(E - (t+g)P) \backslash L(E - (t+g+1)P)$ be such that

$$(0 \mid \mathbf{c_1}) = (f(P_1), \ldots, f(P_n)).$$

Then $\nu_P(f) = t + g$ and by Theorem 1.34 there exists a rational function $h \in \mathbb{F}_q(X)$ such that

- $h(P_i) = 1$ for all $i \geq 2$,
- $\nu_P(h) = -t - g$ and $hf(P) = 1$.

More details can be found in [26, Theorem 1.3.1] and [7, Theorem 13]. □

Remark. Using Theorem 3.14, we obtain a non-degenerated code B that satisfies the conditions from Theorem 3.9. Morevoer, it is not necessary to explicitly find such h. Its existence is enough.

Now we know how to obtain an ECP only knowing the public key (G, t). We note that the attack does not rely on G to have a specific form as long as $m > 3g$. In particular, if $\tilde{G} = BGP$, as it is generally the case in the McEliece encryption scheme, then the linear code generated by $\tilde{G}$ is still an algebraic geometry code defined over the same curve X with the same divisor E and even the points $\mathcal{P}$ are the same, even though they are permuted. However, this does not change the attack and it is still possible to obtain an ECP for the algebraic geometry code generated by $\tilde{G}$.

Remark. There are various ways to improve this attack, either by reducing the number of iterations, weakening the assumptions on $m = \deg E$ or even allowing a larger error correcting capacity, namely

$$t = \left\lfloor \frac{d^* - 1}{2} \right\rfloor.$$

However, this last improvement requires the use of error correcting arrays, which are described in Couvreur, Márquez-Corbella, and Pellikaan [8].

Next, we want to improve the attack by reducing the steps in the iteration of Theorem 3.13. This will reduce the time needed to compute $\mathcal{B}_{t+g}$ significantly. To do so, we consider the following theorem.

Theorem 3.15 ([7, Theorem 10])**.** *If* $\frac{n}{2} > m \geq 2g + \lfloor \frac{S+1}{2} \rfloor + 1$, *then* $\mathcal{B}_S$ *is the solution space to*

$$\begin{cases} \mathbf{z} \in \mathcal{B}_{\lfloor (S+1)/2 \rfloor} \\ \mathbf{z} * \mathcal{B}_0 \subset \mathcal{B}_{\lfloor S/2 \rfloor} * \mathcal{B}_{\lfloor (S+1)/2 \rfloor} \end{cases}.$$

Proof. The proof works analogously to the proof of Theorem 3.13. First we note that $\lfloor \frac{S}{2} \rfloor + \lfloor \frac{S+1}{2} \rfloor = S$ for any $S \in \mathbb{N}$. The assumptions on m allow us to use Corollary 2.9 and hence we have

$$\begin{aligned}\mathcal{B}_S * \mathcal{B}_0 &= C_L(X, \mathcal{P}, 2E - SP) \\ &= C_L\left(X, \mathcal{P}, 2E - \left(\left\lfloor \frac{S}{2} \right\rfloor + \left\lfloor \frac{S+1}{2} \right\rfloor\right) P\right) \\ &= \mathcal{B}_{\lfloor S/2 \rfloor} * \mathcal{B}_{\lfloor (S+1)/2 \rfloor}.\end{aligned}$$

Thus, any $\mathbf{z} \in \mathcal{B}_S$ satisfies the conditions. On the other hand, assume that there exists a $\mathbf{c} \in \mathcal{B}_{\lfloor (S+1)/2 \rfloor} \backslash \mathcal{B}_S$ which satisfies the conditions. Then there exists $f \in L(E - \lfloor (S+1)/2 \rfloor P) \backslash L(E - SP)$ such that $(f(P_1), \ldots, f(P_n)) = \mathbf{c}$, i.e., $\nu_P(f) = \lfloor \frac{S+1}{2} \rfloor$. As before, any $\gamma \geq -\lfloor \frac{S+1}{2} \rfloor$ is an E non-gap at P, in particular 1 is a non-gap and hence there exists $g \in \mathcal{B}_0 \backslash \mathcal{B}_1$, i.e., $\nu_P(g) = 0$. As before, we have $fg \in L(2E - SP)$ since $\mathbf{c}$ satisfies the conditions. But at the same time, we have $\nu_P(fg) = \lfloor \frac{S+1}{2} \rfloor + 0 < S$ for all $S > 1$, which is a contradiction. Now if $S \in \{0, 1\}$, then we have $\mathcal{B}_{\lfloor (S+1)/2 \rfloor} \backslash \mathcal{B}_S = \emptyset$ and hence the theorem holds for all $S \in \mathbb{N}$. □

Similarly as before, we can iterate this theorem to obtain the desired code $\mathcal{B}_{t+g}$. However, this time the algorithm is slightly more complicated. In particular, we assume that $\frac{n}{2} > m \geq \frac{5g+t}{2} + 2$. This ensures that the conditions to apply the theorem are satisfied. Then we proceed as follows.
Input: Generator matrices for the codes $\mathcal{B}_0$ and $\mathcal{B}_1$.
Output: Generator matrices for $\mathcal{B}_{t+g}$ and $\mathcal{B}_{t+g+1}$.
Let $b = \lfloor \log_2 n \rfloor + 1$. Then we note that $2^{b-1} \leq n < 2^b$ and proceed iteratively. For $s = b-2, \ldots, 1$ compute $\mathcal{B}_{\lfloor (t+g)/2^{s-1} \rfloor}$ and $\mathcal{B}_{\lfloor (t+g)/2^{s-1}+1 \rfloor}$ from the codes $\mathcal{B}_{\lfloor (t+g)/2^s \rfloor}$ and $\mathcal{B}_{\lfloor (t+g)/2^s+1 \rfloor}$ by applying Theorem 3.15.

Remark. We quickly want to justify this algorithm and explain why it works. To do so, we proceed by induction, but first we note that for $z \in \mathbb{N}$ and $r \in [0; 1)$, we have

$$\left\lfloor \frac{\lfloor z + r \rfloor}{2} \right\rfloor = \left\lfloor \frac{z}{2} \right\rfloor = \left\lfloor \frac{z + r}{2} \right\rfloor$$

since we have

$$\left\lfloor \frac{z+r}{2} \right\rfloor = \begin{cases} \lfloor \frac{2k}{2} + \frac{r}{2} \rfloor = k & \text{if } z = 2k \text{ is even} \\ \lfloor \frac{2k+1+r}{2} \rfloor = k + \lfloor \frac{1}{2} + \frac{r}{2} \rfloor = k & \text{if } z = 2k+1 \text{ is odd} \end{cases}$$

as $0 \leq \frac{r}{2} < \frac{1}{2}$ since $0 \leq r < 1$. We also immediately observe that

$$\left\lfloor \frac{\lfloor z + r \rfloor + 1}{2} \right\rfloor = \left\lfloor \frac{\lfloor z + r \rfloor}{2} + \frac{1}{2} \right\rfloor \in \left\{ \left\lfloor \frac{z+r}{2} \right\rfloor, \left\lfloor \frac{z+r}{2} \right\rfloor + 1 \right\}.$$

Using these observations, we can verify the base case and inductive step that appear in the algorithm.
Base case: From the definition of $t = \lfloor \frac{m-3g+1}{2} \rfloor$, it follows that $m \geq 2t + 3g$

and hence $\frac{n}{2} > 2t + 3g \geq 2(t+g)$, which implies that $\frac{t+g}{n} < \frac{1}{4}$. First we verify the base case $s = b - 2$. We consider

$$\frac{t+g}{2^{s-1}} = 2\frac{t+g}{2^s} = 2\frac{t+g}{2^{b-2}} = 8\frac{t+g}{2^b} < 8\frac{t+g}{n} < 2.$$

Therefore, it follows that $\left\lfloor \frac{t+g}{2^{s-1}} \right\rfloor \in \{0, 1\}$. Similarly,

$$\frac{t+g}{2^{s-1}} + 1 < 3$$

and hence $\left\lfloor \frac{t+g}{2^{s-1}} + 1 \right\rfloor \in \{0, 1, 2\}$. But for $S \in \{0, 1, 2\}$, we have that $\left\lfloor \frac{S}{2} \right\rfloor, \left\lfloor \frac{S+1}{2} \right\rfloor \in \{0, 1\}$. The codes $\mathcal{B}_0$ and $\mathcal{B}_1$ are known and thus we can apply Theorem 3.15.
Induction hypothesis: We assume that we know the codes

$$\mathcal{B}_{\lfloor (t+g)/2^s \rfloor} \text{ and } \mathcal{B}_{\lfloor (t+g)/2^s + 1 \rfloor}.$$

Inductive step: Using the observation above, it is also easy to verify the inductive step. Assume that we want to compute the code $\mathcal{B}_S$ for $S = \left\lfloor \frac{t+g}{2^{s-1}} + 1 \right\rfloor$. Then we need

$$\left\lfloor \frac{S}{2} \right\rfloor = \left\lfloor \frac{\left\lfloor \frac{t+g}{2^{s-1}} + 1 \right\rfloor}{2} \right\rfloor = \left\lfloor \frac{t+g}{2^s} + \frac{1}{2} \right\rfloor \in \left\{ \left\lfloor \frac{t+g}{2^s} \right\rfloor, \left\lfloor \frac{t+g}{2^s} + 1 \right\rfloor \right\},$$

which we know. Furthermore, we need

$$\left\lfloor \frac{S+1}{2} \right\rfloor = \left\lfloor \frac{t+g}{2^s} + 1 \right\rfloor,$$

which we also know by the induction hypothesis. The case where $S = \left\lfloor \frac{t+g}{2^{s-1}} \right\rfloor$ works completely analogously.

Remark. This algorithm consists of $2\lceil \log_2(t+g) \rceil$ systems of linear equations. For comparison, using Theorem 3.13, we have a total of $t + g$ systems of linear equations. The systems are not exactly the same but they have a similar complexity and thus it is obvious that this is an exponential improvement. We can then use this algorithm to find $\mathcal{B}_{t+g}$ and the rest of the attack remains the same. Most importantly, we still need to apply Theorem 3.14.

3.3 Examples based on Hermitian Curves

In this section, we give some examples regarding the s-closure and the attack described in Section 3.2. To do so, we consider a class of well-known curves that have an easy description.

Definition 3.16 (Hermitian Curve)**.** Let r be the power of a prime number and $q = r^2$. Then the *Hermitian curve* $\mathcal{H}_r$ over $\mathbb{F}_q$ is defined by the affine equation

$$y^r + y = x^{r+1}.$$

This curve has $n = r^3$ affine $\mathbb{F}_q$-rational points and only one point at infinity, namely $P_\infty = (0 : 1 : 0)$. This curve is a smooth plane curve and thus by Theorem 1.23 it has genus

$$g = \frac{r(r-1)}{2}.$$

We take m such that $2g - 2 < m < n$ and define $E = mP_\infty$ and we take as set $\mathcal{P}$ the n finite points, i.e., $\mathcal{P} = \mathcal{H}_r \backslash \{P_\infty\}$. We define the algebraic geometry code $C = C_L(\mathcal{H}_r, \mathcal{P}, E)$.

Remark. Note that if $m_2 \geq m_1$, then we have $L(m_2 P_\infty) \supset L(m_1 P_\infty)$ and thus

$$\dim(C_L(\mathcal{H}_r, \mathcal{P}, m_2 P_\infty)) \geq \dim(C_L(\mathcal{H}_r, \mathcal{P}, m_1 P_\infty)).$$

Example 8. Take $r = 3$. Then we have $n = 3^3 = 27$ and we work over the field $\mathbb{F}_9$. Then the Hermitian curve $\mathcal{H}_3$ has genus $g = 3$ and we take $m = 11$. First, we consider the algebraic geometry code $C = C_L(\mathcal{H}_3, \mathcal{P}, 11P_\infty)$. This code has dimension 9 and size 27. Using Algorithm A.3, we can compute $\overline{C}^2$ and see that C is two-closed, i.e., $\overline{C}^2 = C$. A further examination gives that we have $C^{(3)} = \mathbb{F}_9^{27}$ is the full space. From this, it follows directly that $\overline{C}^3 = \mathbb{F}_9^{27}$ and thus C is not three-closed. We can also consider the subfield-subcode $C' = C \cap \mathbb{F}_3^{27}$. This code is two-closed and three-closed but not four-closed. In fact, $C' \subsetneq \overline{C'}^4 \subsetneq \mathbb{F}_3^{27}$.

Example 9. This example is based on Couvreur, Márquez-Corbella, and Pellikaan [7, Example 1]. We take $r = 7$, thus $n = 7^3 = 343$. We choose $m = 170$ and consider the code $C = C_L(\mathcal{H}_7, \mathcal{P}, 170P_\infty)^\perp$. This code has dimension $k = 193$ and the curve $\mathcal{H}_7$ has genus $g = 21$. It follows that the designed distance is $d = 130$ and the error correcting capacity is $t = 54$. We have implemented the attack from Section 3.2 and successfully found an ECP (A, B) to recover the secret message $\mathbf{m}$ in various instances using magma [3]. Some of the code is listed as algorithms in Appendix B and a more complete version including this particular example is available at [15]. There, we also use Algorithm B.6 to simulate the McEliece encryption scheme and use an implementation of Algorithm 3.7 to retrieve the secret message $\mathbf{m}$.

Example 10. In this example, we want to give an explicit comparison of the algorithms based on Theorem 3.13 and Theorem 3.15 to determine the code $\mathcal{B}_{t+g}$, i.e., we compare Algorithm B.3 and Algorithm B.4. We only consider this part of the attack that computes $\mathcal{B}_{t+g}$. First, we consider the Hermitian curve with $r = 8$ and choose $m = 120$. Then $t+g = 46$ and $2\lceil \log_2(t+g) \rceil = 12$. In this run of the attack, the algorithms took 93 and 308 seconds respectively. This means that in absolute values the attack was $308 - 93 = 215$ seconds faster, which is a relative improvement of a factor approximately 3.3. We also note that

$$\left| \frac{46}{12} - \frac{308}{93} \right| < 0.6.$$

Next, we consider another example of a Hermitian curve with $r = 9$ and $m = 359$. This code has $t + g = 162$ and $2\lceil \log_2(t+g) \rceil = 16$ and the algorithm took

11235 and 1116 seconds respectively. For comparison,

$$\left|\frac{162}{16} - \frac{11235}{1116}\right| < 0.06.$$

In this example, we see that it was possible to reduce the running time from about 3 hours to 18 minutes, which is a significant improvement.

3.3.1 Construction of Generator Matrices

In this section, we give an explicit way to easily construct a generator matrix of the code $C = C_L(\mathcal{H}_r, \mathcal{P}, E)$. For further details see Stichtenoth [25].

Proposition 3.17 ([25, Proposition 1])**.** *For each $0 \leq m < n$, the Riemann-Roch space $L(E) = L(mP_\infty)$ has the basis*

$$\{x^i y^j \mid 0 \leq i, 0 \leq j \leq r-1, ir + j(r+1) \leq m\}.$$

Corollary 3.18. *For $0 \leq m < n$, consider the set*

$$A(m) = \{(i,j) \mid 0 \leq i, 0 \leq j \leq r-1, ir + j(r+1) \leq m\}$$

and for each $(i,j) \in A(m)$ define the vector $\mathbf{u}_{ij}$ as

$$\mathbf{u}_{ij} = (\alpha^i \beta^j)_{(\alpha,\beta)\in\mathcal{H}_r},$$

where $(\alpha, \beta) \in \mathcal{H}_r$ denote the affine points. Then the matrix M with rows $\mathbf{u}_{ij}$ is a generator matrix of the code C.

4 Secret Sharing

In the following, we will introduce the method of secret sharing as first described in Shamir [23]. We then proceed and explain how it can be naturally extended following Cramer, Damgård, and Maurer [11] and Chen and Cramer [4].

4.1 Shamir's Secret Sharing

Definition 4.1 (Threshold Scheme)**.** Consider some data D called the *secret*. We want to divide D into n pieces $D_1, \ldots, D_n$ such that:

1. If somebody knows at least k pieces $D_{j_1}, \ldots, D_{j_k}$, then they can easily compute the secret D.

2. If somebody knows less than or exactly $k-1$ pieces, they learn nothing about D, i.e., they cannot compute D nor eliminate some possible values for D.

This scheme is called (k,n)-*threshold scheme* and the pieces D_i are called the *shares*. In particular, we need $k \leq n$.

Threshold schemes have two main advantages. If less than $n-k$ pieces D_i are lost, then it is still possible to recover the secret D. On the other hand, if an attacker wants to know the secret D, they need to know at least k pieces D_i.

Remark. If we choose $n = 2k-1$, then we have a threshold scheme with very good parameters because the threshold is at $k-1 = \lfloor \frac{n}{2} \rfloor$. Namely, even if $k-1$ pieces are lost we still have $n-(k-1) = k$ pieces left and thus we can still recover the secret D. However, an attacker still needs to know k pieces and an attacker knowing $\lfloor \frac{n}{2} \rfloor = k-1$ or less pieces cannot reconstruct the secret.

Theorem 4.2 (Lagrange Interpolation, [12, Chapter 5.3])**.** *Let K be a field and consider l pairs $(x_1, y_1), \ldots, (x_l, y_l) \in K^2$ such that all x_i are pairwise different. Then there exists a unique polynomial $f(x) \in K[x]_{<l}$ such that $f(x_i) = y_i$ for all i. Moreover, f has the explicit description*

$$f(x) = \sum_{i=1}^{l} y_i f_i(x),$$

where

$$f_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}.$$

Proof. It is clear that $f \in K[x]_{<l}$ as $f_i \in K[x]_{<l}$ for all i. Furthermore, f_i is well-defined because $x_i \neq x_j$ for all $i \neq j$. First we show uniqueness. We assume that there exist $h, g \in K[x]_{<l}$ such that $f(x_i) = g(x_i) = y_i$ for all i. We consider the polynomial $h-g$, which has degree at most $l-1$ but has at least l zeros, namely at every x_i. This is only possible if $h-g$ is the zero polynomial, i.e., $h-g = 0$ and thus $h = g$. Next we verify that f satisfies the property that $f(x_i) = y_i$ for all i. We note that $f_i(x_i) = 1$ and $f_i(x_j) = 0$ for all $i \neq j$ by construction. Hence, as f is a linear combination of the f_i, we obtain that

$$f(x_j) = \sum_{i=1}^{l} y_i f_i(x_j) = y_i.$$

□

Shamir's (k,n)-threshold scheme: This scheme exploits the fact that if we have k points $(x_1, y_1), \ldots, (x_k, y_k)$ such that all x_i are distinct, then there exists a unique polynomial $q(x)$ with degree $k-1$ such that $q(x_i) = y_i$ for all $i = 1, \ldots, k$. Consider now the secret D to be a number. We take a random polynomial $q(x)$ such that $\deg q = k-1$, i.e.,

$$q(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1}$$

and such that $a_0 = D$. We define the pieces D_i as

$$D_1 = q(1), \ldots, D_i = q(i), \ldots, D_n = q(n).$$

More precisely, the piece D_i consists of the value $q(i)$ and its index i. Now, if we know at least k of these pieces we can use interpolation as described in Theorem 4.2 to find the polynomial q and then we can compute the secret $q(0) = D$.

Example 11. Let $D \in \mathbb{N}$ be the secret and let $k, n \in \mathbb{N}$. We choose a prime number p such that $p > n, D$ and consider the field $\mathbb{F}_p = \mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$. We choose the coefficients $a_1, \ldots, a_{k-1} \in \mathbb{F}_p$ uniformly at random and set

$$q(x) = D + a_1 x + \ldots + a_{k-1} x^{k-1}.$$

Then we set the shares as

$$D_i = (i, q(i)), \quad i = 1, \ldots, n.$$

If an attacker knows k or more shares, then they can perform interpolation and recover the secret. However, if the attacker only knows $k-1$ or less shares, then they cannot. Indeed, assume the attacker knows exactly $k-1$ of the shares. Then for all values $D' \in \mathbb{F}_p$, there exists a polynomial q' of degree $k-1$ such that $q'(0) = D'$ and $q'(i) = D_i$ for all D_i known to the attacker. So, there are in fact p possibilities for the secret D, all of which are equally likely and the attacker learns nothing from the knowledge of the D_i.

This scheme is related to Reed-Solomon codes and it was later generalized in McEliece and Sarwate [17] in the following way. Consider the finite field $\mathbb{F}_q$ and take a fixed list $(\alpha_1, \ldots, \alpha_{q-1})$ of all the nonzero elements. We take a message $\mathbf{a} = (a_0, \ldots, a_{k-1}) \in \mathbb{F}_q^k$ and encode it as $\mathbf{D} = (D_1, \ldots, D_{q-1}) \in \mathbb{F}_q^{q-1}$, where

$$D_i = \sum_{j=0}^{k-1} a_j \alpha_i^j.$$

Then the secret is $a_0 = -\sum_{i=1}^{q-1} D_i$ and the shares are the values D_i. This scheme corresponds to an RS-code and hence it is not necessary to know all the entries of $\mathbf{D}$ to recover the whole codeword $\mathbf{D}$ thanks to error-correcting algorithms.

Remark. The equality

$$a_0 = -\sum_{i=1}^{q-1} D_i$$

follows from basic facts about finite fields. Let $\alpha \in \mathbb{F}_q^*$ be a primitive element. Then we obtain the following:

$$\begin{aligned}
\sum_{i=1}^{q-1} D_i &= \sum_{i=1}^{q-1} \left(\sum_{j=0}^{k-1} a_j \alpha_i^j \right) = \sum_{i=1}^{q-1} \left(a_0 + \sum_{j=1}^{k-1} a_j \alpha_i^j \right) \\
&= (q-1)a_0 + \sum_{i=1}^{q-1} \left(\sum_{j=1}^{k-1} a_j \alpha_i^j \right) = -a_0 + \sum_{j=1}^{k-1} \left(\sum_{i=1}^{q-1} a_j \alpha_i^j \right) \\
&= -a_0 + \sum_{j=1}^{k-1} \left(a_j \sum_{i=1}^{q-1} \alpha_i^j \right) = -a_0 + \sum_{j=1}^{k-1} \left(a_j \sum_{l=1}^{q-1} (\alpha^l)^j \right) \\
&= -a_0,
\end{aligned}$$

where in the last two steps we used the fact that we can represent the elements of $\mathbb{F}_q$ with respect to some primitive element α and used a geometric sum, which is zero over $\mathbb{F}_q$.

Remark. This scheme corresponds to an RS-code in the following way. An $[n, k]$-RS-code has as a basis

$$\{(x_1^j, \ldots, x_n^j) \mid 0 \leq j \leq k-1\}.$$

In the context of this scheme, $(x_1, \ldots, x_n)$ correspond to $(\alpha_1, \ldots, \alpha_{q-1})$. This means that we have a $[q-1, k]$-RS-code with generator matrix

$$G = \begin{pmatrix} 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_{q-1} \\ \vdots & & \vdots \\ \alpha_1^{k-1} & \cdots & \alpha_{q-1}^{k-1} \end{pmatrix}.$$

By definition of $\mathbf{D}$, we immediately observe that indeed $\mathbf{a}G = \mathbf{D}$.

This is in particular interesting since it allows for errors to be corrected, i.e., if at most t of the shares of $\mathbf{D}$ are altered, then it is still possible to recover the secret by applying an error-correcting algorithm. It is also possible to consider the whole message $\mathbf{a}$ as the secret since one can recover $\mathbf{a}$ knowing at least k shares D_i.
In the following, we want to give some more background about RS-codes that will explain in more detail why this works.

Lemma 4.3 ([21, Proposition 1.3.12])**.** *Let $C \subset \mathbb{F}_q^n$ be a linear code with parity check matrix H. Then $d(C)$ is equal to the minimal number d such that there exist d columns of H that are linearly dependent, but any $d-1$ columns of H are linearly independent.*

Proof. Denote by h_i the columns of H. We consider an element $0 \neq v \in \mathbb{F}_q^n$ such that $1 \leq \mathrm{wt}(v) \leq d-1$. It follows that

$$H \cdot v^T = \sum_{i=1}^{d-1} v_i h_i = \sum_{v_j \neq 0} v_j h_j = \sum_{j=1}^{d-1} v_{i_j} h_{i_j} \neq 0$$

since $h_{i_1}, \ldots, h_{i_{d-1}}$ are linearly independent by assumption. Thus, it follows that $v \notin C$ and hence $d(C) > d-1$. Next, we take $h_{i_1}, \ldots, h_{i_d}$ such that they are linearly dependent, namely there exist nonzero coefficients $\lambda_{i_1}, \ldots, \lambda_{i_d}$ such that

$$\sum_{j=1}^{d} \lambda_{i_j} h_{i_j} = 0.$$

Therefore, it follows that the vector v with $v_{i_j} = \lambda_{i_j}$ and zero everywhere else satisfies

$$H \cdot v^T = 0.$$

Hence, there exists a codeword with weight $\mathrm{wt}(v) = d$ and thus $d(C) = d$. □

Corollary 4.4 (Singleton Bound, [21, Theorem 2.2.1]). *Let $C \subset \mathbb{F}_q^n$ be a linear code of dimension k. Then*

$$d(C) \leq n - k + 1.$$

Proof. Let H be a parity check matrix of C. Then H has rank $n-k$ and hence any $n-k+1$ columns are linearly dependent. □

Definition 4.5 (MDS-Code). If $d(C) = n - k + 1$, then C is called *maximum distance separable code (MDS-code).*

Theorem 4.6 ([21, Proposition 2.2.5]). *The following are equivalent:*

1. *C is an MDS-code.*
2. *For any parity check matrix H, any $n-k$ columns are linearly independent.*
3. *For any generator matrix G, any k columns are linearly independent.*

Proof. 1 is equivalent to 2 by Lemma 4.3. 2 is equivalent to 3 by the relation of generator and parity check matrices. More precisely, we assume 3. Then any $k \times k$-submatrix has full rank and thus we can bring any generator matrix G into standard form and apply Proposition 1.3. In particular, we consider $G = (I_k|P)$ and obtain $H = (-P^T|I_{n-k})$. Next, we note that their minors are related. In fact, the $k \times k$-minors of G correspond to all minors of P, which again correspond to the $(n-k) \times (n-k)$-minors of H. Indeed, we denote the columns of G in a special way. Namely, we write J_i, $i = 1, \ldots, k$ for the columns of I_k and P_i, $i = k+1, \ldots, n$ for the columns of P. Then any $k \times k$-minor of G has the form

$$\det M = \det(J_{i_1}, \ldots, J_{i_l}, P_{i_{l+1}}, \ldots, P_{i_k}).$$

We can reduce the submatrix M to a smaller matrix M' by first removing all J_{i_j} and then make it a square matrix by removing all rows that share an index with the removed columns J_{i_j}. This means that if we removed the column J_j from M, then we also remove the j-th row from the remaining matrix $(P_{i_{l+1}}, \ldots, P_{i_k})$. This is essentially the same procedure that happens by applying Laplace expansion and hence by formally applying Laplace expansion and iteratively using it l times on the first column each, we see that

$$\det M = \det M' \text{ or } \det M = -\det M'.$$

We apply the same procedure to a submatrix of H but this time Laplace expansion happens from the right. Thus, we see that the $(n-k) \times (n-k)$-minors of H correspond to all minors of $-P^T$, which are the same as the minors of P up to sign. We treat the special case of the identity matrix separately, but it is clear that $\det I_k = \det I_{n-k} = 1$. Hence, we found the desired correspondence of minors. Next, we note that by assumption $\det M \neq 0$ and thus $\det M' \neq 0$. Therefore, by the correspondence of the minors it follows that all the $(n-k) \times (n-k)$-minors of H are nonzero and thus any $n-k$ columns must

be linearly independent. We can obtain an analogous result by assuming 2 and applying Proposition 1.3 in the other direction. Hence, it follows that all full size minors of G are not zero if and only if all full size minors of H are not zero and therefore we obtain that 3 is equivalent to 2. □

Lemma 4.7 (Vandermonde, [21, Lemma 2.2.11])**.** *Let K be a field and let $x_1, \dots, x_k \in K$. We consider the $k \times k$-matrix*

$$V(x_1, \dots, x_k) = \begin{pmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_k \\ \vdots & & \vdots \\ x_1^{k-1} & \cdots & x_k^{k-1} \end{pmatrix}.$$

Then

$$\det V(x_1, \dots, x_k) = \prod_{i>j} (x_i - x_j).$$

Proof. First we note that applying elementary row or column operations will not change the value of the determinant. We start with $V(x_1, \dots, x_k)$ and denote its rows by V_i, $1 \leq i \leq k$. Now we will perform multiple elementary row operations, namely we map

$$V_i \mapsto V_i - x_1 V_{i-1}, \ i = k, \dots, 2.$$

Thus, we obtain a new matrix with the same determinant. This matrix is

$$\tilde{V} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & x_2 - x_1 & \cdots & x_k - x_1 \\ \vdots & \vdots & & \vdots \\ 0 & x_2^{k-1} - x_1 x_2^{k-2} & \cdots & x_k^{k-1} - x_1 x_k^{k-2} \end{pmatrix}.$$

Then we apply Laplace expansion to the first column and by the linearity of the determinant we factor $(x_i - x_1)$ from each column and hence we obtain

$$\det(V(x_1, \dots, x_k)) = \det(\tilde{V}) = \left(\prod_{i=2}^{k} (x_i - x_1) \right) \det \begin{pmatrix} 1 & \cdots & 1 \\ x_2 & \cdots & x_k \\ \vdots & & \vdots \\ x_2^{k-1} & \cdots & x_k^{k-1} \end{pmatrix}.$$

Then we can apply the same procedure to

$$\begin{pmatrix} 1 & \cdots & 1 \\ x_2 & \cdots & x_k \\ \vdots & & \vdots \\ x_2^{k-1} & \cdots & x_k^{k-1} \end{pmatrix}$$

and hence proceed inductively. In the end, we will obtain

$$\det(V(x_1, \dots, x_k)) = \prod_{i=2}^{k} (x_i - x_1) \prod_{i=3}^{k} (x_i - x_2) \dots (x_k - x_{k-1}) = \prod_{i>j} (x_i - x_j).$$

□

Remark. A matrix of the form $V(x_1, \ldots, x_k)$ is called a *Vandermonde matrix*.

Proposition 4.8 ([21, Proposition 2.2.12])**.** *Any RS-code is also an MDS-code.*

Proof. Consider a generator matrix

$$G = \begin{pmatrix} 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_{q-1} \\ \vdots & & \vdots \\ \alpha_1^{k-1} & \cdots & \alpha_{q-1}^{k-1} \end{pmatrix}$$

and any $k \times k$ submatrix

$$V(\alpha_{i_1}, \ldots, \alpha_{i_k}) = \begin{pmatrix} 1 & \cdots & 1 \\ \alpha_{i_1} & \cdots & \alpha_{i_k} \\ \vdots & & \vdots \\ \alpha_{i_1}^{k-1} & \cdots & \alpha_{i_k}^{k-1} \end{pmatrix}.$$

Then by Lemma 4.7

$$\det V(\alpha_{i_1}, \ldots, \alpha_{i_k}) = \prod_{i_j > i_l} (\alpha_{i_j} - \alpha_{i_l}) \neq 0$$

since $\alpha_i \neq \alpha_j$ for all $i \neq j$. Therefore, any k columns are linearly independent and we conclude by Theorem 4.6. □

4.2 Secret Sharing based on AG-Codes

4.2.1 Linear Secret Sharing Schemes

Definition 4.9 (LSSS)**.** We consider a scheme with n players, where each player holds one share. An *adversary structure* $\mathcal{A}$ on the finite player set $\mathcal{U} = \{1, \ldots, n\}$ is a collection of subsets of $\mathcal{U}$ such that if $A' \in \mathcal{A}$ and $A \subset A'$, then it follows that also $A \in \mathcal{A}$. We denote by $\mathcal{A}_{t,n}$ the *t-threshold adversary structure*, i.e., all possible subsets of $\{1, \ldots, n\}$ with size at most t. By $\Gamma_{r,n}$, we denote the *threshold access structure*, i.e., all possible subsets of $\{1, \ldots, n\}$ with size at least r.
A *linear secret sharing scheme (LSSS)* $\mathcal{M}$ over the field $\mathbb{F}_q$ on the player set $\mathcal{U}$ is given by some positive integer $e \in \mathbb{N}$, a sequence of subspaces $V_1, \ldots, V_n \subset \mathbb{F}_q^e$ and a nonzero vector $u \in \mathbb{F}_q^e$. Without loss of generality we can set $u = (1, 0, \ldots, 0)$. We denote by $V_A \subset \mathbb{F}_q^e$ the subspace spanned by V_i, $i \in A$. Then we define the *access structure* $\Gamma(\mathcal{M})$ of $\mathcal{M}$ as the set consisting of all $B \subset \mathcal{U}$ such that $u \in V_B$ and we denote by $\mathcal{A}(\mathcal{M})$ the set consisting of all sets $A \subset \mathcal{U}$ such that $A \notin \Gamma(\mathcal{M})$. An LSSS is said to *reject* an adversary structure $\mathcal{A}$ if $\mathcal{A} \subset \mathcal{A}(\mathcal{M})$.

Secret sharing then works as follows. We assume that all subspaces V_i have dimension 1 and that they have fixed bases $\mathbf{b}_i$. Let $s \in \mathbb{F}_q$ be the secret and choose a random linear map $\phi : \mathbb{F}_q^e \to \mathbb{F}_q$ such that $\phi(u) = s$. We give a $\phi|_{V_i}$ to player i, i.e., the action of ϕ on the chosen basis vector $\mathbf{b}_i$ of V_i. We denote it by $s_i = \phi(\mathbf{b}_i)$, which is a *share* of s.

This definition follows Chen and Cramer [4] and differs from the definition in Cramer, Damgård, and Maurer [11]. In fact, it combines the definitions of MSP and LSSS as seen in Cramer, Damgård, and Maurer [11]. More precisely, an MSP is defined as follows.

Definition 4.10 (Monotone Span Program (MSP))**.** A *monotone span program (MSP)* $\mathcal{M}$ is a triple (K, M, ψ), where $K = \mathbb{F}_q$ is a finite field, M is an $e \times d$ matrix over K such that $e \leq d$ and $\psi : \{1, \ldots, d\} \to \{1, \ldots, n\}$ is a surjective map. In the following, we assume that the number of players n is equal to d, i.e., $M \in Mat_{e \times n}(\mathbb{F}_q)$ and $\psi = id$. We denote by $\mathbf{g}_i$ the i-th column of M and give it to player i. Next, we choose our secret $s \in K$ and a random vector $\rho \in K^{e-1}$ and define $\mathbf{b} = (s, \rho) \in K^e$. For each column i we compute $b_i = \mathbf{g}_i \cdot \mathbf{b}$ and give it to player i. If $A \subset \{1, \ldots, n\}$, then the players A can reconstruct the secret s if and only if $u = (1, 0, \ldots, 0) \in \operatorname{span}(\mathbf{g}_i : i \in A)$. Namely, if $u \in \operatorname{span}(\mathbf{g}_i : i \in A)$, then we can write

$$u = (1, 0, \ldots, 0) = \sum_{i \in A} \lambda_i \mathbf{g}_i$$

and hence

$$s = \mathbf{b} \cdot u = \mathbf{b} \cdot \sum_{i \in A} \lambda_i \mathbf{g}_i = \sum_{i \in A} \lambda_i b_i$$

can be computed by the knowledge of b_i. Now assume that $u \notin \operatorname{span}(\mathbf{g}_i : i \in A)$. It follows that $\dim \operatorname{span}(\mathbf{g}_i : i \in A) < e$. For simplicity and without loss of generality, we may assume that $\{\mathbf{g}_i\}_{i \in A}$ is a basis for $\operatorname{span}(\mathbf{g}_i : i \in A)$. Let $a \in \mathbb{F}_q$ be any element. Then we consider the following system of equations

$$\begin{cases} \mathbf{b}' \cdot \mathbf{g}_i = b_i & \text{for all } \ i \in A \\ \mathbf{b}' \cdot u = a \end{cases}.$$

In this system, we know $\mathbf{g}_i, b_i$ and a and we want to solve for $\mathbf{b}' \in \mathbb{F}_q^e$. Namely, we have a system with at most e equations but exactly e unknowns and we assumed $\{\mathbf{g}_i, u\}$ to be a set of linearly independent vectors. Thus, there exists at least one solution $\mathbf{b}'$. This vector $\mathbf{b}'$ satisfies $\mathbf{b}' \cdot \mathbf{g}_i = b_i$ for all $i \in A$ but $\mathbf{b}' \cdot u = a$. In particular, such a vector $\mathbf{b}'$ exists for any $a \in \mathbb{F}_q$ and moreover, any value $a = s$ is equally likely. To see this, let $\mathbf{f}_j$ be such that $\{u, \mathbf{g}_i, \mathbf{f}_j\}$ is a basis of $\mathbb{F}_q^e$ and let $m = |\{\mathbf{f}_j\}| \geq 0$. Then we fix $a \in \mathbb{F}_q$ and we observe that for all $f = (f_1, \ldots, f_m) \in \mathbb{F}_q^m$ the system

$$\begin{cases} \mathbf{b}' \cdot \mathbf{g}_i = b_i & \text{for all } \ i \in A \\ \mathbf{b}' \cdot \mathbf{f}_j = f_j & \text{for all } \ j = 1, \ldots, m \\ \mathbf{b}' \cdot u = a \end{cases}$$

has exactly one solution since $\{u, \mathbf{g}_i, \mathbf{f}_j\}$ is a basis. It is also clear that for two different vectors $f \neq f'$, we get two different solutions for $\mathbf{b}'$ and hence for every $a \in \mathbb{F}_q$ we have exactly q^m solutions to

$$\begin{cases} \mathbf{b}' \cdot \mathbf{g}_i = b_i & \text{for all } \ i \in A \\ \mathbf{b}' \cdot u = a \end{cases}.$$

Thus, for an adversary $A \in \mathcal{A}(\mathcal{M})$ knowing only $\mathbf{g}_i, i \in A$, any value $a \in \mathbb{F}_q$ such that $\mathbf{b}' \cdot u = a$ appears to be equally likely and it cannot determine the secret s any better than guessing it.

Lemma 4.11. *$A \in \mathcal{A}(\mathcal{M})$ if and only if there exists a linear map $\varphi : \mathbb{F}_q^e \to \mathbb{F}_q$ such that $\varphi|_{V_A} \equiv 0$ but $\varphi(u) = 1$.*

Proof. First note that $A \in \mathcal{A}(\mathcal{M})$ if and only if $u \notin V_A = \operatorname{span}(V_i \ : \ i \in A) \subset \mathbb{F}_q^e$. Now we assume that $u \notin V_A$. We choose a basis $\{B_j\}_{j=1,\ldots,e}$ of $\mathbb{F}_q^e$ such that $B_1 = u$ and $B_2, \ldots, B_l$ is a basis of V_A, where $l = \dim V_A + 1$. Then we set $\varphi(B_1) = \varphi(u) = 1$ and $\varphi(B_j) = 0$ for all $j = 2, \ldots, e$ and extend it linearly. This is clearly a linear map by definition and as $B_1 \notin V_A$, we have that $\varphi|_{V_A} \equiv 0$. On the other hand, assume that such a map φ exists. It is obvious that $u \notin V_A$ since $\varphi(x) = 0$ for all $x \in V_A$ but $\varphi(u) = 1$. □

Remark. Let us assume that a secret s was shared using an LSSS and that we have a set A of players who want to reconstruct s. Then, if $A \in \Gamma(\mathcal{M})$, we have that $u \in V_A$. Hence, we can write u as a linear combination of the vectors $\mathbf{b}_i, i \in A$ and thus learn $s = \phi(u)$ by linearity of ϕ and the knowledge of $s_i = \phi(\mathbf{b}_i)$. On the other hand, if $A \in \mathcal{A}$, we have that $u \notin V_A$ and we apply Lemma 4.11. For any φ such as in Lemma 4.11 and any $a \in \mathbb{F}_q$, we consider the map $\psi_a = \phi + (a - s)\varphi$. In particular, we have

$$\psi_a(u) = \phi(u) + (a - s)\varphi(u) = a$$

but

$$\psi_a|_{V_A} \equiv \phi|_{V_A}.$$

This means that the set of players A cannot determine the secret s any better than guessing it since for each $a \in \mathbb{F}_q$ there exists the same amount of maps ψ_a that coincide with ϕ on the set V_A but produce the secret a instead of s.
Using this observation and Lemma 4.12 below, we immediately obtain the desired property of an MSP described in Definition 4.10.

Lemma 4.12 ([2, Chapter 2.3, Korollar 6])**.** *Let $V = K^n$ be a vector space and denote by $\operatorname{Hom}(V, K)$ the space of all linear maps $\phi : V \to K$. We denote by $\cdot$ the standard inner product of two vectors in V and by $e_1, \ldots, e_n$ the standard basis. Then we have a $1 - 1$ correspondence between vectors in V and linear maps in $\operatorname{Hom}(V, K)$. More precisely, we have the maps*

$$\psi_1 : \operatorname{Hom}(V, K) \to V$$

$$\phi \mapsto (\phi(e_1), \ldots, \phi(e_n))$$

and

$$\psi_2 : V \to \mathrm{Hom}(V, K)$$

$$\mathbf{v} \mapsto \phi_{\mathbf{v}}, \text{ where } \phi_{\mathbf{v}}(\mathbf{x}) = \mathbf{v} \cdot \mathbf{x}$$

such that $\psi_1 \circ \psi_2 = \mathrm{id}_V$ *and* $\psi_2 \circ \psi_1 = \mathrm{id}_{\mathrm{Hom}(V,K)}$.

Proof. First, we note that from linearity it follows immediately that any linear map is uniquely determined by its action on a chosen basis and it is obvious that $\phi_{\mathbf{v}}$ defines a linear map. Indeed, $\mathbf{v} \cdot \lambda\mathbf{x} = \lambda(\mathbf{v} \cdot \mathbf{x})$ and $\mathbf{v} \cdot (\mathbf{x} + \mathbf{y}) = \mathbf{v} \cdot \mathbf{x} + \mathbf{v} \cdot \mathbf{y}$ follow directly from the definition of the inner product. Now let $\mathbf{v} \in V$. Then we have that

$$\psi_1(\psi_2(\mathbf{v})) = \psi_1(\phi_{\mathbf{v}}) = (\mathbf{v} \cdot e_1, \ldots, \mathbf{v} \cdot e_n) = \mathbf{v}.$$

On the other hand, for any $\phi \in \mathrm{Hom}(V, K)$ we have

$$\begin{aligned}\psi_2(\psi_1(\phi))(\mathbf{x}) &= \psi_2(\phi(e_1), \ldots, \phi(e_n))(\mathbf{x}) = (\phi(e_1), \ldots, \phi(e_n)) \cdot \mathbf{x} \\ &= \sum_{i=1}^{n} x_i \phi(e_i) = \sum_{i=1}^{n} \phi(x_i e_i) = \phi\left(\sum_{i=1}^{n} x_i e_i\right) = \phi(\mathbf{x})\end{aligned}$$

by linearity of ϕ. □

Example 12. Shamir's Secret Sharing Scheme is in fact an LSSS. Let us first fix the number of players n and choose a prime number $p > n$. Let $k < n$ and fix a random polynomial $f \in \mathbb{F}_p[x]_{<k}$, namely we write f as

$$f(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1} = \sum_{j=1}^{k} a_{j-1} x^{j-1}.$$

In the following, we consider the standard basis $\{1, x, \ldots, x^{k-1}\}$ of $\mathbb{F}_p[x]_{<k}$ and for all $1 \leq i \leq n$ the evaluation map

$$\mathrm{Ev}_i : \mathbb{F}_p[x]_{<k} \to \mathbb{F}_p$$

$$g \mapsto g(i).$$

We define the vectors

$$\mathbf{b}_i = (\mathrm{Ev}_i(1), \mathrm{Ev}_i(x), \ldots, \mathrm{Ev}_i(x^{k-1}))$$

and player i receives the vector $\mathbf{b}_i$. More precisely, in terms of LSSS this means that our vector space $\mathbb{F}_q^e$ is the space $\mathbb{F}_p^k$ and the vector space V_i given to player i is the space spanned by the vector $\mathbf{b}_i$. The map ϕ acts on the basis vector $\mathbf{b}_i$ as

$$\phi(\mathbf{b}_i) = s_i = f(i) = \sum_{j=1}^{k} a_{j-1} (\mathbf{b}_i)_j = (a_0, a_1, \ldots, a_{k-1}) \cdot \mathbf{b}_i.$$

This means, in addition to the vector $\mathbf{b}_i$ player i obtains the value $f(i) = s_i$. Due to the construction of the vector $\mathbf{b}_i$, it is not secret at all, however s_i is. As before, we set $u = (1, 0, \ldots, 0)$. Now with these prerequisites, we have defined the LSSS. Let us show that it works as intended, namely that any k players can recover the secret $a_0 = s_0$ but any $k-1$ or less players cannot do so. Assume that we have a set $A \subset \mathcal{U} = \{1, \ldots, n\}$ such that $|A| \geq k$. It follows from properties of the Vandermonde matrix and RS-codes that $V_A = \mathbb{F}_p^k$ and hence we have $u \in V_A$. This allows us to recover the secret as follows. Without loss of generality we assume that $|A| = k$ and we write u with respect to the basis $\{\mathbf{b}_i \mid i \in A\}$, namely

$$u = \sum_{i \in A} c_i \mathbf{b}_i$$

for some coefficients $c_i \in \mathbb{F}_p$. It follows that

$$s_0 = f(0) = a_0 = (a_0, a_1, \ldots, a_{k-1}) \cdot u = \sum_{i \in A} c_i (a_0, a_1, \ldots, a_{k-1}) \cdot \mathbf{b}_i = \sum_{i \in A} c_i s_i.$$

Hence, any k or more players can recover the secret s_0. Now assume that $|A| < k$. Without loss of generality we may assume, $|A| = k-1$ and we denote the elements of A by i_j. We note that $u \in V_A$ if and only if there exist $c_{i_1}, \ldots, c_{i_{k-1}}$ such that

$$\sum_{j=1}^{k-1} c_{i_j} \mathbf{b}_{i_j} = u,$$

which is equivalent to saying that

$$\left(\sum_{j=1}^{k-1} c_{i_j} \mathbf{b}_{i_j} \right)_1 = 1 \text{ and } \left(\sum_{j=1}^{k-1} c_{i_j} \mathbf{b}_{i_j} \right)_l = 0, \ l = 2, \ldots k-1.$$

Now we consider the matrix

$$(\mathbf{b}_{i_j})_{i_j \in A} = (\mathbf{b}_{i_1}^T, \ldots, \mathbf{b}_{i_{k-1}}^T) = \begin{pmatrix} 1 & \cdots & 1 \\ i_1 & \cdots & i_{k-1} \\ \vdots & \cdots & \vdots \\ i_1^{k-1} & \cdots & i_{k-1}^{k-1} \end{pmatrix},$$

and we remove the first row and scale each column by i_j^{-1} and obtain a $(k-1) \times (k-1)$ Vandermonde matrix

$$\begin{pmatrix} 1 & \cdots & 1 \\ i_1 & \cdots & i_{k-1} \\ \vdots & \cdots & \vdots \\ i_1^{k-2} & \cdots & i_{k-1}^{k-2} \end{pmatrix}.$$

This matrix has as columns the vectors $i_j^{-1}((\mathbf{b}_{i_j})_2, \ldots, (\mathbf{b}_{i_j})_{k-1})$ and, since it is a Vandermonde matrix with $i_{j_1} \neq i_{j_2}$ for all $j_1 \neq j_2$, it follows that all of its

columns are linearly independent by Lemma 4.7. In other words, the vectors $i_j^{-1}((\mathbf{b}_{i_j})_2, \ldots, (\mathbf{b}_{i_j})_{k-1})$ are linearly independent and hence $((\mathbf{b}_{i_j})_2, \ldots, (\mathbf{b}_{i_j})_{k-1})$ must also be linearly independent. This means that they can only add up to 0 if $c_{i_j} = 0$ for all j, which means that $\left(\sum_{j=1}^{k-1} c_{i_j}\mathbf{b}_{i_j}\right)_1 = 0 \neq 1$ and hence we cannot write u as a linear combination of $\mathbf{b}_{i_j}$. Therefore, we showed that Shamir's Secret Sharing Scheme is indeed an LSSS and we once again verified that it is in fact a threshold scheme.

Definition 4.13 (Strong Multiplication Property)**.** For $\mathbf{x} = (x_1, \ldots, x_e), \mathbf{y} = (y_1, \ldots, y_e) \in \mathbb{F}_q^e$, we denote by $\mathbf{x} \otimes \mathbf{y}$ the element

$$\mathbf{x} \otimes \mathbf{y} = (x_1 \cdot \mathbf{y}, \ldots, x_i \cdot \mathbf{y}, \ldots, x_e \cdot \mathbf{y}) = (x_1y_1, x_1y_2, \ldots, x_ey_e) \in \mathbb{F}_q^{e^2}.$$

By $\widehat{V_i}$, we denote the subspace

$$\widehat{V_i} = V_i \otimes V_i = \text{span}(\mathbf{x} \otimes \mathbf{y} \ : \ \mathbf{x}, \mathbf{y} \in V_i) = \text{span}(\mathbf{b}_i \otimes \mathbf{b}_i) \subset \mathbb{F}_q^{e^2}$$

and we use the notation $\widehat{\mathbf{b}_i} = \mathbf{b}_i \otimes \mathbf{b}_i$. Similarly, we use the notation $\widehat{V}_B$ for the space spanned by the spaces $\widehat{V_i}$, $i \in B$ and we write $\widehat{u} = u \otimes u$. We write $\widehat{\mathcal{M}}$ for the LSSS defined by $\widehat{V_1}, \ldots, \widehat{V_n}$ and $\widehat{u}$. Then $\mathcal{M}$ is said to have the *multiplication property* if $\widehat{u} \in \widehat{V_{\mathcal{U}}}$. $\mathcal{M}$ is said to have the *strong multiplication property* with respect to an adversary structure $\mathcal{A}$ if:

1. $\mathcal{M}$ rejects $\mathcal{A}$, i.e., $\mathcal{A} \subset \mathcal{A}(\mathcal{M})$.
2. For all $B \subset U$ of the form $B = \mathcal{U} \backslash A$ for some $A \in \mathcal{A}$, $\widehat{\mathcal{M}}_B$ has the multiplication property.

Remark. We note that for any set $D \subset \mathcal{U}$ such that $\widehat{u} \in \widehat{V}_D$, we also have $u \in V_D$. Indeed, assume that $\widehat{u} \in \widehat{V}_D$ and write $\mathbf{b}_i$ for the basis vector of V_i, then we note that there exist λ_i such that

$$(1, 0, \ldots, 0) = \widehat{u} = \sum_{i \in D} \lambda_i \widehat{\mathbf{b}_i} = \sum_{i \in D} \lambda_i (b_{i1}b_{i1}, b_{i1}b_{i2}, \ldots, b_{ie}b_{ie}).$$

We truncate all these vectors after e entries and obtain

$$(1, 0, \ldots, 0) = u = \sum_{i \in D} \lambda_i (b_{i1}b_{i1}, \ldots, b_{i1}b_{ie}) = \sum_{i \in D} \lambda_i b_{i1} \mathbf{b}_i$$

and hence $u \in V_D$ and $D \in \Gamma(\mathcal{M})$.

Example 13. We consider the following LSSS over any field K. Let $e = n = 2$ and define

$$V_1 = \text{span}((0, 1)), \quad V_2 = \text{span}((1, 1)).$$

It is clear that $u = (1, 0) \in \text{span}(V_1, V_2)$ and

$$\widehat{V_1} = \text{span}((0, 0, 0, 1)), \quad \widehat{V_2} = \text{span}((1, 1, 1, 1)).$$

Hence, we immediately observe that $\widehat{u} = (1, 0, 0, 0) \notin \text{span}(\widehat{V_1}, \widehat{V_2})$ and thus this LSSS does not have the multiplication property.

Remark. It is clear from the definition that having the strong multiplication property or not does only depend on the choice of our vector spaces V_i and is completely independent of the choice of the linear map ϕ. Thus, it was not necessary to specify a linear map ϕ in the example above.

Lemma 4.14. *Let $A \in \mathcal{A} \subset \mathcal{A}(\mathcal{M})$ and define $B = \mathcal{U} \backslash A$ and write $|B| = d$. Without loss of generality we may assume that $B = \{1, \dots, d\}$. For any linear map ϕ we write $s = \phi(u)$ and $s_i = \phi(\mathbf{b}_i)$, where $V_i = \operatorname{span}(\mathbf{b}_i)$. Then $\mathcal{M}$ has the strong multiplication property with respect to $\mathcal{A}$ if and only if for all such sets B there exists a so called* recombination vector *$\mathbf{r} \in \mathbb{F}_q^d$ such that for any two linear maps ϕ and ϕ' we have*

$$\phi(u)\phi'(u) = ss' = \mathbf{r} \cdot (s_1 s_1', \dots, s_d s_d') = \mathbf{r} \cdot (\phi(\mathbf{b}_1)\phi'(\mathbf{b}_1), \dots, \phi(\mathbf{b}_d)\phi'(\mathbf{b}_d)).$$

Proof. We note that by Lemma 4.12 we can represent the maps ϕ, ϕ' as the inner product with some vector $\mathbf{a}, \mathbf{a}'$ respectively and we note that

$$s_i s_i' = (\mathbf{a} \cdot \mathbf{b}_i)(\mathbf{a}' \cdot \mathbf{b}_i) = \left(\sum_j a_j b_{ij} \right) \left(\sum_k a_k' b_{ik} \right) = \sum_{j,k} a_j a_k' b_{ij} b_{ik} = (\mathbf{a} \otimes \mathbf{a}') \cdot (\mathbf{b}_i \otimes \mathbf{b}_i).$$

Hence, it follows that

$$\begin{aligned}(s_1 s_1', \dots, s_d s_d') &= ((\mathbf{a} \cdot \mathbf{b}_1)(\mathbf{a}' \cdot \mathbf{b}_1), \dots, (\mathbf{a} \cdot \mathbf{b}_d)(\mathbf{a}' \cdot \mathbf{b}_d)) \\ &= ((\mathbf{a} \otimes \mathbf{a}') \cdot (\mathbf{b}_1 \otimes \mathbf{b}_1), \dots, (\mathbf{a} \otimes \mathbf{a}') \cdot (\mathbf{b}_d \otimes \mathbf{b}_d)).\end{aligned}$$

Now, we fix some set B and assume that there exists a vector $\mathbf{r}$ such that for any ϕ, ϕ' we have

$$ss' = \mathbf{r} \cdot (s_1 s_1', \dots, s_d s_d').$$

and thus

$$ss' = (\mathbf{a} \otimes \mathbf{a}')_1 = (\mathbf{a} \otimes \mathbf{a}') \cdot \widehat{u} = \sum_{i \in B} r_i (\mathbf{a} \otimes \mathbf{a}') \cdot \widehat{\mathbf{b}}_i = (\mathbf{a} \otimes \mathbf{a}') \cdot \left(\sum_{i \in B} r_i \widehat{\mathbf{b}}_i \right).$$

This must hold for any $\mathbf{a}, \mathbf{a}'$, which is only possible if $\sum_{i \in B} r_i \widehat{\mathbf{b}}_i = \widehat{u}$. For the other direction, let $\widehat{u} = \sum_{i \in B} \lambda_i \widehat{\mathbf{b}}_i$. Then by setting $\mathbf{r} = (\lambda_1, \dots, \lambda_d)$ we obtain a vector that satisfies the properties. In particular,

$$\sum_{i \in B} \lambda_i s_i s_i' = \sum_{i \in B} \lambda_i (\mathbf{a} \otimes \mathbf{a}') \cdot \widehat{\mathbf{b}}_i = (\mathbf{a} \otimes \mathbf{a}') \cdot \widehat{u} = ss'.$$

□

Corollary 4.15. *An LSSS $\mathcal{M}$ has the multiplication property if and only if there exists a recombination vector $\mathbf{r} \in \mathbb{F}_q^n$ such that for any two linear maps ϕ, ϕ' we have*

$$\phi(u)\phi'(u) = ss' = \mathbf{r} \cdot (s_1 s_1', \dots, s_n s_n') = \mathbf{r} \cdot (\phi(\mathbf{b}_1)\phi'(\mathbf{b}_1), \dots, \phi(\mathbf{b}_n)\phi'(\mathbf{b}_n)).$$

Remark. An adversary structure $\mathcal{A}$ is called *Q3* if for all $A, B, C \in \mathcal{A}$ the union $A \cup B \cup C$ is a proper subset of $\mathcal{U} = \{1, \ldots, n\}$. By Lemma 4.14, it follows that if $A, B \in \mathcal{A}$, then $\mathcal{U} \backslash (A \cup B) \in \Gamma(\mathcal{M})$ and hence it follows that any $\mathcal{A}$ that has the strong multiplication property must also be Q3. To see that, we consider $A, B \in \mathcal{A}$ and set $C = \mathcal{U} \backslash (A \cup B) = (A \cup B)^c$, i.e., the minimal set such that $A \cup B \cup C = \mathcal{U}$. By Lemma 4.11, there exist linear maps ϕ_A, ϕ_B such that

- $\phi_A(\mathbf{x}) = 0$ for all $\mathbf{x} \in V_A$,
- $\phi_B(\mathbf{x}) = 0$ for all $\mathbf{x} \in V_B$,
- $\phi_A(u) = \phi_B(u) = 1$.

In particular, we have $\phi_A(\mathbf{b}_i) = 0$ for all $i \in A$ and $\phi_B(\mathbf{b}_i) = 0$ for all $i \in B$ and thus $\phi_A(\mathbf{b}_i)\phi_B(\mathbf{b}_i) = 0$ for all $i \in A \cup B = C^c$. Suppose now by contradiction that $C \in \mathcal{A}$. Then, by the strong multiplication property and Lemma 4.14 for the set $C^c = A \cup B$, there exists a recombination vector $\mathbf{r}$ such that

$$1 = \phi_A(u)\phi_B(u) = \mathbf{r} \cdot (\phi_A(\mathbf{b}_i)\phi_B(\mathbf{b}_i))_{i \in A \cup B} = \mathbf{r} \cdot (0, \ldots, 0) = 0.$$

This is a contradiction and hence $C \notin \mathcal{A}$, which is equivalent to saying that $\mathcal{U} \backslash (A \cup B) \in \Gamma(\mathcal{M})$.

4.2.2 An LSSS using AG-Codes

We define the LSSS $\mathcal{M}$ in the following way. Let X be a curve over $\mathbb{F}_q$ with genus g and let

$$Q, P_0, P_1, \ldots, P_n$$

be any distinct rational points on X. Let t be an integer such that $1 \leq t < n - 2g$ and define the divisor

$$D = (2g + t) \cdot Q.$$

Thus, it is clear that $\deg D = 2g + t$ and D has disjoint support from $P_0, P_1, \ldots, P_n$. The scheme works as follows. Let $s \in \mathbb{F}_q$ be the secret and select a random function $f \in L(D)$ such that $f(P_0) = s$. We define

$$f(P_1) = s_1, \ldots, f(P_n) = s_n$$

as the shares.

Remark. Before, we denoted by t the error correcting capacity. However, now t is just some chosen integer and it is important not to confuse the two. As described later in Corollary 4.18, t appears as a parameter that determines which sets of players can or cannot recover a secret.

Lemma 4.16 ([4, Lemma 1])**.** *Let E be an $\mathbb{F}_q$-divisor on X with $\deg E = m$ and assume that $d = \dim L(E) > 0$. Then each $f \in L(E)$ is uniquely determined by evaluations of f at any $m+1$ rational points outside the support of E. Moreover,*

for any additional rational point Q_0 distinct from $Q_1, \ldots, Q_{m+1}$ and outside the support of E, there exist $\lambda_1, \ldots, \lambda_{m+1}$ such that for all $f \in L(E)$ we have

$$f(Q_0) = \sum_{i=1}^{m+1} \lambda_i f(Q_i).$$

Proof. Since $f \in L(E)$, it follows that f has no poles outside the support of E and thus the value is defined outside the support of E. Consider any $m+1$ rational points $Q_1, \ldots, Q_{m+1}$ outside the support of E. The map

$$\mathrm{Ev} : L(E) \to \mathbb{F}_q^{m+1}$$

$$f \mapsto (f(Q_1), \ldots, f(Q_{m+1}))$$

is an injective linear map of $\mathbb{F}_q$-vector spaces by the same method as used in the proof of Theorem 1.39. Hence, the preimage $f^{-1}(\{\mathrm{Ev}(f)\}) = \{f\}$ is unique. More precisely, we identify $L(E) \cong \mathbb{F}_q^d$ and for a basis $\{b_1, \ldots, b_d\}$ of $L(E)$, we obtain a basis $\{\mathrm{Ev}(b_1), \ldots, \mathrm{Ev}(b_d), B_{d+1}, \ldots, B_{m+1}\}$ of $\mathbb{F}_q^{m+1}$. We set

$$\chi(\mathrm{Ev}(b_i)) = b_i \text{ and } \chi(B_j) = 0.$$

Thus, χ defines a linear map and we obtain the sequence

$$\mathbb{F}_q^d \xhookrightarrow{\mathrm{Ev}} \mathbb{F}_q^{m+1} \xrightarrow{\chi} \!\!\!\rightarrow \mathbb{F}_q^d \xrightarrow{\mathrm{Ev}_0} \mathbb{F}_q,$$

where $\mathrm{Ev}_0(f) = f(Q_0)$. Moreover, by definition χ satisfies

$$\chi \circ \mathrm{Ev} = \mathrm{id}_{L(E)}.$$

We note that

$$\mathrm{Ev}_0 \circ \chi : \mathbb{F}_q^{m+1} \to \mathbb{F}_q$$

is the composition of two linear maps and thus is itself a linear map. Hence, by Lemma 4.12 there exists a vector $\lambda = (\lambda_1, \ldots, \lambda_{m+1})$ such that for all $\mathbf{x} \in \mathbb{F}_q^{m+1}$ we have

$$\mathrm{Ev}_0 \circ \chi(\mathbf{x}) = \lambda \cdot \mathbf{x}.$$

We consider an element $\mathrm{Ev}(f) = (f(Q_1), \ldots, f(Q_{m+1})) \in \mathbb{F}_q^{m+1}$ and observe that

$$f(Q_0) = \mathrm{Ev}_0 \circ \chi(\mathrm{Ev}(f)) = \lambda \cdot (f(Q_1), \ldots, f(Q_{m+1}))$$

and therefore

$$f(Q_0) = \sum_{i=1}^{m+1} \lambda_i f(Q_i)$$

for all $f \in L(E)$. □

Remark. The evaluations at any $m+1$ points already determine a unique $f \in L(E)$. However, if we have more points, then the evaluations still determine the same f and the proof still holds. So in fact, we can replace $m+1$ with any integer $m' > m$, given that enough points Q_i exist.

This LSSS works analogously to Example 12. We note that the principal divisor of the constant function 1 is $(1) = 0$ and as $D \geq 0$, it follows that $(1) + D \geq 0$ and thus $1 \in L(D)$. We set $B_1 = 1 \in L(D)$. Then we can choose a basis $\mathcal{B}' = \{B_1, B_2', \ldots, B_d'\}$ of $L(D)$, where the dimension $d = g + t + 1$ by Corollary 1.25. For all $i \geq 2$ we set

$$B_i = B_i' - B_i'(P_0)B_1 = B_i' - B_i'(P_0).$$

Then we have the new basis $\mathcal{B} = \{B_1, B_2, \ldots, B_d\}$, which is clearly a basis by construction. We note that it has the following important property:

$$B_1(P_0) = 1 \quad \text{and} \quad B_i(P_0) = 0 \text{ for all } i \geq 2.$$

For an element $g \in L(D)$, we write it in this basis, i.e.,

$$g = \sum_{j=1}^{d} a_j B_j$$

and we define by $\mathbf{a}_g = (a_1, \ldots, a_d)$ the corresponding vector of the coefficients. Thanks to the special choice of the basis $\mathcal{B}$, it follows immediately that for all $g \in L(D)$, we have $a_1 = g(P_0)$. This property is essential for recovering the secret and proving Theorem 4.17. Namely, the LSSS and secret sharing then works as follows. We choose any $f \in L(D)$ and write it in this basis, i.e., we consider the corresponding vector $\mathbf{a}_f$ and hence we have $s = a_1 = f(P_0)$. Similarly as before in Example 12, we define the vectors

$$\mathbf{b}_i = (B_1(P_i), \ldots, B_d(P_i)) \quad \text{for all } i = 1, \ldots, n$$

and we give the vector $\mathbf{b}_i$ to player i. Furthermore, player i receives the value $s_i = f(P_i)$. Then ϕ acts on the basis vector $\mathbf{b}_i$ of V_i as

$$\phi(\mathbf{b}_i) = s_i = f(P_i) = \mathbf{a}_f \cdot \mathbf{b}_i.$$

If we have a set $A \subset \mathcal{U} = \{1, \ldots, n\}$ such that $u = (1, 0, \ldots, 0) \in V_A$, then we can recover the secret $s = f(P_0)$. More precisely, if $u \in V_A$, then we can write it with respect to $\{\mathbf{b}_i \mid i \in A\}$, i.e., let

$$u = \sum_{i \in A} c_i \mathbf{b}_i.$$

Then

$$s = f(P_0) = a_1 = \mathbf{a}_f \cdot u = \mathbf{a}_f \cdot \sum_{i \in A} c_i \mathbf{b}_i = \sum_{i \in A} c_i \mathbf{b}_i \cdot \mathbf{a}_f = \sum_{i \in A} c_i s_i.$$

Remark. This scheme is obviously related to the notion of AG-codes. If we take $\mathcal{P} = (P_1, \ldots, P_n)$, then the vector $\mathbf{s} = (s_1, \ldots, s_n)$ is a codeword of the

code $C_L(X, \mathcal{P}, D)$ and decoding algorithms can be applied. More precisely, the matrix G with columns $\mathbf{b}_i$ is a generator matrix of this code and

$$\mathbf{a}_f G = \mathbf{s}.$$

This means that if we assume G to be known by all players and somebody knows enough shares s_i, it is possible to apply a decoding algorithm and recover both the whole codeword $\mathbf{s}$ and the message $\mathbf{a}_f$ and hence the secret $s = a_1 = \mathbf{a}_{f1}$. Note that such a decoding algorithm can also be applied if some shares are corrupted and that the number of correct shares needed depends on the decoding algorithm.

Example 14. Consider the Hermitian curve $\mathcal{H}_3$ over the field $\mathbb{F}_9 = \mathbb{F}_3[a]_{/(a^2+1)}$ defined by the affine equation

$$Y^3 + Y = X^4.$$

This curve has as only point at infinity $Q = (0 : 1 : 0)$ and it has genus $g = 3$ and $3^3 = 27$ affine points. We choose $t = 2$ and $n = 26$, i.e., the maximal number possible since we also need an additional point P_0. This means that $m = 2g + t = 8$. Recall that by Proposition 3.17, $L(mQ)$ has the (affine) basis

$$\{x^i y^j \mid 0 \leq i, 0 \leq j \leq 2, 3i + 4j \leq 8\} = \{x^i y^j \mid (i, j) \in A(8)\},$$

where

$$A(8) = \{(0,0), (0,1), (0,2), (1,0), (1,1), (2,0)\}.$$

It follows that $L(8Q)$ has the basis

$$\{1, y, y^2, x, xy, x^2\}.$$

If $P_0 = (\alpha, \beta)$, then we can apply a base change by

$$(x, y) \mapsto (x - \alpha, y - \beta),$$

or equivalently in projective coordinates

$$(x : y : z) \mapsto (x - \alpha z : y - \beta z : z).$$

Then it follows that by setting $B_1 = 1$, we have $B_1(P_0) = 1$ and $B_i(P_0) = 0$ for all $i \neq 1$. This means that in this basis $a_1 = f(P_0)$ for all $f \in L(8Q)$ and hence the LSSS works as described before.

Theorem 4.17 ([4, Proposition 1])**.** *For a subset $A \subset \{1, \ldots, n\}$ we denote by D_A the divisor*

$$D_A = \sum_{i \in A} P_i.$$

Then $\mathcal{A}(\mathcal{M})$ consists of all sets A such that

$$\dim L(D - (P_0 + D_A)) < \dim L(D - D_A).$$

Equivalently, $\Gamma(\mathcal{M})$ consists of all sets A such that

$$\dim L(D - (P_0 + D_A)) = \dim L(D - D_A).$$

Proof. By Lemma 4.11 it follows that $A \in \mathcal{A}(\mathcal{M})$ if and only if there exists $g \in L(D)$ such that $g(P_0) = 1$ and $g(P_i) = 0$ for all $i \in A$. Namely, the linear map ϕ_g corresponding to $g \in L(D)$ is

$$\phi_g : \mathbb{F}_q^d \to \mathbb{F}_q,$$

$$\mathbf{x} \mapsto \phi_g(\mathbf{x}) = \mathbf{a}_g \cdot \mathbf{x},$$

where $\mathbf{a}_g = (a_1, \ldots, a_d)$ represents the coefficients of g with respect to the basis $\mathcal{B}$. In particular, as seen before,

$$g(P_i) = \mathbf{a}_g \cdot \mathbf{b}_i = \phi_g(\mathbf{b}_i) = 0 \text{ for all } i \in A$$

and

$$g(P_0) = \phi_g(u) = 1.$$

Hence, we are in the case of Lemma 4.11. Now we want to determine in what cases such a $g \in L(D)$ exists. By definition, D has disjoint support from P_0 and D_A and hence $D - (P_0 + D_A) \leq D - D_A \leq D$, which implies that

$$L(D - (P_0 + D_A)) \subset L(D - D_A) \subset L(D).$$

Any function

$$g' \in L(D - D_A) \backslash L(D - (P_0 + D_A))$$

satisfies $g'(P_0) \neq 0$ and $g'(P_i) = 0$ for all $i \in A$. This is clear as g' must have a zero wherever the divisor is negative. By normalizing g' at P_0, we obtain a function of the desired form g. Clearly,

$$L(D - D_A) \backslash L(D - (P_0 + D_A)) \neq \emptyset$$

if and only if they have different dimensions. The equivalent statement about $\Gamma(\mathcal{M})$ follows immediately from the definition of $\mathcal{A}(\mathcal{M})$ as the complement of $\Gamma(\mathcal{M})$ and the fact that

$$\dim L(D - (P_0 + D_A)) \leq \dim L(D - D_A).$$

□

Corollary 4.18 ([4, Corollaries 1 and 2])**.** *1.* $\mathcal{A}_{t,n} \subset \mathcal{A}(\mathcal{M})$.

2. $\Gamma_{2g+t+1,n} \subset \Gamma(\mathcal{M})$.

Proof. 1. If $|A| = a \leq t$, then

$$\deg(D - (P_0 + D_A)) = (2g + t) - (1 + a) \geq 2g + t - 1 - t = 2g - 1$$

and

$$\deg(D - D_A) = (2g + t) - a \geq 2g + t - t = 2g.$$

Thus by Corollary 1.25, we have

$$\dim L(D - (P_0 + D_A)) = g + t - a < g + t - a + 1 = \dim L(D - D_A).$$

2. By definition, we have $n \geq 2g + t + 1$. If $|A| = a \geq 2g + t + 1$, then

$$\deg(D - D_A) = (2g + t) - a \leq 2g + t - 2g - t - 1 = -1 < 0.$$

Namely, the degree is negative and hence

$$\dim L(D - (P_0 + D_A)) = \dim L(D - D_A) = 0.$$

□

Remark. In other words, this means that if we know the shares corresponding to a set $B \subset \{1, \ldots, n\}$ such that $|B| \geq 2g+t+1$, we can recover the secret. On the other hand, if $|B| \leq t$ we cannot recover the secret. For $t < |B| < 2g+t+1$, anything might happen, including partial reveal of the secret. Such a scheme is sometimes called a *ramp scheme*.

Remark. If we choose the curve X to be $X = \mathbb{P}^1$, then we have $g(X) = 0$. Hence, we obtain $\mathcal{A}_{t,n} \subset \mathcal{A}(\mathcal{M})$ and $\Gamma_{t+1,n} \subset \Gamma(\mathcal{M})$, i.e., we have a threshold scheme. As seen in Example 7, this corresponds to an RS-code, which corresponds to Shamir's Scheme.

Example 15. We recall the scheme from Example 14. Namely, we take the Hermitian curve $\mathcal{H}_3$ over the field $\mathbb{F}_9 = \mathbb{F}_3[a]_{/(a^2+1)}$ defined by the affine equation

$$Y^3 + Y = X^4.$$

We take $t = 2$ and $n = 10$. For simplicity, we take $P_0 = (0, 0) \in \mathcal{H}_3$ and work in affine coordinates. Hence, we consider the space $L(8Q)$ with affine basis

$$\mathcal{B} = \{B_1, \ldots, B_6\} = \{1, y, y^2, x, xy, x^2\}.$$

Therefore by Corollary 4.18, it follows that any set of 2 or less players cannot recover the secret, while any set of 9 or more players can, thus our choice of $n = 10$. We have not chosen the points $P_1, \ldots, P_{10}$, yet but the corollary holds for any choice. We choose the following points:

$$P_1 = (0, a),\ P_2(1, 2),\ P_3 = (a, 2),\ P_4 = (2, 2),\ P_5 = (2, a + 2),$$

$$P_6 = (0, 2a),\ P_7 = (a+1, 1),\ P_8 = (2a, 2),\ P_9 = (2a, a+2),\ P_{10} = (a+1, a+1),$$

where $a \in \mathbb{F}_9$ is such that $a^2 = 2$. Next, we list our basis evaluated at these points, i.e., the basis vectors $\mathbf{b}_i$ corresponding to V_i that are distributed among

the players. Namely, we get

$$\begin{aligned}
\mathbf{b}_1 &= (1, a, 2, 0, 0, 0),\\
\mathbf{b}_2 &= (1, 2, 1, 1, 2, 1),\\
\mathbf{b}_3 &= (1, 2, 1, a, 2a, 2),\\
\mathbf{b}_4 &= (1, 2, 1, 2, 1, 1),\\
\mathbf{b}_5 &= (1, a+2, a, 2, 2a+1, 1),\\
\mathbf{b}_6 &= (1, 2a, 2, 0, 0, 0),\\
\mathbf{b}_7 &= (1, 1, 1, a+1, a+1, 2a),\\
\mathbf{b}_8 &= (1, 2, 1, 2a, a, 2),\\
\mathbf{b}_9 &= (1, a+2, a, 2a, a+1, 2),\\
\mathbf{b}_{10} &= (1, a+1, 2a, a+1, 2a, 2a).
\end{aligned}$$

Next, we choose some function $f \in L(8Q)$. In fact with this set up, we can choose different functions but use the same vectors $\mathbf{b}_i$ to distribute different secrets s. Let us take the function $f = 2a + xy + (a+1)x + 2y^2 + ay$, i.e., we consider the vector

$$\mathbf{a}_f = (2a, a, 2, a+1, 1, 0).$$

Hence, our secret is $s = 2a$. Now we compute $s_i = \mathbf{a}_f \cdot \mathbf{b}_i$ and give the value to player i. Those are

$$s_1 = 2a,\ s_2 = 2a+2,\ s_3 = a+1,\ s_4 = 2,\ s_5 = a+2,$$

$$s_6 = 2a+2,\ s_7 = 0,\ s_8 = a,\ s_9 = 1,\ s_{10} = 2a+2.$$

With this setup, we can now give a concrete example of Corollary 4.18. For example, if we consider the set $A = \{1, \ldots, 9\}$, the corollary ensures that $u \in V_A$. Indeed, we note that if we take the vector $\lambda = (2a+1, a+2, 1, 2a+2, 0, a+1, 0, 0, 0)$, then we have

$$\sum_{i \in A} \lambda_i \mathbf{b}_i = u.$$

We also note that as some entries of λ are 0, we even have that $u \in V_{A'}$, where $A' = \{1, 2, 3, 4, 6\}$. So in fact, one of the player sets that we learn nothing about from the corollary can also reconstruct the secret but is much smaller than the number of players that is guaranteed to be able to reconstruct the secret. Let us check:

$$\begin{aligned}
\sum_{i \in A} \lambda_i s_i &= \sum_{i \in A'} \lambda_i s_i\\
&= 2a(2a+1) + (a+2)(2a+2) + (a+1) + 2(2a+2) + (a+1)(2a+2)\\
&= 2a + 2 + 2 + a + 1 + 2 + a + a\\
&= 2a = s.
\end{aligned}$$

This example shows that we can in fact have a set of players A such that $t < |A| < 2g + t + 1$ such that $u \in V_A$. On the other hand, there can also be such sets A such that $u \notin V_A$. The point is that we do not know how those sets A behave as the corollary does not apply to them. We can also add up to 16 more players by using more points on $\mathcal{H}_3$.

Proposition 4.19 ([4, Proposition 2])**.** *If $3t < n - 4g$, then $\mathcal{M}$ has the strong multiplication property with respect to $\mathcal{A}_{t,n}$.*

Proof. Let $h_1, h_2 \in L(D)$. Then

$$0 \leq ((h_1) + D) + ((h_2) + D) \leq (h_1 h_2) + 2D$$

and hence $h_1 h_2 \in L(2D)$. Thus, it follows that $\mathcal{M}$ has the strong multiplication property if

$$n - t > \deg(2D) = 4g + 2t$$

by Lemma 4.16. Indeed, let $A \in \mathcal{A}_{t,n}$ and consider $B = A^c$. Then $|B| \geq n - t \geq 4g+2t+1$, so in the following we assume $B \subset \{1, \ldots, n\}$ and $|B| = 4g+2t+1$. We consider the linear maps $\phi = \phi_{h_1}$ and $\phi' = \phi_{h_2}$ defined as the inner product with their coefficients $\mathbf{a}_{h_1}$ and $\mathbf{a}_{h_2}$ respectively. We set $s = \phi_{h_1}(u)$ and $s' = \phi_{h_2}(u)$. We have $l = |B| > \deg(2D)$ and therefore we can apply Lemma 4.16. This means that there exists some vector $\mathbf{r} = (\lambda_1, \ldots, \lambda_l)$ such that for any $h \in L(2D)$,

$$h(P_0) = \sum \lambda_i h(P_i) = \mathbf{r} \cdot (h(P_{i_1}), \ldots, h(P_{i_l})).$$

In our case, we have $h = h_1 h_2 \in L(2D)$ and thus it follows that

$$h_1 h_2(P_0) = \mathbf{r} \cdot (h_1 h_2(P_{i_1}), \ldots, h_1 h_2(P_{i_l})) = \mathbf{r} \cdot (s_i s'_i)_{i \in B}.$$

This must hold for any such B and hence we conclude by Lemma 4.14. □

Remark. For Shamir's Secret Sharing Scheme with threshold $t + 1$, we immediately see that it is Q3 if and only if $n \geq 3t + 1$. This scheme corresponds to an RS-code, which itself corresponds to an AG-code over $\mathbb{P}^1$ of the form $C_L(\mathbb{P}^1, \mathcal{P}, tP_\infty)$, where $\mathcal{P}$ is a vector of length n. The curve $X = \mathbb{P}^1$ has genus 0 and hence it follows by Proposition 4.19 that this scheme has the strong multiplication property if $n > 3t$. The strong multiplication property implies the $Q3$ property and hence we conclude that in the case of Shamir's Secret Sharing Scheme, being $Q3$ is equivalent to having the strong multiplication property.

5 Verifiable Secret Sharing and Secure Multi Party Computation

In this section, we first use Shamir's Secret Sharing Scheme to realize what is called VSS and MPC following Gennaro, Rabin, and Rabin [14] and then proceed with the more general construction as seen in Cramer, Damgård, and Maurer [11].

5.1 VSS and MPC

In this section, we consider a setting with n players denoted by $P_1, \ldots, P_n$ and an additional special player D, called the *dealer*, who will share some secret s. When we use the term player, we usually do not mean the dealer. We assume that between any two players, and also the dealer, there is a secure private channel and we also have a public broadcast channel. If some player or the dealer sends a public message through the public broadcast channel, then every other player can read the message and also learns who was the sender of that message. Some players might be corrupted by an *adversary* $\mathcal{A}$, which means that they do not play by the rules. We will call them *bad players*. Furthermore, the dealer D might be corrupted as well in addition to the players corrupted by $\mathcal{A}$. The honest players that are playing by the rules are called *good players*. For the remainder of Section 5.1, we assume that an adversary can corrupt at most t players plus the dealer and we assume it to be *static*, i.e., it can only do so at the beginning of the protocol. In particular an adversary can

- read all the public information shared via the broadcast channel,
- read all the private information given to a corrupted player via the private channel,
- coordinate the actions of the corrupted players, i.e., the adversary can make them refuse to open their share or make them broadcast incorrect values while claiming they were correct.

Such an adversary is called *active*. An adversary is called *passive* if it cannot coordinate the actions of the bad players but can only read their private information. No kind of adversary is able to read the private information of the good players given to them by a good dealer nor can it actively manipulate their output. Furthermore, we assume that the adversary is computationally bounded. This means that it cannot freely solve any problem but only those that are solvable in a feasible amount of time on a given computer. In practice, this means that we assume our adversary to be able to run polynomial time algorithms or the so called probabilistic polynomial time algorithms. In the following, we will see protocols that ensure that an adversary cannot sabotage the sharing of a secret. We also assume that there are more good players than bad players, i.e., since we have at most t bad players that means that we have at least $t + 1$ good players. This assumption is necessary to ensure that the good players can reconstruct a secret as shown later in Example 16.

Definition 5.1 (VSS, [13, Appendix B])**.** A *verifiable secret sharing (VSS)* protocol consists of two phases called *Share* and *Reconstruct*. All players know a set S of possible secrets. After the phase Share, each player P_i outputs a value $ver_i \in \{0, 1\}$ and after the phase Reconstruct, each player outputs a value $s_i \in S$ or the default value $* \notin S$. Such a protocol is called a VSS if for all adversaries $\mathcal{A}$ it satisfies the following properties:

1. If any good player P_i outputs $ver_i = 1$ at the end of Share, then $ver_j = 1$ for all good players P_j.

2. If the dealer D is good, then $ver_i = 1$ for all good players P_i.

3. If a good player P_i outputs $ver_i = 1$ at the end of Share, then there exists a fixed value $\sigma \in S \cup \{*\}$ such that all good players will output $s_i = \sigma$ at the end of Reconstruct. Moreover, if the dealer D is good, then $\sigma = s$.

4. If the secret s is chosen randomly from the set S with cardinality $|S| = q$ and the dealer D is good, then an adversary $\mathcal{A}$ cannot guess at the end of Share the value s with probability better than $\frac{1}{q}$ by a negligible additive factor.

Let us explain this definition in more details. The two phases Share and Reconstruct do exactly what they are named after. During Share, the dealer D will distribute the shares of a chosen secret s. In a VSS protocol, the dealer will also *commit* to those shares, i.e., the dealer will use some commitment function C to prove the correctness of their actions. The function C is known to all players and therefore they can verify the correctness to some degree and will output $ver_i = 1$ if they accept the sharing and $ver_i = 0$ if they reject the sharing. It is still possible for a bad dealer to be accepted, but a good dealer is always guaranteed to be accepted by the good players. In other words, if the dealer is good, then all their actions must be correct and there cannot be any contradictions with the commitment function. Thus, all good players will agree on the correctness and output $ver_i = 1$, which is what property 2 means. Property 1 means that all good players will come to the same conclusion, i.e., a good player accepts the sharing if and only if all other good players also accept it. Property 4 simply means that an adversary does not gain any relevant information after the execution of Share and thus can only guess the secret $s \in S$. Finally, we explain Property 3. After the secret has been shared using the protocol Share, the players will reconstruct it. This means that each player will reveal their share and they will apply some reconstruction algorithm. If the dealer D is good, a VSS requires all good players to reconstruct the correct secret, but even if D is bad they have to reach the same secret after Reconstruct, even if some bad players try to sabotage the protocol. This is either possible by refusing to broadcast their personal shares of the secret or by broadcasting incorrect values. Hence, in practice, there needs to be some way to verify the correctness of such broadcast values. This is where we will once again use the commitment function.
From this definition, it is also clear that VSS is much more complicated than general secret sharing as we need to be able to verify the correctness of the shares and reconstructed secret, whereas before it was enough that the good players could reconstruct the secret while the adversary could not do so. In other words, the schemes we have seen in Section 4 were secure against a passive adversary only and our goal is to generalize them to be secure also against an active adversary.

Remark. With this basic definition of VSS, it is possible that there is in fact no secret. If the dealer D is bad, they might share incorrect shares that do not determine a unique secret. However, in a VSS protocol the players must be able to detect that at the end of Reconstruct. In practice, this means that if the players execute Reconstruct and fail to determine a unique secret $\sigma \in S$, they will conclude that there was no secret and that the dealer was bad and all good players will output the default value $* \notin S$. There are ways to enforce the existence of a secret on the dealer by adding auxiliary protocols such as Protocol 3, which we will see later. However, the definition of VSS does not require that.

In the following, we will explain a simple VSS based on Shamir's Secret Sharing Scheme and thus on interpolation. Let us first recall the most important assumptions. We assume that there is a computationally bounded, static adversary $\mathcal{A}$ that can corrupt up to t players. Furthermore, we assume that we have both a public broadcast channel and private channels between each player.

Protocol 1 ([14, Figure 1]). **Sharing Phase:**

1. We consider some field $K = \mathbb{F}_p$, where $p > n$ is a prime number. The dealer D chooses a secret s and then proceeds as follows:
   (a) The dealer randomly chooses $f, r \in K[x]_{\leq t}$ of degree t such that $f(0) = f_0 = s$.
   (b) The dealer computes the values $\alpha_i = f(i)$ and $\rho_i = r(i)$ and distributes them among the players P_i. More precisely, P_i receives α_i and ρ_i through a private channel.
   (c) The dealer computes the values $A_i = C(\alpha_i, \rho_i)$ for some commitment function C and broadcasts them through a public channel.
2. Player P_i verifies that indeed $A_i = C(\alpha_i, \rho_i)$. If the equation does not hold, player P_i broadcasts a public complaint against the dealer D.
3. If player P_i broadcast a complaint, then D broadcasts the values α_i, ρ_i.
4. If D does not follow some step or broadcasts contradicting information, then D is disqualified. D is also disqualified if at least $t + 1$ players complained, since this implies that D gave wrong values α_i, ρ_i to a good player P_i or broadcast a wrong value A_i, which only happens if D is bad. Else the players conclude that a secret has been shared.

Reconstruction Phase:

1. Each player publicly broadcasts the values α_i, ρ_i.
2. The players take any $t+1$ values that are consistent with the commitment, i.e., values for which $A_i = C(\alpha_i, \rho_i)$. Then they use Theorem 4.2 to interpolate polynomials $\widehat{f}, \widehat{r} \in K[x]_{\leq t}$ that pass through the chosen points.
3. The players compute $\widehat{\alpha}_i = \widehat{f}(i)$ and $\widehat{\rho}_i = \widehat{r}(i)$ and verify that $A_i = C(\widehat{\alpha}_i, \widehat{\rho}_i)$ for all $i = 1, \ldots, n$. If this holds and $\widehat{f}(0) \in S$, then they output $\widehat{f}(0)$, else they output the default value $*$.

Remark. By construction, it is immediately clear that this protocol always satisfies points 1 and 2 in the definition of VSS. This is due to the fact that the decision to accept the sharing or disqualify the dealer happens based on public knowledge only and hence all good players will come to the same conclusion. Furthermore, if the dealer is good, the good players will see all their actions as correct and accept the sharing.

A priori, this protocol is not a VSS since the commitment function C might reveal some information about α_i or some bad player P_j could be able to find incorrect values α'_j, ρ'_j such that $C(\alpha'_j, \rho'_j) = A_j$. This would enable them to severely sabotage the reconstruction as shown in Example 16. However, we can choose C in such a way that the protocol is indeed a VSS.

Definition 5.2 (Homomorphic Commitment)**.** We consider a commitment $C(\alpha, \rho)$ to α with randomness ρ. Then C is called a *homomorphic commitment* if for all $A_1 = C(\alpha_1, \rho_1)$ and $A_2 = C(\alpha_2, \rho_2)$ there exists some ρ such that

$$A_1 A_2 = C(\alpha_1 + \alpha_2, \rho).$$

Example 16. If C is any homomorphic commitment, then we cannot conclude that Protocol 1 is a VSS. Indeed, consider the following scenario. We take $K = \mathbb{F}_5$ and we assume the dealer D to be good and we have a good player P_1 and a bad player P_2. In particular, we have $t = 1$ and $m = 2$. We usually do not allow such a scenario as it is not secure, which we will see in the following. We take as the commitment function C the map

$$C : \mathbb{F}_5^2 \to \mathbb{F}_5$$

$$(x, y) \mapsto y \cdot 2^x.$$

This map is clearly a homomorphic commitment since

$$C(x_1, y_1)C(x_2, y_2) = y_1 2^{x_1} y_2 2^{x_2} = y_1 y_2 2^{x_1 + x_2} = C(x_1 + x_2, y_1 y_2).$$

Furthermore, we consider the polynomials

$$f(x) = 2x + 3 \text{ and } r(x) = x + 4.$$

Thus, it follows that P_1 receives $(\alpha_1, \rho_1) = (0, 0)$ and P_2 receives $(\alpha_2, \rho_2) = (2, 1)$. The dealer D broadcasts publicly the values $A_1 = 0$ and $A_2 = 4$. We assume that all players agree with the sharing and start the reconstruction phase. The good player P_1 broadcasts $(\alpha_1, \rho_1) = (0, 0)$, but P_2 broadcasts $(1, 2)$ instead of $(\alpha_2, \rho_2) = (2, 1)$. Note that $C(1, 2) = 4 = A_2$ and hence the players reconstruct the polynomials

$$\widehat{f}(x) = x + 4 \text{ and } \widehat{r}(x) = 2x + 3$$

by interpolation. So, the polynomials interpolated by the players are not the same as intended by the dealer, in fact they are reversed. The players check if they have $A_i = C(\widehat{\alpha}_i, \widehat{\rho}_i)$ for all i. Namely, they compute $C(0, 0) = 0 = A_1$ and

$C(1,2) = 4 = A_2$. Therefore, they conclude that everything is correct and that the secret shared by the dealer was $\widehat{f}(0) = 4 \neq 3 = f(0)$, i.e., they reconstructed the wrong secret.
Even worse, the bad player might completely refuse to open their share but will still learn the secret. Even a good commitment function cannot prevent that and thus it follows that a scenario with only 2 players, where one of them is corrupt, is terrible. This problem generalizes to any number of players if there are less or equal good than bad players. Hence, it was crucial to assume that there are more good than bad players in the beginning of this section.
So, now we will consider a better setting with t bad players and $t+1$ good players to ensure that reconstruction works. However, we will see that reconstruction can still fail since C was a bad commitment function. To arrive there, we consider the same setting as before, but this time we add one more good player P_3. Then P_3 receives $(\alpha_3, \rho_3) = (4, 2)$ and D broadcasts $A_3 = 2$. Again, we assume that all players accept the sharing and that the bad player P_2 broadcasts the incorrect values $(1, 2)$. If the players use again the values given by P_1 and P_2 they will again interpolate the same incorrect polynomials, but this time they have to verify another commitment. Namely, they compute $C(2,4) = 1 \neq 2 = A_3$. Hence, the bad player was able to disrupt the reconstruction of the secret and the good players will output $*$. In this scenario, the bad player still learns the secret after Reconstruct by interpolating in the values of the good players, while the good players cannot. But what happens if the players choose to interpolate in the values given by the good players? Then they will clearly interpolate the original polynomials f, r and we only need to check what happens at $i = 2$. We have $C(2,1) = A_2$ and hence they will reconstruct the correct secret $s = 3$. We conclude that this commitment function was not good enough since the protocol violates point 4 in the definition of VSS.

This example shows that we need to choose a very good commitment function that is what is called *collision resistant*. Namely, it should be infeasible to find different values such that they output the same commitment.

Theorem 5.3 ([13, Chapter 2.1])**.** *We assume that C satisfies the following properties:*

- *Given $C(x, r)$, it is infeasible to compute any information about x.*
- *C is* collision resistant*. That means, it is infeasible to find x_1, r_1 and x_2, r_2 such that $C(x_1, r_1) = C(x_2, r_2)$. Such a pair (x_1, r_1) and (x_2, r_2) is called a* collision*.*
- *Given x, r and y, everybody can verify if $y = C(x, r)$, i.e., the computation of C does not require any knowledge of a secret key.*

Then Protocol 1 is a VSS.

Proof. We verify that all points in Definition 5.1 are satisfied.

1. The decision to disqualify D or accept the sharing is based on public information. Hence, all good players reach the same decision.

2. If the dealer is good, then all their public actions will be seen as good and all the good players will agree that a secret has been shared.

3. Without loss of generality we may assume that $P_1, \ldots, P_{t+1}$ are good. If $A_i = C(\alpha_i, \rho_i)$ for all i, then set $\sigma = f(0)$. Else we set $\sigma = *$. Thus, σ is fixed at the end of the sharing phase. Now assume by contradiction that at the end of the reconstruction phase the good players output $\tilde{\sigma} \neq \sigma$ by choosing values $\alpha_{i_1}, \ldots, \alpha_{i_{t+1}}$ given out by players P_j such that $C(\alpha_{i_j}, \rho_{i_j}) = A_{i_j}$. This means that we have $\widehat{f}, \widehat{r}$ such that $A_i = C(\widehat{\alpha}_i, \widehat{\rho}_i)$ but $\widehat{f} \neq f$. Hence, there exists some j such that $\widehat{f}(j) \neq f(j)$. Then the pairs $(\widehat{\alpha}_j, \widehat{\rho}_j)$ and (α_j, ρ_j) are a collision for C, which is known either to player P_j or the dealer D. This is a contradiction.

4. If D is good, then the adversary $\mathcal{A}$ sees t points on a polynomial of degree t and all the values A_i as they are public. By assumption on C, the values A_i give no information about α_i.

□

The Discrete Logarithm Problem (DLP): The *discrete logarithm problem (DLP)* is a well known problem in Cryptography that has various applications. Depending on the underlying group, the problem is assumed to be very hard and in general there is no polynomial time algorithm known that can efficiently solve the DLP on a classical computer. However, such algorithms are known for some groups, e.g., $(\mathbb{Z}/n\mathbb{Z}, +)$, and in Shor [24] it is shown that there exists a general algorithm working on a quantum computer. This means that the commitment function based on DLP, which we will see in the following, cannot be used in post-quantum cryptography. Let us now explain, what the DLP is and how we can use it to find a good commitment function. We consider a cyclic group G with generator g, i.e., $G = \langle g \rangle$. We write the group multiplicatively and denote by g^z the element

$$g^z = \prod_{i=1}^{z} g$$

for $z \in \mathbb{N}_{>0}$. Similarly, $g^{-z} = \prod_{i=1}^{z} g^{-1}$ and $g^0 = 1_G$. Then the *discrete logarithm* of an element $h \in G$ with respect to the base g is any number $z \in \mathbb{Z}$ such that $h = g^z$ and is denoted by $z = \log_g(h)$. It is easy to see that $\log_g(h)$ is unique up to adding multiples of the group order. In particular, if the order of G is $|G| = k$ and we know an element z such that $h = g^z$, then we also have $h = g^{z+kw}$ for all $w \in \mathbb{Z}$ as

$$h = g^{z+kw} = g^z g^{kw} = g^z \cdot 1_G = g^z.$$

Then the DLP is the challenge of finding $\log_g(h)$ for some given g, h. As mentioned before, for a good choice of G this problem is assumed to be very hard on a classical computer and hence we may call it infeasible. In the following, we will assume that the DLP cannot be solved by any player. This is a necessary

assumption but it is not unreasonable. Next we will see how we can use this to find an explicit commitment function.
In Example 16, we picked a bad commitment function that was not collision resistant. However, there exist much better functions, for example one based on the discrete logarithm problem. In general, we can build a commitment function based on DLP using any cyclic group G, where the DLP is hard. For example, this works over elliptic curves or large fields of prime order. However, $(\mathbb{Z}/n\mathbb{Z}, +)$ is a bad choice since the Euclidean algorithm can be applied. Thus, in the following we will choose a special setting with two prime fields. Let p, p' be two very large prime numbers such that p' divides $p-1$. Denote by $G_{p'} = \{a \in \mathbb{F}_p \mid a^{p'} = 1\}$ the unique cyclic subgroup of $\mathbb{F}_p^*$ of order p' and let $g \in G_{p'}$ be a generator. For any element $a \in G_{p'}$, we denote by $\log_b(a)$ the discrete logarithm with respect to the base b, i.e., the unique number $r \in \{0, \dots, p'-1\} \cong \mathbb{F}_{p'}$ such that $b^r = a$. This works for any $b \neq 1$. In general, finding the discrete logarithm is very hard and hence we fix some $g, h \in G_{p'}$ and assume that nobody knows $\log_g(h)$. Then we choose the commitment function $C : \mathbb{F}_{p'}^2 \to G_{p'}$ as

$$C(x, y) = g^x h^y.$$

It is obvious that this is a homomorphic commitment, but more importantly, it is also collision resistant under the assumption that nobody can find $\log_g(h)$. Indeed, we first observe that for a fixed x and random y the value $C(x, y)$ is also randomly distributed since the map $y \mapsto g^x h^y$ is an isomorphism. Next, we assume that we have elements $x \neq x' \in \mathbb{F}_{p'}$ and y, y' such that $C(x, y) = C(x', y')$. Then it is clear that we must have $y \neq y'$ since $g^x \neq g^{x'}$. It follows that for a collision (x, y) and (x', y') we have

$$g^x h^y = g^{x'} h^{y'},$$

which is equivalent to

$$g^{x-x'} = h^{y'-y}$$

and hence $\log_g(h) = \frac{x-x'}{y'-y}$. From this, we see directly that knowing a collision implies the knowledge of the discrete logarithm. However, if we know the discrete logarithm and x, y, we can solve $\log_g(h) = \frac{x-x'}{y'-y}$ for y' and obtain a formula for y' depending only on x', which allows us to find a collision (x', y') for any x'. So in fact, knowing a collision is equivalent to knowing the discrete logarithm $\log_g(h)$.
This commitment function has another interesting property. We assume that somebody wants to commit to the same value x in two different ways, i.e., by broadcasting $\beta = C(x, y)$ and $\beta' = C(x, y')$ such that $y \neq y'$. Then anyone who knows an element r such that $\frac{\beta}{\beta'} = h^r$, i.e., $r = \log_h(\frac{\beta}{\beta'}) = y - y'$, can open β as x if and only if they can also open β' as x. In other words, if someone reveals $r = y - y'$, everybody can verify that indeed $\frac{\beta}{\beta'} = h^r$ and hence

$$h^r = \frac{\beta}{\beta'} = \frac{g^x h^y}{g^x h^{y'}},$$

i.e., it follows that it was indeed the same x. On the other hand, the value $r = y - y'$ does not contain any information about x and thus an adversary does not gain any information.
It is also important to note that g, h must be chosen in some random way that ensures the difficulty of the discrete logarithm problem. For example, if the dealer was to choose g, h by themselves, then they could simply set $h = g^z$ and thus would know $\log_g(h) = z$, which would enable them to easily find collisions. Therefore, it is important for all players to jointly agree on some elements g, h. This can be achieved by each player broadcasting some random g_i, h_i and then setting $g = \prod g_i, h = \prod h_i$. This way, the bad players cannot predict the value $\log_g(h)$ and need to solve the discrete logarithm problem to find it. However, the discrete logarithm problem is assumed to be infeasible to solve and thus no player knows $\log_g(h)$.

Remark. In the following when we refer to Protocol 1, we always mean that we use the commitment function $C(x, y) = g^x h^y$. Then Theorem 5.3 implies that Protocol 1 is a VSS under the assumption that nobody can solve the discrete logarithm problem.

Example 17. Consider $p = 29$ and $p' = 7$. In the context of Protocol 1, this means that $K = \mathbb{F}_7$. We find that $G_7 \subset \mathbb{F}_{29}^*$ is

$$G_7 = \{1, 7, 16, 20, 23, 24, 25\}.$$

We take $g = 16$ and $h = 24$ and we assume that we want to commit to $x = 3$ with randomness $y = 5$ and $y' = 2$. It follows that $\beta = C(3, 5) = 16^3 \cdot 24^5 = 20$ and $\beta' = C(3, 2) = 16^3 \cdot 24^2 = 1$. We observe that $\frac{\beta}{\beta'} = 20 = 24^3 = 24^{5-2}$. It is not easy to find a collision without already knowing $\log_{16}(24)$. However, if we know that $\log_{16}(24) = 2$, we only need to find (x', y') such that

$$2 = \frac{3 - x'}{y' - 5}.$$

Now it is easy to see that $C(3, 5) = C(4, 1) = 20$. In fact, one can solve for y' and find that

$$y' = \frac{3 - x'}{2} + 5 = 3 + 3x'.$$

Using this formula, it is possible to find for any $x' \in \mathbb{F}_7$ a value y' such that (x, y) and (x', y') is a collision. Hence, it is important to choose very large prime numbers p and p' to ensure that nobody can compute $\log_g(h)$.

Secure Multi-Party Computation (MPC): In the following, we want to use the discrete logarithm problem and its commitment function to realize what is called *(secure) multi-party computation (MPC)*. In Ben-Or, Goldwasser, and Wigderson [1] and also Cramer, Damgård, and Maurer [11], MPC is explained as follows. We have n players $P_1, \ldots, P_n$ who contribute inputs $x_1, \ldots, x_n$. They want to compute some function given by an arithmetic circuit that consists of addition and multiplication gates. This means that at each gate the players want to compute $a + b$ or $a \cdot b$, where a, b are some input values x_i, x_j or the outcome of a previous gate. The MPC then works in 3 stages.

1. *Input Stage:* Each player P_i shares their input x_i via VSS. A player P_i who fails is disqualified and we take some default value for x_i.

2. *Computation Stage:* The players follow the arithmetic circuit and proceed gate by gate. At each gate they compute $a + b$ or $a \cdot b$ for some a, b. This happens by applying some procedure on the shares and results in shares of $a+b, a \cdot b$ respectively, which are not used for reconstruction yet. If a player is disqualified during this stage, their input value x_i is reconstructed by VSS reconstruction.

3. *Final Stage:* The shares of the final computed value are revealed to all players. In other words, they apply VSS reconstruction to recover the value computed at the last gate.

During the input stage, P_i must also convince all other players that there exists a secret x_i if it is not implicit from the VSS protocol. In our case, Protocol 1 does not imply that and hence we will use an auxiliary protocol. It is also important to note that the values computed at some gate are usually not revealed at that point but simply VSS shared in a way that ensures the correctness of the shares. All MPC protocols we will see in the following achieve the sharing without revealing the value yet. However, the last step in these protocols will be to open some commitments and use the reconstruction procedure to retrieve said value. This step can easily be skipped and be applied to the final value only.
In the following, we will focus mostly on the computation stage and especially the processing of a single gate and we will use the term MPC synonymously. The idea there is that multiple players want to jointly add and multiply two values a and b. In practice, this means that each player gets shares a_i and b_i of some secrets a and b and they want to be able to compute both $a + b$ and $a \cdot b$. The process should not reveal any more to the adversary except the information that is already implied by $a + b, a \cdot b$ respectively. Realizing addition usually is easy. The challenge is to share a and b in such a way that multiplication can be achieved without revealing too much information upon sharing and still remaining correct with an active adversary that tries to sabotage the execution of the protocol. There are multiple ways to achieve that. For example, one can take any LSSS with the strong multiplication property and obtain a secure MPC protocol, which is explained in Section 5.2. But first, we want to continue and show how Protocol 1 can be naturally extended to a secure MPC protocol using the commitment function based on the discrete logarithm problem. To do so, we need three auxiliary protocols.

Remark. In the context of Protocol 1 it is indeed easy to compute $a + b$ using shares a_i, b_i. Assume that a was shared using a polynomial f_a and b was shared using a polynomial f_b, i.e., $f_a(0) = a$ and $f_b(0) = b$. Then it holds that $(f_a + f_b)(0) = a + b$ but we also have that $(f_a + f_b)(i) = f_a(i) + f_b(i) = a_i + b_i$ and hence it is possible to use interpolation to find $f_a + f_b$ from $a_i + b_i$.

First, we want to see how we can compute the product $\alpha\beta$ given two secrets shared by f_α, f_β. For now, we will assume that all players act properly, i.e., they

can be seen as good. Later, we can use Protocol 3 to force the bad players to act properly. Computing $\alpha\beta$ is not trivial. Indeed, we note that $f_\alpha f_\beta(0) = \alpha\beta$ but this is a polynomial of degree $2t$ and it is not random. This problem can be solved by applying both a degree reduction and randomization protocol. By assumption, we have at least $2t+1$ players and let us denote by $f_\alpha(i), f_\beta(i)$ the respective shares given to P_i. We note that the product $f_\alpha f_\beta = f_{\alpha\beta}$ satisfies

$$f_{\alpha\beta}(x) = a_{2t}x^{2t} + \ldots + a_1 x + \alpha\beta$$

for some coefficients $a_j, j = 1, \ldots, 2t$. It is clear from definition that $f_{\alpha\beta}(i) = f_\alpha(i)f_\beta(i)$ for all $i = 1, \ldots, 2t+1$ and hence we can write this as a matrix equation and obtain

$$A \begin{pmatrix} \alpha\beta \\ a_1 \\ \vdots \\ a_{2t} \end{pmatrix} = \begin{pmatrix} f_\alpha(1)f_\beta(1) \\ f_\alpha(2)f_\beta(2) \\ \vdots \\ f_\alpha(2t+1)f_\beta(2t+1) \end{pmatrix},$$

where A is the $(2t+1) \times (2t+1)$ matrix

$$A = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 2^{2t} \\ \vdots & & & \vdots \\ 1 & 2t+1 & \cdots & (2t+1)^{2t} \end{pmatrix}.$$

We note that A is the transpose of a Vandermonde matrix and thus it is invertible by Lemma 4.7. We denote by $\lambda = (\lambda_1, \ldots, \lambda_{2t+1})$ the first row of A^{-1} and note that the vector λ can be easily computed by any player from public knowledge only. Then, from the matrix equation it follows that

$$\alpha\beta = \sum_{i=1}^{2t+1} \lambda_i f_\alpha(i) f_\beta(i).$$

Next, each player chooses a polynomial $h_i(x) \in \mathbb{F}_q[x]_{\leq t}$ of degree t such that $h_i(0) = f_\alpha(i)f_\beta(i)$, which will be used to share $f_\alpha(i)f_\beta(i)$. We set $H(x) = \sum_{i=1}^{2t+1} \lambda_i h_i(x) \in \mathbb{F}_q[x]_{\leq t}$ and note that

$$H(0) = \sum_{i=1}^{2t+1} \lambda_i f_\alpha(i) f_\beta(i) = \alpha\beta.$$

Moreover, the polynomial H is random as we have at least $t+1$ good players randomly choosing a polynomial h_i. Using all this, we obtain the following protocol.

Protocol 2 ([14, Figure 2]). **Input of player** P_i: The shares $f_\alpha(i)$ and $f_\beta(i)$.

1. P_i shares the value $f_\alpha(i)f_\beta(i) = f_{\alpha\beta}(i)$ via a random polynomial $h_i(x)$ of degree t such that $h_i(0) = f_\alpha(i)f_\beta(i)$. Thus, P_j receives the share $h_i(j)$ for $j = 1, \ldots, 2t+1$.

2. Each player P_j computes $H(j) = \sum_{i=1}^{2t+1} \lambda_i h_i(j)$, which is a share of $\alpha\beta$ using λ as above. Namely, λ is the first row vector of the matrix A^{-1}.

3. The players interpolate the values $H(j)$ to reconstruct the secret.

Remark. In this protocol, we use the shares of $2t+1$ players, but it is in fact possible to use all n shares, which will be important later. The matrix A will then be an $n \times (2t+1)$-matrix with left inverse A^{-1} and hence the protocol works for any subset of players that is larger than $2t+1$. More precisely, let A' be the $(2t+1) \times (2t+1)$ submatrix of A as described above. This matrix is invertible and thus in particular defines an injective map. It follows that the larger matrix A must also define an injective map and therefore there exists a left inverse A^{-1} of A. We then find λ as before, namely λ is the first row vector of A^{-1}.

It is clear from construction that this protocol is secure against a passive adversary, but it only works if all the players act properly and hence we want to force the bad players to act properly. This can be achieved by introducing the VSPS property and using a protocol to verify that it is satisfied.

Definition 5.4 (VSPS, [14, Definition 1])**.** A VSS protocol of the form of Protocol 1 is called *verifiable secret and polynomial sharing (VSPS)* if it has the VSPS property. That is, if the VSS sets the value σ, then there exists $f \in K[x]_{\leq t}$ such that $f(0) = \sigma$ and player P_i knows $f(i)$. In other words, the shares and the secret lie on some polynomial of degree at most t.

Remark. We mentioned before that a bad dealer might be able to distribute shares that do not define a unique secret. This is allowed for general VSS but it is in fact rather undesirable. If a VSS has the VSPS property, this cannot happen anymore. The VSPS property ensures that there is a unique secret defined by the shares.

Next, we will see a protocol that lets the players verify that a given VSS has the VSPS property. To arrive there, we first note a few things. Let the dealer D commit to the values $f(i), i = 1, \ldots, n$ with randomness $r(i)$ and let them also commit to the secret $f(0)$ with randomness $r(0)$, i.e., let $A_0 = g^{f(0)}h^{r(0)}$. We now assume that all these committed values $f(i), r(i)$ lie on a polynomial of degree at most t respectively. We assumed in the beginning of the section that we have more good than bad players and thus in particular $n \geq 2t+1$, so let us now assume that $n = 2t+1$. The case where $n > 2t+1$ will be treated later. If D acts correctly, the evaluations corresponding to the commitments $A_0, \ldots, A_t$ and $A_{t+1}, \ldots, A_{2t+1}$ will define the same polynomials f and r. So, we define the polynomials $f^{(1)}, r^{(1)}$ and $f^{(2)}, r^{(2)}$ as the polynomials obtained by interpolating the evaluations at $\mathcal{B}_1 = \{0, \ldots, t\}$ and $\mathcal{B}_2 = \{t+1, \ldots, 2t+1\}$ respectively and we write them as

$$f^{(1)}(x) = a_{1,t}x^t + \ldots + a_{1,0}, \quad r^{(1)}(x) = r_{1,t}x^t + \ldots + r_{1,0}$$

and

$$f^{(2)}(x) = a_{2,t}x^t + \ldots + a_{2,0}, \quad r^{(2)}(x) = r_{2,t}x^t + \ldots + r_{2,0}.$$

As noted before, for a VSPS these two pairs should be the same. The idea of the VSPS-Check is to verify this, but by doing so in a clever way that does not require to actually apply interpolation. More precisely, we take a random value $\delta \in \mathbb{F}_p$ and check if

$$g^{f^{(1)}(\delta)} h^{r^{(1)}(\delta)} = g^{f^{(2)}(\delta)} h^{r^{(2)}(\delta)}.$$

As we have $h = g^z$ for some $z = \log_g(h) \in \mathbb{F}_p$, we obtain the equation

$$f^1(\delta) + z r^{(1)}(\delta) = f^2(\delta) + z r^{(2)}(\delta).$$

The collision resistance of $C(x, y) = g^x h^y$ ensures that the dealer cannot distribute shares that satisfy the equation but such that the polynomials are different. Then the VSPS-Check reduces to a check of each player that the equation holds for a random $\delta \in \mathbb{F}_p$. Thanks to our choice of the commitment function and the public commitments, this check is possible without knowing $f^{(1)}, r^{(1)}, f^{(2)}, r^{(2)}$ as explained in the following protocol.

Protocol 3 (VSPS-Check, [14, Chapter 4.2.1]). Each player chooses a random $\delta \in \mathbb{F}_p$ and verifies that

$$g^{f^{(1)}(\delta)} h^{r^{(1)}(\delta)} = g^{f^{(2)}(\delta)} h^{r^{(2)}(\delta)}$$

using the following observation. We have

$$\begin{aligned} g^{f^{(1)}(\delta)} h^{r^{(1)}(\delta)} &= g^{\sum_{j=0}^{t} a_{1,j} \delta^j} h^{\sum_{j=0}^{t} r_{1,j} \delta^j} \\ &= g^{\sum_{j=0}^{t} \sum_{i=0}^{t} \lambda_{ji} f(i) \delta^j} h^{\sum_{j=0}^{t} \sum_{i=0}^{t} \lambda_{ji} r(i) \delta^j} \\ &= \prod_{i=0}^{t} \left(g^{f(i)} h^{r(i)} \right)^{\Delta_i} \\ &= \prod_{i=0}^{t} A_i^{\Delta_i}, \end{aligned}$$

where $\Delta_i = \sum_{j=0}^{t} \lambda_{ji} \delta^j$ for appropriate Lagrange coefficients λ_{ji}. The same method can be applied to the right hand side for $f^{(2)}, r^{(2)}$ and $i = t{+}1, \ldots, 2t{+}1$ with corresponding Δ_i. Namely,

$$g^{f^{(2)}(\delta)} h^{r^{(2)}(\delta)} = \prod_{i=t+1}^{2t+1} A_i^{\Delta_i}$$

for appropriate Δ_i. Then if some player finds that the equation is not satisfied, they must reveal δ and all other players will check if the equation is indeed violated. This way, bad players cannot sabotage a good dealer.

Remark. The Lagrange coefficients λ_{ji} do not depend on any secret information and can be computed from the knowledge of the numbers i associated with player P_i, which are publicly known. More precisely, λ_{ji} correspond to the coefficients of f_i in Theorem 4.2. Moreover, the protocol can be adapted so that the players jointly agree on a single δ to run the check.

Lemma 5.5. *The probability that a VSS without the VSPS property successfully passes Protocol 3 is at most* $\frac{t}{p}$*. In particular, if* p *is large, then* $\frac{t}{p} \approx 0$ *is negligible and thus we may assume that any VSS passing Protocol 3 is a VSPS.*

Proof. There exists a unique $z \in \mathbb{F}_p$ such that $h = g^z$ and hence

$$g^{f^{(1)}(\delta)} h^{r^{(1)}(\delta)} = g^{f^{(2)}(\delta)} h^{r^{(2)}(\delta)}$$

implies that

$$f^{(1)}(\delta) + zr^{(1)}(\delta) = f^{(2)}(\delta) + zr^{(2)}(\delta).$$

For two distinct polynomials $P \neq Q \in \mathbb{F}_p[x]_{\leq t}$, there are at most t points where they can coincide. Otherwise, assume there are at least $t+1$ such points, then $P - Q$ has at least $t+1$ zeros and thus would be the zero polynomial. So, if we choose a random δ, then the chance that this δ is one of the points where they coincide is at most $\frac{t}{p}$. Therefore, with high probability

$$f^{(1)}(x) + zr^{(1)}(x) = f^{(2)}(x) + zr^{(2)}(x)$$

and hence $(f^{(1)}, r^{(1)}) = (f^{(2)}, r^{(2)})$ by the collision resistance of $C(x, y) = g^x h^y$, which implies the VSPS property. □

Remark. By running Protocol 3 multiple times for different values δ, we can decrease the error probability and in fact if we verify the equation for $t+1$ different elements δ, then the given VSS must be a VSPS with probability exactly 1. However, in practice it is usually not feasible to run so many VSPS-Checks and especially if p is very large, one check may suffice.

Example 18. We consider $P, Q \in \mathbb{F}_7[x]_{\leq 2}$, where

$$P(x) = x^2 + 3x + 5 \text{ and } Q(x) = 6x^2 + x + 1.$$

These polynomials coincide only at $x = 3$. Now, assume that a bad dealer distributes shares such that $f^{(1)} + zr^{(1)} = P$ and $f^{(2)} + zr^{(2)} = Q$. Then if a good player chooses any $\delta \neq 3$ and runs Protocol 3, they will conclude that the equation is violated and the dealer will be disqualified. The probability that a good player chooses the value $\delta = 3$ and thus falsely concludes that this is a VSPS is $\frac{1}{7} < \frac{2}{7}$. However, it is also obvious that there exist pairs P, Q which exhaust the bound.

Remark. Protocol 3 only checks if the values $f(i), r(i), i = 0, \ldots 2t+1$ define the same polynomials f, r respectively. If we have more players, we need to adapt it a bit. This can be easily done by computing more products such as in Protocol 3. In particular, we consider the set $N = \{0, 1, \ldots, n\}$ and the integer $m = \lceil \frac{n+1}{t+1} \rceil$. Then we choose the sets $\mathcal{B}_1 = \{0, \ldots, t\}, \mathcal{B}_2 = \{t+1, \ldots, 2t+1\}, \ldots, \mathcal{B}_m = \{n-t, \ldots, n\}$. It follows that $|\mathcal{B}_i| = t+1$ for all $i = 1, \ldots, m$ and $\cup_{i=1}^m \mathcal{B}_i = N$. Furthermore, $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ for all $i \neq j$ where $i, j \neq m$. We also have $\mathcal{B}_m \cap \mathcal{B}_j = \emptyset$ for all $j = 1, \ldots, m-2$. Lastly, $\mathcal{B}_m \cap \mathcal{B}_{m-1} = \emptyset$ if and only if $\frac{n+1}{t+1} \in \mathbb{Z}$. This means that we always have $\mathcal{B}_1 \cap \mathcal{B}_m = \emptyset$ and thus the polynomials interpolated

by $\mathcal{B}_1$ and $\mathcal{B}_m$ do not use any shared point for interpolation. We then proceed similarly as above. In particular, to check the VSPS property the players will choose a random $\delta \in \mathbb{F}_p$ and verify that

$$\prod_{i \in \mathcal{B}_1} A_i^{\Delta_i} = \prod_{i \in \mathcal{B}_2} A_i^{\Delta_i} = \ldots = \prod_{i \in \mathcal{B}_m} A_i^{\Delta_i}.$$

This method has an error probability of at most $\frac{t}{p}$ by application of Lemma 5.5. More precisely, we assume that some sets $\mathcal{B}_i, \mathcal{B}_j$ do not interpolate the same polynomials and we note that for any two $\mathcal{B}_i, \mathcal{B}_j$ such that $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ we can apply Lemma 5.5. Therefore, the probability of two sets $\mathcal{B}_i, \mathcal{B}_j$ which do not interpolate the same polynomials successfully passing the protocol is at most $\frac{t}{p}$ and it is clear that having more additional checks for the other sets can only decrease that probability but not increase it.

Next, we need a protocol that allows the dealer D to prove that they committed to the product $c = ab$ of some values a, b without revealing them. To do so, we use something known as *zero knowledge (ZK) proof*. The general idea of a ZK proof is that the prover wants to prove some statement without revealing some secret information. It usually consists of some other player, called the verifier, posing a random challenge which must be solved by the prover and the verifier can verify the correctness of this. In particular, if the prover passes a ZK protocol, the verifier will conclude that the prover does indeed know some secret information. However, due to the nature of ZK proofs, we usually have a small probability that a proof will be accepted falsely, but in general this probability is very small and can be further decreased by iterating the protocol for different challenges.

Protocol 4 (ZK Proof, [14, Appendix C]). *Setting:* The dealer D broadcasts three commitments $A = g^\alpha h^\rho, B = g^\beta h^\sigma, C = g^{\alpha\beta} h^\tau$. D wants to prove to all other players that the opening of C is indeed the product of the opening of A and B. As part of that, D must show that they know how to open these commitments, i.e. that A, B, C are proper commitments to $\alpha, \beta, \alpha\beta$. The players proceed as follows:

1. Each player P_i commits to some random value $e_i \in \mathbb{F}_p$.
2. D chooses $x, s_1, s_2, d, s \in \mathbb{F}_p$ and broadcasts $M_1 = g^x h^{s_1}, M_2 = B^x h^{s_2}, M = g^d h^s$.
3. The other players open their commitments by revealing e_i and choose the random challenge $e = \sum e_i \in \mathbb{F}_p$.
4. D broadcasts the values $z = x + e\alpha, w_1 = s_1 + e\rho, w_2 = s_2 + e(\tau - \sigma\alpha), y = d + e\beta, w = s + e\sigma$.
5. All players verify that $g^z h^{w_1} = M_1 A^e, B^z h^{w_2} = M_2 C^e, g^y h^w = M B^e$.

Example 19. The following example thanks to Ivan Damgård shows what can happen if we omit the check of $g^y h^w = M B^e$. We assume that the prover can

open $A = g^\alpha h^\rho$ but not B. In other words, B is not a proper commitment but rather some random element. In theory, it is possible to open B as any $\beta \in \mathbb{F}_p$ but due to the assumption that the DLP is hard, this is not possible in practice. So, let us now assume that B is random and cannot be opened and set $C = B^\alpha h^\tau$ for any τ. Then a prover is able to pass Protocol 4. Indeed, let the prover correctly broadcast randomly chosen x, s_1, s_2 and corresponding $M_1 = g^x h^{s_1}, M_2 = B^x h^{s_2}$. These can be computed for any B. The prover then correctly broadcasts $z = x + e\alpha$ and $w_1 = s_1 + e\rho$ during Step 4 and thus $g^z h^{w_1} = M_1 A^e$ must be satisfied. Furthermore, the prover will broadcast $w_2 = s_2 + \tau e$. This choice of w_2 will lead to the correct equality $B^z h^{w_2} = M_2 C^e$. Indeed,

$$B^z h^{w_2} = B^x B^{e\alpha} h^{s_2} h^{\tau e} = M_2 B^{e\alpha} h^{\tau e} = M_2 (B^\alpha h^\tau)^e = M_2 C^e.$$

This means that the prover was able to pass the protocol without C being a correct commitment. This does not imply that the prover D can open C as $c \neq \alpha\beta$, but in general we do not want such an occurrence regardless. Therefore, we need the verifier to verify the equality $g^y h^w = MB^e$.

Proposition 5.6 ([9, Lemma 2.6])**.** *If C is a commitment to $c = \alpha\beta$, then Protocol 4 will be accepted. Moreover, if C is not a commitment to $c = \alpha\beta$, then the probability that the protocol will be accepted is at most $\frac{1}{p}$.*

Proof. Steps 1 and 3 ensure that no bad player can manipulate the chosen random challenge e. Namely, they cannot influence what e will be during Step 1. This is necessary for the proof to remain zero knowledge. In particular, if D is good, the adversary cannot reconstruct any relevant information from the values broadcast in Step 4. It is easy to verify that all equations in Step 5 are satisfied if the dealer acts correctly. Namely,

1. $g^z h^{w_1} = g^{x+e\alpha} h^{s_1+e\rho} = g^x h^{s_1} g^{e\alpha} h^{e\rho} = M_1 (g^\alpha h^\rho)^e = M_1 A^e$,
2. $B^z h^{w_2} = g^{\beta(x+e\alpha)} h^{\sigma(x+e\alpha)} h^{s_2+e\tau-e\sigma\alpha} = (g^{\beta x} h^{\sigma x} h^{s_2})(g^{e\alpha\beta} h^{e\tau})$
   $= (g^\beta h^\sigma)^x h^{s_2} (g^{\alpha\beta} h^\tau)^e = B^x h^{s_2} C^e = M_2 C^e$,
3. $g^y h^w = g^{d+e\beta} h^{s+e\sigma} = g^d h^s g^{e\beta} h^{e\sigma} = MB^e$.

To show the error probability, we use a method related to the notion of *special soundness*. The main idea is, that if we pose two different challenges $e \neq e'$ during Step 3 and both lead to accept, then it is possible to compute values of the correct form. Protocol 4 satisfies this property, which then immediately implies an error probability of $\frac{1}{p}$. Indeed, pose two different challenges e, e' during Step 3 but let all values from Step 2 be fixed. We divide the equations pairwise and obtain:

1. $g^{z-z'} h^{w_1-w_1'} = A^{e-e'}$,
2. $B^{z-z'} h^{w_2-w_2'} = C^{e-e'}$,
3. $g^{y-y'} h^{w-w'} = B^{e-e'}$.

It follows that we can set $\alpha = \frac{z-z'}{e-e'}, \rho = \frac{w_1-w_1'}{e-e'}, \beta = \frac{y-y'}{e-e'}, \sigma = \frac{w-w'}{e-e'}, c = \alpha\beta, \tau = \frac{w_2-w_2'}{e-e'} + \alpha\sigma$, which then must have the correct form. It also immediately follows that there exists at most 1 value e such that a faulty commitment C to $c \neq \alpha\beta$ can pass Protocol 4. Thus, the error probability is at most $\frac{1}{p}$. □

Remark. Protocol 4 can be iterated to decrease the error probability. However, as seen in the above proof, it is important to run the whole protocol and especially Step 2. If we only run Step 2 once, then an adversary can easily recover a, b by exploiting special soundness.

We can now combine all these protocols to obtain a protocol that realizes a secure multi-party computation. We assume that α and β were shared using Protocol 1 with $C(x, y) = g^x h^y$ and that the VSPS property has been checked. Then the following protocol, which is a slight adaption of the protocol seen in Gennaro, Rabin, and Rabin [13], realizes MPC.

Protocol 5. We assume that $n \geq 3t + 1$.
Input of player P_i: Values $\alpha_i = f_\alpha(i), \beta_i = f_\beta(i), \rho_i = r(i), \sigma_i = s(i)$.
Public input: $A_i = C(\alpha_i, \rho_i) = g^{\alpha_i} h^{\rho_i}, B_i = C(\beta_i, \sigma_i) = g^{\beta_i} h^{\sigma_i}$ for $0 \leq i \leq n$.

1. Each player P_i shares $\alpha_i\beta_i$ using Protocol 1 with $C(x, y) = g^x h^y$ as commitment. That is, set $c_{ij} = f_{\alpha\beta,i}(j), \tau_{ij} = u_i(j)$, where $f_{\alpha\beta,i}, u_i$ are random polynomials of degree t such that $f_{\alpha\beta,i}(0) = \alpha_i\beta_i$.
   Secret Information of P_i: share (c_{ji}, τ_{ji}) of $\alpha_j\beta_j$.
   Public Information: $C_{ij} = g^{c_{ij}} h^{\tau_{ij}}$ for $1 \leq i, j \leq n$ and $C_{i0} = g^{c_{i0}} h^{\tau_{i0}}$ for $1 \leq i \leq n$.

2. The players run a VSPS-Check on P_i's sharing using Protocol 3. If a sharing fails the test, that player must be corrupt and is disqualified. The remaining players proceed without them.

3. P_i uses Protocol 4 to prove in ZK that C_{i0} is a commitment to the product $\alpha_i\beta_i$. If a player fails, they must be corrupt and are disqualified. The remaining players proceed without them.

4. Denote by R the set of remaining players that were not disqualified during Step 2 or 3. We have that $|R| \geq 2t+1$ by assumption since no good player will be disqualified and thus we can find a vector λ such as in Protocol 2. Player P_i computes $\gamma_i = \sum_{j \in R} \lambda_j c_{ji}$, which is a share of $\gamma = \alpha\beta$ via a random polynomial of degree t. They compute also $\tau_i = \sum_{j \in R} \lambda_j \tau_{ji}$ and $C_j = C(\gamma_j, \tau_j) = g^{\gamma_j} h^{\tau_j} = \prod_{l \in R} C_{lj}^{\lambda_l}$ for all $j \in R$.
   Secret Information of P_i: The share γ_i and corresponding randomness τ_i.
   Public information: C_i for $i \in R$.

5. The players open their commitments C_i to reveal γ_i and use interpolation on γ_i as in Protocol 1 to determine the secret $\gamma = \alpha\beta$. The opening of the good players must be accepted and thus we always have enough players to reconstruct γ.

Proposition 5.7 ([14, Theorem 4])**.** *Protocol 5 is secure against an active adversary.*

Proof. First, we note that this protocol must be secure against a passive adversary by construction of Protocol 2. Step 1 works in the same way as Protocol 2. Steps 2 and 3 ensure that the sharing of all players was executed correctly and bad players who fail will be disqualified. By our assumption that $n \geq 3t+1$, we are guaranteed that $|R| \geq 2t+1$ and thus there always exists a vector λ as in Protocol 2. The commitments C_j can be easily computed since the commitment function $C(x,y) = g^x h^y$ is homomorphic and in Step 5 we only use shares that are consistent with the commitment. Since all good players contribute consistent shares, the reconstruction of $\gamma = \alpha\beta$ must succeed. □

Remark. Running many VSPS-Checks can be computationally costly on a standard computer, but this protocol can be improved by reducing the number of VSPS-Checks. The players will omit Step 2 and only run Steps $1, 3, 4$. After that, they will run a VSPS-Check on the C_i. If the check is successful, the players continue with Step 5, else they run Step 2 to determine the bad players. If all players act properly, this reduces the VSPS-Checks that need to be run by factor n. Namely, the players only run a single VSPS-Check instead of n. However, if some bad players try to sabotage the protocol by broadcasting values that do not satisfy the VSPS property, then the players gain no computational advantage as they need go back and run Step 2.

Protocol 5 can be adapted to work for any number of players $3t+1 > n \geq 2t+1$ at the cost of some additional steps. Namely, we need at least $2t+1$ values $\alpha_i\beta_i$ to reconstruct the product $\gamma = \alpha\beta$ and thus a player failing to execute Step 2 or 3 will be a problem. However, there is a method to retrieve the values α_i, β_i for anyone who fails. As before, we assume that α and β were shared by some dealer D using Protocol 1 with $C(x,y) = g^x h^y$ and that the VSPS property has been checked. But now, D also provides the random auxiliary polynomials f_j, F_j of degree t such that $f_j(j) = 0$. D distributes the shares $f_j(i), F_j(i)$ to player P_i and broadcasts the commitments $G_i^j = g^{f_j(i)} h^{F_j(i)}$ for all $1 \leq j \leq n, 0 \leq i \leq n$. Then the players run VSPS-Checks on $\{G_i^j\}_{i=0}^n$ for all $j = 1, \ldots, n$. This ensures the existence of the polynomials f_j, F_j. Next, the commitments $G_j^j = g^{f_j(j)} h^{F_j(j)}$ are opened in public to reveal 0 for all players P_j. If D fails any of this, then D is disqualified and the protocol stops here. With this setup, it is possible to recover the values of bad players failing Step 2 or 3 without revealing the values of the good players.
This works as follows. Assume a player P_j fails Step 2 or 3 and we need to reconstruct their shares α_j, β_j to proceed. All players that are not disqualified yet will open the commitments $A_i G_i^j, B_i G_i^j$ in public to reveal the values $\alpha_i + f_j(i), \beta_i + f_j(i)$. These values can be used to reconstruct the polynomials $f_\alpha + f_j, f_\beta + f_j$ since the shares of all good players will surely be enough to do so. As $f_j(j) = 0$, we have that

$$(f_\alpha + f_j)(j) = \alpha_j \quad \text{and} \quad (f_\beta + f_j)(j) = \beta_j.$$

The values were reconstructed and player P_j who was disqualified will now be openly simulated by the remaining players. There are some important things we want to stress here. First, the adversary has enough information to reconstruct f_j for any good player P_j, but this is no problem since these f_j will never be used. If D is good, the adversary does not have enough information to reconstruct its own f_j that belong to bad players P_j as it misses one point to use interpolation. Thus, it cannot recover any information about f_α, f_β from the reconstruction of $f_\alpha + f_j, f_\beta + f_j$ since a good dealer will choose random polynomials f_j with the only constraint that $f_j(j) = 0$.

Protocol 6. **Input of player** P_i: Values $\alpha_i = f_\alpha(i), \beta_i = f_\beta(i), \rho_i = r(i), \sigma_i = s(i)$.
Public input: $A_i = C(\alpha_i, \rho_i) = g^{\alpha_i} h^{\rho_i}, B_i = C(\beta_i, \sigma_i) = g^{\beta_i} h^{\sigma_i}$ for $0 \leq i \leq n$.

1. Each player P_i shares $\alpha_i \beta_i$ using Protocol 1 with $C(x, y) = g^x h^y$ as commitment. That is, set $c_{ij} = f_{\alpha\beta,i}(j), \tau_{ij} = u_i(j)$, where $f_{\alpha\beta,i}, u_i$ are random polynomials of degree t such that $f_{\alpha\beta,i}(0) = \alpha_i \beta_i$.
   Secret Information of P_i: share (c_{ji}, τ_{ji}) of $\alpha_j \beta_j$.
   Public Information: $C_{ij} = g^{c_{ij}} h^{\tau_{ij}}$ for $1 \leq i, j \leq n$ and $C_{i0} = g^{c_{i0}} h^{\tau_{i0}}$ for $1 \leq i \leq n$.

2. The players run a VSPS-Check on P_i's sharing using Protocol 3. If a sharing fails the test, that player must be corrupt and is disqualified. The remaining players proceed without them.

3. P_i uses Protocol 4 to prove in ZK that C_{i0} is a commitment to the product $\alpha_i \beta_i$. If a player fails, they must be corrupt and are disqualified. The remaining players proceed without them.

4. In this step, the values α_j, β_j belonging to disqualified players are retrieved using f_j, F_j and the method explained above. Then these players will be openly simulated and all commitments given to them by the remaining players are opened in public.

5. Player P_i computes $\gamma_i = \sum_{j=1}^{n} \lambda_j c_{ji}$, which is a share of $\gamma = \alpha\beta$ via a random polynomial of degree t. They compute also $\tau_i = \sum_{j=1}^{n} \lambda_j \tau_{ji}$ and $C_j = C(\gamma_j, \tau_j) = g^{\gamma_j} h^{\tau_j} = \prod_{l=1}^{n} C_{lj}^{\lambda_l}$ for $1 \leq j \leq n$.
   Secret Information of P_i: The share γ_i and corresponding randomness τ_i.
   Public information: C_i for $1 \leq i \leq n$.

6. The players open their commitments C_i to reveal γ_i and use interpolation on γ_i as in Protocol 1 to determine the secret $\gamma = \alpha\beta$.

Remark. Our setting differs a bit from the setting in Ben-Or, Goldwasser, and Wigderson [1] and Cramer, Damgård, and Maurer [11] since during Protocol 6 we do not need the help of the dealer. However, if the dealer is a player P_i such as in Ben-Or, Goldwasser, and Wigderson [1] and Cramer, Damgård, and Maurer [11], we can more easily simplify Protocol 5 to work for $3t + 1 > n \geq 2t + 1$

players. In particular, if a player P_j is disqualified, then the dealer P_i will publicly open the commitments to the share α_j. We proceed similarly with β_j. If a player P_i does not follow that, they must be corrupt too and are also disqualified. In this case, their input value α will be reconstructed by VSS reconstruction. More precisely, the input value of any player who is disqualified will be reconstructed by VSS reconstruction since this value might be important at a later gate. However, in our simplified setting we usually only consider a single gate with two input values α, β.

5.2 Secure MPC using LSSS

In this section, we explain how to obtain an MPC protocol that is secure against an active adversary based on MSP and LSSS as defined in Section 4.2.1. To do so, let us fix some LSSS $\mathcal{M}$ over a finite field $K = \mathbb{F}_q$, where q is not necessarily a prime number. We denote the players by $P_1, \dots, P_n$ and write $\mathcal{U} = \{1, \dots, n\}$. To each player P_i, we associate the vector space $V_i = \text{span}(\mathbf{b}_i) \subset \mathbb{F}_q^e$. We assume that for all i, the vector $\mathbf{b}_i$ associated to player P_i is known to every player. We do not choose a secret s or linear map ϕ yet and we want to stress that $\mathcal{A}(\mathcal{M}), \Gamma(\mathcal{M})$ do not depend on the choice of ϕ. First, we will see a protocol that is only secure against a passive adversary $\mathcal{A} \subset \mathcal{A}(\mathcal{M})$ and then we see how we can improve it to make it secure against any active adversary $\mathcal{A} \subset \mathcal{A}(\mathcal{M})$.

Protocol 7 (Simple MPC, [11, Chapter 6]). We assume that there are values $a, b \in K$ determined by shares $a_1, \dots, a_n, b_1, \dots, b_n$ via the LSSS $\mathcal{M}$. Furthermore, we assume that $\mathcal{M}$ has the multiplication property and that $u = \sum_{i=1}^n \lambda_i \mathbf{b}_i$. Then addition and multiplication work as follows:
Addition: For $i = 1, \dots, n$, P_i computes $a_i + b_i$ and broadcasts it. Then we have $a + b = \sum_i \lambda_i (a_i + b_i)$.
Multiplication: For $i = 1, \dots, n$, P_i computes $a_i \cdot b_i = \tilde{c}_i$.
Resharing Step: P_i randomly chooses a vector $\rho_i \in \mathbb{F}_q^e$ such that its first entry equals $\tilde{c}_i$ and uses it to secret share $\tilde{c}_i$ resulting in shares $c_{i1}, \dots, c_{in}$. Then P_i sends c_{ij} to player P_j.
Recombination Step: For $j = 1, \dots, n$, P_j computes $c_j = \sum_{i=1}^n r_i c_{ij}$, where $\mathbf{r} = (r_1, \dots, r_n)$ is some fixed recombination vector of $\mathcal{M}$. Then the shares $c_1, \dots, c_n$ are broadcast and determine

$$ab = c = \sum_{j=1}^{n} \lambda_j c_j.$$

Remark. The formula $ab = \sum_j \lambda_j c_j$ follows from the multiplication property and the same methods as used in the proof of Lemma 4.14. More precisely, for any set B such that $u = \sum_{j \in B} \lambda_j \mathbf{b}_j$ we have

$$a_i b_i = \sum_{j \in B} \lambda_j c_{ij}$$

and hence

$$ab = \mathbf{r} \cdot (a_1 b_1, \ldots, a_n b_n) = \mathbf{r} \cdot \left(\sum_{j \in B} \lambda_j c_{ij} \right)_{i=1,\ldots,n} = \sum_{i=1}^{n} r_i \sum_{j \in B} \lambda_j c_{ij}$$
$$= \sum_{i=1}^{n} \sum_{j \in B} r_i \lambda_j c_{ij} = \sum_{j \in B} \sum_{i=1}^{n} r_i \lambda_j c_{ij} = \sum_{j \in B} \lambda_j \sum_{i=1}^{n} r_i c_{ij}$$
$$= \sum_{j \in B} \lambda_j c_j.$$

It is important to note that the Resharing Step is essential for the security as we see in the following example.

Example 20. It is not secure to have all players broadcast their values $a_i b_i$ as this could reveal too much about a, b. Indeed, let us consider Shamir's LSSS with $t = 1$ and $n = 2t + 1 = 3$ over $\mathbb{F}_5$. This means that we have $\mathbf{b}_1 = (1, 1), \mathbf{b}_2 = (1, 2), \mathbf{b}_3 = (1, 3)$. We note that

$$3\widehat{\mathbf{b}}_1 + 2\widehat{\mathbf{b}}_2 + \widehat{\mathbf{b}}_3 = \widehat{u}$$

and thus by Corollary 4.15 $\mathbf{r} = (3, 2, 1)$ is a recombination vector. Assume the secrets $a = 1, b = 2$ are shared via $\mathbf{a}_a = (1, 1), \mathbf{a}_b = (2, 1)$, which results in shares

$$a_1 = 2, a_2 = 3, a_3 = 4 \quad \text{and} \quad b_1 = 3, b_2 = 4, b_3 = 0.$$

Then it is clear from the multiplication property that

$$ab = 2 = \sum r_i a_i b_i = (3, 2, 1) \cdot (1, 2, 0).$$

However, broadcasting all values $a_i b_i$ is not secure. Let us assume that P_1 is the bad player who wants to solve for the unknown maps $\mathbf{a}_a = (a, a'), \mathbf{a}_b = (b, b')$. P_1 knows the following information:

- The value ab,
- all values $a_i b_i$,
- a_1 and b_1.

Furthermore, P_1 knows all vectors $\mathbf{b}_i$ and that the sharing happened via some linear maps. This will lead to the following system of equations:

$$\begin{cases} 2 = a_1 = a + a' \\ 3 = b_1 = b + b' \\ 2 = a_2 b_2 = (a + 2a')(b + 2b') \quad (1) \\ 0 = a_3 b_3 = (a + 3a')(b + 3b') \quad (2) \\ 2 = ab \end{cases}.$$

This allows us to express a', b, b' in terms of a and will lead to a quadratic equation which has at most 2 solutions. In particular, we substitute $b = 2a^{-1}, a' = 2 - a, b' = 3 - b = 3 - 2a^{-1}$. Then equations $(1), (2)$ are equivalent to $0 = 4a + 4 + 2a^{-1}, 0 = 2a + 2 + a^{-1}$. By multiplying with a, we obtain two quadratic equations

$$0 = 4a^2 + 4a + 2 \quad \text{and} \quad 0 = a^2 + 2a + 2$$

with solutions $\{1, 3\}, \{1, 2\}$ respectively. Hence, we conclude that $a = 1$ and thus $b = 2$. This means that the secrets a, b were fully revealed to the adversary simply by the knowledge of $a_i b_i$ for all i. This would not happen by only knowing the product ab. More precisely, from the equality $2 = ab$, we only learn that $b = 2a^{-1}$ but the equations $2 = a + a', 3 = b + b'$ still have solutions for any choice of $a \neq 0$. For MPC, we do not want the protocol to reveal more than what is already implied by ab and thus we need the Resharing Step. It is also important to note that in some cases the knowledge of ab might already reveal some information. For example, if $ab = 0$, the adversary learns that $a = 0$ or $b = 0$. This cannot be prevented by any protocol since the aim is to compute this value ab.

Proposition 5.8 ([11, Chapter 6])**.** *Protocol 7 is secure against a passive adversary.*

Proof. Consider some fixed set $A \in \mathcal{A}$. It is clear that A cannot find a, b from the knowledge of $a_i, b_i, i \in A$ only. We proceed by showing that broadcasting $a_i + b_i$ and c_{ij}, c_j does not reveal any relevant information to A other than what is already implied by the values $a + b$ and $a \cdot b$.
Let us first show that the adversary does not learn a, b from the knowledge of $a_i + b_i$ for all i. To do so, we fix some linear map φ such as in Lemma 4.11, i.e., $\varphi(u) = 1$ and $\varphi|_{V_A} \equiv 0$. We then choose any random $\mu \in \mathbb{F}_q$ and consider the maps

$$\psi_a = \phi_a + \mu\varphi$$

and

$$\psi_b = \phi_b - \mu\varphi,$$

where ϕ_a, ϕ_b are the linear maps used to share a, b respectively. Then it is clear that $\psi_a(u) = a + \mu$ and $\psi_b(u) = b - \mu$, while $\psi_a|_{V_A} \equiv \phi_a|_{V_A}$ and $\psi_b|_{V_A} \equiv \phi_b|_{V_A}$. We also note that $\psi_a + \psi_b = \phi_a + \phi_b$ and hence must also be consistent with the values $a_i + b_i$ belonging to good players P_i. Then the adversary cannot guess (a, b) any better than the set $(a + \mu, b - \mu)$ for any $\mu \in \mathbb{F}_q$. In particular, the choice $\mu = 0$ gives the correct secrets a, b. It follows that the adversary cannot guess (a, b) any better than choosing two values that add up to $a + b$.
Let us now treat the multiplication case. We first note that for any $A \in \mathcal{A}$, the set of shares $\{c_{ij}\}_{j \in A}$ is not enough to determine the secret $\tilde{c}_i$ for any good player P_i. Furthermore, by a truncation argument $\widehat{u} \in \widehat{V}_{\mathcal{U}}$ implies that $u \in V_{\mathcal{U}}$ and thus that we have $\operatorname{supp}(\mathbf{r}) \in \Gamma(\mathcal{M})$. This means that at least one coefficient $r_l \neq 0$ in the vector $\mathbf{r}$ corresponds to some good player P_l. We note that the

values c_j are shares of c via the linear map $\rho = \sum r_i \rho_i$ and since $r_l \neq 0$ this map is random. This means that the sharing of c as c_j is also random and thus does not reveal anything. However, it is important to note that there might be some bad case for a weak LSSS. Namely, suppose that $r_i = 0$ for all good players except one, which we will call P_l. This allows the adversary to solve for ρ_l and thus reveal $a_l b_l$. Or in other words, after broadcasting all values c_j, the adversary can find $a_l b_l$ by solving each equation

$$c_j = \sum_{i=1}^{n} r_i c_{ij}$$

for the only unknown c_{lj} and then applying the reconstruction procedure using the values $c_{lj}, j = 1, \ldots, n$. The protocol is still secure in the sense that this information was already implicit from the value $c = ab$ since

$$a_l b_l = r_l^{-1} \left(c - \sum_{i \neq l} r_i a_i b_i \right)$$

and all those $a_i b_i$ belong to bad players and $\mathbf{r}$ is publicly known. Moreover, if two values $r_l, r_{l'}$ belonging to good players are not 0, then the adversary cannot find $\rho_l, \rho_{l'}$. More precisely, the adversary sees the equations

$$c_j - \sum_{i \in A} r_i c_{ij} = r_l c_{lj} + r_{l'} c_{l'j}$$

for all players P_j. The left hand side is known to the adversary, as well as $r_l, r_{l'}$, i.e., the adversary knows the values $r_l c_{lj} + r_{l'} c_{l'j}$ for all $j = 1, \ldots, n$. Then the adversary cannot determine $\rho_l, \rho_{l'}$ by the exact same method as used above in the case of addition. Hence, the adversary cannot guess $\tilde{c}_l = a_l b_l, \tilde{c}_{l'} = a_{l'} b_{l'}$ any better than $\tilde{c}_l + \mu, \tilde{c}_{l'} - \mu$ by application of Lemma 4.11. However, this information was already implied by

$$c - \sum_{i \in A} r_i a_i b_i = r_l a_l b_l + r_{l'} a_{l'} b_{l'}.$$

This then generalizes to any number of good players in $\mathrm{supp}(\mathbf{r})$. Indeed, for some fixed $A \in \mathcal{A}$ and recombination vector $\mathbf{r}$, let $\mathrm{supp}(\mathbf{r}) = B \cup G$ such that $B \subset A$ and $G \cap A = \emptyset$ and such that $|G| \geq 2$. Then, for any $i \in G$ we write ϕ_i for the linear map corresponding to ρ_i and consider the maps

$$\psi_i = r_i \phi_i + \mu_i \varphi$$

for any fixed φ such as in Lemma 4.11 and values μ_i such that $\sum_{i \in G} \mu_i = 0$. Then by Lemma 4.11 and the same methods as above, it is clear that $\psi_i|_{V_A} \equiv r_i \phi_i|_{V_A}$ for all $i \in G$ and $\sum_{i \in G} \psi_i + \sum_{i \in B} r_i \rho_i = \rho$ but $\psi_i(u) = a_i b_i + \mu_i$. Thus, ψ_i are consistent with all information available to the adversary but define the

wrong secrets $a_i b_i + \mu_i$ and therefore the adversary cannot find the correct secrets $a_i b_i$ any better than $a_i b_i + \mu_i$ such that $\sum_{i \in G} \mu_i = 0$. Furthermore, this information was already implied by the value $c = ab$ in the same way as before. It is also clear that an adversary cannot find $a_i b_i$ where $r_i = 0$ and hence it does not have enough information to reconstruct a, b. □

To obtain a protocol that is secure against an active adversary we need some auxiliary protocols and a good commitment scheme. We have already seen commitment functions but let us now formally introduce the notion of a commitment scheme and then explain a special commitment protocol, which we will use.

Definition 5.9 (Commitment Scheme, [11, Chapter 5.2])**.** A *commitment scheme* for a given adversary $\mathcal{A}$ consists of two protocols *Commit* and *Open*. The protocol Commit allows some player P_i, who takes the role of the dealer, to commit to a value a and the protocol Open allows them later to reveal the committed value. The total information stored by all the players after Commit by P_i is denoted by $[a]_i$ and is called the *commitment*. Both Commit and Open may be interactive protocols such that the other players will either accept the outcome or disqualify the dealer. A commitment protocol must be in such a way that an adversary $\mathcal{A}$ cannot find the committed value a before the execution of Open. Furthermore, it must bind the dealer to the committed value, i.e., there is at most 1 value that the dealer can open such that they will be accepted during the Open protocol. In practice, this means that either there is a unique value that can be opened or that the dealer is unable to compute another value that could be accepted due to computational limitations.

Example 21. We have already seen a commitment scheme, namely the commitment function based on the discrete logarithm problem. The dealer will commit to a value x with randomness y using the commitment function $C(x, y) = g^x h^y$. During the Open protocol, they will reveal (x, y) and the other players will verify that indeed $C(x, y) = g^x h^y$. Using the randomness y makes it impossible for an adversary to guess the corresponding value x and under the assumption that solving the discrete logarithm is computationally too costly, the dealer is unable to open a different pair (x', y'). Using the notation as above, we have that $[x]_i = g^x h^y$.

In the following, we will assume that our commitments are always *homomorphic*. In particular, this means that given two commitments $[a]_i$ and $[b]_i$ all players are able to compute $[a + b]_i$ by P_i without interaction with some other player and $[ma]_i$ for any constant $m \in \mathbb{F}_q$. In the context of the commitment function $C(x, y) = g^x h^y$ over $\mathbb{F}_p$, this is possible and translates to

- $m[a]_i = [ma]_i = C(a, r)^m = g^{am} h^{rm}$.
- $[a]_i + [b]_i = [a + b]_i = C(a, r)C(b, s) = g^{a+b} h^{r+s}$.

Now, we will see a very special commitment scheme that is an adaption of the scheme described in Cramer, Damgård, and Maurer [11, Chapter 5.3] and

works by secret sharing the committed value s using our LSSS $\mathcal{M}$. The decision to accept or reject the opening is based on the check if the shares consistently determine the same secret s or not. Furthermore, it is easy to see that this commitment scheme must be homomorphic by the linearity of the underlying LSSS $\mathcal{M}$.

Protocol 8 (Commitment Protocol). We assume that $\mathcal{M}$ is Q3 and that in addition to the players $\{P_1, \ldots, P_n\}$ we also have a dealer D.

Commit Phase:

1. Assume that the dealer D wants to commit to some element $s \in K$. To do so, D chooses a symmetric $e \times e$ matrix R such that $R_{1,1} = s$ and the remaining matrix is random. We denote by $\mathbf{b}_i$ the chosen basis vector of V_i belonging to the player P_i. Then D sends to P_i the vector $\mathbf{u}_i = R\mathbf{b}_i^T$ and the share s_i of s given to the player P_i is defined to be the first entry of $\mathbf{u}_i$, i.e., $s_i = \mathbf{u}_i \cdot (1, 0, \ldots, 0) = \mathbf{u}_i \cdot u$.

2. The share s_i will be shared into further pieces, namely $s_{ij} = \mathbf{b}_j \cdot \mathbf{u}_i$, which we think of as a share of s_i given to the player P_j. We note that

$$\mathbf{b}_j \cdot \mathbf{u}_i = \mathbf{b}_j R \cdot \mathbf{b}_i = \mathbf{b}_i \cdot R\mathbf{b}_j^T = \mathbf{b}_i \cdot \mathbf{u}_j.$$

   Now, the player P_i sends to P_j the value $s_{ij} = \mathbf{b}_j \cdot \mathbf{u}_i$ and P_j compares it to $s_{ji} = \mathbf{b}_i \cdot \mathbf{u}_j$. If the values are not equal, P_j broadcasts the message $\mathit{complaint}(i, j)$.

3. In response to $\mathit{complaint}(i, j)$, the dealer D must broadcast the correct value s_{ij}.

4. If any player finds that the information broadcast by D contradicts what they received in Step 1, they broadcast an *accusation*, i.e., they claim D to be corrupt.

5. In response to an accusation by P_i, D must broadcast all information sent to P_i in Step 1, namely the vector $\mathbf{u}_i$. Next, each player P_j checks if $s_{ij} = s_{ji}$ using the public vector $\mathbf{u}_i$ to compute $s_{ij} = \mathbf{b}_j \cdot \mathbf{u}_i$. If they find a contradiction, they broadcast a complaint.

6. This may lead to further accusations and Steps 3 to 5 are repeated until D contradicts themselves or they are accused by a set of players not in $\mathcal{A}$, in which cases D is clearly corrupt and thus disqualified, or no new complaints occur. If there are no new complaints and only a set of players in $\mathcal{A}$ complained, then the Commit protocol is accepted. This means that all players P_i who did not complain agree on their shares s_i and vector $\mathbf{u}_i$. The players P_j who did complain must accept the vector $\mathbf{u}_j$ broadcast for them during Step 5 and proceed as if they received it during Step 1.

Open Phase:

1. D broadcasts the full set of shares $\{s_i\}$ and the secret s.

2. Each player broadcasts *agree* if this coincides with their share s_i or *complain* if it does not. Denote the set of players who complained by C. If $C \in \Gamma(\mathcal{M})$, then the opening is rejected.

3. If $C \in \mathcal{A}$, we have to check that the secret s is consistent with all shares $\{s_i\}_{i \notin C}$. To do so, we first note that verification of the secret is easy. More precisely, if everything went correct and we have $u = \sum \lambda_i \mathbf{b}_i$, then

$$\sum \lambda_i s_i = \sum \lambda_i \mathbf{u}_i \cdot u = \sum \lambda_i (R\mathbf{b}_i^T) \cdot u = \sum \lambda_i \mathbf{b}_i \cdot (Ru^T) = u \cdot Ru^T = R_{1,1} = s.$$

For any set $B \in \Gamma(\mathcal{M})$ with λ such that $\sum_{i \in B} \lambda_i \mathbf{b}_i = u$, we use the notation $s_B = \sum_{i \in B} \lambda_i s_i$. A secret is determined consistently if and only if for all B such that $B \in \Gamma(\mathcal{M})$ and $B \subset C^c = \mathcal{U} \backslash C$ we have $s_B = s$. If this is satisfied, then the players accept the opening, else they reject it. Moreover, we note that for any $A \in \mathcal{A}$ the set $B' = C^c \backslash A \in \Gamma(\mathcal{M})$ by the Q3 property and $s_{B'} = s$ must be the correct secret if D is not disqualified, which will be justified later.

In other words, the commitment works by secret sharing s and $[s]_D = (s_1, \ldots, s_n)$ are the shares determining s. During Open, the players will check if the shares consistently determine the secret s to accept or reject the opening. The players will also reject the opening if a set of players that is not in $\mathcal{A}$ complained about the opening, i.e., they claim that their received share s_i is different from what the dealer broadcast during the Open protocol. The additional use of $\mathbf{u}_i$ and R will ensure the correctness of the protocol in the presence of an active adversary and especially a bad dealer.
We also note that what happens in Steps 1 and 2 of the Commit phase is just a special way of choosing our linear map ϕ. In fact, by the symmetry of R we have

$$s_i = \mathbf{u}_i \cdot u = (R\mathbf{b}_i^T) \cdot u = \mathbf{b}_i \cdot (Ru^T),$$

i.e., the map ϕ is given by the inner product with Ru^T which is the first column (and also row) vector of R.

Lemma 5.10. *If the dealer is not disqualified during the Commit phase of Protocol 8, then the set of good players consistently define a unique secret s. In other words, this secret s is the value D committed to.*

Proof. Let us denote by G the set of all good players. By assumption, the Commit protocol is accepted. This means that only a set in $\mathcal{A}$ complained and for any good player P_j who accused the dealer D, they broadcast a new vector $\mathbf{u}_j$ that was consistent with the remaining vectors $\mathbf{u}_i$ belonging to good players P_i. Furthermore, all values s_{ij} known to the good players P_i must be consistent. In other words, since s_i was LSSS shared by a good player P_i using the linear map corresponding to $\mathbf{u}_i$, the shares $(s_{ij})_j$ consistently determine the secret s_i as a linear combination, i.e., $s_i = \sum_{j \in B} \lambda_j s_{ij}$ for all sets B such that $B \subset G$ and $B \in \Gamma(\mathcal{M})$. By linearity of LSSS secret sharing the shares $\sum_{j \in B} \lambda_j s_{ij} = s_i$ must consistently determine some secret s. More precisely, let us fix some set

B such that $B \subset G$ and $B \in \Gamma(\mathcal{M})$. Then for all sets B' such that $B' \subset G$ and $B' \in \Gamma(\mathcal{M})$ with corresponding vector λ' we have

$$\begin{aligned} s_B = \sum_{i \in B} \lambda_i s_i &= \sum_{i \in B} \lambda_i \sum_{j \in B'} \lambda'_j s_{ij} = \sum_{i \in B} \sum_{j \in B'} \lambda_i \lambda'_j s_{ij} \\ &= \sum_{j \in B'} \sum_{i \in B} \lambda_i \lambda'_j s_{ij} = \sum_{j \in B'} \lambda'_j \sum_{i \in B} \lambda_i s_{ij} \\ &= \sum_{j \in B'} \lambda'_j s_j = s_{B'}. \end{aligned}$$

This equation only holds due to the fact that $s_{ij} = s_{ji}$ and therefore s_{ij} is both, a share of s_i and s_j. We conclude that for any two sets B, B' such that $B, B' \subset G$ and $B, B' \in \Gamma(\mathcal{M})$ we have $s_B = s_{B'}$ which means that the secret $s = s_B$ is consistently determined by the good players. □

Theorem 5.11. *Protocol 8 describes a commitment scheme. Most importantly, if the dealer D is not disqualified during the Commit phase, then D can open exactly one value s. Moreover, an active adversary cannot sabotage a good dealer and the adversary does not learn the value s before the Open protocol.*

Proof. We first note that the value s the dealer D committed to is fixed as $s = s_G$, which is consistently determined by Lemma 5.10. Hence, if D is not disqualified during the Commit phase, they committed to a unique value s defined by the shares held by good players. This holds even if D was bad. We now assume that D is good. Then, an adversary learns nothing from Steps 2-6 of the Commit phase that it was not already told in Step 1 since a good player will never accuse a good dealer and hence no vector $\mathbf{u}_i$ belonging to a good player P_i is made public. Furthermore, the adversary does not learn anything on the secret s from the information given to it in Step 1. Indeed, let $A \in \mathcal{A}$. Then, the players A learnt the vectors $\mathbf{u}_i = R\mathbf{b}_i^T, i \in A$ in Step 1. Now, let X be any symmetric $e \times e$-matrix such that

$$X\mathbf{b}_i^T = R\mathbf{b}_i^T = \mathbf{u}_i \quad \text{for all } i \in A.$$

We denote by $s' = X_{1,1}$ the entry in its upper left corner. As $A \in \mathcal{A}$, it follows by Lemma 4.11 that there exists a linear map φ such that $\varphi|_{V_A} \equiv 0$ and $\varphi(u) = 1$. Using Lemma 4.12, we consider its corresponding vector, which we denote by $\mathbf{a} = (a_1, \ldots, a_e)$. By construction, it is clear that $a_1 = 1$. Next, we consider the symmetric matrix

$$F = \begin{pmatrix} a_1\mathbf{a} \\ a_2\mathbf{a} \\ \vdots \\ a_e\mathbf{a} \end{pmatrix} = \begin{pmatrix} a_1a_1 & a_1a_2 & \cdots & a_1a_e \\ a_2a_1 & a_2a_2 & \cdots & a_2a_e \\ \vdots & \vdots & & \vdots \\ a_ea_1 & a_ea_2 & \cdots & a_ea_e \end{pmatrix}$$

and again by construction we have that $F_{1,1} = a_1a_1 = 1$. We observe that the matrix $Y = X + (s - s')F$ satisfies

$$Y\mathbf{b}_i^T = R\mathbf{b}_i^T = \mathbf{u}_i \quad \text{for all } i \in A$$

and note that $Y_{1,1} = s$ and is symmetric by construction. Therefore, for any s' the number of possible solutions X is the same and the adversary does not learn anything from the information available to it.
If D is good throughout the protocol, all good players will agree with them and thus opening always succeeds. More precisely, each good player P_i will agree with the values s_i broadcast by D and only bad players could possibly complain. But by the construction of the Open phase, this set cannot force the opening to be rejected and all good players will see that D opened the correct value s during Step 3. At this point again only the bad players could complain and hence the opening will be accepted.
It remains to show that the opening either reveals the secret s or is rejected. If the opening is not rejected during Step 1, we consider the set of all players $\mathcal{U}$ and subtract the set of corrupt players A and the players C who complained about the opening and denote this set by $B = \mathcal{U} \backslash (C \cup A)$. Since the opening was not rejected during Step 1, the set of players who complained C must be in $\mathcal{A}$ and thus the remaining set B cannot be in $\mathcal{A}$ by the Q3 property and hence it follows that $B \in \Gamma(\mathcal{M})$. This set B consists of all the good players that did not complain and thus the shares revealed for them are the same as the ones received during the Commit phase. Therefore, $s_B = s$ and if D tries to open a different value $s' \neq s$, the opening will be rejected during Step 3. It follows that the opening is accepted only if the revealed value is s. We conclude that Protocol 8 describes a commitment scheme. □

Remark. In Protocol 8, we use an additional dealer D who is distinct from the players P_i. We do not have such a dealer in the context of MPC. More precisely, each player P_i can take the role of the dealer. However, the commitment scheme still works by having P_i give a share to themselves. It is clear that nothing can go wrong if D is good, but even if D is bad, the protocol must still work.

Example 22. In the following, we want to give a short example once again based on Shamir's Secret Sharing Scheme explaining this protocol. We consider a scheme with additional dealer D and players P_1, P_2, P_3, P_4, i.e., we have $t = 1$ and $n = 4$. This ensures that $\mathcal{M}$ is $Q3$. We only want to give an example of the Open protocol and thus we assume that during the Commit protocol all players acted properly, even the bad ones. We first assume that D is good and P_3 is bad. We set $K = \mathbb{F}_5$ and thus we have $\mathbf{b}_i = (\mathrm{Ev}_i(1), \mathrm{Ev}_i(x))$, i.e.,

$$\mathbf{b}_1 = (1, 1),$$

$$\mathbf{b}_2 = (1, 2),$$

$$\mathbf{b}_3 = (1, 3),$$

$$\mathbf{b}_4 = (1, 4).$$

It was shown before, but let us once again stress, that no single player can reconstruct a secret s shared via this LSSS, while any set of at least two players can do so. More precisely, we have

$$u = 2\mathbf{b}_1 + 4\mathbf{b}_2 = 2\mathbf{b}_3 + 4\mathbf{b}_1 = 3\mathbf{b}_2 + 3\mathbf{b}_3 = 3\mathbf{b}_1 + 3\mathbf{b}_4 = 2\mathbf{b}_2 + 4\mathbf{b}_4 = 4\mathbf{b}_3 + 2\mathbf{b}_4.$$

Let us now assume that the dealer D wants to commit to $s = 1$. To do so, D chooses a matrix of the form R at random, e.g., let us take

$$R = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}.$$

We compute $\mathbf{u}_i = R\mathbf{b}_i^T$ and it follows that

$$\mathbf{u}_1 = (3,1),\ \mathbf{u}_2 = (0,0),\ \mathbf{u}_3 = (2,4),\ \mathbf{u}_4 = (4,3)$$

and thus

$$s_1 = 3,\ s_2 = 0,\ s_3 = 2,\ s_4 = 4.$$

Next, each player will compute s_{ij} and verify that $s_{ij} = s_{ji}$. We assumed that this succeeds, but even if P_3 tried to sabotage the protocol here, the good dealer would broadcast the correct values $s_{3,i} = s_{i,3}$ and P_3 would achieve nothing. Now, if D wants to open the commitment, D broadcasts all correct shares and even if P_3 complains about their share, the remaining players will see that their shares consistently determine s and accept the opening.
Let us now consider a more interesting example, where the dealer D is bad. The rest of the setting remains the same as before up to the point where D wants to open the commitment. Let us assume that D wants to open a different value than $s = 1$, e.g., $s' = 3$. D reveals the shares $(3, 0, 3, 1)$. The good player P_4 will complain but the bad player P_3 will support the bad dealer and claim that their value 3 was correct. Now, all good players will check if the secret s' is consistent with all possible sets in Γ excluding the player P_4 since they complained. They can verify that $s_{\{1,3\}} = 3$ but they also see that $s_{\{1,2\}} = 1$. Hence, they reject the opening.
We want to conclude this example with the remark that many things can happen during the Commit phase. For example, bad players can try to distribute inconsistent shares s_{ij}. However, if D is good, they will not succeed. On the other hand, if D is bad, then D might try to broadcast vectors $\mathbf{u}_i$ that were not created using a symmetric matrix R. This again cannot succeed thanks to Lemma 5.10. Giving an explicit example would be lengthy and hence is omitted.

Remark. Step 3 of the Open phase might be computationally costly depending on the amount of players participating. There are multiple ways to improve the checks whether the shares determine the secret consistently or not. For example, one can use a probabilistic approach and only check some sets or examine which is an optimal way to choose the sets that need to be checked. However, we will consider a different approach here in the case of an LSSS based on an underlying code such as Shamir's scheme using an RS-code or the more general LSSS using AG-codes. The main idea here is to use the underlying code to check if $\mathbf{s} = (s_1, \ldots, s_n)$ is a codeword or not. If D is good, then $\mathbf{s}$ must be a codeword and if $\mathbf{s}$ is not a codeword, then D was bad and tried to open a wrong value $s' \neq s$.
Let us first consider an LSSS based on Shamir's scheme. For Protocol 8, we need it to be Q3. We note that Shamir's LSSS is Q3 if and only if $n \geq 3t + 1$

and hence let us assume $n = 3t + 1$. In this scheme, the vectors $\mathbf{b}_i$ correspond to the columns of the generator matrix G and the linear map used to share a secret s is some vector $\mathbf{u}$ with first entry $(\mathbf{u})_1 = s$ that is encoded using G, i.e., $\mathbf{u}G = \mathbf{s}$. Denote by C the set of players who complained during Step 2 and let us assume the worst case that $|C| = t$. We consider the matrix G', where we deleted the columns of players who complained. This still defines an RS-code. It follows that the code corresponding to G has distance $2t + 1$, while the code corresponding to G' has distance $t + 1$. Let $\mathbf{s}' = (s_i)_{i \notin C}$. If the opening is accepted, then $\mathbf{s}'$ must be a codeword. If the dealer D is bad, they may broadcast up to t bad values s_i corresponding to bad players P_i who will not complain, i.e., there were t errors and we are looking at $\mathbf{s}' + e$, where $\mathrm{wt}(e) = t$. But we have distance $t + 1$. This ensures that $\mathbf{s}' + e$ cannot be a different codeword and it is enough to check that $\mathbf{s}'$ is a codeword. In practice, this means that players will check that $H'\mathbf{s}'^T = 0$, where H' is a parity check matrix defining the same code as G'. If D opened the correct vector $\mathbf{s}$, then this equation holds. If it is violated, the opening will be rejected.
This method easily generalizes to the case of an AG-code with $n \geq 6g + 3t + 1$ and hence let us assume that $n = 6g + 3t + 1$. By Corollary 4.18, any set in $\mathcal{A}$ has at most size $2g+t$ and thus this scheme is Q3. As before, we have two codes generated by G, G' and their distances are at least $d = 4g+2t+1, d' = 2g+t+1$ by Theorem 1.39. Hence, if there are $2g+t$ bad players with bad shares, $\mathbf{s}'$ cannot be a codeword and therefore it is enough to check if $H'\mathbf{s}'^T = 0$. We also want to note that the choice $n \geq 6g+3t+1$ is too generous. Namely, by Proposition 4.19 we know that a code with $n \geq 4g + 3t + 1$ has the strong multiplication property and thus also is Q3. However, due to the nature of the LSSS based on AG-codes as a ramp scheme, the method from above does not generalize in the same way.
The method explained in this remark can also be adapted a bit. Namely, if we pass Step 1 of the Open phase, we are guaranteed that only a set in $\mathcal{A}$ complained. These may be considered errors as well and we can check that $H\mathbf{s}^T = 0$. Since the code corresponding to H has a larger distance, namely at least $d = 4g + 2t + 1$, and we have at most $4g + 2t$ errors, we can still detect the errors. In this case, a good dealer must be accepted, while a bad dealer will be rejected if they try to open a different value $s' \neq s$. However, in general a bad dealer might still be accepted if they open wrong s_i for good players who complain. This is not possible anymore using the parity check matrix H and the method using H can be seen as stricter. It is also important to note that the protocol must still pass Step 1, otherwise there might be more errors than the distance and the opened shares could determine a different codeword.

Example 23. We want to apply the method explained in the remark above to the setting of Example 22. This means that we have

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix}.$$

If P_3 complains about the opening, we consider the matrix

$$G' = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix}$$

with corresponding parity check matrix

$$H' = \begin{pmatrix} 1 & 1 & 3 \end{pmatrix}.$$

Next, every player will compute $H'(3,0,4)^T = 0$ and the opening is accepted. Similarly in the case where we chose D to be bad and broadcast wrong values for P_3 and P_4, we have

$$G' = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

with

$$H' = \begin{pmatrix} 1 & 3 & 1 \end{pmatrix}.$$

The players will compute $H'(1,3,2)^T = 2 \neq 0$ and reject the opening.
Let us also quickly explain the slightly adapted version. We note that

$$H = \begin{pmatrix} 1 & 0 & 2 & 2 \\ 0 & 1 & 3 & 1 \end{pmatrix}.$$

The checks for $(3,0,2,4)$ and $(3,0,3,1)$ turn out as above. However, if D tries to open the correct value $s = 1$ by broadcasting the vector $(3,0,2,1)$, we will get a different result. Namely, following Protocol 8 this opening will be accepted since P_4 complains and the wrong value $s'_4 = 1$ will be disregarded. On the other hand, we have that $H(3,0,2,1)^T = (4,2)$ and thus the opening will be rejected. A good dealer will still always be accepted, but using the parity check matrix H instead of H' is stricter and will reject some cases, where a bad dealer tried to sabotage the opening but still opened the correct value. In other words, this means that using H, the dealer is forced to broadcast the exact correct vector of shares, otherwise the opening will be rejected. In practice, this can be seen as more convenient since we only need a single parity check matrix for any possible set of players who complained.

Using Protocol 8 as a commitment protocol is not enough to ensure the correctness of MPC with the presence of an active adversary. We need the following two auxiliary protocols. First, we need a *commitment transfer protocol (CTP)* which allows player P_i to transfer a commitment to P_j , i.e., to convert $[a]_i$ into $[a]_j$. Clearly, P_j learns a in the process but the protocol must leak no information to the adversary if P_i and P_j both are good. On the other hand, it must ensure that the committed value remains consistent, even if both P_i and P_j are bad players.
The second protocol we need is called *commitment sharing protocol (CSP)*. It allows player P_i to convert a committed value $[a]_i$ into a set of commitments to shares of a denoted by $[a_1]_1, \ldots, [a_n]_n$. This is realized as $a_i = \mathbf{b}_i \cdot \rho$, where $\mathbf{b}_i$ is the corresponding basis vector of V_i given to player P_i and $\rho = (a, \rho_2, \ldots, \rho_e)$

corresponds to the linear map ϕ. This protocol must still work for bad players P_j and must not leak any information to the adversary if P_j is a good player. In general, the CSP works as follows.

Protocol 9 (CSP, [11, Chapter 5.2]).

1. Let P_i be committed to a as $[a]_i$.
2. P_i chooses a random vector $\rho = (a, \rho_2, \ldots, \rho_e)$ with the only constraint that it has a as its first entry. Then P_i commits to $\rho_2, \ldots, \rho_e$ resulting in $[\rho_2]_i, \ldots, [\rho_e]_i$. In other words, player P_i chooses a random linear map ϕ and commits to its coefficients except the first one, which corresponds to the secret a. We denote by $(a_1, \ldots, a_n)$ the shares resulting from sharing a using the map ϕ.
3. By construction, each a_i is a linear function of the committed values and hence the players can compute $[a_1]_i, \ldots, [a_n]_i$ since we assumed the commitment to be homomorphic. More precisely, $a_j = \mathbf{b}_j \cdot \rho$ and thus we have

$$[a_j]_i = [\mathbf{b}_j \cdot \rho]_i = (\mathbf{b}_j)_1[a]_i + (\mathbf{b}_j)_2[\rho_2]_i + \ldots + (\mathbf{b}_j)_e[\rho_e]_i,$$

   which can be computed by everyone since the commitment is homomorphic.
4. Now, P_i uses CTP to convert $[a_j]_i$ into $[a_j]_j$ for $j = 1, \ldots, n$.

Proposition 5.12. *Protocol 9 satisfies the desired properties of a CSP. Namely, it converts the commitment $[a]_i$ into shares without leaking a to the adversary if P_i is good. Furthermore, the protocol still works as intended, even if P_i is bad.*

Proof. First assume that P_i is good. Then the sharing via ϕ is random and from the commitment the adversary cannot find ρ. More precisely, the adversary knows the shares $a_j, j \in A$, which are not enough to determine the secret a. Then by assumptions on CTP applying CTP does not reveal any additional information to the adversary. Now, assume that P_i is bad. Committing to a and some values ρ_k will always determine a unique linear map ϕ defined by these values. Commitments are homomorphic and thus P_i is unable to distribute inconsistent shares a_j to good players P_j since they can verify the commitment by linearity and the formula as seen in Step 3. Once again by assumptions on CTP, the share a_j, which is now considered to be the secret, must remain consistent. We conclude that this protocol has the desired properties of a CSP. □

The CTP is a crucial part of the CSP but we have not explained it yet. Let us do so now.

Protocol 10 (CTP, [11, Chapter 5.4]).

1. Given a commitment $[a]_i$ realized as in Protocol 8 with P_i in the role of the dealer, P_i sends privately the shares determining a to P_j. If this information is not consistent, then P_j broadcasts the message *"complaint"* and the protocol continues at Step 4.

2. P_j commits to a independently, which will result in $[a]_j$.

3. Using linearity of the commitments, P_j opens the difference $[a]_i - [a]_j$ to reveal 0 using information from Step 1. If P_j fails, the protocol continues at Step 4, else the protocol ends successfully.

4. If we arrive at this step, then either P_i or P_j must be corrupt. Then P_i must open $[a]_i$ in public and we either disqualify P_i, if P_i fails, or continue with a default commitment to a assigned to P_j.

Proposition 5.13. *Protocol 10 satisfies the desired properties of a CTP. Namely, if both players P_i, P_j are good, the adversary learns nothing, but even if both players are bad, the commitment must remain consistent.*

Proof. Let us first justify Step 3. Assume that λ is such that $u = \sum_k \lambda_k \mathbf{b}_k$ and that both players P_i and P_j commit to the same value a using two different matrices R_i, R_j. We denote the commitments as $[a]_i = (a_1^i, \ldots, a_n^i), [a]_j = (a_1^j, \ldots, a_n^j)$, where

$$a_k^l = R_l \mathbf{b}_k^T \cdot u, \quad l = i, j.$$

Then if both players act properly, we have

$$a = \sum_k \lambda_k a_k^i = \sum_k \lambda_k a_k^j$$

and hence

$$0 = \sum_k \lambda_k a_k^i - \sum_k \lambda_k a_k^j = \sum_k \lambda_k \left(a_k^i - a_k^j \right).$$

But this is the same as opening $[a]_i - [a]_j = (a_1^i - a_1^j, \ldots, a_n^i - a_n^j)$ as 0. Thus, we conclude that both commitments $[a]_i$ and $[a]_j$ must be commitments to the same value a. If they are different, it is impossible to open their difference as 0 and hence commitments remain consistent even if both P_i and P_j are bad.
If both players are good, it is clear that opening $[a]_i - [a]_j$ by broadcasting all values $(a_1^i - a_1^j, \ldots, a_n^i - a_n^j)$ does not reveal a by a similar method as in the proof of Proposition 5.8. More precisely, we take any $a' \in \mathbb{F}_q$ and φ as in Lemma 4.11 and consider the maps

$$\psi_i = \phi_i + (a' - a)\varphi \quad \text{and} \quad \psi_j = \phi_j + (a' - a)\varphi,$$

where ϕ_i, ϕ_j are the maps used by P_i, P_j to share a. We note that $\psi_i - \psi_j = \phi_i - \phi_j$ but $\psi_i(u) = \psi_j(u) = a'$. Hence, to the adversary any $a' \in \mathbb{F}_q$ seems equally likely to be the secret a, which means that a remains secret. □

We can use the CSP protocol and our commitment scheme from Protocol 8 to obtain a VSS protocol as explained in the following.

Protocol 11. We consider a dealer D who wants to VSS share a. For the commitment, we use the notation $[a]_D$. Then the protocol works as follows.

1. D commits to a as $[a]_D$.

2. D performs CSP on $[a]_D$. More precisely:

   (a) D chooses a random vector $\rho \in \mathbb{F}_q^e$ such that $\rho_1 = a$ and commits to its entries as $[\rho_2]_D, \ldots, [\rho_e]_D$.

   (b) Each player P_i can compute the commitment $[a_j]_D$ for all j and P_i receives the commitment $[a_i]_i$ by CTP, where P_i also learns their share $a_i = \mathbf{b}_i \cdot \rho$.

3. For the reconstruction, each player P_i opens $[a_i]_i$ and the other players verify the correctness of the opening.

4. Using some reconstruction vector λ such that $u = \sum \lambda_i \mathbf{b}_i$, the players reconstruct $a = \sum \lambda_i a_i$. They only use shares a_i that are consistent with the commitment.

Proposition 5.14. *Protocol 11 is a VSS protocol. Moreover, if the sharing is accepted, then reconstruction will never fail.*

Proof. We verify that 1-4 in Definition 5.1 are satisfied:

1. The good players will decide to output $ver_i = 1$ or $ver_i = 0$ based on public information. More precisely, if they see that a set of players in Γ accused the dealer during Steps 1 or 2, they will decide $ver_i = 0$, else they will output $ver_i = 1$. Hence, they will always agree.

2. It is clear that if the dealer is good, then all good players will see the correctness of the sharing.

3. We assume that all good players P_i accept the sharing. In particular, D is not disqualified and thus there exists a unique secret a they committed to after Step 1. This secret is consistently determined by the shares held by good players. In Step 2(a), there always exists a linear map ϕ corresponding to ρ and a bad dealer cannot choose to distribute wrong shares since the commitment to a_i is computed using the fact that this commitment scheme is homomorphic. Thus at this point, there exists a unique secret a shared via ρ. CTP ensures that the committed values remain consistent. Let us now see what happens during the Reconstruct step. Each player opens their share $[a_i]_i$. A good player will always be accepted, while a bad player will be rejected if they try to open a wrong value. There will be enough good players left to use the reconstruction procedure and compute the correct value a if D was good. Furthermore, it is clear that a must be consistent. We also note that the shares of a_j given to good players P_i must consistently determine a_j for all players P_j. This follows from the definition of CSP. As a consequence, no bad player can contribute an inconsistent share and thus the reconstruction will never fail.

4. We assume that D is good. Then, after Step 1 the adversary knows some shares of a via a random matrix R. We have already seen that they are not enough to find a. From Step 2, the adversary learns shares of ρ_k and

also the shares a_i of a via the linear map corresponding to ρ. These shares are again not enough to find a nor ρ_k. We also note that a good player will never complain about a good dealer during CTP and thus the adversary does not learn anything and at the end of share it cannot determine a. It was also shown before that in fact each element $a' \in \mathbb{F}_q$ is equally likely to be the secret given the information the adversary knows.

□

Example 24. In this example, we want to see how the VSS works, i.e., it consists of examples of the commitment protocol, CSP and CTP. To do so, we recall the setting from Example 22. Namely, let $K = \mathbb{F}_5$, $n = 4$ and $t = 1$ with an additional dealer D. A good dealer is not interesting and as before, we assume that D and P_3 are corrupt. However, for simplicity we assume that D correctly committed to $a = 1$ in the same way as in Example 22. This concludes Step 1. Next, D performs CSP on $[a]_D = [1]_D$. In this case, ρ has only two entries, e.g., take $\rho_2 = 2$. We assume that D correctly commits to $[\rho_2]_D = [2]_D$ using the matrix

$$R_{\rho_2} = \begin{pmatrix} 2 & 3 \\ 3 & 1 \end{pmatrix}.$$

This results in the shares $[2]_D = (0, 3, 1, 4)$. Next, a is LSSS shared using ρ, i.e., P_i receives the share $a_i = \mathbf{b}_i \cdot (1, 2)$. In particular, we have $(a_1, a_2, a_3, a_4) = (3, 0, 2, 4)$. Every player can compute their share of the commitment $[a_i]_D$ without interaction with any other player or the dealer using the linearity of the commitment, i.e, we use the formula

$$[a_i]_D = (\mathbf{b}_i)_1[a]_D + (\mathbf{b}_i)_2[\rho_2]_D = [1]_D + i[2]_D = (3, 0, 2, 4) + i(0, 3, 1, 4).$$

It follows that

$$[a_1]_D = (3, 3, 3, 3),$$
$$[a_2]_D = (3, 1, 4, 2),$$
$$[a_3]_D = (3, 4, 0, 1),$$
$$[a_4]_D = (3, 2, 1, 0).$$

It can be easily verified that these shares consistently determine their respective secret. This is the commitment that D has to use for the CTP protocol. D is not allowed to use a different one. This is due to the fact that these special commitments can be computed by all players and thus must be consistent with the commitments to a and ρ_2. In the next step, D sends to P_i the value a_i and the corresponding shares determining a_i computed as above. Each player P_i will then choose a random symmetric matrix R_i to commit to a_i as in Protocol 8 and then they will open the difference as 0. Let us show this at the example of P_3 assuming that P_3 commits correctly using the matrix

$$R_3 = \begin{pmatrix} 2 & 3 \\ 3 & 0 \end{pmatrix}$$

and then explain the opening as 0. The commitment of P_3 to a_3 consists of the shares $[a_3]_3 = (0, 3, 1, 4)$. Let us first assume that P_3 acts correctly during Open. Then P_3 broadcasts the vector $(3, 1, 4, 2)$ and everybody can verify that these shares determine 0 consistently and see the CTP protocol as correct. For example, we have $2 \cdot 3 + 4 \cdot 1 = 4 \cdot 4 + 2 \cdot 2 = 0$. P_3 cannot broadcast any wrong values since they would not be consistent. If D tried to sabotage the CTP protocol by sending inconsistent shares to a good player P_i, then they would be forced to open $[a_i]_D$ in public, which could lead to the disqualification of D. Let us omit the sharing of the good players via CTP and assume that everybody acted correctly.
For the reconstruction, each player opens their share $[a_i]_i$. If the opening is rejected, this share will be disregarded for the reconstruction of the secret a. However, shares belonging to a good player P_i will always be accepted and all shares that are accepted must be consistent. For example, if P_3 tries to open their share by broadcasting the wrong values $(0, 3, 2, 2)$, then only P_4 will complain. However, during the opening process, the good players will see that there is no consistent secret and they will not use this share. On the other hand, the opening of a_1, a_2, a_4 must succeed and these shares will in fact determine a consistent secret a.

Next, we need one more auxiliary protocol, called *commitment multiplication protocol (CMP)*. Such a protocol starts from commitments $[a]_i, [b]_i, [c]_i$ and allows player P_i to convince the other players that indeed $ab = c$. If P_i remains honest, the protocol must leak no information to the adversary beyond the fact that $ab = c$. Moreover, if $[c]_i$ is opened, the adversary must learn nothing about a, b beyond what is implied by c and the information it holds. In other words, the CMP is a ZK protocol. The following protocol works for any homomorphic commitment scheme as long as it is unconditionally binding.

Protocol 12 (CMP, [11, Chapter 7.1]). We assume that our commitment scheme is homomorphic and *unconditionally binding*. This means that there exists a unique committed value that does not rely on the computational hardness of some underlying problem. In other words, $[a]_i$ contains the unique value a.

1. **Inputs**: The commitments $[a]_i, [b]_i, [c]_i$, where P_i claims that $ab = c$. P_i chooses a random β and makes commitments $[\beta]_i, [\beta b]_i$.

2. The other players jointly generate a random challenge r using standard techniques, i.e., each player broadcasts some value e_j and the players then set $r = \sum e_j$.

3. P_i opens the commitments $r[a]_i + [\beta]_i$ to reveal a value r_1. P_i opens the commitment $r_1[b]_i - [\beta b]_i - r[c]_i$ to reveal 0.

4. If any of these openings fail, the proof is rejected, else it is accepted.

Remark. This protocol can be seen as a generalization of Protocol 4. In this context, we have $\beta = x$ and $r_1 = z$. We also have $[\beta]_i = M_1 = g^x h^{s_1}$ and

$[\beta b]_i = M_2 = B^x h^{s_2}$. Opening and revealing r_1 is equivalent to broadcasting z and w_1 and the opening of 0 is the same as verifying the equality $B^z h^{w_2} = M_2 C^e$. Namely, $g^0 h^{w_2} = M_2 C^e B^{-z}$, which means that the dealer opens $M_2 C^e B^{-z}$ as 0 by revealing w_2. The additional check that $g^y h^w = MB^e$ is not necessary in the context of Protocol 12 since we assumed the commitment to be unconditionally binding.

Proposition 5.15. *If P_i acts properly, P_i will always pass Protocol 12 and the adversary does not learn a, b, c. Moreover, the probability that an incorrect commitment $c \neq ab$ passes the protocol is exactly $\frac{1}{q}$.*

Proof. It is clear that if P_i is good, then r_1 will be random and thus the adversary cannot deduce a, b, c. Now we want to examine what can happen if $c \neq ab$. Then we also want P_i to falsely commit to any $[d]_i$ instead of $[\beta b]_i$. We observe that if $r = \frac{d-\beta b}{ab-c}$, then P_i can open $r_1[b]_i - [d]_i - r[c]_i$ as 0. More precisely,

$$r_1 b - d - rc = (ra+\beta)b - d - rc = \beta b - d + r(ab-c) = \beta b - d + \frac{d-\beta b}{ab-c}(ab-c) = 0.$$

However, as the bad players cannot manipulate the outcome of r, the probability that $r = \frac{d-\beta b}{ab-c}$ is only $\frac{1}{q}$ which is negligible if q is large. In fact, this is the only value that would allow P_i to pass the protocol undetected. Therefore, if after P_i commits in Step 1, the other players impose two different challenges $r \neq r'$ and P_i can solve both, then $ab = c$ with probability exactly 1. But this is impossible to implement since if P_i is good, we have

$$r_1 - r_1' = ra + \beta - (r'a + \beta) = a(r - r')$$

and thus the adversary could compute $a = \frac{r_1 - r_1'}{r - r'}$. So, the best we can do, is to run the whole protocol multiple times, including that P_i commits to different values β, which will then decrease the error probability by factor q each time. □

Protocol 12 may not detect a false commitment. The probability to do so is very low and can be further decreased by iterating the protocol. However, there is a better protocol with an error probability of exactly 0 as seen in Cramer, Damgård, and Maurer [10, Chapter 7.1]. This protocol works only for the commitment scheme explained in Protocol 8 based on an LSSS that has the strong multiplication property and is explained in the following. To do so, we first need the following lemma, which simply is a reformulated version of Lemma 4.14.

Lemma 5.16. *Let $\mathcal{M}$ be an LSSS and assume that s, s' were shared using $\mathcal{M}$ resulting in shares $\mathbf{s} = (s_1, \ldots, s_n), \mathbf{s}' = (s_1', \ldots, s_n')$ respectively. Consider the vector $\mathbf{s} * \mathbf{s}' = (s_1 s_1', \ldots, s_n s_n')$. Then this is a vector of shares of ss' in a new LSSS $\mathcal{M}'$, which can be computed from $\mathcal{M}$. Moreover, if $\mathcal{M}$ has the strong multiplication property, then for any $A \in \mathcal{A}$ we have $B = \mathcal{U} \backslash A \in \Gamma(\mathcal{M}')$.*

Proof. Recall that for $\mathbf{b}_i = (b_{i1}, \ldots, b_{ie})$ we use the notation

$$\widehat{\mathbf{b}}_i = \mathbf{b}_i \otimes \mathbf{b}_i = (b_{i1}b_{i1}, b_{i1}b_{i2}, \ldots, b_{ie}b_{ie}) \in K^{e^2}.$$

These $\widehat{\mathbf{b}}_i$ define a new LSSS $\mathcal{M}'$ over K^{e^2}. Similarly as in the proof of Lemma 4.14, we note that $s_i s'_i$ are shares of ss' via the linear map $\mathbf{a} \otimes \mathbf{a}'$ in the LSSS $\mathcal{M}'$, where $\mathbf{a}, \mathbf{a}'$ are the linear maps used for sharing s, s' respectively. The claim about B follows immediately from the definition of the strong multiplication property. □

Protocol 13. Consider some dealer D who already committed to a, b, c as $[a]_D, [b]_D, [c]_D$ following Protocol 8. D wants to prove that $[c]_D$ is in fact a commitment to $c = ab$. To do so, D proceeds as follows.

1. D uses CSP on $[a]_D$ and $[b]_D$ resulting in commitments $[a_j]_j, [b_j]_j$. We denote the vectors used for sharing a, b by $\rho^a = (a, \rho_2^a, \ldots, \rho_e^a), \rho^b = (b, \rho_2^b, \ldots, \rho_e^b)$ respectively.

2. D computes the new vector $\rho = \rho^a \otimes \rho^b = (ab, \rho_2, \ldots, \rho_{e^2})$ and commits to the entries $\rho_2, \ldots, \rho_{e^2}$. D already committed to the first entry $\rho_1 = ab = c$ before. Using these commitments, every player can compute the commitments
$$[c_j]_D = ([c]_D, [\rho_2]_D, \ldots, [\rho_{e^2}]_D) \cdot \widehat{\mathbf{b}}_j$$
since the commitment is homomorphic.

3. By Lemma 5.16 we have $c_j = a_j b_j$. This can be verified as follows.
   (a) D privately sends all shares defining $[c_j]_D$ to P_j. If P_j sees at this point that $c_j \neq a_j b_j$ or that the shares are inconsistent, proceed with Step (c).
   (b) P_j commits independently to $[c_j]_j = [a_j b_j]_j$. The difference $[c_j]_D - [c_j]_j$ is opened in public to reveal 0. If this fails, proceed with Step (c).
   (c) If we arrive here, then either D or P_j is corrupt. Then D has to open $[c_j]_D$ and P_j has to open $[a_j]_j, [b_j]_j$ in public. If $c_j \neq a_j b_j$ or the opening of $[c_j]_D$ fails, D is disqualified.

Remark. Step 3(b) is important to ensure that D sends the same shares to P_j which were computed during Step 2. There may still be some incorrect values, but they must correspond to a set in $\mathcal{A}$, otherwise the step fails. The method explained in Step 3(b) is the same method as used in Protocol 10 for the CTP but there are different methods to achieve what we need. For example, in Cramer, Damgård, and Maurer [10], each player P_i privately sends their share of $[c_j]_D$ to P_j. Then P_j checks if only a set in $\mathcal{A}$ differs from what they received in Step 3(a) and if so, P_j disregards those values and checks that the remaining values consistently determine $c_j = a_j b_j$. If anything of this fails, we proceed with Step (c) as before.

Proposition 5.17. *A commitment $[c]_D$ to the value c passes Protocol 13 if and only if $c = ab$. Moreover, if D is good, the protocol does not reveal a, b, c to the adversary.*

Proof. First, we note that if D is good, then all their actions are correct and must be accepted. Furthermore, an adversary complaining in Step 3 will only learn the value c_j, which was already known to it. Let us now assume that D is bad. Due to properties of our commitment scheme, there are unique committed values a, b, c. CSP ensures that the shares held by good players determine a unique secret. Step 3 is made in such a way that all good players will check if $c_j = a_j b_j$ and if this check fails, D will be disqualified. Moreover, the shares of c_j opened to P_j must be consistent with the shares computed during Step 2. Thus, all good players must hold correct shares of $c = ab$. It is important to note that in Step 2 the commitments to ρ always define a unique linear map and if D passes the protocol, the shares via this map belonging to the good players P_j are indeed the correct shares $a_j b_j$. This means that the shares of the commitment $[c]_D$ held by good players must consistently determine the secret $c = ab$ by Lemma 5.16. However, the opening of $[c]_D$ may still fail if D does not act properly. □

Now we have all the tools to obtain a secure MPC protocol based on Cramer, Damgård, and Maurer [11, Chapter 7.2]. Namely, we will combine all auxiliary protocols and use our special commitment scheme, which combine to the following protocol.

Protocol 14. We assume $\mathcal{M}$ to have the strong multiplication property and let λ be such that $u = \sum_{i \in B} \lambda_i \mathbf{b}_i$ for some set $B \in \Gamma(\mathcal{M})$. Let $a, b \in \mathbb{F}_q$ be two values such that the players want to compute $a + b$ and $a \cdot b$. We assume that a, b are VSS shared using Protocol 11. So now, each player P_i is committed to shares a_i, b_i as $[a_i]_i, [b_i]_i$. We note that by properties of the commitment scheme, there exist unique committed secrets a, b defined by the good players and by CSP the shares a_i, b_i must be consistent.

Addition: For $i = 1, \ldots, n$, P_i computes $a_i + b_i$ and the players compute $[a_i + b_i]_i$. By linearity of the secret sharing and commitments, $[a_i + b_i]_i$ determine $a + b$. More precisely, each player P_i opens $[a_i + b_i]_i$ as $a_i + b_i$ and then we have

$$a + b = \sum_{i \in B} \lambda_i (a_i + b_i).$$

The commitments ensure correctness and all good players must be accepted. Hence, there always exists an appropriate set $B \in \Gamma(\mathcal{M})$.

Multiplication: For $i = 1, \ldots, n$, P_i computes $a_i b_i = \tilde{c}_i$ and commits to it. Then P_i performs CMP on inputs $[a_i]_i, [b_i]_i, [\tilde{c}_i]_i$ using either Protocol 12 or 13. If a player fails, they are disqualified and the other players continue without them.

Resharing Step: P_i follows Protocol 9 to perform CSP on $[\tilde{c}_i]_i$, resulting in commitments $[c_{i1}]_1, \ldots, [c_{in}]_n$. If a player fails, they are disqualified and the other players proceed without them.

Recombination Step: For $j = 1, \ldots, n$, P_j computes $c_j = \sum_i r_i c_{ij}$ for some fixed recombination vector $\mathbf{r} = (r_1, \ldots, r_n)$, where $r_i = 0$ for all players who failed CMP or the Resharing Step. Such a vector always exists due to the strong

multiplication property. All players compute

$$[c_j]_j = \sum_i r_i [c_{ij}]_j = \left[\sum_i r_i c_{ij} \right]_j .$$

By the multiplication property and linearity of $\mathcal{M}$, the commitments $[c_1]_1, \ldots, [c_n]_n$ determine $c = ab$. Next, the commitments to c_j are opened and if a player fails the opening, they are disqualified. All players compute $c = \sum_{j \in B} \lambda_j c_j$ for some set $B \in \Gamma(\mathcal{M})$ containing no disqualified player by the same methods as in Protocol 7. Since the set of all good players is in $\Gamma(\mathcal{M})$ and no good player will be disqualified, such a set always exists.

Theorem 5.18. *Protocol 14 is secure against an active adversary.*

Proof. First, we note that this protocol is still secure against a passive adversary by the exact same methods as seen in the proof of Proposition 5.8 and the security of all auxiliary protocols involved. Furthermore, the additional steps involving commitments ensure that a bad player cannot open wrong values. If a bad player tries to do so, the opening will be rejected and the player will be disqualified. Due to the strong multiplication property, the bad players are not necessary to reconstruct $c = ab$. □

Remark. In Section 5.1, we were able to generalize Protocol 5 to work for $n \geq 2t + 1$ players. We cannot apply an analogous generalization to Protocol 14, i.e., we cannot make it work if we only assume the multiplication property. The problem is that the commitment scheme in Protocol 8 requires our LSSS $\mathcal{M}$ to be Q3. Since having the strong multiplication property implies being Q3, this assumption is satisfied in Protocol 14. However, having the multiplication property does not imply being Q3, e.g., Shamir's LSSS has the multiplication property for $n \geq 2t + 1$ by application of Protocol 2 but is Q3 if and only if $n \geq 3t+1$. This means that it is impossible to make Protocol 14 work if we only assume the multiplication property as long as we use Protocol 8 as commitment scheme.

We want to conclude this section with some remarks and general thoughts about the protocols described in Section 5.2 based on claims as seen in Cramer, Damgård, and Maurer [11]. First, we note that Protocol 8 is a better commitment scheme than the one based on the DLP in the sense that it does not rely on the hardness of the DLP or any other problem that is assumed to be hard. This means that it will still work on a quantum computer, where the DLP can be solved using the methods described in Shor [24] and thus Protocol 5 is not secure anymore. In fact, it even implies that Protocol 14 is secure against any computationally unbounded adversary.
The second remark we want to make is about how the adversary corrupts players. Recall that up to now we always assumed an adversary to be static, i.e., that it can only corrupt players at the beginning of the protocol. However, there is also what is called an *adaptive* adversary that can corrupt other players at

any point during the protocol as long as the set of all corrupted players A is still an element of the adversary structure $\mathcal{A}$. It follows that all protocols in Section 5.2 still work for an adaptive adversary since there is essentially no difference between P_i being corrupted at some point during the protocol or them being corrupt from the start but acting correctly up to that point. A more precise justification can be found in Cramer, Damgård, and Maurer [11].

References

[1] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness theorems for non-cryptographic fault-tolerant distributed computation". In: *Symposium on the Theory of Computing*. 1988. URL: https://api.semanticscholar.org/CorpusID:207554159.

[2] Siegfried Bosch. *LINEARE ALGEBRA: ein grundkurs mit aufgabentrainer*. Springer, 2021. URL: https://link.springer.com/book/10.1007/978-3-662-62616-0.

[3] Wieb Bosma, John Cannon, and Catherine Playoust. "The Magma algebra system. I. The user language". In: *J. Symbolic Comput.* 24.3-4 (1997). Computational algebra and number theory (London, 1993), pp. 235–265. ISSN: 0747-7171. DOI: 10.1006/jsco.1996.0125. URL: http://dx.doi.org/10.1006/jsco.1996.0125.

[4] Hao Chen and Ronald Cramer. "Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computations over Small Fields". In: *Advances in Cryptology - CRYPTO 2006*. Ed. by Cynthia Dwork. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 521–536. ISBN: 978-3-540-37433-6.

[5] *CoCalc*. URL: https://cocalc.com/ (visited on 08/29/2024).

[6] Alain Couvreur, Philippe Gaborit, Valérie Gauthier-Umaña, Ayoub Otmani, and Jean-Pierre Tillich. "Distinguisher-based attacks on public-key cryptosystems using Reed–Solomon codes". In: *Designs, Codes and Cryptography* 73 (2014), pp. 641–666.

[7] Alain Couvreur, Irene Márquez-Corbella, and Ruud Pellikaan. "A polynomial time attack against algebraic geometry code based public key cryptosystems". In: *2014 IEEE International Symposium on Information Theory*. 2014, pp. 1446–1450. DOI: 10.1109/ISIT.2014.6875072.

[8] Alain Couvreur, Irene Márquez-Corbella, and Ruud Pellikaan. "Cryptanalysis of McEliece Cryptosystem Based on Algebraic Geometry Codes and Their Subcodes". In: *IEEE Transactions on Information Theory* 63.8 (2017), pp. 5404–5418. DOI: 10.1109/TIT.2017.2712636.

[9] Ronald Cramer and Ivan Damgård. "Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free?" In: *Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*. Springer. 1998, pp. 424–441.

[10] Ronald Cramer, Ivan Damgård, and Ueli Maurer. *General Secure Multi-Party Computation from any Linear Secret-Sharing Scheme*. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7fe4535458d691971b0b9502414900660e2c9db9.

[11] Ronald Cramer, Ivan Damgård, and Ueli Maurer. "General Secure Multiparty Computation from any Linear Secret-Sharing Scheme". In: *Advances in Cryptology — EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 316–334. ISBN: 978-3-540-45539-4.

[12] Paul Feldman. "A practical scheme for non-interactive verifiable secret sharing". In: *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. 1987, pp. 427–438. DOI: 10.1109/SFCS.1987.4.

[13] Rosario Gennaro, Michael O Rabin, and Tal Rabin. *Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography*. 1998. URL: https://mit6875.github.io/PAPERS/GennaroRabinRabin.pdf (visited on 03/24/2024).

[14] Rosario Gennaro, Michael O Rabin, and Tal Rabin. "Simplified VSS and fast-track multiparty computations with applications to threshold cryptography". In: *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*. 1998, pp. 101–111.

[15] Jens Kopp. *Attack_magma*. URL: https://github.com/jenskopp99/Attack_magma.git (visited on 09/05/2024).

[16] Irene Márquez-Corbella, Edgar Martínez-Moro, Ruud Pellikaan, and Diego Ruano. "Computational aspects of retrieving a representation of an algebraic geometry code". In: *Journal of Symbolic Computation* 64 (2014). Mathematical and computer algebra techniques in cryptology, pp. 67–87. ISSN: 0747-7171. DOI: https://doi.org/10.1016/j.jsc.2013.12.007. URL: https://www.sciencedirect.com/science/article/pii/S0747717113001740.

[17] R. J. McEliece and D. V. Sarwate. "On sharing secrets and Reed-Solomon codes". In: *Commun. ACM* 24.9 (Sept. 1981), pp. 583–584. ISSN: 0001-0782. DOI: 10.1145/358746.358762. URL: https://doi.org/10.1145/358746.358762.

[18] David Mumford. "Varieties Defined by Quadratic Equations". In: *Questions on Algebraic Varieties*. Ed. by E. Marchionna. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 29–100. ISBN: 978-3-642-11015-3. DOI: 10.1007/978-3-642-11015-3_2. URL: https://doi.org/10.1007/978-3-642-11015-3_2.

[19] Torben Pryds Pedersen. "Non-interactive and information-theoretic secure verifiable secret sharing". In: *Annual international cryptology conference*. Springer. 1991, pp. 129–140.

[20] Ruud Pellikaan. "On decoding by error location and dependent sets of error positions". In: *Discrete Mathematics* 106-107 (1992), pp. 369–381. ISSN: 0012-365X. DOI: https://doi.org/10.1016/0012-365X(92)90567-Y. URL: https://www.sciencedirect.com/science/article/pii/0012365X9290567Y.

[21] Ruud Pellikaan, Xin-Wen Wu, Stanislav Bulygin, and Relinde Jurrius. *Codes, cryptology and curves with computer algebra*. Cambridge University Press, 2017.

[22] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.1)*. 2021. URL: https://www.sagemath.org.

[23] Adi Shamir. "How to share a secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://doi.org/10.1145/359168.359176.

[24] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172. eprint: https://doi.org/10.1137/S0097539795293172. URL: https://doi.org/10.1137/S0097539795293172.

[25] H. Stichtenoth. "A note on Hermitian codes over GF(q/sup 2/)". In: *IEEE Transactions on Information Theory* 34.5 (1988), pp. 1345–1348. DOI: 10.1109/18.21267.

[26] Henning Stichtenoth. *Algebraic function fields and codes*. Vol. 254. Springer Science & Business Media, 2009.

[27] Henning Stichtenoth. "Über die Automorphismengruppe eines algebraischen Funktionenkörpers von Primzahlcharakteristik: Teil II: Ein spezieller Typ von Funktionenkörpern". In: *Archiv der Mathematik* 24 (1973), pp. 615–631.

[28] M.A. Tsfasman, S.G. Vlăduţ, and D. Nogin. *Algebraic Geometric Codes: Basic Notions: Basic Notions*. Mathematical surveys and monographs Bd. 1. American Mathematical Society, 2007. ISBN: 9780821843062. URL: https://books.google.ch/books?id=kif0BwAAQBAJ.

A SageMath Codes

In this section, we give the codes used in Section 3.3 based on SageMath [22]. For convenience, we used its implementation via the online tool CoCalc [5]

Algorithm A.1. The following function is used to generate an AG-code based on a Hermitian curve as defined in Definition 3.16. It takes the inputs r and m and outputs the generator matrix G of the code $C_L(\mathcal{H}_r, \mathcal{P}, E)$. We used this code for various examples, most notably Example 8.

```
def Hermitian_code(r,m):
    n=r^3
```

```
    q=r^2
    F=GF(q,"a")
    x,y=F["x,y"].gens()
    X=Curve(x^(r+1)-y^r-y)
    K=X.function_field()
    pls=K.places()
    Q,=X.places_at_infinity()
    pls.remove(Q)
    E=m*Q
    C=codes.EvaluationAGCode(pls,E)
    G=C.generator_matrix()
    return G
```

Remark. If we want the field $\mathbb{F}_q$ to use a particular representation such as in Example 15, we need to specify this in the construction, i.e., we use the following line of code:

```
    F.<a>=GF(q,modulus=x^2+1)
```

Algorithm A.2. The following function takes as input the generator matrices A, B of two codes over the same base field F with same length n and computes the generator matrix S of the Schur product of these codes as defined in Definition 2.1.

```
def Schur(A,B,F):
    n=A.ncols()
    k_1=A.nrows()
    k_2=B.nrows()
    if A==B:
        d=(k_1+1)*k_1/2
    else:
        d=k_1*k_2
    H=zero_matrix(F,d,n)
    l=0
    if A==B:
        for i in range(k_1):
            for j in range(i,k_1):
                g_i=A[i,:]
                g_j=B[j,:]
                g=g_i.elementwise_product(g_j)
                H[l,:]=g
                l=l+1
    else:
        for i in range(k_1):
            for j in range(k_2):
                g_i=A[i,:]
                g_j=B[j,:]
```

```
                g=g_i.elementwise_product(g_j)
                H[l,:]=g
                l=l+1
    C=codes.LinearCode(H)
    S=C.generator_matrix()
    return S
```

Algorithm A.3. The following code first defines the function *closed* which, given the correct input values corresponding to $C^{(s-1)}, C^{(s)}$, can compute any $\overline{C}^s$ as defined in Definition 2.11. For example, *twoclosed* will compute $\overline{C}^2$. All actions are performed via the respective generator matrices. We also use the function *Schur* as described in Algorithm A.2. This algorithm is used for the computations in Example 8.

```
def closed(G,G_2,F):
    C_2=codes.LinearCode(G_2)
    H=C_2.parity_check_matrix()
    H=H.transpose()
    k=G.nrows()
    W=VectorSpace(F,n)
    for i in range(k):
        g_i=vector(G[i,:])
        b=diagonal_matrix(F,g_i)
        K=(b*H).kernel()
        W=W.intersection(K)
    W=matrix(W.basis())
    return W

def twoclosed(G,F):
    G_2=Schur(G,G,F)
    W=closed(G,G_2,F)
    return W
```

B Magma Codes

While trying to implement the attack seen in Section 3.2, we noticed that neither CoCalc [5] nor the implementation of SageMath [22] at the university worked particularly well. Therefore, we decided to try it with a different programming language, namely the one used in Couvreur, Márquez-Corbella, and Pellikaan [7]: magma [3]. In this section, we give some of the codes used in Section 3.3 using *Magma V2.27-7*. A more complete version compiling all our functions and including Example 9 can also be found at [15]. We also want to note that magma does not require indents or line breaks. However, they improve readability, hence we used them.
At this point we also want to recall that the attack only works if $m = \deg E$

satisfies certain conditions, i.e.,

$$\frac{n}{2} > m > 3g.$$

We always check that this is satisfied before running a particular example.

Algorithm B.1. The following function computes the Schur product and works analogously to Algorithm A.2. As before, it requires as the inputs two generator matrices and gives as an output a generator matrix of the corresponding Schur product.

```
function Schur(A,B,F)
    n:=Ncols(A);
    k_1:=Nrows(A);
    k_2:=Nrows(B);
    l:=1;
        d:=k_1*k_2;
        H:=ZeroMatrix(F,d,n);
        for i in {1..k_1} do
            for j in {1..k_2} do
                g_i:=Vector(RowSubmatrix(A,i,1));
                g_j:=Vector(RowSubmatrix(B,j,1));
                g:=g_i*g_j;
                H[l]:=g;
                l:=l+1;
            end for;
        end for;
    H:=EchelonForm(H);
    RemoveZeroRows(~H);
    return H;
end function;
```

Algorithm B.2. This function computes from a generator matrix G the linear code D, which is the same as the code B_1 defined in Section 3.2.

```
function find_B1(G,F)
    k_1:=Nrows(G);
    n:=Ncols(G);
    B:=G;
    for i in {1..k_1} do
        if G[i,1] ne 0 then
            B[i]:=Vector(ZeroMatrix(F,1,n));
        end if;
    end for;
    RemoveZeroRows(~B);
    D:=LinearCode(B);
    return D;
end function;
```

Algorithm B.3. This function computes an ECP (A, B) using Theorem 3.13. It takes as inputs a generator matrix G over the field F and the error correcting capacity t. The output values correspond to $\mathcal{B}_{t+g}$ and $\mathcal{B}_{t+g+1}$. This function also uses the functions *Schur* as defined in Algorithm B.1 and *find_B1* as defined in Algorithm B.5 below.

```
function ECP(G,F,t)
    G_2:=Schur(G,G,F);
    k_1:=Nrows(G);
    k_2:=Nrows(G_2);
    n:=Ncols(G);
    m:=k_2-k_1;
    g:=k_2-2*k_1+1;
    B_0:=LinearCode(G);
    B_1:=find_B1(G,F);
    B_minus:=B_0;
    B_s:=B_1;
    V:=VectorSpace(F,n);
    for s in {2..(t+g+1)} do
        G_s:=GeneratorMatrix(B_s);
        G_minus:=GeneratorMatrix(B_minus);
        G_s2:=Schur(G_s,G_s,F);
        B_s2:=LinearCode(G_s2);
        H_s2:=ParityCheckMatrix(B_s2);
        W:=VectorSpace(B_s);
        for j in {1..(Nrows(G_minus))} do
            b:=DiagonalMatrix(F,ElementToSequence(G_minus[j]));
            K:=b*(Transpose(H_s2));
            L:=Kernel(K);
            W:=W meet L;
        end for;
        B_minus:=B_s;
        J:=Matrix(Basis(W));
        B_s:=LinearCode(J);
    end for;
    return B_minus, B_s;
end function;
```

Algorithm B.4. The function *ECP_alt* is an implementation of the alternative method seen in Theorem 3.15. We also need the auxiliary function *ECP_alt_aux*. In short, the function *ECP_alt_aux* computes $\mathcal{B}_S$ from $\mathcal{B}_{\lfloor (S+1)/2}$ and $\mathcal{B}_{S/2}$ and the function *ECP_alt* then uses this iteratively. In the end, it also outputs the codes $\mathcal{B}_{t+g}$ and $\mathcal{B}_{t+g+1}$ and similarly as above, it also uses the functions *Schur* as defined in Algorithm B.1 and *find_B1* as defined in Algorithm B.5.

```
function ECP_alt_aux(F, G, B_s_half, B_splus1_half)
    G_s_half:=GeneratorMatrix(B_s_half);
```

```
    G_splus1_half:=GeneratorMatrix(B_splus1_half);
    G_Schur:=Schur(G_s_half,G_splus1_half,F);
    B_Schur:=LinearCode(G_Schur);
    H_Schur:=ParityCheckMatrix(B_Schur);
    W:=VectorSpace(B_splus1_half);
    for j in {1..(Nrows(G))} do
        b:=DiagonalMatrix(F,ElementToSequence(G[j]));
        K:=b*(Transpose(H_Schur));
        L:=Kernel(K);
        W:=W meet L;
    end for;
    J:=Matrix(Basis(W));
    B_s:=LinearCode(J);
    return B_s;
end function;

function ECP_alt(G,F,t)
    G_2:=Schur(G,G,F);
    k_1:=Nrows(G);
    k_2:=Nrows(G_2);
    n:=Ncols(G);
    m:=k_2-k_1;
    g:=k_2-2*k_1+1;
    B_0:=LinearCode(G);
    B_1:=find_B1(G,F);
    b:=Floor(Log(2,n))+1;
    list_index:=[0,1];
    list_codes:=[B_0,B_1];
    L:=SetToSequence({1..(b-2)});
    Reverse(~L);
    for s in L do
        S:=Floor((t+g)/(2^(s-1)));
        if Index(list_index, S) eq 0 then
            s_1:=Floor(((t+g)/(2^(s-1))+1)/2);
            s_2:=Floor((t+g)/(2^s));
            i_1:=Index(list_index, s_1);
            i_2:=Index(list_index,s_2);
            B_tg:=ECP_alt_aux(F,G,list_codes[i_2],list_codes[i_1]);
            Append(~list_index, S);
            Append(~list_codes, B_tg);
        end if;
        S:=Floor((t+g)/(2^(s-1))+1);
        if Index(list_index, S) eq 0 then
            s_1:=Floor(((t+g)/(2^(s-1))+2)/2);
            s_2:=Floor(((t+g)/(2^(s-1))+1)/2);
            i_1:=Index(list_index, s_1);
```

```
            i_2:=Index(list_index,s_2);
            B_tgplus:=ECP_alt_aux(F,G,list_codes[i_2],list_codes[i_1]);
            Append(~list_index, S);
            Append(~list_codes, B_tgplus);
        end if;
    end for;
    return B_tg, B_tgplus;
end function;
```

Algorithm B.5. The following two functions compute a non-degenerated code using Theorem 3.14 and find the pair (A, B) using Corollary 3.10 respectively. In particular, the function *nondeg* takes as input the linear codes *B_tg, B_tgplus* and outputs the generator matrix H of the desired form. The function *pair* takes as inputs the generator matrix G as well as the generator matrix B of some code such as in Corollary 3.10. It then computes the corresponding generator matrix A. The function *pair* also uses the function *Schur* as defined in Algorithm B.1.

```
function nondeg(B_tg,B_tgplus)
    G:=GeneratorMatrix(B_tg);
    M:=0;
    for i in {1..Nrows(G)} do
        if G[i] notin VectorSpace(B_tgplus) then
            M:=M+1;
            break;
        end if;
    end for;
    H:=GeneratorMatrix(B_tgplus);
    v:=G[M];
    v[1]:=1;
    VerticalJoin(~H,v);
    return H;
end function;

function pair(F,B,G)
    C:=LinearCode(G);
    H:=ParityCheckMatrix(C);
    S:=Schur(B,H,F);
    D:=LinearCode(S);
    A:=ParityCheckMatrix(D);
    return A,B;
end function;
```

The above functions described in the algorithms can then be combined to realize the attack seen in Section 3.2 and be used for Example 9. At [15] we also give an implementation of Algorithm 3.7 including a very basic implementation of an erasure decoding algorithm. It also contains the following function that generates an example based on Hermitian curves.

Algorithm B.6. The following function takes as inputs a prime power r and the degree of the divisor m. Similarly to Algorithm A.1, it generates the code $C_L(\mathcal{H}_r, \mathcal{P}, E)$ and also some secret message M with added error vector such as in the McEliece encryption scheme. For the purpose of the example, it also returns the error correcting probability t as well as the correct message M, so that we can later easily verify the correctness of the decoding.

```
function generate_example(r,m)
    g:=((r-1)*r) div 2;
    d:=m-2*g+2;
    t:=(d-g-1) div 2;
    n:=r^3;
    k:=n-(m-g+1);
    F<a>:=GF(r^2);
    P2<x,y,z> := ProjectiveSpace(F, 2);
    f:=y^r*z+y*z^r-x^(r+1);
    X:=Curve(P2,f);
    pls:=Places(X,1);
    p:=X ! [0,1,0];
    P:=Places(p);
    O:=P[1];
    Exclude(~pls, O);
    E:=m*(DivisorGroup(X)!O);
    C:=AlgebraicGeometricCode(pls,E);
    G:=ParityCheckMatrix(C);
    C:=LinearCode(G);
    V:=VectorSpace(F,k);
    M:=Random(V);
    c:=M*G;
    e:=Vector(ZeroMatrix(F,1,n));
    L:={1..n};
    for i in {1..t} do
        l:=Random(L);
        Exclude(~L,l);
        a:=Random(F);
        e[l]:=a;
    end for;
    R:=c+e;
    return G,F,R,t,M;
end function;
```