HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Engineering Physics and Mathematics

Lasse Leskelä

IMPLEMENTING ARITHMETIC FOR ELLIPTIC CURVE CRYPTOSYSTEMS

Master's Thesis

Supervisor: Professor Olavi Nevanlinna
Instructor: PhD Valtteri Niemi

Helsinki, 11th January 1999

Tekijä: Lasse Leskelä

Työn nimi: Aritmetiikan toteuttaminen elliptisen käyrän kryptosysteemeihin

English title: Implementing Arithmetic for Elliptic Curve Cryptosystems

Päivämäärä: 11. tammikuuta 1999 **Sivumäärä:** 87

Osasto: Teknillisen fysiikan ja matematiikan osasto

Professuuri: Mat-1 Matematiikka

Työn valvoja: Professori Olavi Nevanlinna

Työn ohjaaja: FT Valtteri Niemi

Työ käsittelee elliptisiin käyriin pohjautuvien kryptosysteemien tehokasta toteuttamista. Näiden systeemien suorituskykyä tarkastellaan vertailemalla äärellisten kuntien ja elliptisten käyrien aritmetiikan eri laskentamenetelmiä. Lisäksi tehdään katsaus kehittyneimpiin algoritmeihin elliptisen käyrän diskreetin logaritmin ongelman ratkaisemiseksi, mikä luo matemaattista pohjaa olettamukselle, että elliptisiin käyriin perustuvilla julkisen avaimen kryptosysteemeillä voidaan saada aikaan vahva salaus suhteellisen lyhyillä avaimilla.

Elliptisten käyrien laaja matemaattinen teoria juontaa juurensa algebrallisesta geometriasta ja lukuteoriasta. Lyhyyden vuoksi muutama lause, joiden todistaminen vaatisi syvällisempää tietämystä näiltä aloilta, on esitetty suoraan ilman todistusta. Näitä lauseita tarvitaan elliptisen käyrän ryhmärakenteen selvittämiseksi, mikä puolestaan on kyseiseen käyrään perustuvan kryptosysteemin turvallisuuden kannalta merkittävä tekijä.

Elliptisen käyrän ryhmäoperaatio voidaan laskea suorittamalla muutama laskutoimitus kunnassa, jonka päälle kyseinen käyrä on rakennettu. Koska kryptografiassa käytettävät kunnat ovat äärellisiä, tästä seuraa, että äärellisen kunnan nopeat laskenta-algoritmit ovat tärkeitä suunniteltaessa elliptisen käyrän aritmetiikan tehokkaita toteutuksia. Tämän vuoksi äärellisiä kuntia käsitellään työssä yksityiskohtaisesti, painottuen kuntiin, joiden karakteristika on 2.

Mikä tekee karakteristikan 2 äärellisistä kunnista kiinnostavia on se tosiseikka, että nämä voidaan tulkita vektoriavaruuksiksi yli kunnan $\mathbb{F}_2 = \{0, 1\}$. Kyseiset kunnat voidaan näin ollen luontevasti esittää kiinteän pituisina bittijonoina, mikä puolestaan johtaa hyvin nopeisiin kuntaoperaatioiden toteutuksiin käyttäen logiikkapiirejä tai yleiskäyttöisiä mikroprosessoreita. Työssä esitetään kuvaus eräistä aritmetiikan toteutuksista suurille karakteristikan 2 kunnille.

Elliptisen käyrän kryptosysteemeissä käytettävistä laskutoimituksista tärkein on käyrän pisteen monikerran laskeminen, mikä on analoginen toimenpide potenssiin korotuksen kanssa kertolaskunotaatiolla varustetussa ryhmässä. Työssä esitetään muutama nopea yleisen ryhmän potenssiinkorotusalgoritmi sekä tekniikoita, jotka hyödyntävät elliptisen käyrän pisteiden esitystä projektiivisen tason homogeenisissa koordinaateissa. Yhdistämällä nämä tekniikat äärellisen kunnan algoritmeihin päädytään tehokkaisiin elliptisen käyrän pisteen monikerran laskentamenetelmiin. Työssä vertaillaan näiden menetelmien kompleksisuutta ja tarkastellaan eräitä tehtyjä ohjelmisto- ja mikropiiritoteutuksia.

Avainsanat: elliptiset käyrät, elliptisen käyrän kryptosysteemit, äärelliset kunnat, diskreetin logaritmin ongelma, toteutukset

Ei lainata ennen: **Työn sijaintipaikka:**

Author: Lasse Leskelä

Title of thesis: Implementing Arithmetic for Elliptic Curve Cryptosystems

Finnish title: Aritmetiikan toteuttaminen elliptisen käyrän kryptosysteemeihin

Date: 11th January 1999 **Pages:** 87

Department: Department of Engineering Physics and Mathematics

Chair: Mat-1 Mathematics

Supervisor: Professor Olavi Nevanlinna

Instructor: PhD Valtteri Niemi

This thesis describes how elliptic curve cryptosystems can be efficiently implemented. The performance of these systems will be studied by comparing various methods for efficient computation in finite fields and elliptic curve groups. In addition, the latest state-of-the-art algorithms for solving the elliptic curve discrete logarithm problem will be reviewed. This will provide mathematical evidence for the assumption that elliptic curve public key cryptosystems may provide strong security with relatively small key lengths.

The mathematical theory of elliptic curves is rich, having its roots in algebraic geometry and number theory. To keep the size of the presentation within reasonable bounds, some theorems whose proof would involve deeper knowledge of their respective areas are taken here for granted. These theorems are needed in exploring the group structure of an elliptic curve over a finite field, which is a key factor in determining the security of a cryptosystem built on this curve.

The elliptic curve group operation can be performed by computing some operations in the field on which the curve is built. Since the fields used in cryptography are finite, this means that fast algorithms for finite field arithmetic play a crucial role in designing efficient implementations of elliptic curve arithmetic. Because of their practical importance, finite fields will be discussed in great detail, with emphasis on fields of characteristic 2.

Finite fields with characteristic 2 are attractive from the implementation point of view, since these fields can be considered as vector spaces over the field $\mathbb{F}_2 = \{0, 1\}$. Thus, these fields can be naturally regarded as bit strings of fixed length. This in turn allows very fast implementations of field operations using logic circuits or general purpose microprocessors. Some efficient hardware architectures and software algorithms for performing arithmetical operations in large finite fields will be presented here.

The key operation in elliptic curve cryptosystems is the scalar multiplication of an elliptic curve point, which is analogous to exponentiation in multiplicative groups. Some algorithms for fast exponentiation in general groups will be described together with some more specific techniques that exploit the representation of elliptic curve points using homogeneous coordinates in the projective plane. These techniques are combined with algorithms for finite field arithmetic, resulting in efficient architectures for computing scalar multiples of elliptic curve points. The complexity of these architectures will be compared and a summary of some existing implementations in software and hardware will be presented.

Keywords: elliptic curves, elliptic curve cryptosystems, finite fields, discrete logarithm problem, implementations

Not borrowable till: **Library code:**

Preface

I would like to thank Olavi Nevanlinna for his supervision and encouragement with this work. Furthermore, I wish to express my gratitude to Valtteri Niemi for his expert guidance, and Asko Vilavaara for arranging an excellent working environment at the Nokia Research Center, Helsinki.

I am also grateful to numerous other colleagues in the Mobile Networks laboratory, especially Jari Juopperi for his valuable advice with many practical issues. Special thanks go out to Kaisa Nyberg and Ville Heikkala for engaging in many helpful discussions on various mathematical topics that came up during the work.

Helsinki, 11th January 1999

Lasse Leskelä

Contents

Chapter 1

Introduction

1.1 Motivation

An *elliptic curve* is a mathematical object built on a field. What makes these curves interesting is that they share a special algebraic property – a group structure. In 1985, Koblitz [Kob87b] and Miller [Mil86a] independently came up with the idea of considering elliptic curves over finite fields to construct public key cryptosystems suitable for digital signatures and key agreement protocols.

Cryptosystems based on the use of elliptic curve groups have two advantages over the more traditional public key schemes:

- The great diversity of elliptic curves over finite fields provides a large supply of naturally occurring finite groups. This enhances security by making it easy for the users of the cryptosystem to frequently change their encryption parameters.
- The absence of subexponential-time algorithms for inverting the encryption function implies potentially equivalent security as other public key systems, but with much shorter keys. Using shorter keys means smaller bandwidth and memory requirements, which in turn makes it possible to integrate strong encryption into platforms with very limited computational resources, such as smart cards.

The drawback with the use of elliptic curves is the fact that one group operation on a curve corresponds to several operations in the underlying finite field. Furthermore, since the fields used in cryptography are very large, it is crucial to find methods for efficient computation in finite fields and elliptic curve groups. This is the goal of the thesis.

The representation of this work has been tried to keep self-contained in the sense that the potential reader is assumed only minimal mathematical background. That is, only some basic facts about number theory, linear algebra and group theory are necessary. Appendix A will serve as a quick tutorial on the necessary facts about finite fields required for understanding the text. Hopefully this approach will make the mathematics of elliptic curves over finite fields more accessible to a wider audience.

1.2 Thesis Outline

Chapter 2 introduces the basic terms and definitions used in classical and public key cryptography. The aim is to provide a general description of cryptographic primitives based on the discrete logarithm problem in a finite cyclic group and to keep the presentation clear using as little mathematical details as possible.

Chapter 3 is an introduction to elliptic curves, with emphasis on curves over finite fields. A set of theorems from algebraic geometry is taken here for granted in order to construct a practical interface between the two worlds of mathematical abstractions and cryptographic applications. For the sake of brevity, only those facts about elliptic curves which are needed to understand the security and implementation aspects are presented here.

Security issues of elliptic curve cryptosystems are discussed in Chapter 4. This is accomplished by reviewing the best currently known algorithms for solving the discrete logarithm problem in a general finite cyclic group. Then follows a brief description of latest attempts to tackle this problem in elliptic curve groups. The chapter closes with a discussion on the size of keys needed to provide the desired level of security using elliptic curve public key cryptosystems.

Chapter 5 describes how finite field arithmetic can be efficiently performed. First, different alternatives for representing the field elements are given. Then various algorithms for multiplying and inverting elements in finite fields are discussed. Some hardware architectures are also included.

In Chapter 6, all issues concerning the implementation of arithmetical operations used in elliptic curve cryptosystems are summarized. The first part deals with general aspects of cryptosystems, which is followed by a discussion on practical methods to generate suitable elliptic curves, and how to compute their cardinality. Then some efficient methods for exponentiation in a finite cyclic group are presented. These methods are further combined with algorithms from Chapter 5 to implement the scalar multiplication of elliptic curve points. The end of the chapter summarizes some existing software and hardware implementations reported in the literature.

Final comparison of various solutions for implementing the whole is given in Chapter 7. Some recommendations for further research are also included there.

Chapter 2

Public Key Cryptography

2.1 Classical Cryptography

The classical purpose of cryptography is to let two people, Alice and Bob, communicate securely over an insecure transmission channel. One way to accomplish this is to first use a secure channel, e.g. a courier or a face-to-face contact, to agree upon a common secret piece of information called the *secret key* and then use this knowledge to transmit data in an encrypted form over a channel that can be listened. The message to be scrambled is called *plaintext* and the result of encrypting is called *ciphertext*. Mathematically, the idea of the ciphering procedure can be formulated as follows.

Definition 2.1. Let $\mathcal{P}$, $\mathcal{C}$ and $\mathcal{K}$ denote the finite sets of all possible plaintexts, ciphertexts and keys, respectively. A *cryptosystem* is a quintuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ with the property: For each $k \in \mathcal{K}$, there exists an *encryption rule* $E_k \in \mathcal{E}$, $E_k : \mathcal{P} \to \mathcal{C}$, and a corresponding *decryption rule* $D_k \in \mathcal{D}$, $D_k : \mathcal{C} \to \mathcal{P}$ that satisfy

$$D_k(E_k(P)) = P \quad \text{for every } P \in \mathcal{P}. \tag{2.1}$$

The claimed property (2.1) guarantees that the original plaintext can always be recovered once the decryption rule is known.

Alice and Bob can now use a cryptosystem to communicate securely as follows. First, Alice and Bob secretly agree upon a common key $k \in \mathcal{K}$. When Alice wants to send Bob a message $P \in \mathcal{P}$ she simply computes $C = E_k(P)$ and sends this to Bob. Now Bob can recover the original message by applying the decryption rule D_k to C. Figure 2.1 illustrates this.

To be of any use in practice, a cryptosystem should clearly have some additional properties. First, it should be computationally feasible to apply the algorithms for encryption and decryption. To provide security, it should be difficult for an eavesdropper who gains knowledge of C to obtain the plaintext P or the key k. The latter property should hold even in the case the eavesdropper knows everything about the cryptosystem, except the particular key being used. This requirement is known as the *Kerkhoff's principle*.

This classical scheme of choosing a common secret key between two users and using

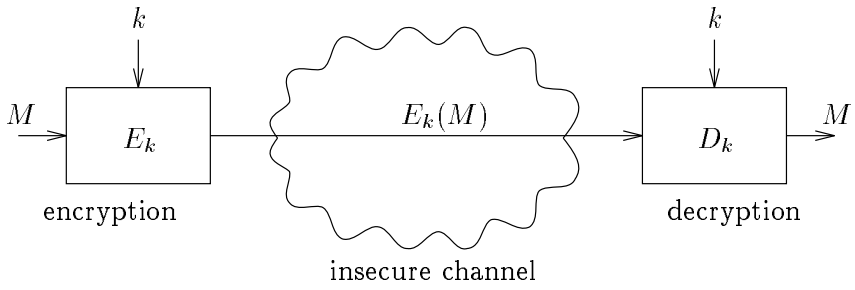

Figure 2.1: Symmetric ciphering procedure.

this to compute E_k and D_k is nowadays called *symmetric* or *private key cryptography*. Although it seems to fulfill the basic need for privacy, certain problems have arisen that limit its usefulness.

- *Key distribution.* Formerly, the main users of cryptography were military and diplomatic organizations. For them it was no problem to use couriers to exchange keys. Today, many people may want to communicate securely without even knowing each other, for example via the Internet. It is clearly infeasible to use couriers in this situation.
- *Key management.* In a network of n users, where each one wants to communicate securely with everyone, $n(n-1)/2$ keys are required. While the use of networks for commercial purposes is rapidly growing, this brings on serious problems with storing all the keys confidentially.

In addition, the use of digital signatures is not possible using private key cryptography.

2.2 The Diffie–Hellman Key Exchange

The first two to offer a solution to the key distribution problem of symmetric cryptography were Diffie and Hellman [DH76] in 1976. They proposed a protocol for agreeing upon a secret key without using any secure communications channels. This is how it works in its simplest form:

1. Alice and Bob publicly select a multiplicative cyclic group G and an element $g \in G$.
2. Alice produces a random integer a and sends g^a to Bob.
3. Bob generates a random integer b and sends g^b to Alice.
4. After receiving g^b, Alice computes $(g^b)^a$.
5. Similarly, Bob computes $(g^a)^b$.

Now, Alice and Bob share a common group element g^{ab} which will play the role of the secret key between them. Can an outsider figure out the key by listening the

channel, i.e., is it possible for someone to determine g^{ab} from the knowledge of g^a, g^b and g? This is referred as the *Diffie–Hellman problem.*

The Diffie–Hellman problem is closely related to an another problem, called the *discrete logarithm problem.* That is, find an integer x that satisfies $g^x = y$, where g and y are known elements of a finite (multiplicatively written) group. It is easy to see that if one possesses an algorithm to solve the discrete logarithm problem, he/she can also solve the Diffie–Hellman problem. A lot of effort has been devoted to proving the converse, which would imply that the both problems are equally difficult to solve. For cyclic groups of certain orders this has been done, but in general this issue still involves some open questions. For a discussion on the topic, see [MW98].

To securely exchange keys applying the Diffie–Hellman protocol it is essential to find finite groups where the discrete logarithm problem is difficult to solve. It is worth noting that since the group is finite, solving is always possible using exhaustive trial-and-error method. The point is to construct large groups where solving the discrete logarithm problem using the best available computing power takes very long time. At the same time, to perform the Diffie–Hellman protocol, exponentiation should be feasible to compute. This idea of relative computability led to the invention of public key cryptography.

2.3 Public Key Cryptosystems

A function $f : \mathcal{P} \to \mathcal{C}$ is called a *one-way*, if evaluating values of f can be done efficiently, but inverting f is "hard", i.e., no feasible algorithm for computing preimages $f^{-1}(\{C\}), C \in \mathcal{C}$ is known. A more accurate meaning to this intuitive definition will be developed in Chapter 4. To discuss public key cryptography, still two more terms should be introduced. A *trapdoor* for a function f is a piece of information that enables one to compute inverses of f efficiently. A *trapdoor one-way function* is a mapping $f : \mathcal{P} \to \mathcal{C}$ having a trapdoor (that is not easy find out) such that without knowing the trapdoor f looks like a one-way function.

Let $\mathcal{E}$ be a family of bijective trapdoor one-way functions,

$$\mathcal{E} = \{f_K : \mathcal{P} \to \mathcal{C} \mid K \in \mathcal{K}\}$$

and denote by $\mathcal{D}$ the set

$$\{f_K^{-1} : \mathcal{C} \to \mathcal{P} \mid K \in \mathcal{K}\},$$

respectively. Now it is clear that the quintuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ constitutes a cryptosystem. Also, for each $K \in \mathcal{K}$, the encrypting rule is the function $f_K : \mathcal{P} \to \mathcal{K}$. In addition, to be able to decrypt in practice, a trapdoor $t(K)$ corresponding to the key K used is required. The key K is called the *public key* and the corresponding trapdoor $t(K)$ is called the *private key.*

In a *public key cryptosystem*, every user U is assigned a public key K_U and a private key $t(K_U)$. The private key is kept secret while the public key is generally available for any other user. Bob can privately send a message $P \in \mathcal{P}$ to Alice by obtaining her public key K_A from a public directory and then encrypting his message using this

key. Since Alice is the only person who knows the secret trapdoor, no one except her can decrypt the ciphertext. This is the fundamental idea of public key cryptography.

Public key cryptosystems have many other applications. To mention one, digital signatures can be created by reversing the ciphering procedure described above. That is, if Alice wants to convince someone that she is the author of the message P, she may apply her decrypting rule to P using her private key (we assume here that $\mathcal{P} = \mathcal{C}$). Her signature will then be $S_A(P) = f_{K_A}^{-1}(P)$. Now anyone, after receiving the message and the signature, can verify the signature by checking if $f_{K_A}(S_A(P)) = P$, using Alice's public key. Again, Alice is the only person with the necessary knowledge of producing messages that map correctly under encrypting with f_{K_A}. Efficient digital signature protocols include some additional techniques, such as application of *hash functions*; a class of cryptographic functions used to compress the message being signed. Other applications include for example zero-knowledge proofs, secure elections and digital cash; see [Sch96, Sti95].

2.4 Cryptographic Primitives Based on the Discrete Logarithm Problem

2.4.1 ElGamal Cryptosystem

Let us assume we have a (multiplicative) group G of order n generated by g, where taking discrete logarithms with respect to g is difficult. In other words, we assume the mapping

$$\mathbb{Z} \ni k \mapsto g^k \in G$$

to be one-way. We will discuss the validity of this assumption in Chapter 4. In the following we will show how this particular one-way function can be used to construct cryptographic primitives.

The Diffie–Hellman key exchange introduced in Section 2.2 is an example of a protocol that exploits this one-way property. From practical point of view, it has the drawback that one can not freely choose the particular secret key to be agreed for symmetric ciphering.

One way to address this issue was invented by ElGamal [ElG85]. Assume $k_A, k_B \in \mathbb{Z}$ are Alice's and Bob's private keys and $V_a = g^{k_A}, V_B = g^{k_B}$ are their corresponding public keys. Now, if Alice wants to transmit Bob a secret message $M \in G$, she may encrypt her message by evaluating the function

$$f : G \ni M \mapsto (V_A, MV_B^{k_A}) \in G \times G$$

and sending $f(M)$ to Bob. Here k_B will be the trapdoor, since knowing k_B and $f(M)$, one can compute

$$V_a^{k_B} = g^{k_A k_B} = V_B^{k_A}.$$

Hence, denoting $f(M) = (f_1, f_2)$, M is recovered by computing

$$M = f_2(f_1^{k_B})^{-1}.$$

Here we have assumed that inverses in the group G can be efficiently computed. In later chapters we'll see that this is a feasible assumption.

What about the one-wayness of f? Suppose an eavesdropper gets to know $f(M) = (V_a, MV_B^{k_A})$. Can he figure out M? Since we assume group inversion to be easy, knowledge of M is equivalent to knowing $V_B^{k_A}$. But this is equivalent to solving the Diffie–Hellman problem discussed in Section 2.2, because $V_B^{k_A} = g^{k_A k_B}$.

The ElGamal cryptosystem presented above has one undesirable property. There is one person who can invert f without k_B, namely Alice. This doesn't sound like too serious a problem, since Alice originally knew M. However, it means that this system is not suitable for digital signatures. Another thing to note is that Alice's private/public key pair is used only to mask Bob's public key, an as such is in no way related to Alice's identity. Thus Alice's key pair could be as well replaced by a random one-time key pair. In fact, this should always be done, since otherwise there is an efficient attack against this cryptosystem [Sch96].

2.4.2 The Digital Signature Algorithm

Below we will describe a digital signature scheme that can be applied in an arbitrary cyclic group G of prime order. This scheme is based on the Digital Signature Standard (DSS) proposed by U.S. National Institute of Technology (NIST) [fST91]. As before, we assume G is generated by g. Assume further that the order of G is a prime r. Also, let ψ be a bijection from G to the set $\mathbb{Z}_r = \{0, 1, \ldots, r-1\}$ that can be easily computed, and denote the image $\psi(x)$ of $x \in G$ by $\overline{x}$.

Suppose m is a positive integer representing the message Alice wants to sign. This is how the signing goes:

1. Generate a one-time key pair (u, V), where $1 < u < r$ is a random integer and $V = g^u$.
2. Solve $0 \leq v \leq r-1$ from
$$m = -k_A \overline{V} + uv \pmod{r}, \tag{2.2}$$
where k_A is Alice's private key. If $v = 0$, go back to Step 1.
3. Output the pair (V, v) as Alice's signature for m.

To verify the signature with Alice's public key $V_A = g^{k_A}$ one performs the following steps.

1. Compute $h = v^{-1} \mod r$.
2. Compute $h_1 = mh \mod r$, and $h_2 = \overline{V}h \mod r$.
3. Compute $V' = g^{h_1} V_A^{h_2}$.
4. Verify that $V' = V$.

The verification works because from (2.2) we have

$$V' = g^{h_1}(g^{k_A})^{h_2} = g^{mh}g^{k_A\overline{V}h} = g^{(m+k_A\overline{V})h} = g^{(uv)h} = g^u = V.$$

If someone now wants to forge Alice's signature for m, he/she would need to find a pair (u, v) such that

$$g^u = g^{mw}V_A^{\overline{g^u}w}, \quad w = v^{-1} \mod r$$

holds. By first trying to fix u and solve for a suitable v (equivalently w) is a discrete logarithm problem. On the other hand, fixing first v and looking for the correct u becomes a mixed exponential congruence problem in u for which no efficient algorithm is known. So we may say that the security of this signature scheme is closely related to the discrete logarithm problem in G.

Chapter 3

Elliptic Curves

In this chapter we represent a brief introduction to elliptic curves and write down some fundamental facts about their algebraic structure. We will follow quite closely the representation given in Menezes' book [Men93], complementing it with geometric characterization of curves and filling in some details. Readers unfamiliar with field theory are recommended to consult Appendix A for a quick tutorial on the topic.

3.1 Background and Definitions

3.1.1 Geometric Approach

A set of points $(x, y) \in \mathbb{R}^2$ for which

$$y^2 = x^3 + ax + b, \quad a, b \in \mathbb{R}, \tag{3.1}$$

has classically been called an *elliptic curve*. This type of curves were historically involved in determining the arc length of an ellipse – that is why they are been called "elliptic".

Figure 3.1 shows a way to define an operation for two points P and Q on an elliptic curve. First, we draw a line through P and Q. We see that this line intersects the curve in exactly one more point, labeled R. We define the result of the operation to be the mirror image R' of R with respect to x-axis. This result will be from now on called the *sum of P and Q* and denoted $P + Q = R'$.

It is clearly seen that the operation outlined above is commutative, that is,

$$P + Q = Q + P.$$

By carefully drawing new points and lines to Figure 3.1 it is also possible to guess or at least hope that the operation be associative:

$$(P + Q) + S = P + (Q + S).$$

So, it looks like we have found a new interesting algebraic structure within this curve. However, some problems arise. For example, what would be the sum of R and R' in

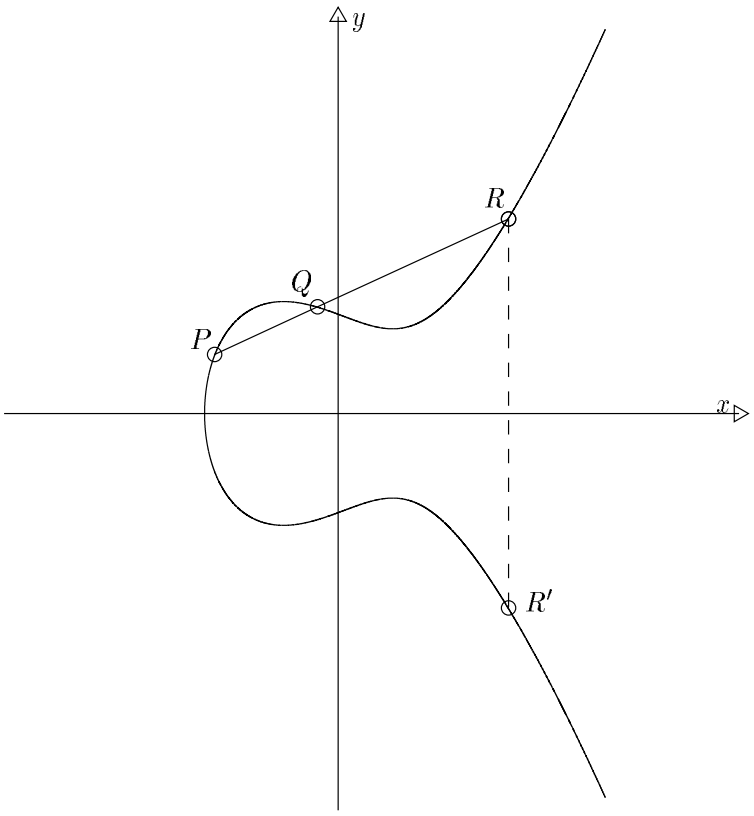

Figure 3.1: Adding two points on the elliptic curve $y^2 = x^3 - 2x + 4$.

Figure 3.1? It is clear from the figure that no point of $\mathbb{R}^2$ other than R and R' will satisfy equation (3.1). We seem to have run out of points.

Let us not give up yet. If $\mathbb{R}^2$ is too small a space, why not try to fix this by embedding this plane and the curve into some bigger structure? Geometrically, we would like to try adding some extra material to the furthest edges of the plane – a horizon. To do this, we need some machinery from algebraic geometry.

3.1.2 The Projective Plane

Since the algebraic concepts to be introduced depend only on the field structure of the coordinate axes of the plane, we will fix $\mathbb{K}$ to denote any field in what follows. Define two nonzero points in the affine space $\mathbb{K}^3$ to be equivalent if they are scalar multiples of each other. This is clearly an equivalence relation. The equivalence class corresponding to the point $(X, Y, Z) \in \mathbb{K}^3 \setminus \{(0, 0, 0)\}$ will be the set

$$(X{:}Y{:}Z) = \{(\lambda X, \lambda Y, \lambda Z) \in \mathbb{K}^3 \mid \lambda \in \mathbb{K}\},$$

that is, a line in $\mathbb{K}^3$ passing through the origin.

Definition 3.1. The *projective plane* $\mathbb{P}^2(\mathbb{K})$ *over* $\mathbb{K}$ is the set of all lines in $\mathbb{K}^3$ that intersect the origin:

$$\mathbb{P}^2(\mathbb{K}) = \{(X{:}Y{:}Z) \mid (X, Y, Z) \in \mathbb{K}^3 \setminus \{(0, 0, 0)\}\}.$$

A point $(X:Y:Z)$ in the projective plane can be uniquely represented by any nonzero point of the line $(X:Y:Z) \subseteq \mathbb{K}^3$. The set of nonzero points of the line is consequently called the set of *projective (homogeneous) coordinates* for the point $(X:Y:Z)$.

If $Z \neq 0$, then $(X:Y:Z) = (X/Z : Y/Z : 1)$. Thus, the projective plane has a decomposition into two parts

$$\mathbb{P}^2(\mathbb{K}) = \{(X:Y:1) \mid X,Y \in \mathbb{K}\} \cup \{(X:Y:0) \mid (X,Y,0) \in \mathbb{K}^3 \setminus \{(0,0,0)\}\}. \tag{3.2}$$

The former part can be interpreted as an affine plane $\mathbb{K}^2$ while the latter could be called the "line at the infinity". So the projective plane looks like an ordinary plane with its edges at the infinity glued to the horizon, see Figure 3.2.

3.1.3 The Weierstrass Equation

From now on, $\overline{\mathbb{K}}$ shall be the *algebraic closure* (see Definition A.28 in Appendix A) of $\mathbb{K}$. A *Weierstrass equation* is a homogeneous equation of degree 3 of the form

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3, \tag{3.3}$$

where $a_1, \ldots, a_6$ are elements of $\overline{\mathbb{K}}$. With this equation we associate a polynomial $F \in \overline{\mathbb{K}}[X,Y,Z]$:

$$F(X,Y,Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3.$$

Since every monomial of F has degree 3,

$$F(\lambda X, \lambda Y, \lambda Z) = \lambda^3 F(X,Y,Z),$$

and thus for nonzero $\lambda \in \overline{\mathbb{K}}$, $F(\lambda X, \lambda Y, \lambda Z) = 0$ if and only if $F(X,Y,Z) = 0$. By this observation, we can define a point $P = (X:Y:Z) \in \mathbb{P}^2(\overline{\mathbb{K}})$ to be a *zero of* F if $F(X,Y,Z) = 0$. In that being the case, we write $F(P) = 0$.

The Weierstrass equation is said to be *smooth* if for all points $P = (X:Y:Z) \in \mathbb{P}^2(\overline{\mathbb{K}})$ that satisfy $F(P) = 0$ the gradient of F at P is nonzero, i.e., at least one of the partial derivatives (all of which monomials are of degree 2)

$$\frac{\partial F}{\partial X}(P), \frac{\partial F}{\partial Y}(P), \frac{\partial F}{\partial Z}(P)$$

is nonzero. Note that in a field with no metric structure we can formally define the derivative of polynomial by the familiar rule

$$\frac{\partial X^j}{\partial X} = jX^{j-1}, \quad j = 1, 2, \ldots$$

Now we are ready to give a rigorous definition of an elliptic curve.

Definition 3.2. An *elliptic curve* E is the set of all solutions of a smooth Weierstrass equation. If the coefficients $a_1, \ldots, a_6$ belong to $\mathbb{K}$, then E is said to be *defined over* K, and we denote this by $E/\mathbb{K}$.

By examining the Weierstrass equation, we see that there is exactly one point in E with $Z = 0$, the point $(0 : 1 : 0)$. By comparing with the decomposition (3.2), this point is seen to be the only point of E outside the affine plane $\overline{\mathbb{K}}^2$. This point is called the *point at infinity* and denoted by $\mathcal{O}$.

For the points of E in the affine plane $Z \neq 0$, using affine (non-homogeneous) coordinates $x = X/Z$ and $y = Y/Z$ we may rewrite the Weierstrass equation as

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{3.4}$$

The elliptic curve is thus the set of solutions of (3.4) in $\overline{\mathbb{K}}^2$, together with the point $\mathcal{O}$. Figure 3.2 illustrates this.

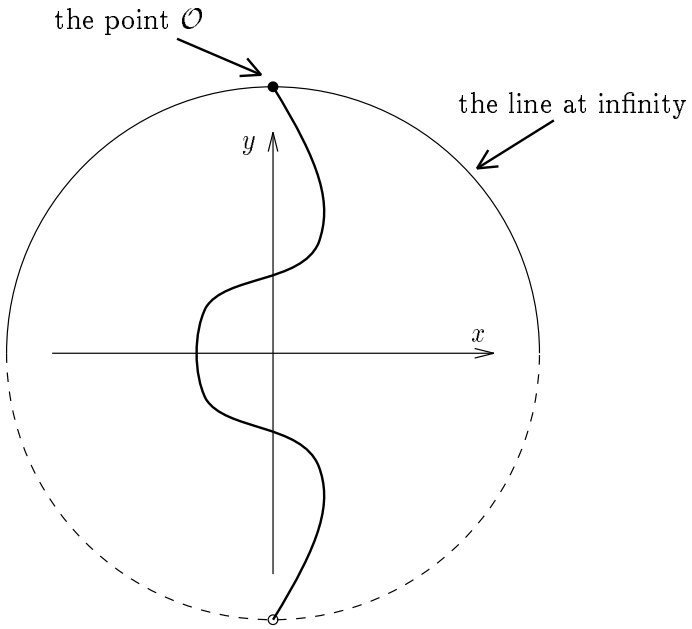

Figure 3.2: An elliptic curve sketched in the projective plane.

Definition 3.3. Let $E/\mathbb{K}$ be an elliptic curve defined over $\mathbb{K}$. The set of $\mathbb{K}$-*rational points of* E is the set of points on E whose coordinates lie in $\mathbb{K}$, together with the point $\mathcal{O}$. The set of $\mathbb{K}$-rational points of E is denoted by $E(\mathbb{K})$:

$$E(\mathbb{K}) = (E/\mathbb{K} \cap \mathbb{K}^2) \cup \{\mathcal{O}\}.$$

Two elliptic curves $E_1/\mathbb{K}$ and $E_2/\mathbb{K}$ are defined to be *isomorphic*, if they are isomorphic as *projective varieties*. However, the clear explanation of these concepts would need a considerable amount of background from algebraic geometry, which is beyond the scope of this thesis. Fortunately, we have an equivalent characterization in more tractable terms [Sil86, Chapter III]:

Theorem 3.4. *Two elliptic curves $E_1/\mathbb{K}$ and $E_2/\mathbb{K}$, given by the equations*

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{3.5}$$

$$y^2 + b_1xy + b_3y = x^3 + b_2x^2 + b_4x + b_6 \tag{3.6}$$

are isomorphic over $\mathbb{K}$ *if and only if there exists* $c_1, c_2, c_3, c_4 \in \mathbb{K}$, $c_1 \neq 0$, *such that the change of variables*

$$(x, y) \mapsto (c_1^2 x + c_2, c_1^3 y + c_1^2 c_3 x + c_4)$$

transforms equation (3.5) to (3.6). The relationship of isomorphism is an equivalence relation.

3.2 Basic Properties

3.2.1 The Group Law

Now we have all the machinery we need to complete the discussion about the algebraic operation introduced in Section 3.1. Observe that the familiar definitions from analytic geometry for *line* and *tangent* in $\mathbb{R}^2$ can be generalized without any modifications to the (non-geometric) plane $\mathbb{K}^2$.

Definition 3.5. Let E be an elliptic curve induced by the affine Weierstrass equation (3.4). For all $P, Q \in E \setminus \{\mathcal{O}\}$; $P = (x_1, y_1), Q = (x_2, y_2)$ define:

1. $\mathcal{O} + P = P$ and $P + \mathcal{O} = P$.
2. $-\mathcal{O} = \mathcal{O}$.
3. $-P = (x_1, -y_1 - a_1 x_1 - a_3)$
4. If $Q = -P$, then $P + Q = \mathcal{O}$.
5. If $Q = P$, then let R be the point of intersection of the curve E with its tangent at P and define $P + Q = -R$.
6. If $Q \neq P$ and $Q \neq -P$, then let R be the point where the line $\overline{PQ}$ through P and Q intersects the curve (or, if the line $\overline{PQ}$ is tangent to the curve at P, let $R = P$, and respectively, if $\overline{PQ}$ is tangent to the curve at Q, let $R = Q$). Define the sum $P + Q = -R$.

The point at infinity will consequently act as an identity under the operation defined. For a nonnegative integer n, we define

$$nP = \underbrace{P + \cdots + P}_{n},$$

and similarly, $nP = |n|\,(-P)$ for $n < 0$. The next theorem states the most important fact about elliptic curves.

Theorem 3.6. *Let* E *be an elliptic curve. Then the operation given above is well-defined. With respect to this operation,* E *is an abelian group with identity element* $\mathcal{O}$.

The proof of the above theorem has one difficult part, the associativity law. An algebraic argument doing the task using divisor theory can be found for example in [Sil86, Section III.3].

To see how the group operation works, we will follow the presentation in [Sil86, Section III.3] and write down explicit formulae for computing sums of points in an arbitrary field. For convenience, denote

$$f(x,y) = x^3 + a_2x^2 + a_4x + a_6 - y^2 - a_1xy - a_3y,$$

whence, the affine Weierstrass equation is equivalent to

$$f(x,y) = 0. \tag{3.7}$$

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be points in $E \setminus \{\mathcal{O}\}$, and denote their sum by $P+Q = (x_3, y_3)$. If $x_1 = x_2$ and $y_1 = -y_2 - a_1x_2 - a_3$, then $P+Q = \mathcal{O}$. Otherwise the line $\overline{PQ}$ through P and Q (tangent to the curve E if $P = Q$) has the equation

$$y = \lambda x + \beta.$$

By substituting this into equation (3.7), we get a polynomial equation of degree 3 in variable x. Since we are working in an algebraically closed field, the monic polynomial $f(x, \lambda x + \beta)$ splits as

$$f(x, \lambda x + \beta) = (x - x_1)(x - x_2)(x - x_3), \tag{3.8}$$

where x_3 is the coordinate we are looking for. By extracting both sides of (3.8) and comparing coefficients for x^2, we get

$$x_1 + x_2 + x_3 = \lambda^2 + a_1\lambda - a_2.$$

This gives us x_3, and substituting it to the equation of the line we get

$$y_3 = \lambda x + \beta.$$

After a short calculation, the values for λ and β can be seen to be

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}, \quad \beta = \frac{y_1x_2 - y_2x_1}{x_2 - x_1},$$

if $x_1 \neq x_2$, and

$$\lambda = \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}, \quad \beta = \frac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3},$$

if $x_1 = x_2$.

The above formulae imply that the geometrically defined elliptic curve operation can be computed performing just a few arithmetic operations in the underlying field. Especially, the next fact is now clear.

Theorem 3.7. *If E is defined over $\mathbb{K}$, then $E(\mathbb{K})$ is a subgroup of $E/\mathbb{K}$.*

3.2.2 The Discriminant and the j-Invariant

There are two important quantities that characterize elliptic curves. To define them, we assume that our elliptic curve E is given by the Weierstrass equation (3.4) and define

$$\begin{aligned} d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4 + a_1a_3 \\ d_6 &= a_3^3 + 4a_6 \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \\ \Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \qquad (3.9) \\ j(E) &= (d_2^2 - 24d_4)^3/\Delta. \qquad (3.10) \end{aligned}$$

Definition 3.8. The quantity Δ defined by (3.9) is called the *discriminant* of the Weierstrass equation and $j(E)$ in equation (3.10) is called the *j-invariant* of the elliptic curve.

The next two theorems [Sil86, Section III.1] show the significance of these quantities and that the j-invariant is well defined for all elliptic curves.

Theorem 3.9. *The Weierstrass equation (3.3) is smooth, if and only if $\Delta \neq 0$.*

Theorem 3.10. *If two elliptic curves $E_1/\mathbb{K}$ and $E_2/\mathbb{K}$ are isomorphic over $\mathbb{K}$, then $j(E_1) = j(E_2)$. If $\mathbb{K}$ is an algebraically closed field, the converse also holds.*

3.2.3 Curves over $\mathbb{K}$, $\mathrm{char}(\mathbb{K}) \neq 2, 3$

If an elliptic curve is defined over a field $\mathbb{K}$ with a character $\mathrm{char}(\mathbb{K})$ that is neither 2 nor 3, we will see that the Weierstrass equation (3.4) gets a much neater form. First, if $\mathrm{char}(\mathbb{K}) \neq 2$, division by 2 is possible, and we can apply a change of variables in accordance with Theorem 3.4:

$$(x, y) \mapsto \left(x, y - \frac{a_1}{2}x - \frac{a_3}{2}\right).$$

The resulting curve $E'/\mathbb{K}$ will have the defining equation

$$y^2 = x^3 + b_2x^2 + b_4x + b_6.$$

If in addition, $\mathrm{char}(\mathbb{K}) \neq 3$, making a consecutive change of variables

$$(x, y) \mapsto \left(\frac{x - 3b_2}{36}, \frac{y}{216}\right)$$

will yield another elliptic curve $E''/\mathbb{K}$ over $\mathbb{K}$, with a Weierstrass equation

$$y^2 = x^3 + ax + b, \quad a, b \in \mathbb{K}. \qquad (3.11)$$

This is exactly the equation (3.1) with $\mathbb{K}$ in place of $\mathbb{R}$ in Section 3.1. So we see that in a field with characteristic $\mathrm{char}(\mathbb{K}) \neq 2, 3$, every elliptic curve $E/\mathbb{K}$ over $\mathbb{K}$ is isomorphic to a one represented by (3.11).

Elliptic curves over fields with $\mathrm{char}(\mathbb{K}) \neq 2, 3$ of interest include those over $\mathbb{C}$, $\mathbb{R}$, $\mathbb{Q}$ and the finite fields $\mathbb{F}_p$, where p is a prime greater than 3. Especially, properties of elliptic curves over $\mathbb{Q}$ are deeply related to many big questions in number theory. For example, in 1995 Andrew Wiles [Wil95, WT95] succeeded to prove the famous Fermat's Last Theorem by proving a conjecture about elliptic curves over the rationals. From the implementation point of view though, the most interesting curves seem to be the elliptic curves over finite fields of characteristic 2. Consequently, the rest of this thesis will strongly emphasize on them.

3.3 Elliptic Curves over Finite Fields

Throughout this section, $\mathbb{F}_q$ will stand for the finite field with $q = p^l$ elements, where p is a prime and l is a positive integer.

3.3.1 Number of Points

Assume that an elliptic curve $E/\mathbb{F}_q$ over $\mathbb{F}_q$ is given by equation (3.4). We wish to determine the number of $\mathbb{F}_q$-rational points of $E/\mathbb{F}_q$, which is denoted by $\#E(\mathbb{F}_q)$. For fixed $x \in \mathbb{F}_q$, the elements $y \in \mathbb{F}_q$ that satisfy (3.4) are zeros of the polynomial

$$f(y) = y^2 + (a_1x + a_3)y - (x^3 + a_2x^2 + a_4x + a_6).$$

Since this polynomial has degree 2, we see that for each x, there are at most two possible y in $\mathbb{F}_q$ such that $(x, y) \in E(\mathbb{F}_q)$. So we get an upper bound

$$\#E(\mathbb{F}_q) \leq 2q + 1.$$

Heuristically, we could assume that the parameters of the above quadratic polynomial are uniformly distributed over $\mathbb{F}_q$ as x varies over $\mathbb{F}_q$. Since it can be shown that a quadratic equation with random parameters is solvable in $\mathbb{F}_q$ with probability $1/2$, it seems reasonable to assume that the total number of points of $E(\mathbb{F}_q)$ is not far from $q + 1$. Consequently, we now fix a symbol t such that

$$\#E(\mathbb{F}_q) = q + 1 - t. \tag{3.12}$$

The next fact [Sil86, Section V.1] known as *Hasse's Theorem* validates our reasoning.

Theorem 3.11. *Let $E/\mathbb{F}_q$ be an elliptic curve over the finite field $\mathbb{F}_q$. Then $|t| \leq 2\sqrt{q}$.*

Using the symbol t, we can now characterize an important class of elliptic curves over finite fields in a convenient way [Men93]. For more general definitions, see [Sil86, Section V.3] or [Har77, Section IV.4].

Definition 3.12. The elliptic curve $E/\mathbb{F}_q$ is said to be *supersingular*, if t is divisible by p. Otherwise, it is called *non-supersingular*.

If an elliptic curve is defined over $\mathbb{F}_q$, it makes also perfect sense to look for solutions to the corresponding Weierstrass equation in an extension field $\mathbb{F}_{q^k}$. Then $E(\mathbb{F}_q)$ becomes a subgroup of $E(\mathbb{F}_{q^k})$. As a corollary of Hasse's Theorem we get the following estimate (see Corollary A.39 in Appendix A).

Corollary 3.13. *Let $E/\mathbb{F}_q$ be an elliptic curve defined over $\mathbb{F}_q$ and $E/\mathbb{F}_{q^k}$, $k = 1, 2, \ldots$, be extensions of $E/\mathbb{F}_q$ obtained by the above procedure. Then*

$$\#E(\overline{\mathbb{F}}_q) = \#E(\bigcup_{k=1}^{\infty} \mathbb{F}_{q^k}) = \lim_{k\to\infty} \#E(\mathbb{F}_{q^k}) = \infty.$$

Proof. By Theorem 3.11, we have a lower bound

$$\#E(\mathbb{F}_{q^k}) \geq q^k + 1 - 2\sqrt{q^k} = (q^{k/2} - 1)^2 \to \infty,$$

as $k \to \infty$. □

The *Weil Theorems*, see [Sil86, Section V.2], when applied to elliptic curves can be used to determine the exact number of points in $E(\mathbb{F}_{q^k})$ when E has its coefficients in $\mathbb{F}_q$ [Kob87a].

Theorem 3.14. *Let $E/\mathbb{F}_q$ be an elliptic curve defined over $\mathbb{F}_q$ and let α and β be complex numbers determined from the factorization*

$$1 - tT + qT^2 = (1 - \alpha T)(1 - \beta T),$$

where t is as in (3.12). Then, for $k = 1, 2, \ldots$, the number of $\mathbb{F}_{q^k}$-rational points in the extension field $E/\mathbb{F}_{q^k}$ is

$$\#E(\mathbb{F}_{q^k}) = q^k + 1 - \alpha^k - \beta^k$$

3.3.2 Group Structure

To discuss the group structure of elliptic curves, we need some terminology concerning abelian groups.

Definition 3.15. Let $G_1, \ldots G_r$ be subgroups of an abelian group G. Their *sum* is defined to be the set

$$G_1 + \cdots + G_r = \{g_1 + \cdots + g_r \mid g_i \in G_i, \quad i = 1, \ldots, r\}.$$

The sum of $G_1, \ldots, G_r$ is called *direct*, if its each element is uniquely represented in the form $g_1 + \cdots + g_r$ with $g_i \in G_i$; that is,

$$g_1 + \cdots + g_r = g_1' + \cdots + g_r'$$

if and only if $g_i = g_i'$ for all $i = 1, \ldots, r$. We denote direct sums by

$$\bigoplus_{i=1}^{r} G_i = G_1 \oplus \cdots \oplus G_r.$$

It is clear that the direct sum $G_1 \oplus \cdots \oplus G_r$ is isomorphic to the cartesian product group $G_1 \times \cdots \times G_r$ with component-wise addition.

We denote by $\mathbb{Z}_n$ the additive group of integers modulo n. Note that every finite cyclic group of order n is isomorphic to $\mathbb{Z}_n$. The following is a fundamental result from the theory of abelian groups [Nic93, Chapter 7].

Theorem 3.16. *Every finite abelian group G is a direct sum of cyclic groups. There exists integers $s, n_1, \ldots, n_s$ uniquely determined by G, such that n_{i+1} divides n_i for each i and*

$$G \simeq \mathbb{Z}_{n_1} \oplus \cdots \oplus \mathbb{Z}_{n_s}. \tag{3.13}$$

The integer s in equation (3.13) is called the *rank* of G and the corresponding s-tuple $(n_1, \ldots, n_s)$ is called the *type* of G. If H is an abelian group with type $(m_1, \ldots, m_r)$ such that $r \leq s$ and m_i divides n_i for each i, we say that H has *smaller type* than G.

Definition 3.17. Let E be an elliptic curve defined over $\overline{\mathbb{F}}_q$. If a point $P \in E(\overline{\mathbb{F}}_q)$ satisfies $nP = \mathcal{O}$, it is called an *n-torsion point*. The set of n-torsion points on the curve $E(\overline{\mathbb{F}}_q)$ is denoted by $E[n]$.

Note that the above definition allows $E[n]$ to include points that are not $\mathbb{F}_q$-rational even if E is defined over $\mathbb{F}_q$.

The core information we need about the group structure of elliptic curves is collected in the next theorem [Sil86, Men93].

Theorem 3.18. *Let E be an elliptic curve defined over $\mathbb{F}_q$.*

- *$E(\mathbb{F}_q)$ is an abelian group of rank 1 or 2. That is, $E(\mathbb{F}_q) \simeq \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$, where n_2 divides n_1, and furthermore n_2 divides $q - 1$.*
- *If $\gcd(n, q) = 1$, then $E[n] \simeq \mathbb{Z}_n \oplus \mathbb{Z}_n$. If $n = p^e$, then*

$$E[n] \simeq \begin{cases} \{\mathcal{O}\} & \text{if } E \text{ supersingular,} \\ \mathbb{Z}_{p^e} & \text{if } E \text{ non-supersingular.} \end{cases}$$

Note that according to the notation above, $E(\mathbb{F}_q)$ is cyclic if and only if $n_2 = 1$.

We will not go into proof of Theorem 3.18, since it involves rather heavy machinery from algebraic geometry. Instead we'll use the above results to derive two necessary conditions for rationality of n-torsion points. This check will be useful in Chapter 4.

Theorem 3.19. *Let $E/\mathbb{F}_q$ be an elliptic curve and $\gcd(n, q) = 1$. Assume $E[n] \subseteq E(\mathbb{F}_q)$. Then*

1. *n^2 divides $\#E(\mathbb{F}_q)$ and*
2. *n divides $q - 1$.*

Proof. Consequence 1 is clear after observing that $E[n]$ is a subgroup of E. For the second part, we use Theorem 3.18 to write $E(\mathbb{F}_q) \simeq \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$, where n_2 divides n_1. Let $\prod_i p_i^{c_i}$ be the prime factorization of n_2 and $\prod_i p_i^{d_i} \prod_j q_i^{e_i}$ be the corresponding factorization for n_1 with $d_i \geq c_i$. Then we can write

$$\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2} = \bigoplus_i \mathbb{Z}_{p_i^{d_i}} \oplus \bigoplus_j \mathbb{Z}_{q_j^{d_j}} \oplus \bigoplus_i \mathbb{Z}_{p_i^{c_i}} = \bigoplus_i G(p_i) \oplus \bigoplus_j G(q_i),$$

where

$$G(p_i) = \mathbb{Z}_{p_i^{c_i}} \oplus \mathbb{Z}_{p_i^{d_i}}, \quad G(q_i) = \mathbb{Z}_{q_i^{e_i}}.$$

Similarly, if n factors into primes as $\prod_k r_k^{f_k}$, we may write

$$\mathbb{Z}_n \oplus \mathbb{Z}_n = \bigoplus_k G(r_k), \quad \text{with } G(r_k) = \mathbb{Z}_{r_k^{f_k}} \oplus \mathbb{Z}_{r_k^{f_k}}.$$

From the theory of abelian groups [Nic93, Chapter 7] we know that if $\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$ has a subgroup of the form $\mathbb{Z}_n \oplus \mathbb{Z}_n$, then its every r_k-*primary* subgroup $G(r_k)$ has to be a subgroup of either $G(p_i)$ or $G(q_j)$ for some i or j. However, the (noncyclic) group $G(r_k)$ can't be a subgroup of any (cyclic) group $G(q_i)$, since subgroups of a cyclic group are cyclic [Nic93]. Thus, for every k, $G(r_k)$ is a subgroup of $G(p_i)$ for some i. While it is also true [Nic93, Chapter 7] that subgroups of $G(p_i)$ have smaller type than $G(p_i)$, we conclude that $r_k^{f_k} = p_i^{d_i'}$ for some i with $d_i' \leq c_i$. Since this holds for every factor r_k of n, we see that n divides n_2. Using once more Theorem 3.18, n_2 divides $q-1$ and hence, so does also n. □

3.4 Binary Elliptic Curves

3.4.1 Adding Points on Non-Supersingular Binary Curves

Now let us turn our attention to elliptic curves over a field of characteristic 2, we'll call these *binary elliptic curves*. Assume $E/\mathbb{F}_{2^m}$ is given by equation (3.4). Since doubling annihilates elements in $\mathbb{F}_{2^m}$, formula (3.10) for the j-invariant gets very nice form:

$$j(E) = a_1^{12}/\Delta.$$

Combining this with the following fact [Sil86, Chapter V], we get a simple criterion for supersingularity.

Theorem 3.20. *An elliptic curve over a finite field $\mathbb{F}_q$ of characteristic 2 is supersingular if and only if $j(E) = 0$*

In the non-supersingular case, i.e., $a_1 \neq 0$, we can make a change of variables according to Theorem 3.4:

$$(x, y) \mapsto \left(a_1^2 x + \frac{a_3}{a_1}, a_1^3 y + \frac{a_1^2 a_4 + a_3^3}{a_1^3} \right).$$

This will result in the simplified Weierstrass equation

$$y^2 + xy = x^3 + ax^2 + b, \quad a, b \in \mathbb{F}_{2^m}, \quad b \neq 0, \tag{3.14}$$

with $\Delta = b$ and $j(E) = 1/b$.

Specializing the addition formulae derived in Section 3.2 we get simplified addition formulae for binary non-supersingular elliptic curves.

Theorem 3.21. *Let $E/\mathbb{F}_{2^m}$ be an elliptic curve induced by equation (3.14). Let $P = (x_1, y_1) \in E/\mathbb{F}_{2^m}$. Then $-P = (x_1, y_1 + x_1)$. If $Q = (x_2, y_2) \in E/\mathbb{F}_{2^m}$ and $Q \neq -P$, then $P + Q = (x_3, y_3)$, where*

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a \tag{3.15}$$

and

$$y_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1. \tag{3.16}$$

If $P = Q$, then $P + Q = 2P = (x_3, y_3)$, where

$$x_3 = x_1^2 + \frac{b}{x_1^2} \tag{3.17}$$

and

$$y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3. \tag{3.18}$$

3.4.2 Addition Formulae for Supersingular Curves

If $j(E) = 0$, so is also a_1, and the admissible change of variables

$$(x, y) \mapsto (x + a_2, y)$$

modifies the corresponding Weierstrass equation to

$$y^2 + ay = x^3 + bx + c, \quad a, b, c \in \mathbb{F}_{2^m}, \quad a \neq 0. \tag{3.19}$$

For this curve, $\Delta = a^4$.

In a similar manner as with non-supersingular curves, we write down explicit formulae for adding points for these curves.

Theorem 3.22. *Let $E/\mathbb{F}_{2^m}$ be an elliptic curve given by equation (3.19). Let $P = (x_1, y_1) \in E/\mathbb{F}_{2^m}$. Then $-P = (x_1, y_1 + a)$. If $Q = (x_2, y_2) \in E/\mathbb{F}_{2^m}$ and $Q \neq -P$, then $P + Q = (x_3, y_3)$, where*

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + x_1 + x_2$$

and

$$y_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + y_1 + a.$$

If $P = Q$, then $P + Q = 2P = (x_3, y_3)$, where

$$x_3 = \frac{x_1^4 + b^2}{a^2}$$

and

$$y_3 = \left(\frac{x_1^4 + b}{a}\right)(x_1 + x_3) + y_1 + a.$$

3.4.3 Isomorphism Classes of Binary Curves

To complete our categorization of binary elliptic curves, we will count the number of their isomorphism classes and write down a representative for each class. Towards this end, next lemma will be useful (for the definition of trace, see Section A.4.3 in Appendix A).

Lemma 3.23. *Equation*

$$x^2 + cx + d = 0, \quad c, d \in \mathbb{F}_{2^m}, \quad c \neq 0 \tag{3.20}$$

has a solution in $\mathbb{F}_{2^m}$ if and only if the trace of $c^{-2}d$, $\mathrm{Tr}(c^{-2}d) = 0$. If $\alpha \in \mathbb{F}_{2^m}$ is one solution to (3.20), then the other solution is $\alpha + c$.

Proof. Assume $\alpha \in \mathbb{F}_{2^m}$ satisfies (3.20). Then, since squaring over fields of characteristic 2 is linear, it immediately seen that $\alpha + c$ is also a solution. Multiplying (3.20) by c^{-2} and applying the trace function, we get

$$0 = \mathrm{Tr}(0) = \mathrm{Tr}((c^{-1}\alpha)^2 + c^{-1}\alpha + c^{-2}d) = \mathrm{Tr}(c^{-2}d).$$

Assume next that $\mathrm{Tr}(c^{-2}d) = 0$ and let β be a root of the polynomial $x^2 + cx + d$ in $\overline{\mathbb{F}}_{2^m}$. Then we have

$$\begin{aligned} 0 = \mathrm{Tr}(c^{-2}d) &= \mathrm{Tr}(c^{-1}\beta + (c^{-1}\beta)^2) \\ &= \sum_{i=0}^{m-1} (c^{-1}\beta)^{2^i} + \sum_{i=0}^{m-1} [(c^{-1}\beta)^2]^{2^i} = c^{-1}\beta + (c^{-1}\beta)^{2^m}. \end{aligned}$$

Hence $(c^{-1}\beta)^{2^m} = c^{-1}\beta$, which implies by Corollary A.40 that $c^{-1}\beta \in \mathbb{F}_{2^m}$ and thus also $\beta \in \mathbb{F}_{2^m}$. □

Theorem 3.24. *Assume two non-supersingular elliptic curves E_1 and E_2, given by equations*

$$y^2 + xy = x^3 + a_1x^2 + b_1, \quad b_1 \neq 0, \tag{3.21}$$

$$y^2 + xy = x^3 + a_2x^2 + b_2, \quad b_2 \neq 0, \tag{3.22}$$

respectively, are isomorphic. Then

- *$a_2 = a_1 + s + s^2$ for some $s \in \mathbb{F}_{2^m}$,*
- *$b_2 = b_1$.*

The isomorphism maps rational points of E_1 according to $(x, y) \mapsto (x, y + sx)$.

Proof. Recall that by Theorem 3.4, we have constants $c_1, c_2, c_3, c_4 \in \mathbb{F}_{2^m}$, $c_1 \neq 0$, such that the change of variables

$$(x, y) \mapsto (c_1^2 + c_2, c_1^3 y + c_1^2 c_3 x + c_4)$$

transforms equation (3.21) to equation (3.22). Substituting these new variables into (3.21) yields

$$c_1^6 y^2 + c_1^5 xy + c_1^3 c_2 y = c_1^6 x^3 + c_1^4(c_3^2 + c_3 + c_2 + a_1)x^2 + c_1^2(c_2 c_3 + c_4 + c_2^2)x + c_4^2 + c_2 c_4 + c_2^3 + a_1 c_2^2 + b_1. \quad (3.23)$$

Comparing coefficients of equations (3.21) and (3.23), we get relations

$$\begin{aligned} c_1 &= 1, \\ c_2 = c_4 &= 0, \\ a_2 &= a_1 + c_3 + c_3^2, \\ b_2 &= b_1. \end{aligned}$$

This proves our contention. □

Combining Lemma 3.23 and Theorem 3.24 we get a comprehensive list of isomorphism classes of non-supersingular binary elliptic curves.

Corollary 3.25. *The number of equivalence classes of non-supersingular elliptic curves over* $\mathbb{F}_{2^m}$ *equals* $2(2^m - 1)$. *Let* γ *be an element of* $\mathbb{F}_{2^m}$, *for which* $\mathrm{Tr}(\gamma) = 1$. *Then*

$$\{y^2 + xy = x^3 + ax^2 + b \mid a \in \{0, \gamma\}, b \in \mathbb{F}_{2^m}^*\} \quad (3.24)$$

is a set of representatives for these isomorphism classes.

Proof. Let E be a non-supersingular elliptic curve given by

$$y^2 + xy = x^3 + ax^2 + b, \quad a, b \in 2^m, \quad b \neq 0.$$

Now if $\mathrm{Tr}(a) = 0$, let s be a solution of equation

$$0 = a + x + x^2$$

in $\mathbb{F}_{2^m}$, which exists, by Lemma 3.23. Now we see that E be isomorphic to E', given by equation

$$y^2 + xy = x^3 + b.$$

Similarly, if $\mathrm{Tr}(a) = 1$, we let s be a solution of

$$\gamma = a + x + x^2$$

in $\mathbb{F}_{2^m}$. We then find out that E is isomorphic to a curve given by

$$y^2 + xy = x^3 + \gamma x^2 + b.$$

So every non-supersingular elliptic curve over $\mathbb{F}_{2^m}$ is isomorphic to one given by equation of the form (3.24).

To prove the uniqueness of this representation, assume E_1 and E_2 are isomorphic curves given by equations of type (3.21) and (3.22), respectively, with $a_1, a_2 \in \{0, \gamma\}$ and $b_1, b_2 \in \mathbb{F}_{2^m}$. Then by Theorem 3.24, $b_1 = b_2$ and further $a_2 = a_1 + s + s^2$ implies that $\mathrm{Tr}(a_1) = \mathrm{Tr}(a_2)$. Hence $a_1 = a_2$ and we are done. □

In essentially the same way, a similar treatment can be carried out for supersingular curves. The results are summarized in the next two theorems. For detailed proofs, see [Men93, Chapter 3].

Theorem 3.26. *If m is odd, there are 3 isomorphism classes of supersingular elliptic curves over $\mathbb{F}_{2^m}$. A representative from each class is*

$$\begin{aligned} y^2 + y &= x^3 \\ y^2 + y &= x^3 + x \\ y^2 + y &= x^3 + x + 1. \end{aligned}$$

Theorem 3.27. *If m is even, the number of isomorphism classes of supersingular elliptic curves over $\mathbb{F}_{2^m}$ equals 7. Let γ be a non-cube in $\mathbb{F}_{2^m}$ and δ an element in $\mathbb{F}_{2^m}$ such that*

$$\delta + \delta^{2^2} + \delta^{2^4} + \cdots + \delta^{2^{m-2}} \neq 0.$$

Let $\alpha, \beta, \omega \in \mathbb{F}_{2^m}$ be such that $\mathrm{Tr}(\gamma^{-2}\alpha) = \mathrm{Tr}(\gamma^{-4}\beta) = \mathrm{Tr}(\omega) = 1$. Then a representative of each class is:

$$\begin{aligned} y^2 + \gamma y &= x^3 \\ y^2 + \gamma y &= x^3 + \alpha \\ y^2 + \gamma^2 y &= x^3 \\ y^2 + \gamma^2 y &= x^3 + \beta \\ y^2 + y &= x^3 \\ y^2 + y &= x^3 + \delta x \\ y^2 + y &= x^3 + \omega. \end{aligned}$$

Chapter 4

The Discrete Logarithm Problem

4.1 Complexity of Discrete Exponentiation

We begin by defining some useful complexity theoretic notation.

Definition 4.1. Let f and g be two functions from the set of positive integers to $\mathbb{R}$ that take on positive values for all $n \geq n_0$ for some $n_0 > 0$.

- $f(n) = O(g(n))$, if there exists a positive constant c and a positive integer n_0 such that $0 \leq f(n) \leq cg(n)$ holds for all $n \geq n_0$.
- $f(n) = o(g(n))$, if for any positive constant c there exists a positive integer n_0 such that $0 \leq f(n) \leq cg(n)$ for $n \geq n_0$.

Intuitively, $f(n) = O(g(n))$ means that g is an asymptotic upper bound of f. If in addition $f(n) = o(g(n))$, this upper bound g of f is not tight in the sense that g will asymptotically bound f, even if f is scaled by an arbitrary large scalar factor. Notation $f = O(1)$ means that f is bounded, while $f = o(1)$ is a shorthand for expressing $\lim f(n) = 0$, $n \to \infty$.

For comparing performance of different algorithms, we can categorize them by the following definition. By the *size* of the input for an algorithm we mean the number of bits required to represent the input in ordinary binary notation. The *running time* of an algorithm is the number of "elementary" operations executed.

Definition 4.2. An algorithm with worst-case running time $f(n)$, where n is the input size is said to be

- *polynomial-time*, if $f(n) = O(n^\nu)$ for some positive constant ν,
- *subexponential-time*, if $f(n) = e^{g(n)}$ for some g with $g(n) = o(n)$,
- *exponential-time*, if $f(n) = e^{g(n)}$ for some g with $g(n) = O(n^\nu)$, where $\nu > 0$.

A randomized algorithm is said to be *probabilistic* polynomial-time (respectively subexponential-time) if its expected running time can be bounded by a polynomial (subexponential) function in the size of the input. We further categorize the

subexponential-time algorithms using the notation

$$L[n, c, \alpha] = O(e^{(c+o(1))(\ln n)^{\alpha}(\ln \ln n)^{1-\alpha}}),$$

where n is the input (with size $O(\ln n)$), c is a constant and $0 \leq \alpha \leq 1$. Note that $L[n, c, 0]$ is polynomial-time, while $L[n, c, 1]$ is fully exponential in $\ln n$.

Using the above terminology, we may now characterize one-way functions to be such that direct evaluation of their values can be done in polynomial-time, while no subexponential-time algorithm for inverting them is known.

The cryptosystems presented in Sections 2.2 and 2.4 all relied on the assumption that discrete exponentiation in a finite group is one-way. Let us check first the assumption about the direct computation of values. Suppose we want to raise an element $g \in G$ to the kth power, where G is a finite group of order n. We may suppose $1 \leq k < n$. After writing k in binary form,

$$k = \sum_{i=0}^{r-1} k_i 2^i, \quad k_i \in \{0, 1\},$$

we immediately find out that

$$g^k = \prod_{i \in I} g^{2^i},$$

where I represents the set of nonzero bit positions of k. So exponentiation in G is accomplished rather rapidly by repeated squaring, with at most $\lceil 2 \log k \rceil$ group operations. Regarding n as the size of the input, we see that discrete exponentiation in G takes $O(\log n)$ operations, i.e. is polynomial-time.

The next sections are devoted to investigating the validity of the second assumption made in claiming one-wayness of exponentiation by exploring best known algorithms for computing discrete logarithms.

4.2 General Methods for Finding Discrete Logarithms

From now on, G will stand for a finite cyclic group of order n generated by g.

Definition 4.3. Let α be an element of G. The problem of finding the unique integer $0 \leq k < n$, for which $g^k = \alpha$ is called the *discrete logarithm problem (DLP)*. The solution is denoted

$$k = \log_g \alpha,$$

and called the *discrete logarithm of α to the base g*.

The first thing to try when faced with the of problem determining a discrete logarithm might be to start computing powers of g until the right one appears. However, the worst case running-time for this method is exponential $O(n)$.

Observe also, that the mapping

$$\mathbb{Z}_n \ni k \mapsto g^k \in G$$

is a group isomorphism between G and $(\mathbb{Z}_n, +)$, the group of integers modulo n. In $(\mathbb{Z}_n, +)$, exponentiation to the kth power means scalar multiplication modulo n, and taking discrete logarithms reduces to computing inverses modulo n, which takes only polynomial time in n using the extended Euclidean algorithm [Ros88]. However, the fact that the groups $(\mathbb{Z}_n, +)$ and G are isomorphic does not imply that the DLP is equally easy in both groups – this is just another way to formulate the DLP: find an efficiently computable group isomorphism between G and $(\mathbb{Z}_n, +)$.

4.2.1 Square-Root Methods

Square-root methods use precomputation tables to cut down the amount of exponentiations in solving the DLP. For example, the so-called *baby-step giant-step method* works as follows. Denote $\nu = \lceil \sqrt{n} \rceil$. If $k = \log_g \alpha$, then k can be uniquely written as $k = j\nu + i$, with $0 \leq i, j < \nu$. The first step is to precompute a list of pairs (i, g^i) and sort this list by the second component. Next, for each j in the range $0 \leq j < \nu$, compute $\alpha g^{-j\nu}$ and use binary search to check if this equals g^i for some i. When such integer j is found, we see that

$$\alpha = g^{i+j\nu}.$$

Hence we have the desired value, $\log_g \alpha = i + j\nu$. The number of operations this method takes is $O(\sqrt{n} \log n)$.

A slight refinement of this method is *Pollard's ρ-method* [Pol78]. By introducing some probabilistic behavior, the expected running time is conjectured to be $O(\sqrt{n})$. Using the present computing technology, both these methods become infeasible, when n is larger than 10^{40}.

Pollard's method can be further improved by assuming that the prime factorization of the order of the group is at hand:

$$n = \prod_{i=1}^{r} p_i^{\lambda_i},$$

where p_i are distinct primes and λ_i positive integers. The result is obtained by first solving the unknown modulo $p_i^{\lambda_i}$ and then using the Chinese Remainder Theorem to write down the result. This *Pohlig—Hellman method* requires $O(\sum_{i=1}^{r} \lambda_i (\log n + \sqrt{p_i} \log p_i))$ group operations [PH78] and is thus efficient if the order of the group is a *smooth integer*, that is, all primes dividing n are relatively small.

4.2.2 The Index Calculus Method

The *index calculus method* is based on the use of a *factor basis*; a subset $\Gamma = \{\gamma_1, \ldots, \gamma_r\}$ of G. This method also involves a precomputation step. The general idea looks as follows. Fix a random integer $s \in \mathbb{Z}$ and try to express g^s using the factor basis Γ:

$$g^s = \prod_{i=1}^{r} \gamma_i^{g_i}. \tag{4.1}$$

If such integers s, g_i are found, then by taking logarithms, we see that

$$s = \sum_{i=1}^{r} g_i \log_g \gamma_i \pmod{n}.$$

Our aim is to generate a solvable system of linear equations by repeatedly choosing suitable integers s. Proceeding this way, we can in principle find the logarithms $\log_g \gamma_i$ of the factor basis.

At the second stage, we again try random integers s to get an equality

$$g^s \alpha = \prod_{i=1}^{r} \gamma_i^{\alpha_i}. \tag{4.2}$$

If we are successful, this will yield the result

$$\log_g \alpha = \sum_{i=1}^{r} \log \gamma_i - s \mod n.$$

The above description of the method leaves few open questions. First, the selection of Γ should be such, that it generates as large subgroup of G as possible, while at the same time the number of elements in Γ should be kept small in order the precomputation tables be small. Also, it is not clear in general how to efficiently create equalities of type (4.1) and (4.2). For multiplicative groups of finite fields some usable algorithms have been proposed. For example, Coppersmith's algorithm [Cop84] solves the DLP in $\mathbb{F}_{2^m}^*$ with heuristic running-time $L[2^m, c, 1/3]$, where $1.3507 \leq c \leq 1.4047$. The best algorithm for this group with rigorously proved expected running-time $L[2^m, \sqrt{2}, 1/2]$ is due to G. Simmons [Sim91].

4.3 The Elliptic Curve Discrete Logarithm Problem

4.3.1 MOV Reduction

The reason for cryptographic interest concerning elliptic curves is that solving the DLP in the group $E(\mathbb{F}_q)$ of $\mathbb{F}_q$-rational points of an elliptic curve seems extremely difficult. We reformulate this with respect to additive notation.

Definition 4.4. Let $E/\mathbb{F}_q$ be an elliptic curve over the field $\mathbb{F}_q$ and S and T points in $E(\mathbb{F}_q)$. Assume that the order of T is n. The *elliptic curve discrete logarithm problem (ECDLP)* is to find and integer $0 \leq k \leq n-1$ such that

$$S = kT.$$

If such k exists, it is called the *discrete logarithm* of S with respect to T and denoted

$$k = \log_T S.$$

For the rest of this chapter, T will be a fixed point of order n on an elliptic curve $E(\mathbb{F}_q)$.

Menezes, Okamoto and Vanstone [MOV93, Men93] have invented a way to reduce the ECDLP in the group $E(\mathbb{F}_q)$ to DLP in $\mathbb{F}_{q^m}^*$. They use a mapping from $E[n]\times E[n]$ to $\overline{\mathbb{F}_q^*}$, called the *Weil pairing* to produce a group isomorphism between groups $\langle T\rangle$ and μ_n, where μ_n is the subgroup of nth roots of unity in $\mathbb{F}_{q^m}$. Here m will be chosen to be the smallest integer such that $E[n] \subseteq E(\mathbb{F}_{q^m})$. This m exists, since by Theorem (3.18) $E[n]$ is finite, and by Corollary 3.13,

$$\lim_{m\to\infty} \#E(\mathbb{F}_{q^m}) = \infty.$$

The isomorphism between $\langle T\rangle$ and μ_n is fixed after finding a suitable point $Q \in E[n]$ and fixing m [Men93]. After fixing these two parameters, the isomorphism can be computed in probabilistic polynomial time in $\log q$ using Miller's algorithm [Mil86a]. However, finding suitable Q and m is not easy in general. If $E/\mathbb{F}_q$ is supersingular, is has been shown [Men93] that $m \leq 6$, which makes determining Q and m possible in probabilistic polynomial time (in $\log q$). Recalling that the DLP in $\mathbb{F}_q^*$ was seen to be solvable in subexponential time, we summarize these observations in the following theorem.

Theorem 4.5. *[Men93] Let T be an element of order n in a supersingular elliptic curve $E(\mathbb{F}_q)$ and let $S \in \langle T\rangle$. If q is a prime, or if q is a prime power $q = p^l$, where p is fixed, then $\log_T S$ can be determined in probabilistic subexponential time in $\log q$.*

What about the non-supersingular case, is there a subexponential-time algorithm for solving the ECDLP then? According to Section 4.2.2, we assume that the best algorithm to solve the DLP in $\mathbb{F}_q^*$ has complexity $L[q, c, 1/3]$. The MOV reduction thus yields a probabilistic complexity $L[q^m, c, 1/3]$. We can now write down two necessary conditions for the MOV reduction to solve the ECDLP in subexponential-time:

1. A necessary condition for $L[q^m, c, 1/3]$ to be subexponential in $\ln q$ is that $m \leq (\ln q)^2$.
2. For $E[n] \subseteq E(\mathbb{F}_{q^m})$ to hold, $q^m - 1$ must be divisible by n, see Theorem 3.19. (We may assume $\gcd(n, q) = 1$ [Men93].)

For large random n, finding an m that fulfills both of the above conditions seems highly unlikely [Men93, Kob91]. Thus, for most non-supersingular elliptic curves, the MOV reduction is of no practical use in solving the ECDLP.

4.3.2 Index/Xedni Calculus on Elliptic Curves

For the DLP in $\mathbb{F}_q^*$, the most efficient solving method is the index calculus that was seen to run in subexponential-time. What are the chances of adapting this method to elliptic curve groups? The major question is how to choose the factor basis? In the finite field $\mathbb{F}_p$, there is a natural choice of regarding points of $\mathbb{F}_p$ as integers. In this case a natural factor basis would be to take the set of r smallest primes. Then writing down equations for powers of the generator reduces to integer factorization, which has been widely studied and for which efficient algorithms have

been developed [LL93]. For elliptic curves there seems to be no feasible candidate for a factor basis.

Miller [Mil86b] and Silverman [SS98] have investigated the possibilities of doing index calculus on elliptic curves defined over $\mathbb{F}_p$ ($p > 3$) by lifting them to curves over $\mathbb{Q}$. To see what this means, let us fix an elliptic curve $E_p/\mathbb{F}_p$ parameterized by coefficients $a_p, b_p \in \mathbb{F}_p$. Further, denote by ρ the reduction map that maps integers to $\mathbb{F}_p$,

$$\rho : \mathbb{Z} \ni x \mapsto x \mod p \in \mathbb{F}_p$$

For clarity, we label images of ρ by $\rho(a) = a'$. Denote by $\mathbb{Q}_p$ the set of rational numbers whose denominator is not divisible by p. Now ρ can be extended to $\mathbb{Q}_p$ by defining

$$\rho : \mathbb{Q}_p \ni r/s \mapsto (s')^{-1} r' \in \mathbb{F}_p.$$

Now, given a curve $E/\mathbb{Q}$ parameterized by $a, b \in \mathbb{Q}_p$, we can determine a reduction of this curve by mapping a, b to $a', b' \in \mathbb{F}_p$. The resulting parameters a', b' now determine an elliptic curve E' over $\mathbb{F}_p$. This is called the *reduction of* $E/\mathbb{Q}$ *modulo* p. *Lifting the curve* $E_p/\mathbb{F}_p$ *to* $E/\mathbb{Q}$ means finding an elliptic curve E over $\mathbb{Q}$ such that E' equals $E_p/\mathbb{F}_p$. Note that this lifting procedure is not a well-defined map. In fact, there are infinitely many choices for a suitable $E/\mathbb{Q}$.

Now we are ready to give a brief outline of how the index calculus could be used to compute $\log_T S$, where $S, T \in E_p(\mathbb{F}_p)$. As before, n is the order of T.

1. Try to lift $E_p/\mathbb{F}_p$ to $E/\mathbb{Q}$ such that $E/\mathbb{Q}$ has a reasonably large number of linearly independent points $P_1, \ldots P_r \in E(\mathbb{Q})$.

2. For $j = 1, 2, \ldots$, compute $jT \in E_p(\mathbb{F}_p)$ and try to lift this to a point $T_j \in E(\mathbb{Q})$. This means finding a point $T_j = (x_j, y_j) \in E(\mathbb{Q})$ such that $T_j' = (x_j', y_j')$ equals jT in $E_p(\mathbb{F}_p)$. If this is successful, try to write T_j with respect to the factor basis $\{P_1, \ldots, P_r\}$;

$$T_j = \sum_{i=1}^{r} n_{ij} P_i, \quad n_{ij} \in \mathbb{Z}. \tag{4.3}$$

3. Since reducing coordinates modulo p preserves the group structure, after r successful steps in Step 3, we have r linear equations

$$jT = T_j' = \left(\sum_{i=1}^{r} n_{ij} P_i \right)' = \sum_{i=1}^{r} n_{ij} P_i' \quad \text{in } E_p(\mathbb{F}_p).$$

Assuming $P_i' \in \langle T \rangle$, we can write $P_i' = (\log_T P_i')T$. Hence we get a system of r linear equations modulo n:

$$j = \sum_{i=1}^{r} n_{ij} \log_T P_i' \pmod{n},$$

which can be solved for $\log_T P_i'$.

4. Next try to lift $S, S+T, S+2T, \ldots$ to $E(\mathbb{Q})$, say that $S+\nu T$ lifts to $S_\nu \in E(\mathbb{Q})$, i.e., $S'_\nu = S + \nu T$. If we now can write

$$S_\nu = \sum_{i=1}^{r} s_i P_i, \quad s_i \in \mathbb{Z}, \tag{4.4}$$

we get as in Step 3:

$$\log_T S + \nu = \sum_{i=1}^{r} s_i \log_T P'_i \pmod{n},$$

which gives the desired solution.

Could the above procedure be implemented in practice? As Miller and Silverman point out [Mil86b, SS98], a number of serious difficulties arise immediately.

First, it is difficult to find elliptic curves $E/\mathbb{Q}$ with a large number of rational points of small height. Here the *height* of a point $P \in E(\mathbb{Q})$ is roughly understood as being the number of bits required to represent it. Also, elliptic curves over $\mathbb{Q}$ tend usually have small rank. By the *rank* of $E/\mathbb{Q}$ we mean the nonnegative integer d in the relation

$$E(\mathbb{Q}) \simeq E_{tors}(\mathbb{Q}) \times \mathbb{Z}^d,$$

where the *torsion subgroup* $E_{tors}(\mathbb{Q})$ (the set of points of finite order) is finite. This relation is justified by the famous *Mordell–Weil Theorem* [Sil86, Chapter VIII].

Theorem 4.6. *The group $E(\mathbb{Q})$ is finitely generated.*

The problem with low rank elliptic curves is that the average height of points lifted from $E(\mathbb{F}_p)$ to $E(\mathbb{Q})$ appears to be exponentially large in $\log p$. Another major obstacle is the problem of lifting points $P \in E(\mathbb{F}_p)$ to $E(\mathbb{Q})$, which is a very difficult problem in itself, possibly even more difficult than the ECDLP. To summarize, we quote Silverman and Suzuki [SS98]:

> - - our theoretical and numerical work fully supports Miller's conclusion that the natural generalization of the index calculus to the elliptic curve discrete logarithm problem yields an algorithm which is less efficient than a brute-force search algorithm.

One recent attempt to avoid the problems appearing in index calculus has been made by Silverman [Sil98]. Instead of trying desperately to find elliptic curves of high rank over $\mathbb{Q}$, a completely different approach, dubbed *xedni calculus* is taken. It begins by choosing points $P_1, \ldots, P_r \in E(\mathbb{F}_p)$ and lifting them to points $Q_1, \ldots, Q_r \in E(\mathbb{Q})$ having integer coordinates. Then an elliptic curve $E/\mathbb{Q}$ is constructed, requiring it to go through the points $Q_1, \ldots, Q_r$. This way, the difficulty of lifting points will be avoided. Next, it is checked if $Q_1, \ldots, Q_r$ are linearly dependent. If they are, the ECDLP is almost certainly solved [Sil98]. Thus the algorithm works, if the constructed curve $E(\mathbb{Q})$ has *smaller* than the expected rank. However (cryptographers: read *luckily*), the points $Q_1, \ldots, Q_r$ will usually be linearly independent. Can we consequently declare the (non-supersingular) ECDLP as intractable for good? The answer is no. Active research is currently being made to modify the xedni calculus to give better performance. Mathematicians won't give up that easily.

4.4 Security of Cryptosystems Based on the ECDLP

4.4.1 Cryptographically Secure Elliptic Curves

Summing up the observations made in Section 4.3, we conclude that the best algorithm known to date for solving the ECDLP is the Pollard-ρ method, with few exceptions. The special cases for which there exists a quicker algorithm are listed below.

1. If $E[n] \subseteq E(\mathbb{F}_{q^m})$ for small m, where n is the order of the point $T \in E(\mathbb{F}_q)$, the MOV reduction yields a probabilistic subexponential-time algorithm for computing $\log_T$.
2. If n is smooth, i.e., it has no large prime factors, then the Pohlig–Hellman algorithm makes solving the ECDLP feasible.
3. A recent result concerning the so-called *anomalous elliptic curves* over $\mathbb{F}_p$ (curves, for which $\#E(\mathbb{F}_p) = p$) states that for this class of curves there exists a polynomial-time algorithm to solve the ECDLP [SA97, Sem98].

Since the running time of the Pollard-ρ algorithm is exponential $O(\sqrt{n})$, we conclude that constructing public key cryptosystems according to Section 2.4 over the group $\langle T \rangle \subseteq E(\mathbb{F}_q)$ yields secure cryptographic primitives, when the underlying curve and field are chosen so that none of the above conditions hold. Condition 1 is checked by use of Theorem 3.19. Conditions 2 and 3 are also easily verified once the cardinality $\#E(\mathbb{F}_q)$ is known. Easily implementable algorithms for checking these requirements are listed in the draft IEEE P1363 standard [IEE98].

4.4.2 Choosing the Key Length

The security of a cryptosystem is parametrized by the key length. For elliptic curve cryptosystems, the private key is the integer $0 < k < n$, where n is the order of the generator point $T \in E(\mathbb{F}_q)$. Usually, n is of the same size as q. To determine the key length required to provide the desired level of security, one needs to estimate the computing power of the potential attacker.

field size (bits)	size of n (bits)	computing power (mips-years)
163	160	$9.6 \cdot 10^{11}$
191	186	$7.9 \cdot 10^{15}$
239	234	$1.6 \cdot 10^{23}$
359	354	$1.5 \cdot 10^{41}$
431	426	$1.0 \cdot 10^{52}$

Table 4.1: Computing power required to solve a single ECDLP with the Pollard ρ-method.

Computing power will be measured in *mips-years*, the amount of computation performed by 1 mips machine in a year. Conventionally, 1 mips machine is defined to be equal to the DEC VAX 11/780 in computing power (the latest PC-generation has estimated power of 300 mips). Odlyzko has estimated [Odl95] that if 0.1% of the whole world's computing power were put to work through the Internet for a year, there could be $2 \cdot 10^9$ mips-years available in 2004, and 10^{11}–10^{13} mips-years in 2014.

In the light of Odlyzko's forecasts, it seems reasonable to require that a today's cryptosystem resist an attack of at least 10^{12} mips-years. For elliptic curves, some approximations [cer97] have been made concerning the hardness of the ECDLP. These results are listed in Table 4.1.

For comparison, Table 4.2 (due to Odlyzko [Odl95]) lists the approximated computing power required to solve the problem of factoring an integer n using the *general number field sieve* [LL93], the best known factoring algorithm. RSA, currently the most widely used public key cryptosystem, is based upon this problem [RSA78]. The key lengths used in the RSA cryptosystem are the size of n.

size of n (bits)	computing power (mips-years)
512	$3 \cdot 10^4$
768	$2 \cdot 10^8$
1024	$3 \cdot 10^{11}$
1280	$1 \cdot 10^{14}$
1536	$3 \cdot 10^{16}$
2048	$3 \cdot 10^{20}$

Table 4.2: Computing power required to factor an integer n using the general number field sieve.

Comparing the above tables, we see that elliptic curve cryptosystems can offer equal level of security with the 1024-bit RSA using much shorter 160-bit keys. Therefore, the use of elliptic curves provides a very interesting alternative to traditional methods in public key cryptography. However, it should not be forgotten that the 160/1024-ratio may either increase of decrease in the future, if better algorithms for factoring integers or solving the ECDLP are found.

Chapter 5

Algorithms for Finite Field Arithmetic

The reader unfamiliar with finite fields is advised to consult Appendix A before proceeding with this chapter.

5.1 Representation of Finite Fields

5.1.1 Basic Operations

We know that the number of elements of a finite field is a power of a prime. Thus from now on, q will stand for the number of elements of a finite field, $q = p^l$, where p is a prime and l is a positive integer.

Since the field $\mathbb{F}_{q^k}$ is a k-dimensional vector space over $\mathbb{F}_q$, there exists a basis $\{\alpha_0, \ldots, \alpha_{k-1}\} \in \mathbb{F}_{q^k}$ such that an arbitrary element $a \in \mathbb{F}_{q^k}$ can be uniquely presented as

$$a = a_0\alpha_0 + \cdots + a_{k-1}\alpha_{k-1}, \quad a_i \in \mathbb{F}_q.$$

Elements of $\mathbb{F}_{q^k}$ may thus be viewed as vectors with k $\mathbb{F}_q$-components. We denote this by $a = (a_{k-1}, a_{k-2}, \ldots, a_0)$. If $b = (b_{k-1}, \ldots, b_0)$ is another element of $\mathbb{F}_{q^k}$, we see that the sum of a and b is

$$a + b = \sum_{i=0}^{k-1}(a_i + b_i)\alpha_i = (a_{k-1} + b_{k-1}, \ldots, a_0 + b_0).$$

Hence addition in $\mathbb{F}_{q^k}$ with respect to any basis over $\mathbb{F}_q$ looks always similar. Clearly, this holds also for subtraction in $\mathbb{F}_{q^k}$. The zero of $\mathbb{F}_{q^k}$ is always given by $(0, \ldots, 0)$; a vector with k zeros.

Let $c = (c_{k-1}, \ldots, c_0)$ denote the product of a and b in $\mathbb{F}_{q^k}$. Then

$$c = ab = \sum_{i=0}^{k-1} a_i\alpha_i \sum_{j=0}^{k-1} b_j\alpha_j = \sum_{i,j=0}^{k-1} a_i b_j M_{i,j},$$

where

$$M_{i,j} = \alpha_i \alpha_j = \sum_{t=0}^{k-1} m_{i,j}^{(t)} \alpha_t$$

for some elements $m_{i,j}^{(t)} \in \mathbb{F}_q$. From this representation it is clear that the way multiplication is computed depends heavily on the structure of the basis. In the following section we will investigate how multiplication behaves in different bases.

5.1.2 Bases of Finite Fields

Polynomial Bases

Theorems A.21 and A.22 in Appendix A induce the simplest way to represent the elements of a finite field $\mathbb{F}_{q^k}$. Let $P(x)$ be an irreducible polynomial over $\mathbb{F}_q$ of degree k (this exists by Theorem A.44). Then $\mathbb{F}_{q^k} = \mathbb{F}_q(\theta)$, where θ is a root of $P(x)$ in $\mathbb{F}_{q^k}$. Furthermore, the set $\{1, \theta, \theta^2, \ldots, \theta^{k-1}\}$ is a basis of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$. As the Theorem A.21 says, $\mathbb{F}_{q^k}$ may analogously be viewed as $\mathbb{F}_q[x]/(P)$, the field of polynomials over $\mathbb{F}_q$ modulo $P(x)$. This observation justifies the name of this basis.

Definition 5.1. The set $\{1, \theta, \theta^2, \ldots, \theta^{k-1}\}$ of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$, where θ is a root of an irreducible polynomial $P(x)$ over $\mathbb{F}_q$ is called the *polynomial basis* of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$. The defining polynomial $P(x)$ is called the *field polynomial* of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$.

Example 5.2. Consider the polynomial $P(x) = x^4 + x + 1$ over $\mathbb{F}_2$. Since $P(0) = P(1) = 1$, $P(x)$ has no roots in $\mathbb{F}_2$ and thus by Lemma A.23 no affine factors in $\mathbb{F}_2[x]$. The only irreducible second order polynomial over $\mathbb{F}_2$ is $x^2 + x + 1$. By a direct check it can be seen that this does not divide $P(x)$. So we conclude that $P(x)$ is irreducible over $\mathbb{F}_2$. A root θ of $P(x)$ spans now a polynomial basis over $\mathbb{F}_{2^4}$ and the elements of $\mathbb{F}_{2^4}$ can be represented by vectors with four bits.

$$a = (a_3, a_2, a_1, a_0) = \sum_{i=0}^{3} a_i \theta^i, \quad a_i \in \mathbb{F}_2.$$

The product of $a = (1, 0, 0, 1)$ and $b = (0, 1, 1, 0)$ is given by

$$\begin{aligned} ab &= (\theta^3 + 1)(\theta^2 + \theta) \\ &= \theta^5 + \theta^4 + \theta^2 + \theta \\ &= \theta^4(\theta + 1) + \theta^2 + \theta \\ &= (\theta + 1)(\theta + 1) + \theta^2 + \theta \\ &= \theta^2 + 1 + \theta^2 + \theta = \theta + 1, \end{aligned}$$

where, by definition, $\theta^4 = \theta + 1$. Thus the product of a and b equals $(0, 0, 1, 1)$.

Note that the above method of first guessing a polynomial and then testing for its reducibility is not computationally feasible in large fields. [MBG+93, Chapter 3] describes in detail general methods for constructing irreducible polynomials over finite fields. For the special case of fields with characteristic 2, easily implementable algorithms for doing this are given in [IEE98]. Several precomputed tables of irreducible polynomials with minimal number of terms are reported in the literature, see for example [LN97, IEE98].

Normal Bases

Another interesting basis for a finite field can be generated by repeatedly applying the qth power map.

Definition 5.3. If for some $\alpha \in \mathbb{F}_{q^k}$, the set

$$\{\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{k-1}}\}$$

is linearly independent, that set is called a *normal basis* of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$.

Following the tradition, an element $a \in \mathbb{F}_{q^k}$ represented with respect to a normal basis

$$a = \sum_{i=0}^{k-1} a_i \alpha^{q^i}, \quad a_i \in \mathbb{F}_q$$

is associated the vector notation $a = (a_0, a_1, \ldots, a_{k-1})$, reversing the notation used for polynomial basis.

Example 5.4. Since $P(x) = x^3 + x^2 + 1$ has no roots in $\mathbb{F}_2$, it is irreducible in $\mathbb{F}_2[x]$. Let θ be a root of $P(x)$ in $\mathbb{F}_{2^3}$. By definition, $\theta^3 = \theta^2 + 1$, so $\theta^4 = \theta^2 + \theta + 1$. Now it is easy to see that θ, θ^2 and θ^4 are linearly independent over $\mathbb{F}_2$, i.e., they form a normal basis of $\mathbb{F}_{2^3}$ over $\mathbb{F}_2$. If $a = (a_0, a_1, a_2) \in \mathbb{F}_{2^3}$, the square of a will be

$$a^2 = (a_0\theta + a_1\theta^2 + a_2\theta^4)^2 = a_0\theta^2 + a_1\theta^4 + a_2\theta^8 = a_2 + a_0\theta^2 + a_1\theta^4,$$

i.e., squaring in $\mathbb{F}_{2^3}$ using this representation is a simple cyclic shift $(a_0, a_1, a_2)^2 = (a_2, a_0, a_1)$. This remarkable property makes the use of normal bases quite appealing for applications.

Using Galois theory, it can be proved [Lan65, Section 8] that every finite extension of a finite field has a normal basis.

5.1.3 Duality in Finite Fields

In linear algebra, the *dual* of a vector space V over K is defined to be the set of linear maps from V to K. We will next show a practical interpretation of this in the context of finite fields. First we note that the map

$$\mathbb{F}_{q^k} \times \mathbb{F}_{q^k} \ni (a, b) \to \mathrm{Tr}(ab) \in \mathbb{F}_q$$

is a bilinear form, i.e., linear with respect to both of its arguments. We denote this mapping by $\mathrm{Tr}(ab) = \langle a, b \rangle$.

Theorem 5.5. *Every linear map L from $\mathbb{F}_{q^k}$ to $\mathbb{F}_q$ can be uniquely represented by an element $\beta \in \mathbb{F}_{q^k}$ such that*

$$L(\alpha) = \langle \beta, \alpha \rangle$$

for all $\alpha \in \mathbb{F}_{q^k}$.

Proof. For any $\beta \in \mathbb{F}_{q^k}$, the map

$$L_\beta : \mathbb{F}_{q^k} \ni \alpha \mapsto \langle \beta, \alpha \rangle \in \mathbb{F}_q$$

is obviously linear. Furthermore, for $\gamma \neq \beta$, $L_\gamma(\alpha) - L_\beta(\alpha) = \mathrm{Tr}((\gamma - \beta)\alpha) \neq 0$ for some $\alpha \in \mathbb{F}_{q^k}$ since trace is surjective by Theorem A.46. Thus every element $\beta \in \mathbb{F}_{q^k}$ yields a different linear map L_β. On the other hand, every linear map L from $\mathbb{F}_{q^k}$ to $\mathbb{F}_q$ is uniquely given by assigning to a given fixed basis α_i of $\mathbb{F}_{q^k}$ arbitrary values $L(\alpha_i) \in \mathbb{F}_q$. This way we see that the total number of linear mappings from $\mathbb{F}_{q^k}$ to $\mathbb{F}_q$ is q^k. The mappings L_β already exhaust all of these, thereby proving the theorem. □

After fixing our bilinear form $\langle a, b \rangle = \mathrm{Tr}(ab)$, we may regard $\mathbb{F}_{q^k}$ as its own dual. Transmitting duality using the trace function is not the only alternative; one could have chosen the ordinary inner product similar to that of $\mathbb{R}^n$ as well. But in order to compute an inner product, an *a priori* basis of the vector space in question is required. This explains our choice of the bilinear form. It depends only on the structure of the underlying field of scalars, not on the choice of basis representation.

Definition 5.6. Given two bases $\{\alpha_i\}$ and $\{\beta_i\}$ of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$, $\{\beta_i\}$ is said to be a *dual basis* of $\{\alpha_i\}$, if $\langle \alpha_i, \beta_j \rangle = \delta_{ij}$, where

$$\delta_{ij} = \begin{cases} 0, & i \neq j, \\ 1, & i = j, \end{cases}$$

is the Kronecker delta. If the basis $\{\alpha_i\}$ is dual of itself it is called *self-dual.*

Theorem 5.7. *For any basis $\{\alpha_i\}$ of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$ there exists a uniquely determined dual basis.*

Proof. Let

$$\sum_{i=0}^{k-1} a_i \alpha_i, \quad a_i \in \mathbb{F}_q$$

be the representation of $\alpha \in \mathbb{F}_{q^k}$ with respect to $\{\alpha_i\}$. Now the coefficients a_i are actually linear functions of α. So, by Theorem 5.5 there exists $\beta_i \in \mathbb{F}_{q^k}$ such that $a_i = \langle \beta_i, \alpha \rangle$, i.e.,

$$\alpha = \sum_{i=0}^{k-1} \langle \beta_i, \alpha \rangle \alpha_i$$

for all $\alpha \in \mathbb{F}_{q^k}$. For $\alpha = \alpha_j$,

$$\alpha_j = \sum_{i=0}^{k-1} \langle \beta_i, \alpha \rangle \alpha_i,$$

which implies $\langle \alpha_j, \beta_i \rangle = \delta_{ij}$. The set $\{\beta_i\}$ is linearly independent, since

$$\sum_{i=0}^{k-1} b_i \beta_i = 0, \quad b_i \in \mathbb{F}_q$$

implies

$$0 = \langle \sum_{i=0}^{k-1} b_i \beta_i, \alpha_j \rangle = \sum_{i=0}^{k-1} b_i \langle \beta_i, \alpha_j \rangle = \sum_{i=0}^{k-1} b_i \delta_{ij} = b_j$$

for any b_j. □

We state below a few theorems about the existence of dual bases without proof.

Theorem 5.8. *The dual basis of a normal basis is a normal basis [MBG+93].*

Theorem 5.9. *If k is odd, or if $k = 2 \pmod 4$ and q is even, then there exists a self-dual normal basis of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$ [LW88].*

Theorem 5.10. *For $k \geq 2$, there does not exist a self-dual polynomial basis of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$ [JMV90].*

5.1.4 Logarithm Tables

A vector space construction is not the only feasible choice for representing field elements. Theorem A.42 in Appendix A guarantees that the nonzero elements of the field $\mathbb{F}_q$ can be expressed as powers of a specific element γ, the generator of $\mathbb{F}_q^*$. Thus we have:

$$\mathbb{F}_q = \{0\} \cup \{\gamma, \gamma^2, \ldots, \gamma^{q-1}\}.$$

After fixing a generator γ of $\mathbb{F}_q^*$, we shall create a table containing all discrete logarithms of elements $\mathbb{F}_q^*$ to the base γ, recall Definition 4.3. As was seen in Section 4.2, there are no very efficient ways to compute discrete logarithms in a large finite field. However, assuming q to be small we can use brute force and compute all powers of γ beforehand. Then we can store a table of all pairs $(a, \log_\gamma a)$ into memory.

The use of a logarithm table makes computation in $\mathbb{F}_q$ easy. For example, to multiply two nonzero elements a and b one can proceed in the following way:

1. Find $i = \log_\gamma a$ and $j = \log_\gamma b$ from the table.
2. Calculate $\nu = i + j \mod q - 1$.
3. Find $c = \gamma^\nu = ab$ from the table and return $c = ab$.

Inverting $a \neq 0$ is just as trivial:

1. Find $i = \log_\gamma a$ using the table.
2. Calculate $\nu = -i \mod q - 1$.
3. Find and return $c = \gamma^\nu = a^{-1}$.

The only drawback with this method is its huge memory requirement. For example, for $q = 2^k$ the logarithm table needs $O(k2^k)$ bits of memory.

5.2 Multiplication

5.2.1 Polynomial Basis Algorithms

A Two-Step Algorithm

Consider a finite field $\mathbb{F}_{q^k}$ represented with respect to a polynomial basis generated by θ, a root of the irreducible polynomial

$$P(x) = x^k + \sum_{i=0}^{k-1} p_i x^i, \quad p_i \in \mathbb{F}_q.$$

Let $a, b \in \mathbb{F}_{q^k}$,

$$a = \sum_{i=0}^{k-1} a_i \theta^i, \quad b = \sum_{i=0}^{k-1} b_i \theta^i, \quad a_i, b_i \in \mathbb{F}_q,$$

and denote their product by $c = ab$.

A convenient way to express the multiplication procedure is to associate with a and b the polynomials

$$A(x) = \sum_{i=0}^{k-1} a_i x^i, \quad B(x) = \sum_{i=0}^{k-1} b_i x^i,$$

respectively. These polynomials represent the equivalence classes in $\mathbb{F}_q[x]/(P)$ generated by a and b, respectively, according to Theorem A.21. We then multiply $A(x)$ and $B(x)$ in $\mathbb{F}_q[x]$: $C'(x) = A(x)B(x)$. Now reduction modulo $P(x)$ yields a polynomial in $\mathbb{F}_q[x]$ with degree less than k:

$$C(x) = C'(x) \mod P(x) = \sum_{i=0}^{k-1} c_i x^i.$$

By Theorem A.21, the coefficients $c_i \in \mathbb{F}_q$ are indeed the ones we were looking for:

$$ab = c = \sum_{i=0}^{k-1} c_i \theta^i.$$

Thus the result may be obtained by performing the following two steps:

1. Ordinary polynomial multiplication in $\mathbb{F}_q[x]$.
2. Reduction modulo the field polynomial $P(x)$.

A direct way to perform the first step would be to simply write out the expression for $C'(x)$,

$$C'(x) = A(x)B(x) = \left(\sum_{i=0}^{k-1} a_i x^i \right) \left(\sum_{j=0}^{k-1} b_j x^j \right) = \sum_{i=0}^{2k-2} c'_i x^i, \tag{5.1}$$

from which the coefficients c'_i are seen to be

$$\begin{aligned}
c'_0 &= a_0 b_0 \\
c'_1 &= a_1 b_0 + a_0 b_1 \\
&\vdots \\
c'_{k-1} &= a_{k-1} b_0 + a_{k-2} b_1 + \cdots + a_1 b_{k-2} + a_0 b_{k-1} \\
c'_k &= a_{k-1} b_1 + a_{k-2} b_2 + \cdots + a_1 b_{k-1} \\
&\vdots \\
c'_{2k-3} &= a_{k-1} b_{k-2} + a_{k-2} b_{k-1} \\
c'_{2k-2} &= a_{k-1} b_{k-1}.
\end{aligned}$$

In the second step we need to write down the expressions $x^k, \ldots, x^{2k-2}$ in (5.1) using powers of x not larger that $k-1$. While the mapping that yields this representation is obviously linear, we can implicitly define the *reduction matrix* $\mathbf{R} = (r_{i,j}) \in (\mathbb{F}_q)^{(k-1)\times k}$ by requiring

$$\begin{pmatrix} x^k \mod P(x) \\ x^{k+1} \mod P(x) \\ \vdots \\ x^{2k-2} \mod P(x) \end{pmatrix} = \begin{pmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,k-1} \\ r_{1,0} & r_{1,1} & \cdots & r_{1,k-1} \\ \vdots & \vdots & & \vdots \\ r_{k-2,0} & r_{k-2,1} & \cdots & r_{k-2,k-1} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ \vdots \\ x^{k-1} \end{pmatrix}. \tag{5.2}$$

The reduction matrix depends solely on the defining irreducible polynomial $P(x)$. To compute its entries, we note that

$$x^k \mod P(x) = -\sum_{j=0}^{k-1} p_j x^j,$$

which gives us the first column of $\mathbf{R}$:

$$r_{0,j} = -p_j, \quad j = 0, \ldots, k-1$$

The rest follows by recursion; for $1 \le i \le k-2$,

$$\begin{aligned}
x^{k+i} \mod P(x) &= x x^{k+(i-1)} \mod P(x) \\
&= x \sum_{j=0}^{k-1} r_{i-1,j} x^j \mod P(x) \\
&= \sum_{j=1}^{k-1} r_{i-1,j-1} x^j + r_{i-1,k-1} x^k \mod P(x) \\
&= \sum_{j=1}^{k-1} r_{i-1,j-1} x^j + r_{i-1,k-1} \sum_{j=0}^{k-1} r_{0,j} x^j \\
&= \sum_{j=0}^{k-1} (r_{i-1,j-1} + r_{i-1,k-1} r_{0,j}) x^j,
\end{aligned}$$

where for convenience, we agree $r_{i,-1} = 0$. Observing that by (5.2)

$$x^{k+i} \mod P(x) = \sum_{j=0}^{k-1} r_{i,j} x^j,$$

we get the following lemma.

Lemma 5.11. *The reduction matrix* $\mathbf{R}$ *in (5.2) can be computed from the polynomial* $P(x)$ *by*

$$\begin{aligned} r_{0,j} &= -p_j, \quad j = 0, \ldots, k-1; \\ r_{i,j} &= r_{i-1,j-1} - p_j r_{i-1,k-1}, \quad i = 1, \ldots, k-2, \quad j = 0, \ldots, k-1; \end{aligned}$$

where $r_{i,-1} = 0$.

We are now ready to write an explicit formula for the reduced product $C(x)$:

$$\begin{aligned} C(x) = \sum_{s=0}^{2k-2} c'_s x^s \mod P(x) &= \sum_{s=0}^{k-1} c'_s x^s + \sum_{i=0}^{k-2} c'_{k+i} \left(\sum_{j=0}^{k-1} r_{i,j} x^j \right) \mod P(x) \\ &= \sum_{s=0}^{k-1} \left(c'_s + \sum_{i=0}^{k-2} c'_{k+i} r_{i,s} \right) x^s. \end{aligned}$$

Or in matrix form, the desired result is

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{k-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 & r_{0,0} & \cdots & r_{k-2,0} \\ 0 & 1 & \cdots & 0 & r_{0,1} & \cdots & r_{k-2,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & r_{0,k-1} & \cdots & r_{k-2,k-1} \end{pmatrix} \begin{pmatrix} c'_0 \\ \vdots \\ c'_{k-1} \\ c'_k \\ \vdots \\ c'_{2k-2} \end{pmatrix}.$$

The complexity of the above matrix multiplication is: $k(k-1)$ additions and multiplications (by a constant $r_{i,j}$) in $\mathbb{F}_q$. Since the first step takes k^2 (general multiplications) and $(k-1)^2$ additions in $\mathbb{F}_q$, the total computational complexity of this multiplication algorithm in $\mathbb{F}_{q^k}$ is:

- $2k^2 - k$ multiplications in $\mathbb{F}_q$,
- $2k^2 - 3k + 1$ additions in $\mathbb{F}_q$.

The Karatsuba–Ofman Algorithm

The complexity estimates for the two-step algorithm derived above can be made much lower in practice. The first thing is to note that by a careful choice of the field polynomial $P(x)$ the reduction matrix $\mathbf{R}$ will be sparse [Paa96]. That is the reason why we would like to perform the multiplication step with less than $O(k^2)$ operations. This can be achieved by using an algorithm first described by Karatsuba

and Ofman [KO63]. The method is based on splitting the multiplicands to lower their order. Towards this end, we assume both our our polynomials are of degree less than k, where $k = 2^t$.

We begin with splitting the multiplicands as follows:

$$A(x) = x^{\frac{k}{2}} \sum_{i=0}^{\frac{k}{2}-1} a_{\frac{k}{2}+i} x^i + \sum_{i=0}^{\frac{k}{2}-1} a_i x^i = x^{\frac{k}{2}} A_1(x) + A_0(x).$$
$$B(x) = x^{\frac{k}{2}} \sum_{i=0}^{\frac{k}{2}-1} b_{\frac{k}{2}+i} x^i + \sum_{i=0}^{\frac{k}{2}-1} b_i x^i = x^{\frac{k}{2}} B_1(x) + B_0(x).$$

Using this decomposition, we can express the product of $A(x)$ and $B(x)$ as

$$C(x) = D_0(x) + x^{\frac{k}{2}}[D_1(x) - D_0(x) - D_2(x)] + x^k D_2(x),$$

where

$$\begin{aligned} D_0(x) &= A_0(x)B_0(x), \\ D_1(x) &= [A_0(x) + A_1(x)][B_0(x) + B_1(x)], \\ D_2(x) &= A_1(x)B_1(x). \end{aligned}$$

So we have reduced the original multiplication to 3 multiplications and 4 summations of polynomials of degree $\leq k/2 - 1$. Naturally, we may continue this way and split next the polynomials $A_0, A_1, A_0 + A_1$ together with the corresponding parts of $B(x)$. This may further be continued until finally, the degree of polynomials to be multiplied is 0. The algorithm takes $t = \log k$ steps. By counting the operations involved, we get an estimate for the computational complexity of this algorithm. For a detailed proof, see [Paa96].

Theorem 5.12. *Let $A(x)$ and $B(x)$ be two polynomials in $\mathbb{F}_q[x]$ having degrees less than k. Then their product can be computed using the Karatsuba–Ofman algorithm with*

- *the number of $\mathbb{F}_q$-multiplications $\#\otimes = \lceil k \rceil^{log3}$.*
- *the number of $\mathbb{F}_q$-additions $\#\oplus \leq 6\lceil k \rceil^{log3} - 8\lceil k \rceil + 2$.*

Observing that $\log 3 \sim 1.58$, we see that this yields much better performance than using the ordinary polynomial multiplication with $O(k^2)$ subfield operations.

Squaring in a Field of Characteristic 2

We conclude our discussion on multiplication in polynomial basis by observing that squaring in a field of characteristic 2 can be done a little bit more efficiently than multiplying an element by itself using the multiplication algorithms discussed above. Let

$$a = \sum_{i=0}^{k-1} a_i \theta^i, \quad a_i \in \mathbb{F}_q,$$

where q is a power of 2. Then squaring becomes a linear operation, and hence we may write

$$a^2 = \sum_{i=0}^{k-1} a_i^2 \theta^{2i}, \quad a_i \in \mathbb{F}_q.$$

This gives the square of a in the polynomial basis $\{1, \theta, \ldots, \theta^{k-1}\}$ after multiplication with the reduction matrix $\mathbf{R}$.

5.2.2 Multiplication with Respect to a Normal Basis over $\mathbb{F}_2$

Let $\beta, \beta^2, \beta^4, \ldots \beta^{2^{k-1}}$ be a normal basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$. As already observed in Example 5.4, squaring an element $a = (a_0, a_1, \ldots, a_{k-1})$ given in normal basis is cyclic shifting, i.e., $a^2 = (a_{k-1}, a_0, \ldots, a_{k-2})$. To find the product $c = (c_0, \ldots, c_{k-1})$ of $a = (a_0, \ldots, a_{k-1})$ and $b = (b_0, \ldots, b_{k-1})$, we write

$$c = \left(\sum_{i=0}^{k-1} a_i \beta^{2^i} \right) \left(\sum_{j=0}^{k-1} b_j \beta^{2^j} \right) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_i b_j \beta^{2^i + 2^j}.$$

We will next express elements $\beta^{2^i+2^j}$ in the normal basis,

$$\beta^{2^i+2^j} = \sum_{t=0}^{k-1} \lambda_{i,j}^{(t)} \beta^{2^t},$$

with $\lambda_{i,j}^{(t)} \in \mathbb{F}_2$. Thus we have for the coordinates of our unknown,

$$c_t = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \lambda_{i,j}^{(t)} a_i b_j, \quad t = 0, 1, \ldots, k-1. \tag{5.3}$$

We may write $c_t = c_t(a, b)$, viewing c_t as a bilinear mapping from the coordinate spaces of a and b to $\mathbb{F}_2$. Because we are dealing with a normal basis, it follows that

$$c_t(a, b) = (c^{2^{k-t}})_0 = (a^{2^{k-t}} b^{2^{k-t}})_0 = c_0(a^{2^{k-t}}, b^{2^{k-t}}).$$

Thus we may compute the coordinates c_t of c from equation (5.3) with $t = 0$, by first applying a t-fold cyclic shift to a and b:

$$c_t = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \lambda_{i,j}^{(0)} a_{i+t} b_{j+t}, \quad t = 0, 1, \ldots, k-1, \tag{5.4}$$

where for convenience, subscripts are considered modulo k. Hence the product of a and b with respect to the basis $\{\beta, \beta^2, \ldots, \beta^{2^{k-1}}\}$ is determined by finding out the matrix $\Lambda = (\lambda_{i,j}^{(0)}) \in (\mathbb{F}_2)^{k \times k}$.

The *complexity of the normal basis* $\{\beta, \beta^2, \ldots, \beta^{2^{k-1}}\}$ is defined to be the number of nonzero entries in Λ. We denote this by $\mathcal{C}_\beta$. It is clear that $\mathcal{C}_\beta \leq k^2$. A lower bound $2k - 1 \leq \mathcal{C}_\beta$ has been determined in [MOVW89]. This suggests the following definition.

Definition 5.13. A normal basis $\{\beta, \beta^2, \ldots, \beta^{2^{k-1}}\}$ of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$ is called *optimal*, if $\mathcal{C}_\beta = 2k - 1$.

There are two well-known constructions of optimal normal bases [MOVW89] by Mullin et al. The next two theorems present these. For proofs, see for example [MBG+93].

Theorem 5.14. *Suppose $k+1$ is a prime and 2 is a primitive element in the field $\mathbb{Z}_{k+1}$. Then the k non-unit $(k+1)$th roots of unity form an optimal normal basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$.*

Theorem 5.15. *Let $2k+1$ be a prime and assume that either*

1. *2 is primitive in $\mathbb{Z}_{2k+1}$, or*
2. *$2k+1 = 3 \pmod 4$ and 2 generates the set of quadratic residues in $\mathbb{Z}_{2k+1}$ (i.e., every nonzero element having a square root in $\mathbb{Z}_{2k+1}$ is a power of 2).*

Let γ be an element generating the set of $(2k+1)$th roots of unity in $\mathbb{Z}_{2k+1}$. Then $\beta = \gamma + \gamma^{-1}$ generates an optimal normal basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$.

The basis corresponding to Theorem 5.14 is called a *type I* optimal normal basis, and the optimal normal basis constructed according to Theorem 5.15 is called *type II*. S. Gao and H. Lenstra [GL92] have proved that all optimal bases of $\mathbb{F}_{2^k}$ are indeed either of type I or type II. Table 5.1 lists the values of $150 \leq k \leq 200$ for which there exists an optimal normal basis over $\mathbb{F}_{2^k}$ [MBG+93]. The values marked with ♡ indicate existence of type I optimal normal basis, otherwise the optimal normal basis is of type II.

155	174	186
158	178♡	189
162♡	179	191
172♡	180♡	194
173	183	196♡

Table 5.1: Values of $150 \leq k \leq 200$ for which there exists an optimal normal basis in $\mathbb{F}_{2^k}$.

5.3 Inversion

5.3.1 Algorithms Using Exponentiation

To invert a nonzero element α in the finite field $\mathbb{F}_{q^k}$, one practical method is to use the *finiteness* property of the field. Multiplying α by itself will eventually produce 1. So we just have to stop multiplying after the correct amount of iterations. More rigorously, by Lemma A.36

$$\alpha^{q^k - 1} = 1$$

which immediately gives

$$\alpha^{-1} = \alpha^{q^k-2}.$$

Writing the exponent as powers of q we get

$$\begin{aligned} q^k - 2 &= -1 + (q-1)\frac{q^k-1}{q-1} \\ &= -1 + (q-1)(1 + q + q^2 + \cdots + q^{k-1}) \\ &= q - 2 + (q-1)(q + q^2 + \cdots + q^{k-1}), \end{aligned}$$

so

$$\alpha^{-1} = \alpha^{q-2} \prod_{i=1}^{k-1} \left(\alpha^{q-1}\right)^{q^i}.$$

For the most important case, $q = 2$, this implies

$$\alpha^{-1} = \prod_{i=1}^{k-1} \alpha^{2^i},$$

so inversion in $\mathbb{F}_{2^k}$ can be done by $O(k)$ squarings and multiplications in $\mathbb{F}_{2^k}$.

For the case of binary fields, $q = 2$, Itoh and Tsujii [IT88] have presented a drastic improvement to this algorithm. It is based on writing

$$\alpha^{-1} = \alpha^{2^k-2} = \left(\alpha^{2^{k-1}-1}\right)^2.$$

We can continue splitting the exponent by writing

$$2^{k-1} - 1 = \begin{cases} \left(2^{(k-1)/2} - 1\right)\left(2^{(k-1)/2} + 1\right), & k \text{ odd}, \\ 2\left(2^{(k-2)/2} - 1\right)\left(2^{(k-2)/2} + 1\right) + 1, & k \text{ even}. \end{cases}$$

In the case k is odd, this yields

$$\alpha^{2^{k-1}-1} = \left(\alpha^{2^{(k-1)/2}-1}\right)^{2^{(k-1)/2}+1}.$$

Thus, if we know $\alpha^{2^{(k-1)/2}-1}$, we get $\alpha^{2^{k-1}-1}$ by one multiplication and few repeated squarings, assuming k is odd. Similarly, in the case of even k, prior knowledge of $\alpha^{2^{(k-2)/2}-1}$ gives $\alpha^{2^{k-1}-1}$ after two multiplications and some squarings. Continuing this way, we get an efficient iteration.

Example 5.16. Let $k = 16$. Now

$$\begin{aligned} 2^{15} - 1 &= 2(2^7 - 1)(2^7 + 1) + 1, \\ 2^7 - 1 &= 2(2^3 - 1)(2^3 + 1) + 1, \\ 2^3 - 1 &= 2(2 + 1) + 1. \end{aligned}$$

So, we get α^{-1} by computing

$$\begin{aligned} \beta &= \alpha^{2(2+1)+1}, \\ \gamma &= \beta^{2(2^3+1)}\alpha, \\ \delta &= \gamma^{2(2^7+1)}\alpha, \\ \delta^2 &= \alpha^{-1}. \end{aligned}$$

This requires 6 multiplications and 15 squarings in $\mathbb{F}_{2^{16}}$.

Using induction, one can prove the following result about the multiplicative complexity of this method. For details, see [ABMV93].

Theorem 5.17. *The number of multiplications required in computing* α^{-1} *for* $\alpha \in \mathbb{F}_{2^k}^*$ *by the Itoh–Tsujii algorithm is equal to*

$$\lfloor \log(k-1) \rfloor + H_w(k-1) - 1 = O(\log k).$$

Here $H_w(k-1)$ *stands for the* Hamming weight *of* $k-1$*, the number of ones in the binary representation of* $k-1$*.*

Note that the previous theorem does not include the count for the number of squarings. If we are representing our field with respect to a normal basis over $\mathbb{F}_2$, this is justified, since squaring is a cyclic shift and can thus be regarded "free" (recall Example 5.4). Using the polynomial basis, the Itoh–Tsujii algorithm does not look that attractive. However, a recent result [GP98] shows that the Itoh–Tsujii algorithm can be efficiently adopted to polynomial bases as well.

5.3.2 Euclidean Inversion

Another approach to inversion is to apply *Euclid's algorithm* to polynomials over a finite field. Assume the field $\mathbb{F}_{q^k}$ is given in polynomial basis with an irreducible field polynomial $P(x) \in \mathbb{F}_q[x]$ of degree k. As in Section 5.2, write the element to be inverted as a polynomial

$$A(x) = \sum_{i=0}^{k-1} a_i x^i, \quad a_i \in \mathbb{F}_q.$$

We can formulate the problem of inverting $A \neq 0$ as follows: find a polynomial $B(x) \in \mathbb{F}_q[x]$ with $\deg B < k$ such that

$$A(x)B(x) = 1 \pmod{P(x)},$$

or

$$A(x)B(x) + Q(x)P(x) = 1$$

for some $Q(x) \in \mathbb{F}_q[x]$. To solve $B(x)$ from this equation, we apply an analogue of *Euclid's algorithm* from elementary number theory [Nic93], called the *extended Euclid's algorithm.* The following description of this algorithm is taken from [WBV+96]. Denote by $\ell(F)$ the leading coefficient of F.

1. initialize polynomials $B \leftarrow 1$, $C \leftarrow 0$, $F \leftarrow A$ and $G \leftarrow P$.
2. while $\deg F \neq 0$ do
   (a) if $\deg F < \deg G$, then exchange F, G and exchange B, C.
   (b) $j \leftarrow \deg F - \deg G$, $\alpha \leftarrow \ell(F)/\ell(G)$.
   (c) $F \leftarrow F - \alpha x^j G$, $B \leftarrow B - \alpha x^j C$.
3. return $B/\ell(F)$.

In the above algorithm the relations $F = BA \pmod P$ and $G = CA \pmod P$ are maintained. In each iteration the degree of the longer of F and G is decreased. In [WBV+96] a modification of this method that yields better performance in some cases was also presented. The modified version is called the *almost inverse* algorithm and it is based on the work [SOOS95] of Schröppel et al.

5.4 Methods Exploiting Subfield Structure

5.4.1 Hybrid Multiplication

When implementing a model in practice, one has to deal with finite computational resources – that is, limited memory and speed. To come up with an economically feasible implementation, it is often very efficient to equip the model with some kind of parameters that can be adjusted to find a setting that yields optimal performance. Finite fields have one quite natural choice for this kind of parameter.

Definition 5.18. A finite field $\mathbb{F}_{p^k}$ is called *composite*, if $k = nm$ for some prime p and integers $n, m > 1$.

A composite field $\mathbb{F}_{p^{nm}}$ is clearly isomorphic to $\mathbb{F}_{(p^n)^m}$, so it is an m-dimensional vector space over $\mathbb{F}_{p^n}$. To represent the elements of a composite field, one may choose an irreducible polynomial $P(x) \in \mathbb{F}_p[x]$ of degree n and another irreducible polynomial $Q(x) \in \mathbb{F}_{p^n}[x]$ of degree m. Then elements of $\mathbb{F}_{p^k}$ can be viewed as polynomials modulo $Q(x)$,

$$A = \sum_{i=0}^{m-1} a_i x^i,$$

where $a_i \in \mathbb{F}_{p^n}$, i.e., a_i are considered polynomials modulo $P(x)$. Note that the representation for the underlying subfield does not change anything in the polynomial basis multiplication algorithms derived in Section 5.2, since there we only assumed that subfield calculations can be carried out in some way. This open structure allows one to choose a different method for subfield multiplication.

In software, a simple and efficient solution is to represent both fields with respect to a polynomial basis and use table look-up for multiplication in the subfield $\mathbb{F}_{p^n}$, and extend this to $\mathbb{F}_{p^{nm}}$ by applying some variant of the Karatsuba–Ofman algorithm in the ring $\mathbb{F}_{p^n}[x]$. A detailed analysis of this approach is provided in [GP97]. Section 5.5 gives examples of hardware implementations of the hybrid multiplication.

5.4.2 Inversion Using Subfields

Analogously with the trace function defined in the Appendix A, we define another important mapping from $\mathbb{F}_{q^k}$ to $\mathbb{F}_q$.

Definition 5.19. Let α be an element of $\mathbb{F}_{q^k}$. The *norm* of α over $\mathbb{F}_q$ is given by

$$\mathrm{N}(\alpha) = \prod_{i=0}^{k-1} \alpha^{q^i}.$$

Since

$$\frac{q^k - 1}{q - 1} = 1 + q + q^2 + \cdots + q^{k-1},$$

we find out that

$$\mathrm{N}(\alpha) = \alpha^{\sum_{i=0}^{k-1} q^i} = \alpha^{q^k - 1/q - 1}.$$

Thus,

$$(\mathrm{N}(\alpha))^{q-1} = \alpha^{q^k - 1} = 1$$

for every $\alpha \neq 0$, which implies that $\mathrm{N}(\alpha)$ is always an element of the subfield $\mathbb{F}_q$ of $\mathbb{F}_{q^k}$.

The norm function may by viewed as an operator that contracts $\mathbb{F}_{q^k}$ to $\mathbb{F}_q$ without damaging the multiplicative structure. This can be used to reduce the inversion operation to subfield, an idea originally found by Itoh and Tsujii [IT88]. For a nonzero $\alpha \in \mathbb{F}_{q^k}$, write

$$\alpha^{-1} = \left(\mathrm{N}(\alpha)\frac{\alpha}{\mathrm{N}(\alpha)}\right)^{-1} = \mathrm{N}(\alpha)^{-1}\frac{\mathrm{N}(\alpha)}{\alpha} = \mathrm{N}(\alpha)^{-1}\prod_{i=1}^{k-1}\alpha^{q^i}.$$

Thus α^{-1} is obtained in four steps:

- Compute $\beta = \prod_{i=1}^{k-1}\alpha^{q^i}$.
- Compute $\alpha\beta = \mathrm{N}(\alpha)$.
- Invert $\mathrm{N}(\alpha) \in \mathbb{F}_q$.
- Calculate $\beta\,\mathrm{N}(\alpha)^{-1} = \alpha^{-1}$.

We see from the above that inversion in $\mathbb{F}_{q^k}$ can be reduced to $\mathbb{F}_q$ by m multiplications and $m - 1$ exponentiations to the qth power. Optimizing the exponentiation as in Theorem 5.17, we get the following result [IT88].

Theorem 5.20. *Let $\alpha \in \mathbb{F}_{q^k}^*$, where $k = nm$. Then α^{-1} can be computed by performing one inversion in $\mathbb{F}_{q^n}$, $m - 1$ qth power exponentiations in $\mathbb{F}_{q^k}$ and*

$$\lfloor \log(m-1) \rfloor + H_w(m-1)$$

multiplications in $\mathbb{F}_{q^k}$.

The best performance of this algorithm is got by using a normal basis representation, taking qth powers by cyclic shifting. A discussion of adapting this method also to fields given by a polynomial basis is given in [GP97]. When this approach is taken, the subfield inversion can be done for example by table look-ups.

5.5 Some Hardware Implementations for Fields of Characteristic 2

5.5.1 The Mastrovito Bit-Parallel Multiplier

$\mathbb{F}_{2^k}$-multipliers can be divided in two classes, bit-serial and bit-parallel. In a *bit-serial* multiplier, the input is fed into the multiplier one bit at a time, and the time required to compute the output is $O(k)$, measured with respect to the number of clock cycles of the device. A *bit-parallel* multiplier takes all its input simultaneously and the corresponding result comes out in time $O(1)$, which means that its computation time remains constant as k as grows (the size of the device grows instead). We shall next give an overview of a bit-parallel multiplier architecture of Mastrovito [Mas91].

Let $a, b \in \mathbb{F}_{2^k}$ be given in polynomial basis generated by the field polynomial

$$P(x) = x^k + \sum_{i=0}^{k-1} p_i x^i, \quad p_i \in \mathbb{F}_2,$$

$$a = \sum_{i=0}^{k-1} a_i \theta^i, \quad b = \sum_{i=0}^{k-1} b_i \theta^i, \quad a_i, b_i \in \mathbb{F}_2,$$

where $P(\theta) = 0$. As before, denote $c = (c_{k-1}, \ldots, c_0) = ab$. Viewing $\mathbb{F}_{2^k}$ as a vector space spanned by $\{1, \theta, \ldots, \theta^{k-1}\}$, we may regard multiplication by a as a linear map on $(\mathbb{F}_2)^k$. Let $\mathbf{Z} = (z_{i,j})$ be its matrix representation, i.e.,

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{k-1} \end{pmatrix} = \begin{pmatrix} z_{0,0} & z_{0,1} & \cdots & z_{0,k-1} \\ z_{1,0} & z_{1,1} & \cdots & z_{1,k-1} \\ \vdots & \vdots & & \vdots \\ z_{k-1,0} & z_{k-1,1} & \cdots & z_{k-1,k-1} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{k-1} \end{pmatrix}. \tag{5.5}$$

Clearly, the entries of the product matrix $\mathbf{Z}$ are determined by a and $P(x)$. Define

$$u(n) = \begin{cases} 1, & n \geq 0, \\ 0, & n < 0. \end{cases}$$

A lengthy but straightforward calculation [Mas91] gives

$$z_{i,0} = a_i, \quad i = 0, \ldots, k-1, \tag{5.6}$$

$$z_{i,j} = u(i-j)a_{i-j} + \sum_{s=0}^{j-1} r_{j-1-s,i} a_{k-1-s}, \quad i = 0, \ldots, k-1, \quad j = 1, \ldots, k-1, \tag{5.7}$$

where $r_{i,j}$ are the entries of the reduction matrix modulo $P(x)$, computed in Section 5.2.1.

We shall next present an example of a multiplier over $\mathbb{F}_{2^4}$, taken from [Mas91]. For the irreducible polynomial $P(x) = x^4 + x + 1$ over $\mathbb{F}_2$, the reduction matrix is given by

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

Let $a = (a_3, a_2, a_1, a_0)$ be given in the polynomial basis generated by $P(x)$. Then the product matrix corresponding to a and $P(x)$ will be by equation (5.6),

$$\mathbf{Z} = \mathbf{Z}(a, P(x)) = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 + a_3 & a_2 + a_3 & a_1 + a_2 \\ a_2 & a_1 & a_0 + a_3 & a_2 + a_3 \\ a_3 & a_2 & a_1 & a_0 + a_3 \end{pmatrix}.$$

A hardware architecture corresponding to this matrix is given in Figure 5.1. Note that multiplication in $\mathbb{F}_2$ corresponds to the logical AND operation, while $\mathbb{F}_2$-addition may be considered as the logical XOR (exclusive or).

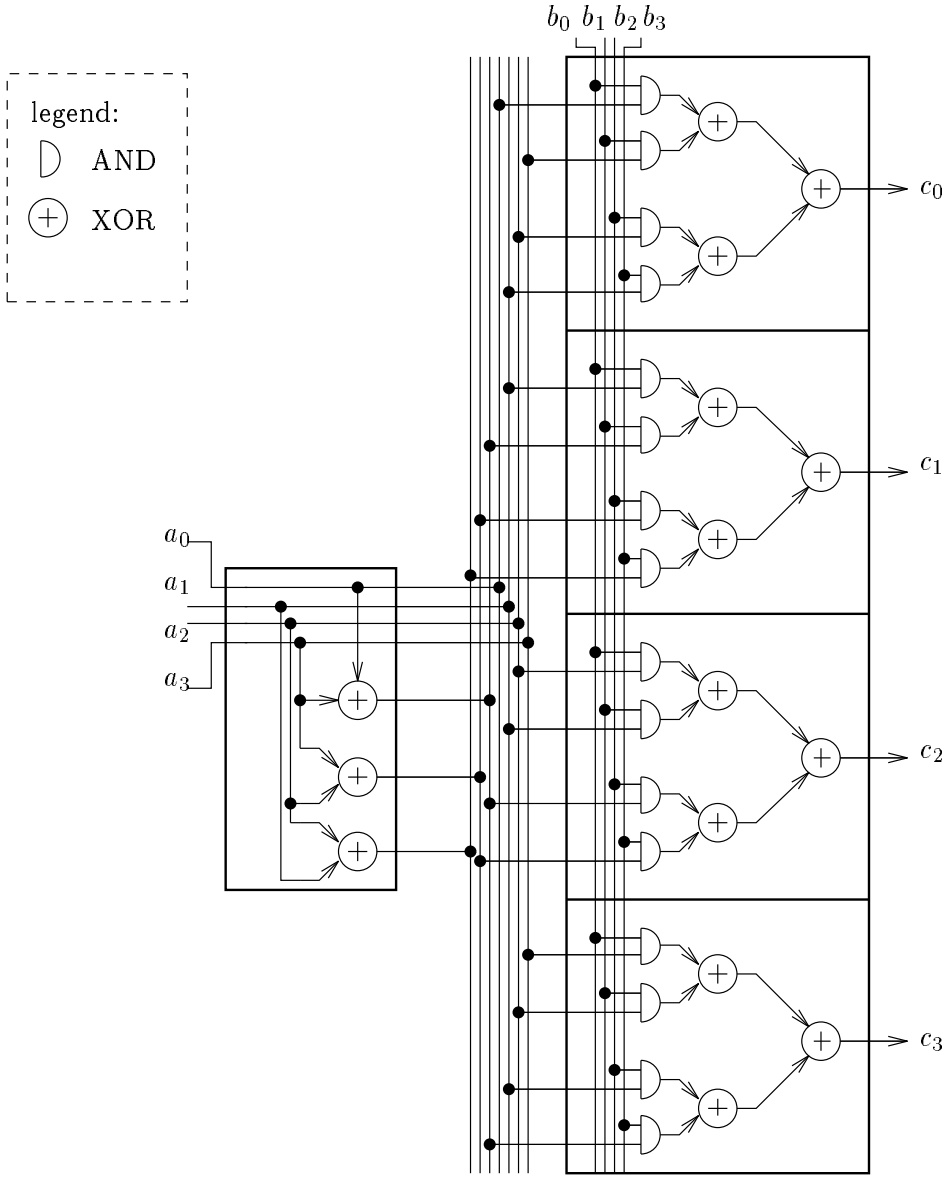

Figure 5.1: A Mastrovito multiplier over $\mathbb{F}_{2^4}$.

From the figure we see that this multiplier consists of 16 AND gates and 15 XOR gates. This count is called the *space complexity* of the multiplier. From equation (5.5)

we see that the number of $\mathbb{F}_2$-multiplications required is always k^2. Let us denote by $\mathcal{C}(k)$ the space complexity of a hardware device. For the $\mathbb{F}_{2^k}$-multiplier, it has been proved in [Mas91] that

$$2k^2 - k + 1 \leq \mathcal{C}(k) \leq 3k^2 - 3k + 1,$$

that is, $\mathcal{C}(k) = O(k^2)$. If the field polynomial is a trinomial of the form

$$P(x) = x^k + x + 1,$$

then the exact space complexity is $\mathcal{C}(k) = 2k^2 - 1$. Some fields for which this kind of irreducible trinomials exists are listed in [LN97].

Note that the Mastrovito multiplier architecture can be easily extended to composite fields. That is, the above multiplication scheme works just as well when the subfield $\mathbb{F}_2$ is replaced by some other subfield $\mathbb{F}_{2^n}$ of $\mathbb{F}_{2^k}$, where n divides k. For example, let $\mathbb{F}_{2^4}$ be given as above by the field polynomial

$$P(x) = x^4 + x + 1,$$

and let ω be a root of $P(x)$ in $\mathbb{F}_{2^4}$. In fact, ω is a generator of $\mathbb{F}_{2^4}$, which can be verified by directly computing ω^i for $i = 1, \ldots, 15$. Now the polynomial

$$Q(y) = y^2 + y + \omega^{14}$$

is irreducible over $\mathbb{F}_{2^4}$ [Paa96]. (Here for convenience, we have represented the coefficients of $Q(y)$ as powers of the generator ω.) Hence we have a representation of $\mathbb{F}_{2^8}$ by affine polynomials over $\mathbb{F}_{2^4}$ modulo $Q(x)$. To find $c = (c_1, c_0) = ab$, with $a = (a_1, a_0), b = (b_1, b_0) \in \mathbb{F}_{2^8}$, write in $\mathbb{F}_{2^4}[y]/Q(y)$:

$$\begin{aligned} A(y)B(y) &= (a_0 + a_1 z) + (b_0 + b_1 z) \\ &= a_0 b_0 + (a_1 b_0 + a_0 b_1) z + a_1 b_1 z^2 \\ &= a_0 b_0 + \omega^{14} a_1 b_1 + [a_0 b_0 + a_0 b_1 + a_1 b_0] z. \end{aligned}$$

This gives us

$$\begin{pmatrix} c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} a_0 + a_1 & a_1 \\ \omega^{14} a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_0 \end{pmatrix},$$

which can be implemented using an architecture similar to that of Figure 5.1.

5.5.2 Two Normal Basis Bit-Serial Multipliers

Let $a = (a_0, \ldots, a_{k-1})$ and $b = (b_0, \ldots, b_{k-1}) \in \mathbb{F}_{2^k}$ be given in normal basis $\{\beta, \beta^2, \ldots, \beta^{2^{k-1}}\}$ of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$. Then the product $c = (c_0, \ldots, c_{k-1}) = ab$ is given by

$$c_t = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \lambda_{i,j}^{(0)} a_{i+t} b_{j+t}, \quad t = 0, 1, \ldots, k-1,$$

where $\lambda_{i,j}^{(0)} \in \mathbb{F}_2$ are as in equation (5.4).

We will now describe two bit-serial hardware architectures for implementing this multiplication algorithm [MO83, AMOV91]. The following example from [AMOV91] will serve as our prototype field.

Example 5.21. The polynomial $P(x) = x^5 + x^2 + 1$ is irreducible over $\mathbb{F}_2$. Let α be a root of $P(x)$ in $\mathbb{F}_{2^5}$, and let $\beta = \alpha^3$. Then $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}\}$ is a normal basis of $\mathbb{F}_{2^5}$ over $\mathbb{F}_2$ [AMOV91]. Performing the required calculations, using the identity $\alpha^5 = \alpha^2 + 1$, we find

$$\Lambda = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix},$$

so the complexity $\mathcal{C}_\beta$ of this normal basis equals 15.

Massey and Omura [MO83] suggested the use a linear feedback shift register to implement the normal basis multiplication. A *linear feedback shift register of length* k is a register that stores k values in its memory locations and performs a cyclic shift (possibly combined with some logical operation) to these values every clock cycle. If each of the k memory locations contains n bits, we say that the linear feedback shift register has *width* n. Let us denote by f the function that calculates the first coordinate c_0 of $c = ab$. For the matrix Λ of Example 5.21, we have

$$\begin{aligned} f(a,b) &= (a_0, a_1, a_2, a_3, a_4)\Lambda(b_0, b_1, b_2, b_3, b_4)^T \\ &= a_0(b_2 + b_3 + b_4) + a_1(b_3 + b_4) + a_2(b_0 + b_3) \\ &\quad + a_3(b_0 + b_1 + b_2 + b_4) + a_4(b_0 + b_1 + b_3 + b_4). \end{aligned}$$

The computation of $f(a,b)$ can be implemented by a similar circuit as that in Figure 5.1. The idea of Massey and Omura was to use this circuit in connection with two linear feedback shift registers to compute the coordinates of c [MO83]. This is illustrated in Figure 5.2. In the beginning the two shift registers are loaded with

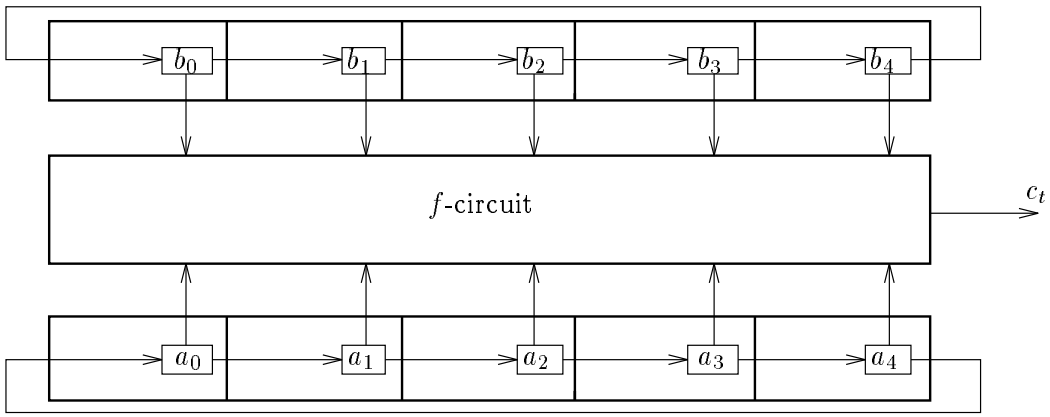

Figure 5.2: The Massey–Omura bit-serial multiplier in $\mathbb{F}_{2^5}$.

$(a_0, a_1, a_2, a_3, a_4)$ and $(b_0, b_1, b_2, b_3, b_4)$, respectively. During the first clock cycle, the circuit for f calculates and outputs c_0. In the beginning of the second clock cycle, the shift registers contain $(a_4, a_0, a_1, a_2, a_3)$ and $(b_4, b_0, b_1, b_2, b_3)$, and thus the f-circuit will output c_4. Proceeding this way, all the coordinates of c are produced in 5 clock cycles.

The Massey–Omura multiplier is easy to implement when the field size is small. However, taking k larger that 150, the f-circuit becomes very complex, due to its irregular nature. To address this problem, Agnew, Mullin, Onyszchuk and Vanstone [AMOV91] suggested another way to implement the normal basis multiplication. They use three linear feedback shift registers, where the lowermost and uppermost registers contain the coordinates of a and b, respectively, and the the one in the middle will hold the coordinates of c in the end of the computation. This way they get a regular structure – one that consists of several identical blocks. This kind of regularity makes the architecture scalable to (cryptographically interesting) fields $\mathbb{F}_{2^k}$ with $k \geq 150$. A detailed analysis and description of the suggested architecture is given in [AMOV91]. For our prototype case, Example 5.21, the corresponding circuit is represented in Figure 5.3. The shift register in the middle is initialized with zeros, and the other registers are loaded with the coordinates of a and b, as with the Massey–Omura multiplier. After $k = 5$ clock cycles, the shift register in the middle will contain the coordinates of c. The reader may verify the correct functionality of the circuit as an exercise.

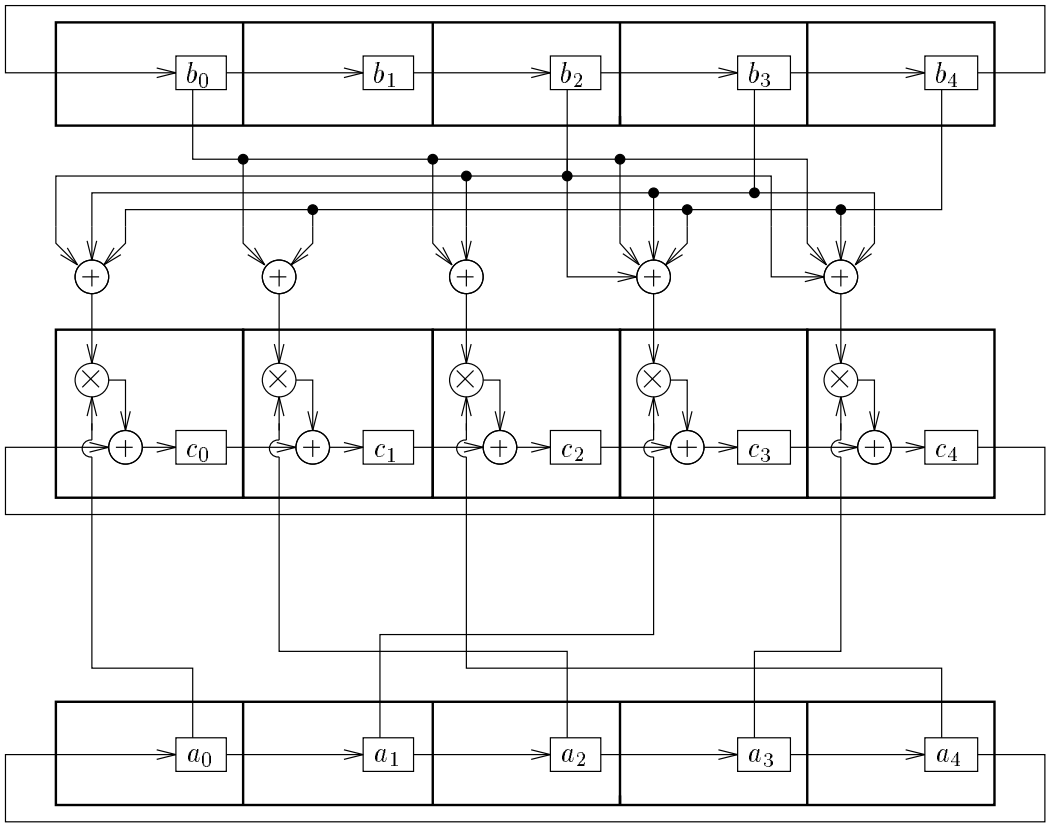

Figure 5.3: A bit-serial multiplier in $\mathbb{F}_{2^5}$.

5.5.3 A Hybrid Multiplier

The bit-parallel multiplier for $\mathbb{F}_{2^k}$ presented in Section 5.5.1 has space complexity $O(k^2)$. This means that these kind of multipliers are not applicable for fields $\mathbb{F}_{2^k}$ with $k \geq 150$, which are used in cryptography. On the other hand, bit-serial multipliers

have $O(k)$ time complexity, which is undesirable in some applications. We will now describe a hybrid multiplication architecture for composite fields $\mathbb{F}_{2^k}$, $k = nm$, which is capable of exploring the time–space trade-off in a flexible manner. This original idea of Mastrovito [Mas91] has been further developed and analyzed by Paar et al. in [PSR97].

Let $\mathbb{F}_{2^n}$ be given by the field polynomial

$$P(x) = x^n + \sum_{i=0}^{n-1} p_i x^i, \quad p_i \in \mathbb{F}_2,$$

and $Q(y)$ be an irreducible polynomial over $\mathbb{F}_{2^n}$ of degree m,

$$Q(y) = y^m + \sum_{i=0}^{n-1} q_i x^i, \quad q_i \in \mathbb{F}_{2^m}.$$

Then we have a representation for the composite field $\mathbb{F}_{2^k}, k = nm$, by polynomials in $\mathbb{F}_{2^n}[y]$ modulo $Q(y)$, with degree less than m. Operating over the field $\mathbb{F}_{2^n}$, we may multiply polynomials modulo $Q(y)$ using a linear feedback shift register of length m and width n according to Figure 5.4. The circuit is initialized by load-

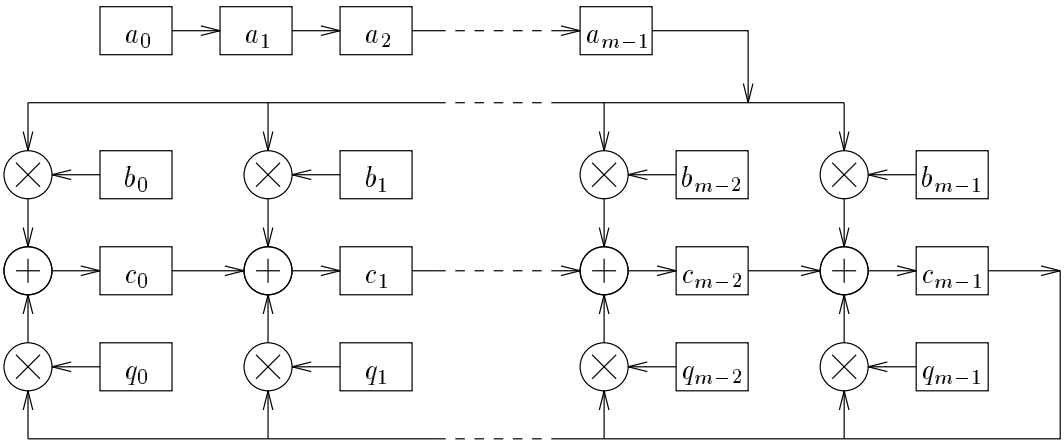

Figure 5.4: A hybrid multiplier in $\mathbb{F}_{2^{nm}}$.

ing the values $b_0, \ldots, b_{m-1}$ and $q_0, \ldots, q_{m-1}$ into their corresponding registers and resetting the memory locations of the shift register in the middle to zero. The computation starts with feeding a_{m-1} to the circuit. All the lines in the figure are n bits wide, and all computation is carried out in $\mathbb{F}_{2^n}$ (for example, the architecture described in Section 5.5.1 can be used for $\mathbb{F}_{2^n}$-multiplication). After m clock cycles, the linear feedback shift register contains the values $c_0, \ldots, c_{m-1}$. Note that if $n = 1$, the structure degenerates into one of the classical bit-serial architectures for $\mathbb{F}_{2^m}$-multiplication, which have been widely used in public key cryptosystems.

To compare this hybrid structure with the classical bit-serial case ($n = 1$), we may use Figure 5.4 to see the space complexity of the bit-serial multiplier to be $\mathcal{C}_{bs}(k) = 3k$. The corresponding time complexity is $\mathcal{T}_{bs}(k) = k$. To get an estimate for the hybrid architecture, we assume that the space complexity of the subfield multiplier is $2n^2 - 1$ [Mas91]. Optimizing the hybrid structure by choosing suitable field polynomials [PSR97], we get for the total space complexity, $\mathcal{C}_{hyb}(k) = (2n + 1 - 1/n)k$,

and for the corresponding time complexity, $\mathcal{T}_{hyb}(k) = k/n$. For example,this means $\mathcal{C}_{hyb} \sim 16\,\mathcal{C}_{bs}$ and $\mathcal{T}_{hyb} \sim \mathcal{T}_{bs}/8$ with $n = 8$. Hence, by choosing a suitable n, one may exploit the time–space trade-off paradigm in a very flexible way.

Chapter 6

Implementing Elliptic Curve Cryptosystems

6.1 General Aspects

When implementing a cryptosystem, different types of cryptographic techniques can be categorized abstractly as follows [IEE98]:

- *Protocols* are sequences of operations performed by multiple parties to achieve some security goal. They may be used to develop secure applications if implemented correctly.
- *Schemes* are building blocks for protocols. They combine cryptographic primitives with some additional techniques, such as message encoding algorithms and hash functions to provide security in the complexity-theoretic sense.
- *Primitives* are the basic mathematical operations from the cryptographical point of view. The signature generation and verification procedures (see Section 2.4) implemented using elliptic curve groups are examples of primitives.

Developing public key cryptosystems involves many (nontrivial) decisions, such as choosing a *public key infrastructure*, i.e., deciding how public keys are distributed and agreed. However, since these are vast subjects as themselves and thus beyond the scope of this thesis, we'll restrict our attention only on the implementation of primitives.

Primitives are implemented using algorithms. In the context of elliptic curves, we may consider two classes:

1. *Low-level algorithms* are used to implement the arithmetical operations in the finite field.
2. *High-level algorithms* use finite field algorithms to implement the group operation on the elliptic curve and to generate new elliptic curve groups.

Implementations of algorithms for finite fields were already discussed in Chapter 5. In the next sections we represent some efficient implementations of high-level algorithms.

Another issue is to choose the type of the field on which our elliptic curves will be treated. Note that, once the underlying field is fixed, we still have a great variety of abelian groups to choose from for our cryptosystem. Practically there are two choices, determined by the characteristic of the field. One choice is to use a field $\mathbb{F}_p$, where p is a prime of size $\sim 10^{60}$. Then the field arithmetic will reduce to integer arithmetic modulo p. Another feasible alternative is to use a field of type $\mathbb{F}_{p^m}$, with p a small prime. Then the choice $p = 2$ appears to be the most attractive one, since then the field elements can be represented by m-bit strings. Further, adding two elements in $\mathbb{F}_{2^m}$ can be done bitwise, which is very efficient in practice. Also, there exist many efficient hardware implementations for arithmetic in $\mathbb{F}_{2^m}$, as was seen in Chapter 5. For these reasons, we will focus on the binary non-supersingular elliptic curves in what follows.

6.2 Group Generation

Once the field is fixed, we need an elliptic curve defined on it, and a generator point $T \in E(\mathbb{F}_q)$. It is preferable to choose a generator with prime order r such that r^2 won't divide $\#E(\mathbb{F}_q)$. Then we see using Theorem 3.18 that either $E[r] \simeq \mathbb{Z}_r$ (if $\gcd(r, q) > 1$), or $E[r] \simeq \mathbb{Z}_r \oplus \mathbb{Z}_r$ (the case $\gcd(r, q) = 1$). In the latter case, our condition guarantees that $E[r]$ is not a subgroup of $E(\mathbb{F}_q)$. From these observations we conclude that in both cases we have $E[r] \cap E(\mathbb{F}_q) = \langle T \rangle$. Moreover, every element in $\langle T \rangle$ except $\mathcal{O}$ will have order r. Since $\langle T \rangle$ is a subgroup of $E(\mathbb{F}_q)$, and so r divides $\#E(\mathbb{F}_q)$, we need to construct a curve whose order is divisible by the large prime r.

Given an elliptic curve $E/\mathbb{F}_q$ with Weierstrass equation

$$y^2 + xy = x^3 + ax^2 + b, \quad a, b \in \mathbb{F}_{2^m}, \quad b \neq 0, \tag{6.1}$$

how do we find out its order $\#E(\mathbb{F}_q)$? One simple way is to first fix $x \neq 0$ and look for solutions y to the above equation. As Lemma 3.23 guarantees, equation (6.1) has two solutions $y \in \mathbb{F}_{2^m}$, if and only if

$$\mathrm{Tr}(x + a + x^{-2}b) = 0,$$

and no solutions otherwise. Thus we get the total number of points in $E(\mathbb{F}_q)$ (taking points $\mathcal{O}$ and $(0, \sqrt{b})$ into account) as

$$\begin{aligned} \#E(\mathbb{F}_{2^m}) &= 2 + \sum_{x \in \mathbb{F}_{2^m}^*} \left(1 + (-1)^{\mathrm{Tr}(x+a+x^{-2}b)}\right) \\ &= q + 1 + (-1)^{\mathrm{Tr}(a)} \sum_{x \in \mathbb{F}_{2^m}^*} (-1)^{\mathrm{Tr}(x+x^{-2}b)}. \end{aligned}$$

However, since the above sum goes through $2^m - 1$ elements, this simple algorithm is exponential in m and hence not feasible when dealing with large fields. To construct an elliptic curve with almost prime order over a large field, there are in principle three approaches to choose from [IEE98]:

1. If $q = 2^m$, where m is divisible by a small integer d, then select a curve over $\mathbb{F}_{2^d}$ and compute its order by the above procedure. Then consider the extension of this curve over $\mathbb{F}_{2^m}$ and use Theorem 3.14 to find $\#E(\mathbb{F}_{2^m})$.
2. Fix an *a priori* suitable order and construct a curve of that order.
3. Select an elliptic curve at random, and use a sophisticated algorithm to compute $\#E(\mathbb{F}_q)$. Repeat this process until an appropriate order is found.

The first method can be easily implemented using a *Lucas sequence*, see [IEE98]. Constructing a curve according to the second alternative is not that simple. One way to accomplish this is to use *complex multiplication* [Mor91, LZ94]. It is based on constructing elliptic curves whose orders share a certain number theoretic property. A description how complex multiplication can be implemented is also available in [IEE98].

The third alternative is the most difficult one to implement efficiently in practice. It is based on a polynomial-time algorithm invented in 1985 by R. Schoof [Sch85]. In its original form it was very slow, having running time $O(\log^8 q)$. Recently, many improvements and modifications have been made [Ler97], resulting in more feasible running times $O(\log^6 q)$. These techniques use deep methods from algebraic geometry, e.g., computation of isogenies on elliptic curves [Sil86].

From the implementation point of view, the first alternative looks most attractive in its simplicity. However, that method restricts the choice of the parameters a, b to the subfield $\mathbb{F}_{2^d}$ of $\mathbb{F}_{2^m}$. This means that elliptic curves generated using Lucas sequences share a special property, which *might* make them vulnerable to a cryptographic attack. Same kind of observations hold for complex multiplication methods. Thus, to avoid all potential risks of the described kind, the most secure choice of curve generation algorithm seems to be the most general one.

When we have a curve with order $\#E(\mathbb{F}_q) = vr$, with r a large prime, this prime should still be checked to satisfy the MOV condition on Page 31. If this check is passed, we then proceed to find a point T having order r:

1. Generate a random point $P \in E(\mathbb{F}_q) \setminus \{\mathcal{O}\}$. (This is easy with help of Lemma 3.23, see [IEE98].)
2. Set $T \leftarrow vP$.
3. If $T = \mathcal{O}$ go back to Step 1.
4. Output T.

The above procedure works, since $rT = (vr)P = \mathcal{O}$, which implies that the order of T divides r. As r is prime and $T \neq \mathcal{O}$, the order of the point T is r.

6.3 Group Operation

6.3.1 Adding and Doubling Points

Let E be a non-supersingular elliptic curve over $\mathbb{F}_{2^m}$, given by equation

$$y^2 + xy = x^3 + ax^2 + b, \quad a, b \in \mathbb{F}_{2^m}, \quad b \neq 0,$$

with $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ being two points in $E(\mathbb{F}_{2^m}) \setminus \{\mathcal{O}\}$. For convenience, we rewrite the addition formulae (3.15) – (3.18) derived in Section 3.4. For point addition with $P \neq Q$ we have $P + Q = (x_3, y_3)$, where

$$\begin{aligned} x_3 &= \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a, \\ y_3 &= \left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1. \end{aligned} \tag{6.2}$$

Doubling the point is done by $2P = (x_3, y_3)$, with

$$\begin{aligned} x_3 &= x_1^2 + \frac{b}{x_1^2}, \\ y_3 &= x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3. \end{aligned} \tag{6.3}$$

Now we can count the number of field operations required to perform these group operations. Since addition of elements in $\mathbb{F}_{2^m}$ takes very short time compared to multiplication and inversion, we count only the latter two. From the above equations we see that the point addition takes 2 multiplications, 1 squaring and 1 inversion in the field $\mathbb{F}_{2^m}$. Similarly, doubling can be accomplished by 3 multiplications, 2 squarings and 1 inversion. In Chapter 5 we saw that inverting $\mathbb{F}_{2^m}$-elements is much more costly than multiplication. So it seems that field inversion becomes a bottleneck when performing operations in the group $E(\mathbb{F}_{2^m})$.

We can avoid the costly inversion present in equations (6.2) and (6.3) by resorting to projective coordinates [Men93]. Assume $P \neq Q$ are two nonzero rational points on a non-supersingular curve $E/\mathbb{F}_{2^m}$ given in projective coordinates, $P = (x_1 : y_1 : z_1), Q = (x_2 : y_2 : z_2)$; recall Definition 3.2 in Section 3.1. To compute the sum $R = P + Q$, we write $P = (x_1/z_1 : y_1/z_1 : 1)$ and $Q = (x_2/z_2 : y_2/z_2 : 1)$. Thus $(x_1/z_1, y_1/z_1)$ and $(x_2/z_2, y_2/z_2)$ correspond to the affine coordinates of P and Q, respectively. We may then apply formula (6.2) to compute $R = (x_3', y_3')$. We get

$$\begin{aligned} x_3' &= \frac{B^2}{A^2} + \frac{B}{A} + \frac{A}{z_1 z_2} + a, \\ y_3' &= \frac{B}{A}(x_1/z_1 + x_3) + x_3 + y_1/z_1, \end{aligned}$$

where $A = x_1 z_2 + x_2 z_1$ and $B = y_1 x_2 + y_2 z_1$.

Viewing (x_3', y_3') again in projective coordinates, we have $R = (x_3' : y_3' : z_3')$ with $z_3' = 1$. Now we take advantage of the extra "space" we have available in the projective plane $\mathbb{P}^2(\mathbb{F}_{2^m})$. That is, we choose a point (x_3, y_3, z_3) from $(\mathbb{F}_{2^m})^3$ to represent the

line $R \subseteq (\mathbb{F}_{2^m})^3$ such that the denominators of x_3' and y_3' will be cancelled out. Examining the above equations, we see that letting $z_3 = z_1 z_2 A^3$ is a good choice. Then a simple calculation yields

$$\begin{aligned} x_3 &= z_1 z_2 A^3 x_3' = AD, \\ y_3 &= z_1 z_2 A^3 y_3' = CD + A^2(y_1 z_2 A + x_1 z_2 B), \\ z_3 &= z_1 z_2 A^3, \end{aligned}$$

where $C = A + B$ and $D = z_1 z_2 BC + A^2(A + z_1 z_2 a)$. A similar calculation for computing $2P = (x_3 : y_3 : z_3)$ can easily be done, resulting in

$$\begin{aligned} x_3 &= EF, \\ y_3 &= x_1^4 E + F(x_1^2 + y_1 z_1 + E), \\ z_3 &= E^3, \end{aligned}$$

with $E = x_1 z_1$ and $F = x_1^4 + z_1^4 b$. This way, we can perform both addition and doubling in the group $E(\mathbb{F}_{2^m})$ without having to deal with inverting in $\mathbb{F}_{2^m}$. This looks a little bit too easy and in fact it is. Namely, to check if two points given in projective coordinates are equal means checking if they are scalar multiples of each other, which obviously requires inversion in $\mathbb{F}_{2^m}$.

However, in practice we only need to compute kT for $T = (x_T, y_T) \in E(\mathbb{F}_{2^m})$. Then we may proceed by considering the projective representation of T, $T = (x_T : y_T : 1)$, and computing kT by repeatedly adding and doubling T in projective coordinates using some method of Section 6.3.2. Then we transform the result $(x_{kT} : y_{kT} : z_{kT})$ back to affine representation by inverting z_{kT}. This method thus requires one inversion in $\mathbb{F}_{2^m}$ for the *whole* procedure of computing kT. An additional advantage is that we may assume $z_2 = 1$ by suitably choosing the order of summation, see Section 6.3.2.

The cost of avoiding the inverse operation is paid in terms of increased number of field multiplications. By the above formulas, computing $P + Q$ (assuming $z_2 = 1$) needs 13 multiplications and 1 squaring in $\mathbb{F}_{2^m}$, and for doubling the point P, 7 multiplications and 5 squarings are enough.

6.3.2 Scalar Multiplication

Binary Algorithm

The fundamental mathematical operation used in elliptic curve cryptosystems is *scalar multiplication*, i.e., computing kT for $T \in E(\mathbb{F}_q)$ and $0 < k < r$ (the word *exponentiation* from the terminology of multiplicative groups is also often used). As already seen in Section 4.1, this can be done by writing k in binary form as

$$k = \sum_{i=0}^{t-1} k_i 2^i, \quad k_i \in \{0, 1\},$$

and performing the following steps:

1. Set $R \leftarrow T$.

2. If $k_0 = 1$, set $S \leftarrow T$, else $S \leftarrow \mathcal{O}$
3. For $i = 1$ to $t - 1$ do
   (a) set $R \leftarrow 2R$.
   (b) if $k_i = 1$, set $S \leftarrow S + R$.
4. Output S.

Hence this method requires $t - 1 = \lfloor k \rfloor - 1$ doublings and $H_w(k) - k_0$ additions. On average, this method requires $3r/2$ group operations. Many modifications and extensions to this method have been invented. One of them is the 2^m*-ary method* [Knu81], where k is represented with respect to the base $b = 2^m$. It works in principle the same way as the binary algorithm, except that precomputation of the values $T_i = iT$ for $i = 1, \ldots, b-1$ is required. The 2^m-ary method has further refinements, reducing the required amount of precomputation [MvOV97]. Guajardo [Gua97] suggested a speed-up to the 2^m-ary method by working out direct formulas for $4T, 8T, \ldots, 2^{m-1}T$, thus reducing the required amount of field inversions when using affine coordinates. Other modifications to the binary algorithm utilizing precomputation include the *sliding window method*, see [Koc95].

Addition–Subtraction Chains

The algorithms described above were originally developed for modular integer arithmetic [Knu81]. After all, there appears to be no difference in elliptic curve scalar multiplication and exponentiating integers using multiplication modulo n as the elementary operation. However, there is one exception. While computing inverses of integers modulo n is much harder than the direct multiplication, group inverses of elliptic curve points are available practically at no cost. This suggests a new kind of approach to implement exponentiation on an elliptic curve.

One way to formally discuss different exponentiation methods is to introduce the following concept.

Definition 6.1. An *addition chain of length* r for a positive integer k is an $(r+1)$-tuple $(k_0, \ldots, k_r)$ of positive integers such that $k_0 = 1$, $k_r = k$ and for every $1 \leq i \leq r$ there exist integers $a(i), b(i)$ such that

$$0 \leq a(i), b(i) < i, \quad k_i = k_{a(i)} + k_{b(i)}. \tag{6.4}$$

Thus, given an addition chain for k of length r, we may compute kT with r additions.

Example 6.2. For $k = 15$, we may write $k_0 = 1, k_1 = k_0 + k_0 = 2$ and

$$15 = 8 + 7 = (4 + 4) + (4 + 3) = [(k_1 + k_1) + (k_1 + k_1)] + [(k_1 + k_1) + (k_1 + k_0)].$$

Hence we see that $(1, 2, 3, 4, 7, 8, 15)$ is an addition chain of length 6 for 15. This is in accordance with the count we got for the binary algorithm, $\lfloor \log k \rfloor - 1 + H_w(k) - k_0 = 6$.

However, on elliptic curves it looks more efficient to compute $15T = 2^4T - T$. This is the idea of an *addition–subtraction chain.*

Definition 6.3. An *addition subtraction chain* for a positive integer k is defined the same way as in Definition 6.1, replacing condition (6.4) with

$$0 \leq a(i), b(i) \leq i, \quad k_i = \pm k_{a(i)} \pm k_{b(i)}.$$

Now, $(k_0, \ldots, k_5) = (1, 2, 4, 8, 16, 15)$ is an addition–subtraction chain for $k = 15$, so 5 operations is enough for finding out $15T$. This idea, applied to elliptic curves by F. Morain can be described as follows: replace k with $k = k_+ - k_-$ so that computing k_+T and k_-T requires less operations than the evaluation of kT. Morain describes two algorithms for doing this, yielding average speed-ups of 8.33 % and 11.11 % respectively, compared to the ordinary binary algorithm [MO90]. Further improvements to these methods have been suggested recently; they are summarized in [Gor98].

6.4 Existing Implementations

6.4.1 Implementations in Software

There exist four software implementations [SOOS95, Bea96, WBV+96, Gua97] of cryptosystems using non-supersingular elliptic curves over $\mathbb{F}_{2^m}$ with $m \geq 150$ reported in the academic literature known by the author. We proceed to give a brief discussion on the implementation of their arithmetical operations.

The earliest implementation of a cryptosystem using non-supersingular elliptic curves over $\mathbb{F}_{2^m}$ with $m \geq 150$, was made by Schröppel et al. in 1995 [SOOS95]. They used a polynomial basis over the field $\mathbb{F}_{2^{155}}$ with $P(x) = x^{155} + x^{62} + 1$ as the field polynomial. The field multiplication was implemented using the two-step algorithm presented in Section 5.2.1. For squaring, a linear map described also in Section 5.2.1 was used. Inversion was accomplished by the almost inverse algorithm [SOOS95]. The elliptic curve group operations were implemented using affine coordinates and a version of the 2^m-ary method, see Page 59. The performance of the system was optimized by letting $a = 0$ in the defining equation (6.1) of the curve. This way the scalar multiplication in $E(\mathbb{F}_{2^{155}})$ took 7.8 ms on a 175-MHz DEC Alpha 3000

In 1996, two implementations for elliptic curve arithmetic over $\mathbb{F}_{2^{176}}$ were independently developed by Beauregard [Bea96] and De Win et al. [WBV+96]. Both implementations use a composite field architecture, representing the field elements as polynomials over $\mathbb{F}_{2^{16}}$ modulo a field polynomial of degree 11. In both works, the field multiplication and squaring were done similarly as in [SOOS95], using table look-ups to perform the computations in the subfield. In addition, both of the implementations used the binary method in affine coordinates for the elliptic curve scalar multiplication. The only considerable difference between the two implementations was the computation of inverses. In [WBV+96], extended Euclidean and almost inverse algorithms (see Page 45) were used, while in [Bea96], the algorithm of Section 5.3.1 was optimized for the polynomial basis representation to accomplish the task. The corresponding time estimates for performing the scalar multiplication are 72 ms for [WBV+96] (on a 133-MHz Pentium), and 123 ms for [Bea96] (175-MHz DEC Alpha 3000).

In 1997, an implementation based on the architecture of [Bea96] was presented in [Gua97], introducing three improvements to the previous work. First, the ordinary polynomial multiplication was replaced by the Karatsuba–Ofman algorithm described in Section 5.2.1. For the field inversion, the subfield reduction in Section 5.4.2 was utilized. In addition, the scalar multiplication was made efficient by using the 2^m-ary method with direct formulae for computing $4T, 8T, \ldots$ [Gua97]. Using a DEC Alpha 3000, the scalar multiplication took 68 ms. The timing results of these implementations are summarized in Table 6.1. Note that these reported timings are not directly comparable (since for example in [WBV+96], a different processor was used in computation).

implementation	group	timing
[SOOS95]	$E(\mathbb{F}_{2^{155}})$ with $a = 0$	7.8 ms
[Bea96]	$E(\mathbb{F}_{2^{176}})$	123 ms
[WBV+96]	$E(\mathbb{F}_{2^{176}})$	72 ms
[Gua97]	$E(\mathbb{F}_{2^{176}})$	68 ms

Table 6.1: Time required to perform elliptic curve scalar multiplication.

6.4.2 Two Hardware Architectures

There have been very few hardware implementations of elliptic curves reported in the literature. In this section, we'll discuss two of them.

A $\mathbb{F}_{2^{155}}$-Coprocessor for Elliptic Curve Arithmetic

In [ABMV93], a VLSI (very large scale integrated circuit) implementation of an arithmetic processor for the field $\mathbb{F}_{2^{155}}$ is described, and its suitability to perform elliptic curve arithmetic is discussed. The goal of the design was to achieve good performance for elliptic curve scalar multiplication under strict area requirements in order to allow the coprocessor to be integrated on a smart card. Similar considerations were presented in [MV90, MV93].

The field is represented using an optimal normal basis over $\mathbb{F}_{2^{155}}$ and multiplication is implemented as in Section 5.5.2. Multiplication requires three registers to store the multiplicands and the result. Every register consists of five independent 32-bit storage blocks, so every block contains one extra bit that is not used. Since the multiplication of elements needs lot of interconnection of blocks, 32-bit wide data paths are used. Inversion is done by a variant of the Itoh–Tsujii algorithm. The implementation runs at 40 MHz, which means a timing of 3.9 μs for multiplication in $\mathbb{F}_{2^{155}}$. The architecture consists of 11 000 gates, which is roughly 15 % of the area of currently available smart card chips.

In [ABMV93], the authors further discuss how elliptic curve scalar multiplication could be done using this coprocessor. They suggest two alternatives. One is to use a supersingular curve over $\mathbb{F}_{2^{310}}$, where doubling on the curve can be done without field inversion. The field $\mathbb{F}_{2^{310}}$ can be viewed as an extension of $\mathbb{F}_{2^{155}}$, and so arithmetic in $\mathbb{F}_{2^{310}}$ is easily implemented using the $\mathbb{F}_{2^{155}}$-processor. For these elliptic curves, a

throughput rate of 444 kilobits per second was estimated. (Note that in this case, supersingular curves can be used for secure public key schemes, since now a MOV reduction leads us to a DLP in a finite field of size $\sim 2^{1000}$.) If shorter key lengths are desired, another alternative is to use a non-supersingular elliptic curve over $\mathbb{F}_{2^{155}}$ and projective coordinates yielding an estimated throughput rate of 60 kbits/s (the timing for a single scalar multiplication is roughly 5.2 ms) using the coprocessor architecture. This VLSI design could be used to implement an elliptic curve crypto engine on a smart card by using the smart card's microprocessor to control the coprocessor.

An Elliptic Curve Cryptosystem on Reconfigurable Hardware

In a recent work by M. Rosner [Ros98], an architecture to perform elliptic curve scalar multiplication on reconfigurable hardware was developed. The system was based on the use of field programmable gate arrays (FPGAs). This choice allows the user of this crypto engine to change all parameters of the cryptosystems, *including* those determining the underlying field and its representation. Thus FPGAs combine the flexibility of software solutions with the speed of hardware circuits.

The field arithmetic in [Ros98] was constructed using composite fields of type $\mathbb{F}_{2^{8m}}$, considering finite fields with $\mathbb{F}_{2^8}$ as a subfield. For the subfield arithmetic, two alternative approaches were suggested, a Mastrovito multiplier for $\mathbb{F}_{2^8}$ and a composite type multiplier using $\mathbb{F}_{2^4}$ as a subfield. Both of which were described in Section 5.5.1. These parallel subfield multipliers were used as building blocks to implement the hybrid multiplier of Section 5.5.3. For squaring, no optimizations were used.

The group operations were done using projective coordinates and binary exponentiation on a non-supersingular elliptic curve. Using automatic synthesis tools an estimate of 4.5 ms for scalar multiplication in $E(\mathbb{F}_{2^{168}})$ (with 38 kbits/s throughput rate) was made. The area estimate for the corresponding architecture was roughly 50 000 gates, which won't fit to the area constraints for current smart cards.

Chapter 7

Discussion

7.1 Conclusions

In this thesis it was shown that the use of elliptic curves in cryptography provides a noteworthy alternative to traditional public key cryptosystems. This goal was fulfilled in three steps:

- A mathematical frame of reference for discussing the security and implementation aspects of elliptic curve cryptosystems was constructed by giving a comprehensive introduction to finite field arithmetic and elliptic curves over finite fields.
- The latest state-of-the-art algorithms for solving the elliptic curve discrete logarithm problem were reviewed. This way, affirmative evidence was found for believing that elliptic curves, when carefully chosen, may provide strong security with relatively small key lengths.
- Several different approaches for implementing arithmetic of elliptic curves were compared. Reviewing the latest software and hardware implementations, it was seen that elliptic curve cryptosystems can be efficiently implemented in practice.

Implementing elliptic curves involves many choices. The most fundamental decision concerns the field characteristic, which determines the exact form of the equation of the curve. If special hardware equipment is used for the field arithmetic, then characteristic 2 appears to be the natural choice due to the fact that the existing hardware technology is almost completely based on two-value logic. In software, where general purpose microprocessors often have good support for modular integer arithmetic, using fields of large characteristic might be a feasible solution. Many recent software implementations are made for elliptic curves over fields $\mathbb{F}_p$, where p is a large prime. An interesting alternative could be trying to implement arithmetic in fields of characteristic 127 or $2^{31} - 1$; two primes that could neatly fit into existing microprocessor technology.

If a small characteristic for the field is chosen, several feasible alternatives for representing the field elements are available. The most dominant choice is to decide

whether the ground field is represented with respect to a polynomial or a normal basis. Multiplication in polynomial basis can be done very efficiently, e.g., by using the Karatsuba–Ofman algorithm. However, this requires storage of temporary results, i.e., it is most efficiently implementable in software. Using the extended Euclidean algorithm, inversion in polynomial basis can also be performed very quickly. This also requires a general purpose microprocessor and software to control it.

In hardware, it was seen that the use of an optimal normal basis yields the best performance for field multiplication. However, using a normal basis means that efficient algorithms based on polynomial arithmetic, such as the extended Euclidean algorithm for inversion, are not available. A new method of using a so-called *palindromic representation* suggested by Blake et al. can combine the good features of both the polynomial and normal basis representations. In [BRS98], they established an analogy between a special set of polynomials modulo $x^{2k-1}-1$ and type II optimal normal bases over $\mathbb{F}_{2^k}$. This representation could allow great flexibility for mixed software/hardware implementations of elliptic curve arithmetic.

Besides the speed of the implementation, space requirement aspects deserve attention. Since the fields used in cryptography are very large, the required amount of memory and registers is a crucial factor in practice. From this point of view, the use of three coordinates for representing a one-dimensional curve in the projective plane seems a waste of memory. For transmitting an elliptic curve point, it is actually enough to transfer just the x-coordinate with one extra bit, which can be used to recover the correct y-coordinate in question [Ser98]. If space consumption is the most important factor to be optimized, then the most feasible solution is to use bit-serial hardware circuits for field arithmetic. Respectively, in the opposite situation with practically unlimited memory resources, a software implementation using a lot of precomputation for elliptic curve points seems to be the most efficient solution. Hybrid architectures exploiting the subfield structure of composite fields may offer a feasible alternative between the two extreme approaches in some environments.

7.2 Further Research

During the research that was carried out in preparing this thesis, many interesting topics about implementing elliptic curve arithmetic that came up had to be left out in order to keep the size of the presentation within reasonable bounds. Some of these ideas which could make interesting subjects for further study are listed below:

- Utilizing duality in finite fields could yield more efficient realizations for field operations. Dual bases over finite fields have been intensively studied from the pure mathematical point of view. However, reported applications using duality in fields of cryptographically interesting size seem to be rare.
- Using two (or even three) coordinates for performing computations on a one-dimensional elliptic curve seems inefficient. Trying to find better coordinate systems in the projective plane specially suitable for elliptic curves might result in improved space–time performance.
- Investigating opportunities for combining the advantages of both the polynomial and optimal normal bases as in [BRS98] could produce very practical

representations for finite fields. This would involve a thorough study of polynomials over finite fields.

- The formulae for computing the elliptic curve group operation presented in this thesis are directly derived from the classical geometric representation of the curve. Alternative ways for computing scalar multiples of elliptic curve points could perhaps be deduced by using more advanced methods from algebraic geometry, or by embedding elliptic curves into some mathematical object with a richer algebraic structure.

Appendix A

Field Theoretic Background

The basic definitions and facts about finite fields that are needed in the thesis are presented in this appendix. For most parts, we will follow the representation given in the book of Lidl and Niederreiter [LN97].

A.1 Basic Definitions

We begin by reviewing the concept of a group.

Definition A.1. A *group* is a set G with a binary operation $\circ$ on G having the properties:

- *Associativity*: for every $a, b, c \in G$, $a \circ (b \circ c) = (a \circ b) \circ c$.
- *Existence of identity*: there is an element $e \in G$ such that $a \circ e = e \circ a = a$, for all $a \in G$.
- *Existence of inverses*: for each $a \in G$, there exists an element $a^{-1} \in G$ such that $a \circ a^{-1} = a^{-1} \circ a = e$.

The group G is called *abelian*, if it also satisfies:

- *Commutativity*: for all $a, b \in G$, $a \circ b = b \circ a$.

If G is finite, then the number of its elements is called the *order* of G, labeled $\#G$. Two groups G_1 and G_2 are said to be *isomorphic*, denoted $G_1 \simeq G_2$, if there exists a bijection $\phi : G_1 \to G_2$ such that $\phi(a \circ b) = \phi(a) \circ \phi(b)$, for all $a, b \in G_1$.

It is easy to verify, that the identity element of a group is unique. In this thesis, two kinds of notation for binary operations are used:

- Using the *additive notation*, the operation is denoted by the symbol $+$, and the corresponding identity element is denoted by 0. Groups with this notation are called *additive groups* and the operation is called "addition".

- In *multiplicative notation*, no symbol for the operation is reserved. Instead, the result of the operation with arguments a and b is expressed simply as ab. The identity element with respect to this operation is denoted by 1 and the word "multiplication" will refer to this operation.

Definition A.2. Let G be a (multiplicatively written) group and g an element in G. The set

$$\langle g \rangle = \{g^k \mid k \in \mathbb{Z}\}$$

is called the *cyclic subgroup of G generated by g*. The smallest positive integer n for which $g^n = 1$ (if exists) is called the *order* of g. If $G = \langle g \rangle$ for some $g \in G$, we say that G is a *cyclic group* and that g is a *generator* of G.

We will need only one theorem from group theory. Its proof can be done easily using cosets [Nic93], but we omit it for brevity.

Theorem A.3. *Let H be subgroup of a finite group G. Then $\#H$ divides $\#G$.*

The next consequence is obvious.

Corollary A.4. *Let g be an element of a finite group G. Then $\#\langle g \rangle$ divides $\#G$.*

Using the group terminology developed above, fields can be easily defined.

Definition A.5. A *ring* is a set $\mathbb{K}$ together with two binary operations called *addition* and *multiplication*, such that

- $\mathbb{K}$ is an abelian group with respect to addition.
- Multiplication is associative, i.e., $a(bc) = (ab)c$ for all $a, b, c \in \mathbb{K}$.
- The two binary operations are connected to each other by *distributivity*; that is, $a(b + c) = ab + ac$ and $(b + c)a = ba + ca$ for all $a, b, c \in \mathbb{K}$.

A ring $\mathbb{K}$ is called a *field*, if it has the additional property:

- The set $\mathbb{K} \setminus \{0\}$ of nonzero elements in $\mathbb{K}$ forms an abelian group under multiplication.

Examples of fields include the rational numbers $\mathbb{Q}$, the real numbers $\mathbb{R}$ and the complex numbers $\mathbb{C}$, having the familiar rules for addition and multiplication.

To identify fields sharing the same algebraic structure, we introduce the concept of being isomorphic.

Definition A.6. Let $\mathbb{K}$ and $\mathbb{L}$ be rings. A mapping $\phi : \mathbb{K} \mapsto \mathbb{L}$ is said to be a *homomorphism*, if it preserves addition and multiplication. That is,

$$\phi(a + b) = \phi(a) + \phi(b) \quad \text{and} \quad \phi(ab) = \phi(a)\phi(b)$$

for all $a, b \in \mathbb{K}$. The mapping ϕ is called an *isomorphism*, if in addition it is bijective. In that case $\mathbb{K}$ and $\mathbb{L}$ are said to be *isomorphic* as rings. We indicate this by $\mathbb{K} \simeq \mathbb{L}$. Two fields are defined to be isomorphic, if they are isomorphic as rings.

Definition A.7. Let $\mathbb{K}$ be a field. A *polynomial* $f(x)$ over $\mathbb{K}$ is a formal sum

$$f(x) = \sum_{i=0}^{n} a_i x^i, \quad n \geq 0, \tag{A.1}$$

where the coefficients a_i are elements of $\mathbb{K}$, and x is a symbol not belonging to $\mathbb{K}$, called an *indeterminate* over $\mathbb{K}$. The set of all polynomials over $\mathbb{K}$ is denoted by $\mathbb{K}[x]$.

Multiplication and addition of polynomials over a field $\mathbb{K}$ are defined the same way as with the polynomials over the real numbers. One can verify by a routine calculation that using these operations, the polynomials over a field $\mathbb{K}$ form a ring. The additive identity of this ring is the polynomial with all coefficients equaling zero. This polynomial is called the *zero polynomial.*

Analogously, the *degree* of a nonzero polynomial f, labeled $\deg f$, is the largest power of x in (A.1) with nonzero coefficient. For the zero polynomial, we agree that $\deg 0 = -\infty$. In a *monic* polynomial, the largest power of x has coefficient 1. Polynomials with degree ≤ 0 are called *constants.*

Definition A.8. Let f and g be polynomials over the field $\mathbb{K}$. We say that g *divides* f, if there is a polynomial $h \in \mathbb{K}[x]$ such that $f = hg$. A non-constant polynomial $f \in \mathbb{K}[x]$ is called *irreducible* over $\mathbb{K}$, if the only polynomials dividing f and having lower degree than f are the constants. The polynomials over $\mathbb{K}$ that are not irreducible over $\mathbb{K}$ are called *reducible.*

The irreducible polynomials over $\mathbb{K}$ play the role of primes in $\mathbb{K}[x]$, as the next theorem will guarantee. For a proof, consult any introductory textbook in abstract algebra, for example [Nic93].

Theorem A.9. *Any monic polynomial $f \in \mathbb{K}[x]$ of positive degree can be written in the form*

$$f = \prod_{i=0}^{r} p_i^{e_i},$$

where p_i are distinct monic irreducible polynomials in $\mathbb{K}[x]$, and e_i are positive integers. This factorization is unique apart from the order of the factors.

In the sequel, we will need yet another elementary result about polynomials, called the *division algorithm* [Nic93].

Theorem A.10. *Let f and g be polynomials over a field $\mathbb{K}$. If $g \neq 0$, then there are uniquely determined polynomials q and r in $\mathbb{K}[x]$ such that $f = qg + r$, where* $\deg r < \deg g$.

The unique remainder r of f divided by $g \neq 0$ will subsequently be denoted $r = f \mod g$.

Definition A.11. In $\mathbb{K}[x]$, we say that f is *equal to g modulo p*, if p divides $f - g$. In that case we write $f = g \pmod{p}$.

It is immediately seen that this relation is an equivalence relation. Thus, our polynomial p has partitioned the ring $\mathbb{K}[x]$ into distinct equivalence classes, denoted by

$$[f] = \{g \in \mathbb{K}[x] \mid g = f \pmod{p}\}.$$

The set of all equivalence classes is called the *factor ring* of $\mathbb{K}[x]$ modulo p and is denoted by

$$\mathbb{K}[x]/(p) = \{[f] \mid f \in \mathbb{K}[x]\}.$$

For two equivalence classes $[f]$ and $[g] \in \mathbb{K}[x]/(p)$, define their *sum* and *product* as

$$[f] + [g] = [f + g]$$
$$[f][g] = [fg].$$

A lengthy but straightforward check shows that these operations are well defined and that the factor ring $\mathbb{K}[x]/(p)$ really is a ring. To verify the next important result involves a little more work [Nic93].

Theorem A.12. *The factor ring $\mathbb{K}[x]/(p)$ is a field if and only if p is irreducible over $\mathbb{K}$.*

Let us now take a closer look at the elements of the factor ring. Using the division algorithm, every polynomial $f \in \mathbb{K}[x]$ can be expressed as

$$f = qp + r, \quad q, r \in \mathbb{K}[x], \quad \deg r < \deg p.$$

This implies that $f = r \pmod{p}$ and thus, $[f] = [r]$. In addition, if $\tilde{r}$ is another polynomial over $\mathbb{K}$ with $[f] = [\tilde{r}]$ and $\deg \tilde{r} < \deg p$, then p divides $r - \tilde{r}$. This can be true only if $\tilde{r} = r$, since $\deg(r - \tilde{r}) < \deg p$. These observations lead to the following theorem.

Theorem A.13. *The set*

$$S = \{r \mid r \in \mathbb{K}[x],\ \deg r < \deg p\}$$

forms a set of representatives of the factor ring $\mathbb{K}[x]/(p)$. That is, the map

$$\phi : S \ni r \mapsto [r] \in \mathbb{K}[x]/(p)$$

is bijection. The inverse of ϕ is calculated by the division algorithm, $\phi^{-1}([f]) = f \bmod p$.

A.2 Field Extensions

A useful method for obtaining finite fields is to start with small fields and use them to construct larger fields extending the original ones.

Definition A.14. Let $\mathbb{F}$ be a field. A subset $\mathbb{K}$ of $\mathbb{F}$ that is itself a field under the operations of $\mathbb{F}$, is called a *subfield* of $\mathbb{F}$. Respectively, $\mathbb{F}$ is called an *extension* of $\mathbb{K}$.

It is interesting to note that a subfield $\mathbb{K}$ of $\mathbb{F}$ can be interpreted as a field of *scalars* for $\mathbb{F}$, that is, $\mathbb{F}$ can be considered as a *vector space* over $\mathbb{K}$.

Definition A.15. Let $\mathbb{F}$ be an extension field of $\mathbb{K}$. If the dimension of the vector space $\mathbb{F}$ over $\mathbb{K}$ is finite, $\mathbb{F}$ is called a *finite extension* of $\mathbb{K}$. This dimension, denoted by $[\mathbb{F} : \mathbb{K}]$, is then called the *degree* of $\mathbb{F}$ over $\mathbb{K}$.

The means to explore the structure of field extensions is to consider polynomials over the underlying subfields.

Definition A.16. Let θ belong to an extension field $\mathbb{F}$ of $\mathbb{K}$. The element θ is said to be *algebraic* over $\mathbb{K}$, if there exists a nonzero polynomial f in the ring $\mathbb{K}[x]$ such that $f(\theta) = 0$. The extension field $\mathbb{F}$ is called *algebraic* over $\mathbb{K}$, if all its elements are *algebraic* over $\mathbb{K}$.

To motivate our survey, we note that a great deal of extensions tend to be algebraic.

Theorem A.17. *Every finite extension of* $\mathbb{K}$ *is algebraic over* $\mathbb{K}$.

Proof. Let $\mathbb{F}$ be a finite extension of $\mathbb{K}$ with degree $n = [\mathbb{F} : \mathbb{K}]$ and let θ be an element of $\mathbb{F}$. Now the set $\{1, \theta, \ldots, \theta^n\}$ is linearly dependent over $\mathbb{K}$; that is, there exist scalars $a_0, \ldots a_n \in \mathbb{K}$ such that

$$a_0 + a_1\theta + \cdots + a_n\theta^n = 0.$$

This means that θ is indeed a root of a polynomial in $\mathbb{K}[x]$. □

Assume that θ is algebraic over a subfield $\mathbb{K}$ of $\mathbb{F}$. Consider the set of polynomials

$$Z(\theta) = \{f \in \mathbb{K}[x] \setminus \{0\} \mid f(\theta) = 0\}.$$

We see that this set is nonempty and that all polynomials in it have positive degree. Thus, we can choose a polynomial $m \in Z(\theta)$ having least degree in $Z(\theta)$. Without loss of generality, we may assume it to be monic. In the next theorem, we will prove that this m is uniquely determined.

Definition A.18. Let $\theta \in \mathbb{F}$ be algebraic over the subfield $\mathbb{K}$ of $\mathbb{F}$. The monic polynomial $m \in \mathbb{K}[x]$ of minimal degree such that $m(\theta) = 0$ is called the *minimal polynomial* of θ over $\mathbb{K}$. The degree of m is called the *degree* of θ over $\mathbb{K}$.

Theorem A.19. *Let* $\theta \in \mathbb{F}$ *be algebraic over* $\mathbb{K}$ *with a minimal polynomial* $m \in \mathbb{K}[x]$. *Then:*

- *The element* θ *uniquely determines* m.
- *The polynomial* m *is irreducible in* $\mathbb{K}[x]$.

Proof. To prove uniqueness, assume that $\tilde{m}$ is another monic polynomial with $\tilde{m}(\theta) = 0$ and minimal degree $\deg \tilde{m} = \deg m$. Using the division algorithm of Theorem A.10, we can write $\tilde{m} = qm + r$ in $\mathbb{K}[x]$, with $\deg r < \deg m$. Since now

$$r(\theta) = \tilde{m}(\theta) - q(\theta)m(\theta) = 0,$$

$r \neq 0$ would contradict the assumption made on the degree of $\tilde{m}$. So $\tilde{m}$ divides m and the same argument shows that m also divides $\tilde{m}$. So, we can find q_1 and q_2 in $\mathbb{K}[x]$ such that $\tilde{m} = q_1 m$ and $m = q_2 \tilde{m}$, or

$$(1 - q_1 q_2)m = 0.$$

Now a comparison of the leading coefficients in the above equation shows that $q_1 q_2 = 1$. This show that q_1 and q_2 are constants, and recalling that m and $\tilde{m}$ were monic, we get $\tilde{m} = m$.

The irreducibility of m is seen by trying to write m as a product $m = fg$ in $\mathbb{K}[x]$, with $\deg f, \deg g < \deg m$. Then the fact $f(\theta)g(\theta) = m(\theta) = 0$ implies that either $f(\theta) = 0$ or $g(\theta) = 0$, contradicting the definition of m. □

To describe how algebraic elements are connected with their minimal polynomials, we need one more definition. Note that any nonempty intersection of subfields of a given field $\mathbb{F}$ is again a subfield of $\mathbb{F}$.

Definition A.20. Let $\mathbb{K}$ be a subfield of $\mathbb{F}$ and $\theta_1, \ldots, \theta_n \in \mathbb{F}$. Then the field constructed by intersecting all subfields of $\mathbb{F}$ that contain $\theta_1, \ldots, \theta_n$ and $\mathbb{K}$ is called the subfield of $\mathbb{F}$ *generated over* $\mathbb{K}$ by the elements $\theta_1, \ldots \theta_n$. We denote this by $\mathbb{K}(\theta_1, \ldots, \theta_n)$. If $\theta \in \mathbb{F}$, the extension $\mathbb{K}(\theta)$ is called a *simple extension* of $\mathbb{K}$ in $\mathbb{F}$.

We are now ready to formulate the key theorem of this section. The proof [Nic93], though not complicated, is omitted, since it requires some auxiliary facts about rings that are not required elsewhere in this work.

Theorem A.21. *Let θ be algebraic of degree n over $\mathbb{K}$ and let m be the minimal polynomial of θ over $\mathbb{K}$. Then:*

- *The map $\phi : \mathbb{K}[x]/(m) \ni [f] \mapsto f(\theta) \in \mathbb{K}(\theta)$ is an isomorphism.*
- *$[\mathbb{K}(\theta) : \mathbb{K}] = n$ and the set $\{1, \theta, \ldots, \theta^{n-1}\}$ is a basis of $\mathbb{K}(\theta)$ over $\mathbb{K}$.*

The previous theorem shows that an algebraic element from an extension field $\mathbb{F}$ of the given field $\mathbb{K}$ can be used to construct a finite extension of $\mathbb{K}$. However, what we really want to do is to extend a given field $\mathbb{K}$ – without any *a priori* reference to any fields containing $\mathbb{K}$ as a subfield.

Theorem A.22. *Let p be an irreducible polynomial of degree n over the field $\mathbb{K}$. Then $\mathbb{K}$ has (after identifying $\mathbb{K}$ with one of its isomorphic copies) a simple extension $\mathbb{K}(\theta)$ with θ being a root of p.*

Proof. Denote $\mathbb{F} = \mathbb{K}[x]/(p)$. By Theorem A.12, $\mathbb{F}$ is a field. Also, it is easy to see that the map $\iota : a \mapsto [a]$ mapping constant polynomials over $\mathbb{K}$ into $\mathbb{K}[x]/(p)$ is one-to-one and preserves multiplication and addition on $\mathbb{K}$. So we see that the fields $\mathbb{K}$ and $\widetilde{\mathbb{K}} = \iota(\mathbb{K})$ are isomorphic. It is also clear that $\widetilde{\mathbb{K}}$ is a subfield of $\mathbb{F}$.

Denote $\theta = [x] \in \mathbb{F}$. Assume p is given as

$$p(x) = \sum_{i=0}^{n} a_i x^i, \quad a_i \in \mathbb{K}.$$

Identifying coefficients a_i as elements of $\widetilde{\mathbb{K}}$, p can be interpreted to lay in $\widetilde{\mathbb{K}}[x]$ and we may compute in $\mathbb{F}$:

$$p(\theta) = \sum_{i=0}^{n} a_i \theta^i = \sum_{i=0}^{n} [a_i][x]^i = [\sum_{i=0}^{n} a_i x^i] = [p] = [0] = 0 \in \mathbb{F}.$$

Write $p = \alpha m$, where $\alpha \in \mathbb{K}$ and m is monic. Since $m(\theta) = 0$, and m is irreducible over $\mathbb{K}$ and so also over $\widetilde{\mathbb{K}}$, it is seen that m is the minimal polynomial of θ over $\widetilde{\mathbb{K}}$. By Theorem A.21, $\widetilde{\mathbb{K}}(\theta)$ is a finite extension of $\widetilde{\mathbb{K}}$ of degree n, and $p(\theta) = 0$. Since isomorphic fields have exactly the same structure, we may identify $\mathbb{K}$ with $\widetilde{\mathbb{K}}$ and $\mathbb{K}(\theta)$ with $\widetilde{\mathbb{K}}(\theta)$, and we are done. □

A.3 Splitting Fields

In this section, we want to extend the field $\mathbb{K}$ so that all polynomials over $\mathbb{K}$ have maximal number of roots. To see what this means, the next two results are needed.

Lemma A.23. *For nonzero $f \in \mathbb{K}[x]$ and $a \in \mathbb{K}$, $f(a) = 0$ if and only if the polynomial $x - a$ divides f.*

Proof. Use the division algorithm to write $f(x) = (x-a)q(x)+r(x)$, with $r, q \in \mathbb{K}[x]$ and $\deg r < \deg(x-a)$. Thus, r is a constant. By a direct substitution, this constant is seen to be $f(a)$, which proves our contention. □

Definition A.24. Let $a \in \mathbb{K}$ be a root of the polynomial $f \in \mathbb{K}$. The largest integer k such that $(x-a)^k$ divides f is called the *multiplicity of b*. If $k = 1$, then b is called a *simple root*, otherwise b is said to be a *multiple* root.

Lemma A.25. *In a field $\mathbb{K}$, a polynomial f of degree $n \geq 0$ may have at most n distinct roots.*

Proof. Let $a_1, \ldots, a_m$ be the distinct roots of f in $\mathbb{K}$. Now f is divisible by the irreducible polynomial $(x - a_i)$, for $i = 1, \ldots, m$. These affine polynomials occur in the canonical factorization of f into irreducible polynomials according to Theorem A.9. Hence, $(x - a_1) \cdots (x - a_m)$ divides f. By comparing degrees, we get $m \leq n$. □

Now we can state what we want. For a polynomial f in $\mathbb{K}[x]$, we want an extension of $\mathbb{K}$ where we can express f as a product of affine polynomials of degree 1. It is clear that in such an extension field, f would have the maximum number of roots.

Definition A.26. Let f be a non-constant polynomial in $\mathbb{K}[x]$ and $\mathbb{F}$ an extension field of $\mathbb{K}$. We say that f *splits* in $\mathbb{F}$, if f has a decomposition into linear factors over $\mathbb{F}$:

$$f(x) = a(x - \alpha_1) \cdots (x - \alpha_n), \quad a \in \mathbb{K}, \quad \alpha_i \in \mathbb{F}.$$

The field $\mathbb{F}$ is a *splitting field* of f over $\mathbb{K}$, if f splits in $\mathbb{F}$ and if moreover, $\mathbb{F} = \mathbb{K}(\alpha_1, \alpha_2, \ldots, \alpha_n)$.

The definition guarantees us that a splitting field $\mathbb{F}$ of f over $\mathbb{K}$ is economical in the sense that no proper subfield of $\mathbb{F}$ that is an extension of $\mathbb{K}$ contains all the roots of f (counting multiplicity). By first factoring f into irreducible polynomials and then applying the extension procedure used in the proof of Theorem A.22, we see that every polynomial has a splitting field. It can be proven [Nic93] that this splitting field is unique up to isomorphism. The next theorem summarizes these observations.

Theorem A.27. *Assume f is a polynomial of positive degree in $\mathbb{K}[x]$. Then there is a splitting field of f over $\mathbb{K}$. Furthermore, any two splitting fields of f over $\mathbb{K}$ are isomorphic under an isomorphism which fixes $\mathbb{K}$.*

It is a remarkable fact that splitting fields for arbitrary collections of polynomials over a field $\mathbb{K}$ exist. Especially, we may consider the field where all polynomials in $\mathbb{K}[x]$ split.

Definition A.28. An extension $\overline{\mathbb{K}}$ of the field $\mathbb{K}$ is called an *algebraic closure* of $\mathbb{K}$, if $\overline{\mathbb{K}}$ is an algebraic extension of $\mathbb{K}$ with the property that every non-constant polynomial in $\overline{\mathbb{K}}[x]$ splits in $\overline{\mathbb{K}}$.

We conclude this section with the following deep theorem [Lan65, Section 7.2] about fields. Since its proof contains considerable use of set theory (e.g., Zorn's Lemma), we omit it.

Theorem A.29. *Every field $\mathbb{K}$ has an algebraic closure. Any two algebraic closures of $\mathbb{K}$ are connected via an isomorphism fixing elements of $\mathbb{K}$.*

A.4 Finite Fields

A.4.1 Subfield Structure

Analogously with polynomials, we define modular arithmetic on $\mathbb{Z}$, the field of integers. Let n be a fixed positive integer.

Definition A.30. We say that the integer a is equal to b modulo n, if n divides $b - a$. We indicate this by writing $a = b \pmod{n}$. The unique remainder $0 \leq r < n$ after division of a by $n \neq 0$ is denoted $r = a \mod n$.

This relation is immediately seen to be an equivalence. We denote the equivalence classes with respect to this relation by

$$[a] = \{b \mid b = a \pmod{n}\},$$

while $\mathbb{Z}/(n)$ stands for the collection of all equivalence classes. As with polynomials, addition and multiplication in $\mathbb{Z}/(n)$ are defined by

$$\begin{aligned} [a] + [b] &= [a + b], \\ [a][b] &= [ab]. \end{aligned} \tag{A.2}$$

Denote by $\mathbb{Z}_n$ the set $\{0, 1, \ldots, n-1\}$. For convenience, note that the mapping from $\mathbb{Z}_n$ into $\mathbb{Z}/(n)$ given by $\phi : a \mapsto [a]$ is bijective. Hence we can use ϕ to transfer the structure of $\mathbb{Z}/(n)$ to $\mathbb{Z}_n$ by defining for $a, b \in \mathbb{Z}_n$:

$$\begin{aligned} a + b &= \phi^{-1}(\phi(a) + \phi(b)), \\ ab &= \phi^{-1}(\phi(a)\phi(b)). \end{aligned} \tag{A.3}$$

We summarize the basic results we need from elementary number theory [Ros88].

Theorem A.31. *The set $\mathbb{Z}_n$ with the binary operations induced by formulae (A.2) and (A.3) is a ring. Further, if p is a prime, then $\mathbb{Z}_p$ is a field.*

Let us now take a look at the structure of a general finite field $\mathbb{F}$. Compute iteratively the sequence $a_1 = 1$, $a_{k+1} = a_k + 1$, $k = 1, 2, \ldots$ Since all a_i can't be distinct, there are integers, $k > l > 0$ such that $a_k = a_l$. This implies that, for a positive integer $(k - l)$,

$$(k-l)1 = \underbrace{1 + \cdots + 1}_{k-l} = 0.$$

Definition A.32. Let $\mathbb{F}$ be a field. If $\mathbb{F}$ is finite, then the least positive integer p with the property $p1 = 0$ is called the *characteristic* of $\mathbb{F}$, denoted by $p = \operatorname{char} \mathbb{F}$. If $n1 \neq 0$ for all $n \geq 1$, then the characteristic of $\mathbb{F}$ is defined to be 0.

It is easily seen that the characteristic $p \neq 0$ of a field is prime, since $p = st$ with $1 < s, t < p$ implies that either $s1$ or $t1$ equals zero, contradicting the minimality of p. Also, we see that every subfield of the finite field $\mathbb{F}$ with $\operatorname{char} \mathbb{F} = p$ contains 0 and 1, and hence by the closure of addition, the distinct elements $0, 1, \ldots, p-1$. This set is easily seen to be isomorphic to $\mathbb{Z}_p$. We summarize this in the next theorem.

Theorem A.33. *Let $\mathbb{F}$ be a finite field with characteristic p. Then $\mathbb{F}$ contains (an isomorphic copy of) the field $\mathbb{Z}_p$ as a subfield.*

We will reserve the notation $\mathbb{F}_q$ for a finite field with q elements. The next lemma is useful.

Lemma A.34. *Let a finite field $\mathbb{F}_{q'}$ contain $\mathbb{F}_q$ as a subfield. Then $q' = q^m$ for some positive integer m.*

Proof. $\mathbb{F}_{q'}$ can be seen as a vector space over $\mathbb{F}_q$. Since $\mathbb{F}_{q'}$ is finite, its dimension m is necessarily finite. So every element of $\mathbb{F}_{q'}$ can written as $k_1 f_1 + \cdots + k_m f_m$, $k_i \in \mathbb{F}_q$, with respect to some basis $f_i \in \mathbb{F}_{q'}$. Since each k_i may have one of q values, the total number of elements in $\mathbb{F}_{q'}$ is q^m. □

Combining results A.33 and A.34 we get immediately the following important consequence.

Corollary A.35. *Let $\mathbb{F}$ be a finite field. Then $\mathbb{F}$ has p^n elements, where the prime p is the characteristic of $\mathbb{F}$ and $n = [\mathbb{F} : \mathbb{F}_p]$ is the degree of $\mathbb{F}$ over its subfield $\mathbb{F}_p$, where $\mathbb{F}_p$ is isomorphic to $\mathbb{Z}_p$.*

The next lemma is needed in establishing the existence of finite fields.

Lemma A.36. *Let $\mathbb{F}_{q^k}$ be an extension of a field $\mathbb{F}_q$. Then the mapping $\sigma : \alpha \mapsto \alpha^q$ is a homomorphism that fixes elements of the subfield $\mathbb{F}_q$.*

Proof. Since the group of nonzero elements of $\mathbb{F}_q$ has $q-1$ elements, it is clear that $\alpha^{q-1} = 1$ for every nonzero $\alpha \in \mathbb{F}_q$. Hence, $\alpha^q = \alpha$ for all $\alpha \in \mathbb{F}_q$. The fact that σ preserves multiplication in $\mathbb{F}_{q^k}$ is clear. For the sum, write using the binomial theorem,

$$(\alpha + \beta)^q = \sum_{i=0}^{q} \binom{q}{i} \alpha^i \beta^{q-i}.$$

A short computation shows that all binomial coefficients expect the first and the last one above are multiplies of the characteristic of the underlying field. Hence, the above equation reduces to

$$(\alpha + \beta)^q = \alpha^q + \beta^q.$$

□

Theorem A.37. *For every prime p and every positive integer there exists a finite field with $q = p^n$ elements. Any two fields with the same number of elements are isomorphic.*

Proof. Consider the polynomial $Q(x) = x^q - x$, which can be decomposed as

$$Q(x) = \prod_{i=1}^{q} (x - \alpha_i), \quad \alpha_i \in \overline{\mathbb{F}}_p.$$

From this it follows that if all of the α_i above are not distinct, then $Q(x)$ is divisible by $(x - \alpha_i)^2$ for some i, i.e.,

$$Q(x) = (x - \alpha_i)^2 Q_0(x)$$

for some $Q_0(x) \in \overline{\mathbb{F}}_p[x]$. Taking (formal) derivatives from both sides, the above equation implies that $x - \alpha_i$ divides $Q'(x)$ in $\overline{\mathbb{F}}_p[x]$. This is impossible, since $Q'(x) = qx^{q-1} - 1 = -1$. Hence, $Q(x)$ has q distinct roots $\alpha_1, \ldots, \alpha_q$ in $\overline{\mathbb{F}}_p$. Denote this set by $\mathbb{K}$. By a direct check, we see that $\mathbb{K}$ is a subfield of $\overline{\mathbb{F}}_p$. Thus, $\mathbb{K}$ is a field with q elements. To prove the uniqueness part, let $\mathbb{F}$ be a finite field with $q = p^n$ elements. We know that $\mathbb{F}$ contains $\mathbb{F}_p \simeq \mathbb{Z}_p$ as a subfield. Consider again the polynomial $Q(x) = x^q - x$ over the field $\mathbb{F}_p$. By Lemma A.25, it has at most q roots. But Lemma A.36 tells that every element of $\mathbb{F}$ is a root of Q and so we can split Q in $\mathbb{F}$ as

$$Q(x) = \prod_{\alpha \in \mathbb{F}} (x - \alpha).$$

Thus $\mathbb{F}$ is a splitting field of $Q(x) = x^q - x$ over $\mathbb{F}_p$. Taking Theorem A.27 into account, we are done. □

Since all finite fields with q elements are isomorphic, we may identify them all and just speak of *the* finite field $\mathbb{F}_q$.

Definition A.38. The finite field $\mathbb{F}_q$ (or more rigorously, the equivalence class of isomorphic fields with q elements) is called the *Galois field* of order q.

Corollary A.39. *The algebraic closure $\overline{\mathbb{F}}_q$ of a finite field $\mathbb{F}_q$ is given by*

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

Proof. By definition, $\overline{\mathbb{F}}_q$ is an algebraic extension of $\mathbb{F}_q$. For $\theta \in \overline{\mathbb{F}}_q$, denote its degree by $k(\theta)$. Since by Theorem A.21 we have $[\mathbb{F}_q(\theta) : \mathbb{F}_q] = k(\theta)$, $\mathbb{F}_q(\theta)$ is isomorphic to $\mathbb{F}_{q^{k(\theta)}}$. Thus we get,

$$\overline{\mathbb{F}}_q = \bigcup_{\theta \in \overline{\mathbb{F}}_q} \{\theta\} \subseteq \bigcup_{\theta \in \overline{\mathbb{F}}_q} \mathbb{F}_q(\theta) \simeq \bigcup_{\theta \in \overline{\mathbb{F}}_q} \mathbb{F}_{q^{k(\theta)}} \subseteq \bigcup_{k=1}^{\infty} \mathbb{F}_{q^k}.$$

On the other hand, if $\theta \in \bigcup_{k=1}^{\infty} \mathbb{F}_{q^k}$, then $\theta \in \mathbb{F}_{q^k}$ for some $k \geq 1$. It is clear that θ belongs to the splitting field of its minimal polynomial over $\mathbb{F}_q$. Now by Theorem A.29 (identifying isomorphic fields), we see that $\theta \in \overline{\mathbb{F}}_q$, hence

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

□

Corollary A.40. *Let α be an element in some extension field of $\mathbb{F}_q$. Then $\alpha \in \mathbb{F}_q$ if and only if $\alpha^q = \alpha$.*

Proof. The necessity condition follows directly from Lemma A.36. Assume $\alpha^q = \alpha$. Then α belongs to the splitting field of $x^q - x$ over $\mathbb{F}_p$. But as we saw in the proof of Theorem A.37, this splitting field is $\mathbb{F}_q$. □

Using the above machinery, the subfield structure of finite fields is quickly explored.

Theorem A.41. *Let $\mathbb{K}$ be a subfield of $\mathbb{F}_{p^n}$. Then $\mathbb{K}$ has p^d elements, where d is a positive integer dividing n. Conversely, for every positive divisor d of n, there is exactly one subfield of $\mathbb{F}_{p^n}$ with p^d elements.*

Proof. Assume $\mathbb{K}$ is a subfield of $\mathbb{F}_{p^n}$. Then $\mathbb{K}$ also has characteristic p, and so may be identified with $\mathbb{F}_{p^d}$ for some integer $d > 0$. By Lemma A.34, $p^n = (p^d)^e$ for some positive integer e, i.e., d divides n. Conversely, if d divides n, it is quickly verified that the polynomial $x^{p^d} - x$ divides $x^{p^n} - x$ in $\mathbb{F}_p[x]$. Thus $\mathbb{F}_{p^n}$ contains as a subfield the splitting field of $x^{p^d} - x$, which has p^d elements, as was seen in the proof of Theorem A.37. If there were two or more distinct subfields of $\mathbb{F}_{p^n}$ with p^d elements, they would together contain more than p^d roots of $x^{p^d} - x$, which contradicts the result of Lemma A.25. □

As an example, the subfield structure of the field $\mathbb{F}_{2^{176}}$ is sketched in Figure A.1.

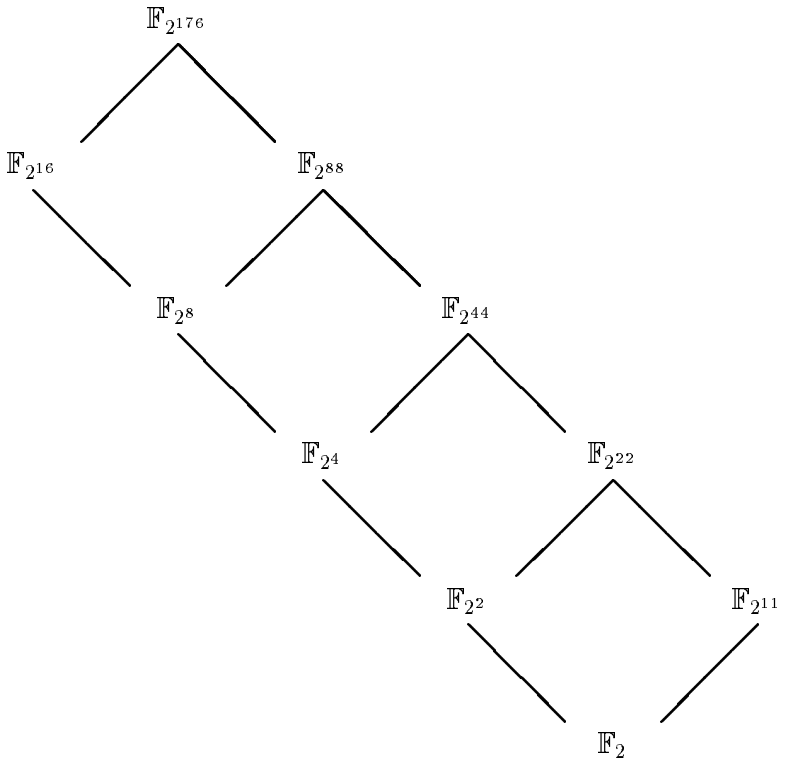

Figure A.1: The subfield structure of $\mathbb{F}_{2^{176}}$

A.4.2 Primitive Elements

For a finite field $\mathbb{F}$, we denote by $\mathbb{F}^*$ the set of nonzero elements in $\mathbb{F}$. By definition, $\mathbb{F}^*$ is a group. For a finite field, the group structure is simple.

Theorem A.42. *For a finite field* $\mathbb{F}_q$*, the group* $\mathbb{F}_q^*$ *is cyclic.*

Proof. Let $h = q - 1$. Since the case $q = 2$ is trivial, we assume $h \geq 2$. Factor h into primes

$$h = \prod_{i \in I} p_i^{e_i},$$

where $e_i \geq 1$ for i in the (finite) index range I. For $i \in I$, fix a nonzero element $a_i \in \mathbb{F}_q$ such that $a_i^{h/p_i} \neq 1$. This can be done, since the polynomial $x^{h/p_i} - 1$ has at most h/p_i roots for $i \in I$. Denote $b_i = a_i^{h/p_i^{e_i}}$. We see that $b_i^{p_i^{e_i}}$ equals 1, while $b_i^{p_i^{e_i-1}} \neq 1$. Thus the multiplicative order of b_i is $p_i^{e_i}$ for $i \in I$. We next show that

$$b = \prod_{i \in I} b_i$$

generates $\mathbb{F}_q^*$. Assume that the order of b is a proper divisor of h. Then for some $j \in I$,

$$1 = b^{h/p_j} = \prod_{i \in I} b_i^{h/p_j}. \tag{A.4}$$

However, for $i \in I$, $i \neq j$, $p_i^{e_i}$ divides h/p_j, which implies $b_i^{h/p_j} = 1$ for every $i \in I$, $i \neq j$. From equation (A.4) we now see that $b_j^{h/p_j} = 1$. This contradicts the fact that the order of b_j is $p_j^{e_j}$. □

Definition A.43. A generator ξ of the multiplicative group $\mathbb{F}_q^*$ is called a *primitive element* of $\mathbb{F}_q$.

With the help of primitive elements, we can now easily establish the existence of irreducible polynomials of any degree over finite fields.

Theorem A.44. *For any finite field $\mathbb{F}_q$ and any positive integer k there exists an irreducible polynomial over $\mathbb{F}_q$ of degree k.*

Proof. Consider the extension field $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$. Let ξ be a primitive element of $\mathbb{F}_{q^k}$. Since $\mathbb{F}_q(\xi)$ contains 0 and all powers of ξ, we know that $\mathbb{F}_q(\xi) = \mathbb{F}_{q^k}$. With the help of Theorems A.19 and A.21 we conclude that the minimal polynomial of ξ is irreducible and has degree k. □

A.4.3 The Trace Function

We shall introduce a mapping that characterizes the connection between a finite field $\mathbb{F}_{q^k}$ and its subfield $\mathbb{F}_q$.

Definition A.45. For $\alpha \in \mathbb{F}_{q^k}$, the *trace* of α over $\mathbb{F}_q$ is defined by

$$\mathrm{Tr}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha) = \alpha + \alpha^q + \alpha^{q^2} + \cdots + \alpha^{q^{k-1}}.$$

If p is a prime, the trace of α over $\mathbb{F}_p$ is called the *absolute trace*. When the context is clear, we omit the subscript from the notation, and denote the absolute trace of α simply by $\mathrm{Tr}(\alpha)$.

The essence of the trace function is that it preserves the vector space structure of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$ and especially, it shrinks the field $\mathbb{F}_{q^k}$ into $\mathbb{F}_q$. The next theorem lists these facts.

Theorem A.46. *The trace* Tr *is a linear transformation from $\mathbb{F}_{q^k}$ onto $\mathbb{F}_q$. That is, $\mathrm{Tr}(\mathbb{F}_{q^k}) = \mathbb{F}_q$, and for any $\alpha, \beta \in \mathbb{F}_{q^k}$ and $c \in \mathbb{F}_q$:*

- $\mathrm{Tr}(\alpha + \beta) = \mathrm{Tr}(\alpha) + \mathrm{Tr}(\beta)$,
- $\mathrm{Tr}(c\alpha) = c\,\mathrm{Tr}(\alpha)$.

Proof. The linearity of trace follows immediately from Lemma A.36. Let $\alpha \in \mathbb{F}_{q^k}$. Recall that by the same lemma, the qth power map is a homomorphism. So we may compute

$$(\mathrm{Tr}(\alpha))^q = (\sum_{i=0}^{k-1} \alpha^{q^i})^q = \sum_{i=0}^{k-1} \alpha^{q^{i+1}}.$$

The Lemma A.36 further implies $\alpha^{q^k} = \alpha$, so we get $(\mathrm{Tr}(\alpha))^q = \mathrm{Tr}(\alpha)$. We conclude that $\mathrm{Tr}(\alpha)$ is a root of $x^q - x$. By Theorem A.37 we know that the set of all roots of $x^q - x$ is $\mathbb{F}_q$. Thus, $\mathrm{Tr}(\mathbb{F}_{q^k}) \subseteq \mathbb{F}_q$. To see that the trace is also onto, by linearity it is enough to show that there is an element in $\mathbb{F}_{q^k}$ with nonzero trace. But for $\alpha \in \mathbb{F}_{q^k}$, $\mathrm{Tr}(\alpha) = 0$ is equal to α being a root of the polynomial

$$x^{q^{k-1}} + \cdots + x^q + x.$$

Since this polynomial may have at most q^{k-1} roots, we are done. □

Appendix B

Table of Notation

Notation	Meaning	See page
$\mathbb{P}^2$	projective plane	10
$(X:Y:Z)$	point in $\mathbb{P}^2$	10
E	elliptic curve	11
$E/\mathbb{K}$	elliptic curve defined over $\mathbb{K}$	11
$\mathcal{O}$	point at infinity	11
$E(\mathbb{K})$	$\mathbb{K}$-rational points of $E/\mathbb{K}$	12
Δ	discriminant	15
$j(E)$	j-invariant	15
$E[n]$	n-torsion points of E	18
O	big-O	24
o	little-o	24
log	logarithm to the base 2	25
$\log_g$	discrete logarithm to the base g	25,27
$L[n,c,\alpha]$	$O(e^{(c+o(1))(\ln n)^{\alpha}(\ln\ln n)^{1-\alpha}})$	25
μ_n	nth roots of unity	27
ln	logarithm to the base e	28
$\mathbb{Q}_p$	rationals with denominators not divisible by p	29
$E_{tors}(\mathbb{Q})$	torsion subgroup of $E(\mathbb{Q})$	30
$\langle\cdot,\cdot\rangle$	bilinear form	35
δ_{ij}	Kronecker delta	36
$\circ$	binary operation	67
$\#G$	number of elements in G	67
$\simeq$	isomorphism	67,68
$\langle g\rangle$	cyclic group generated by g	68
$\mathbb{Z}$	integers	68
$\mathbb{Q}$	rational numbers	68
$\mathbb{R}$	real numbers	68
$\mathbb{C}$	complex numbers	68
deg	degree of a polynomial	69
$\mathbb{K}[x]$	polynomial ring	69
$\mathbb{K}[x]/(p)$	factor ring of $\mathbb{K}[x]$ modulo p	69
mod n	remainder after division by n	69, 74
(mod n)	equality modulo n	69, 74

$[\mathbb{F} : \mathbb{K}]$	degree of $\mathbb{F}$ over $\mathbb{K}$	70
$\mathbb{K}(\theta_1, \ldots, \theta_n)$	extension of $\mathbb{K}$ generated by $\theta_1, \ldots, \theta_n$	72
$\mathbb{K}(\theta)$	simple extension of $\mathbb{K}$	72
$\overline{\mathbb{K}}$	algebraic closure of $\mathbb{K}$	74
char	characteristic	75
$\mathbb{Z}_n$	integers modulo n	75
$\mathbb{F}_q$	finite field with q elements	77
$\mathbb{F}^*$	set of nonzero elements of $\mathbb{F}$	78
Tr	trace	79

Bibliography

[ABMV93] G.B. Agnew, T. Beth, R.C. Mullin, and S.A. Vanstone. Arithmetic operations in $GF(2^m)$. *Journal of Cryptology*, 6:3–13, 1993.

[AMOV91] G.B. Agnew, R.C. Mullin, I.M. Onyszchuk, and S.A. Vanstone. An implementation for a fast public-key cryptosystem. *Journal of Cryptology*, 3:63–79, 1991.

[Bea96] Dan Beauregard. Efficient algorithms for implementing elliptic curve public-key schemes. Master's thesis, Worcester Polytechnic Institute, U.S., 1996.

[BRS98] I. Blake, R. Roth, and G. Seroussi. Efficient arithmetic in $GF(2^n)$ through palindromic representation. Hewlett Packard Laboratories, available from `http://www.hpl.hp.com/`, July 1998.

[cer97] Remarks on the security of the elliptic curve cryptosystem, September 1997. Certicom, available from `http://www.certicom.com/`.

[Cop84] D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30:587–594, 1984.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[fST91] National Institute for Standards and Technology. A proposed federal information processing standard for digital signature standard (DSS). Technical report, FIPS, 1991.

[GL92] S. Gao and H. Lenstra. Optimal normal bases. *Designs, Codes and Cryptography*, 2:315–323, 1992.

[Gor98] D. M. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, 27:129–146, 1998.

[GP97] Jorge Guajardo and Christof Paar. Efficient algorithms for elliptic curve cryptosystems. In *Crypto'97*, pages 342–356. Springer–Verlag, 1997.

[GP98] Jorge Guajardo and Christof Paar. Fast inversion in composite galois fields $GF((2^n)^m)$. *ISIT 1998*, 1998.

[Gua97] Jorge Guajardo. Efficient algorithms for elliptic curve cryptosystems. Master's thesis, Worcester Polytechnic Institute, U.S., 1997.

[Har77] Robin Hartshorne. *Algebraic Geometry*. Springer–Verlag, first edition, 1977. ISBN 3-540-90244-9.

[IEE98] IEEE P1363 Working Group. *Standard Specifications for Public Key Cryptography*, September 1998. Draft; latest version available from `http://www.grouper.ieee.org/groups/1363/`.

[IT88] Toshiya Itoh and Shigeo Tsujii. A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases. *Information and Computation*, 78(3):171–177, 1988.

[JMV90] D. Jungnickel, A. Menezes, and S. A. Vanstone. On the number of self-dual bases of $GF(q^n)$ over $GF(q)$. *Proceedings of American Mathematical Society*, 109:23–29, 1990.

[Knu81] D. E. Knuth. *Seminumerical Algorithms*, volume II of *The Art of Computer Programming*. Addison–Wesley, second edition, 1981.

[KO63] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Sov. Phys.-Dokl.*, 7(7):595–596, 1963.

[Kob87a] Neal Koblitz. *A Course in Number Theory and Cryptography*. Springer–Verlag, second edition, 1987. ISBN 3-540-94293-9.

[Kob87b] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[Kob91] Neal Koblitz. Elliptic curve implementation of zero-knowledge blobs. *Journal of Cryptology*, 4:207–213, 1991.

[Koc95] C. K. Koc. Analysis of sliding window techniques for exponentiation. *Computers and Mathematics with Applications*, 30(10):17–24, 1995.

[Lan65] Serge Lang. *Algebra*. Addison–Wesley, fourth edition, 1965.

[Ler97] Reynald Lercier. *Algorithmique Des Courbes Elliptiques Dans Les Corps Finis*. PhD thesis, École Polytechnique, France, 1997.

[LL93] A. Lenstra and H. Lenstra, editors. *The Development of the Number Field Sieve*. Number 1554 in Lecture Notes in Mathematics. Springer–Verlag, 1993.

[LN97] R. Lidl and H. Niederreiter. *Finite Fields*. Cambridge University Press, second edition, 1997.

[LW88] A. Lempel and M.J. Weinberger. Self-complementary normal bases in finite fields. *SIAM Journal of Discrete Mathematics*, 1:193–198, 1988.

[LZ94] G. Lay and H. Zimmer. Constructing elliptic curves with given group order over finite fields. In *Algorithmic Number Theory: First International Symposium*, volume 877, pages 250–263. Springer–Verlag, 1994.

[Mas91] Edoardo D. Mastrovito. *VLSI Architectures for Computations in Galois Fields*. PhD thesis, Linköping University, Sweden, 1991.

[MBG+93] A. Menezes, I. Blake, XuHong Gao, R. C. Mullin, S. A. Vanstone, and Tomik Yaghoobian. *Applications of Finite Fields.* Kluwer Academic Publishers, first edition, 1993. ISBN 0-7923-9282-5.

[Men93] A. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, first edition, 1993. ISBN 0-7923-9368-6.

[Mil86a] Victor S. Miller. Short programs for functions on curves. Unpublished manuscript, 1986.

[Mil86b] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology – Crypto '85*, pages 417–426. Springer–Verlag, 1986.

[MO83] J. Massey and J. Omura. *Computational Method and Apparatus for Finite Field Arithmetic*, June 1983. European Patent No. 0080528.

[MO90] F. Morain and Jorge Olivos. Speeding up computations on an elliptic curve using addition–subtraction chains. *Theoretical Informatics and Applications*, 24(6):531–433, 1990.

[Mor91] F. Morain. Building cyclic elliptic curves modulo large primes. In *Advances in Cryptology – Eurocrypt '91*, volume 547, pages 328–336. Springer–Verlag, 1991.

[MOV93] A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39:1639–1646, 1993.

[MOVW89] R.C. Mullin, I.M. Onyszchuk, S.A. Vanstone, and R. Wilson. Optimal normal bases in $GF(p^n)$. *Discrete Applied Mathematics*, 22:149–161, 1989.

[MV90] A. Menezes and S.A. Vanstone. The implementation of elliptic curve cryptosystems. In *Advances in Cryptology – Auscrypt '90*, pages 2–13, 1990.

[MV93] A. Menezes and S.A. Vanstone. Elliptic curve cryptosystems and their implementation. *Journal of Cryptography*, 6:209–224, 1993.

[MvOV97] A. Menezes, P.C. van Oorschott, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, first edition, 1997.

[MW98] Ueli Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In *Advances in Cryptology – Eurocrypt '98*, pages 72–84. Springer–Verlag, 1998.

[Nic93] W. Keith Nicholson. *Intoduction to Abstract Algebra.* PWS Publishing Company, first edition, 1993. ISBN 0-534-93189-8.

[Odl95] A. Odlyzko. The future of integer factorization. *CryptoBytes – The Technical Newsletter of RSA Laboratories*, 1(2):5–12, 1995. Also available from `http://www.rsa.com/`.

[Paa96] Christof Paar. A new architecture for a parallel finite field multiplier with low complexity based on composite fields. *IEEE Transactions on Computers*, 45(7):856–861, 1996.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over $GF(p)$. *IEEE Transactions on Information Theory*, 24: 106–110, 1978.

[Pol78] J. Pollard. Monte carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

[PSR97] Christof Paar and Pedro Soria-Rodriguez. Fast arithmetic architectures for public-key algorithms over galois fields $GF((2^n)^m)$. In *Eurocrypt '97*, pages 363–378. Springer–Verlag, 1997.

[Ros88] Kenneth Rosen. *Elementary Number Theory and Its Applications*. Addison–Wesley, second edition, 1988. ISBN 1-201-11958-7.

[Ros98] Martin Rosner. Elliptic curve cryptosystems on reconfigurable hardware. Master's thesis, Worcester Polytechnic Institute, U.S., 1998.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[SA97] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. Preprint; available from `http://www.ams.org/`, October 1997.

[Sch85] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Mathematics of Computation*, 44:483–494, 1985.

[Sch96] Bruce Schneier. *Applied Cryptograhpy*. John Wiley & Sons, Inc., second edition, 1996. ISBN 0-471-11709-9.

[Sem98] I. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Mathematics of Computation*, 67:353–356, 1998.

[Ser98] G. Seroussi. Compact representation of elliptic curve points over $\mathbb{F}_{2^n}$. Available from `http://www.hpl.hp.com/`, 1998.

[Sil86] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*. Springer–Verlag, first edition, 1986. ISBN 3-540-96203-4.

[Sil98] Joseph H. Silverman. The xedni calculus and the elliptic curve discrete logarithm problem. Preprint, August 1998.

[Sim91] G. Simmons. Contemporary cryptology: The science of information integrity. *IEEE Press*, 1991.

[SOOS95] R. Schröppel, H. Orman, S. O'Malley, and O. Spatscheck. Fast key exhange with elliptic curve systems. In *Advances in Cryptology, Proceedings Crypto'95*, pages 43–56. Springer–Verlag, 1995.

[SS98] Joseph H. Silverman and Joe Suzuki. Elliptic curve discrete logarithms and the index calculus. AsiaCrypt '98, to appear, 1998.

[Sti95] Douglas R. Stinson. *Cryptography – Theory and Practice*. CRC Press, first edition, 1995. ISBN 0-8493-8521-0.

[WBV+96] Erik De Win, Antoon Bosselaers, Servaas Vandenberghe, Peter De Gersem, and Joos Vandewalle. A fast software implementation for arithmetic operations in $GF(2^n)$. In *Advances in Cryptology – Asiacrypt '96*, pages 65–76. Springer–Verlag, 1996.

[Wil95] Andrew Wiles. Modular elliptic curves and fermat's last theorem. *Annals of Mathematics*, 141(3):443–551, 1995.

[WT95] Andrew Wiles and Richard Taylor. Ring-theoretic properties of certain hecke algebras. *Annals of Mathematics*, 141(3):553–572, 1995.