Pairing-based cryptography

Maas, M.

Award date:
2006

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computing Science

MASTER'S THESIS

Pairing-Based Cryptography

by

Martijn Maas

Eindhoven, Januari 2004

Supervisor:
Prof. dr. ir. H.C.A. van Tilborg

Advisors:
Dr. B.M.M. de Weger
Drs. G. Schmitz
Dr. ir. P.A.H. Bours

Acknowledgements

I would like to take this opportunity to express my gratitude to some people who were involved in this project. First of all, I owe thanks to Henk van Tilborg for being my overall supervisor and arranging current and previous projects. I would like to thank Benne de Weger, who was closely involved in the writing of this report, for the fruitful discussions and useful advises. Thanks is due to Hans Cuijpers for being on my committee. Patrick Bours and Gido Schmitz deserve thanks for supervising the project and editing this report. Special thanks goes out to Gido, who put a great deal of effort in teaching me all about elliptic curves and making this project into a valuable and enjoyable experience.

On a more personal note, I would like to mention some people who mean a lot to me. First of all Vincent, who provided me with a bed to sleep in and, more importantly, a place to call home in The Hague. Thanks goes out to Ard Jan and Maarten who were always there for me – especially whenever I needed a drink after a train ride across country. Last but by no means least, I would like to thank Ciska and Hans for their unconditional support.

Contents

Chapter 1

Introduction

This report is the result of my graduation project in completion of the Master program Industrial and Applied Mathematics at the Eindhoven University of Technology (TU/e). It has been written in order to obtain the degree of Master of Science. The project has been carried out at the Netherlands National Communications Security Agency (NLNCSA), which is part of the General Intelligence and Security Service (GISS) in Leidschendam.

1.1 Project Background

Elliptic curves have been a subject of research for a long time already. They naturally occur in the study of congruent numbers and Diophantine equations. Initially, researchers were mainly interested in finding points on elliptic curves over infinite fields such as the field of rational or real numbers. The study of curves over finite fields, which at first sight seem to form rather boring abelian groups, aided in finding such points. In 1985, however, elliptic curves over finite fields found an application of their own in cryptography. Koblitz and Miller independently realized that discrete logarithm-based cryptosystems might provide better security when defined on the group of points on an elliptic curve rather than the conventional multiplicative group of a finite field. Or alternatively, they figured, elliptic curves could enable shorter keys, while providing a similar level of security. Since then, a lot of research effort has been put in elliptic curve cryptography and numerous cryptosystems have been proposed. Some of them, however, proved less secure than initially assumed, as the structure of the proposed elliptic curves provides tools to attack the system.

This is where pairings first come into play. A pairing in this context is a function that takes as input two points on an elliptic curve and outputs an element of some multiplicative abelian group. Furthermore, a pairing satisfies some special properties, the most important of which is bilinearity. Due to these special properties, pairings are hard to construct. The two pairings that are known at present are the Weil pairing and the Tate pairing. In 1993, Menezes et al. discovered that the Weil pairing can be used to attack discrete logarithm-based systems on a certain class of elliptic curves; the so-called MOV-reduction. One year later, Frey and Rück used the Tate pairing to describe a similar attack, called the FR-reduction. This cryptanalytic use was the only known application of pairings for a long time. In 2000, however, Joux discovered that pairings can be used as cryptographic building blocks as well. The bilinearity of the pairings enables many cryptosystems with interesting properties. Joux's discovery spurred an extensive research into new applications based on pairings. The large number of articles on pairing-based cryptography that have appeared since 2000 indicates the tremendous amount of research effort put into this subject. An excellent reference is Barreto's 'Pairing-Based Crypto Lounge' [4].

Undoubtedly the most striking application of pairings is the realization of identity-based cryptography. In this variant of public-key cryptography, already proposed by Shamir in 1984, users'

public keys are derived from their identity and the corresponding secret keys are generated by some trusted party. The fact that public keys are linked to the users' identity guarantees the authenticity and therefore takes away the need for certificates as in conventional public-key cryptosystems. Some satisfactory identity-based signature schemes were proposed soon after Shamir described the concept of identity-based crypto. The implementation of an identity-based encryption scheme, however, remained an open problem until Joux's discovery of the constructive use of pairings. In 2001, Boneh and Franklin were able to design an efficient identity-based encryption scheme through pairings. (Simultaneously, Cocks came up with a non pairing-based solution; we do not deal with this scheme.) Soon identity-based signature schemes appeared that are compatible with the encryption scheme by Boneh and Franklin, thus yielding a complete and fully functional solution to the open problem put by Shamir. Besides identity-based systems, numerous other pairing-based schemes with interesting properties have appeared, such as an efficient key agreement protocol and a signature scheme with short signatures.

1.2 Project Description

It is of great importance for the NLNCSA to keep abreast of the recent developments in cryptography. Therefore, the aim of this project is to gain knowledge of the current status of pairing-based crypto in general and identity-based crypto in particular. This includes the mathematics involved on one hand and the potential applications on the other. The fact that in this subject challenging mathematics lies close to real-life applications (in contrast to most interesting mathematics, where usually a large gap is present between theory and practice) is what makes the assignment in our opinion very appealing.

The mathematics involves divisor theory on elliptic curves and consists of number theoretical and algebraic aspects. Since the constructive application of pairings was discovered as recently as 2000, there has not yet appeared a thorough and sound treatise of the theory. Instead, many articles have appeared, often treating (part of) the subject from a different point of view, making different assumptions and using a different notation. We try to give an overview of currently known theory and expose the relation between the different mathematical aspects. Further, we give some proofs that appear to be absent in the literature and support the theory with some examples.

In order to deal with the practice of pairings in cryptography, we describe the setting to which pairings apply, namely discrete logarithm-based cryptography on elliptic curves. We examine the definition, security and application of identity-based cryptography, in comparison with the conventional PKI implementation of asymmetric cryptography. Next, we discuss the encryption scheme by Boneh and Franklin and expose its relation with the classic ElGamal encryption scheme. Lastly, we perform a literature search for applications of pairings, mainly focussing on identity-based applications, but also treating non identity-based applications.

In addition, the NLNCSA asks for a small computer implementation of the pairings, in order to clarify the theory and expose the problems one encounters when implementing. We have written a program in *Mathematica*, which step by step defines the functions needed for the computation of the Weil and the Tate pairing. We have put the stress on the expository function of the implementation rather than the efficiency. The working of the pairings is demonstrated with some small examples.

1.3 Report Outline

As more or less suggested by the project description, this report is divided into two parts. Part I (Chapters 2-6) deals with the mathematical theory of the pairings, where we try to keep the fact that we are interested in cryptographic applications at the back of our minds. Part II (Chapters 7-9) is devoted to the applications of pairings in cryptography. We have tried to keep both parts

independently readable. For this purpose, Part II abstractly defines pairings on elliptic curves as bilinear maps on additive groups, without going into detail how these pairings can be constructed. We therefore believe that, except for some specific examples, Part II can be read without knowledge of the Weil and Tate pairing or even elliptic curves.

Below we give the outline of Parts I and II. We summarize the main results of both parts and make some concluding remarks in Chapter 10. The code of the *Mathematica* implementation is given in Appendix A.

Part I
In Chapter 2, we first state some facts about elliptic curves and functions on elliptic curves. Next we deal with divisor theory on elliptic curves, which lies at the heart of the definitions of the Weil and the Tate pairing.

Chapter 3 focusses on the Weil pairing. The relation between two slightly different definitions of the Weil pairing is exposed and the specific properties of the Weil pairing are given. Then an efficient way to compute the Weil pairing is described and clarified with an example.

The Tate pairing is discussed in Chapter 4. We deal with the definition and properties of this pairing. Next we give the algorithm for computing the Tate pairing, along with an example. We then investigate its relation with the Weil pairing. Hence, readers are advised to read Chapter 3 prior to Chapter 4. Lastly, we outline how the implementation of this algorithm can be optimized.

With the definition of the pairings out of the way, we discuss some aspects that turn out to be important in cryptographic applications. Chapter 5 covers the notion of the embedding degree of an elliptic curve. We give a lower bound for the embedding degree and describe the additional conditions for both pairings. Next we investigate which curves have suitable embedding degree. Chapter 5 elaborates on Chapters 3 and 4.

Distortion maps on elliptic curves are covered in Chapter 6. We give the definition and discuss on which curves such maps can exist. Next we explain how they can be used to modify the Weil and the Tate pairing and describe how these modifications can be used in cryptographic applications. Chapter 6 can be read independently from Chapter 5, but refers to Chapters 3 and 4. This is depicted in Figure 1.1.

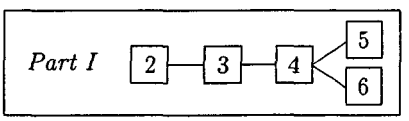

Figure 1.1: Outline Part I

Part II
In Chapter 7, we give a general introduction to elliptic curve cryptography. For this purpose, we first describe the discrete logarithm problem and related problems, on which this type of cryptography is based. Next, we discuss how these problems apply to elliptic curves and what the consequences are of using elliptic curves. Apart from some specific examples, no knowledge of elliptic curves is required. Lastly, we focus on how pairings can be used as building blocks for cryptographic applications. Pairings are defined in a general way, hence readers do not need to have any knowledge of these pairings.

Chapter 8 covers the area of identity-based cryptography. After giving the definition, we compare identity-based cryptography to conventional asymmetric cryptography, which makes use of

public-key infrastructures. Next we describe how pairings can be used to realize a functional implementation of identity-based cryptosystems. In this chapter, we frequently refer to Chapter 7.

Besides identity-based cryptography, pairings enable several other applications. These are dealt with in Chapter 9. The two applications that get the most attention are an efficient multi-party key exchange and a signature scheme with short signatures. Chapter 9 uses the matter covered in Chapter 7, but can be read independently from Chapter 8; this is depicted in Figure 1.2.

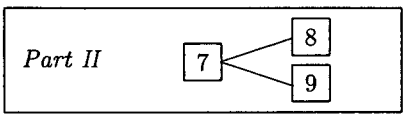

Figure 1.2: Outline Part II

Chapter 2

Preliminaries

In this Chapter, we describe divisor theory on elliptic curves. For this purpose, we briefly give some facts about elliptic curves in Section 2.1. Unless stated otherwise, the results in this section come from [76]. Next we discuss functions on elliptic curves and the multiplicity of their zeros and poles in Sections 2.2 and 2.3. We are then ready to deal with divisor theory in Section 2.4 and a way to compute divisors in practice in Section 2.5. For details on divisor theory, the reader is referred to [54, 76].

2.1 Elliptic Curves

Let K denote a field and $\overline{K}$ its algebraic closure. Throughout the report, we write K^* for $K/\{0\}$. Consider the homogeneous function over the projective plane $\mathbb{P}^2(\overline{K})$ given by

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

where $a_1, a_2, a_3, a_4, a_6 \in \overline{K}$. An *elliptic curve* E is defined to be the set of solutions to $F(X,Y,Z) = 0$ in the projective plane $\mathbb{P}^2(\overline{K})$, where points are equivalent to their multiples, i.e. $(X : Y : Z) \sim (\lambda X : \lambda Y : \lambda Z)$ for $\lambda \in \overline{K}^*$. There is exactly one point in E with $Z = 0$, namely $(0 : 1 : 0)$, which we will call the *point at infinity* and denote by $\mathcal{O}$. By convention, we require the curve to be *non-singular*, i.e. for all points $P \in E$ the three partial derivatives $\partial F/\partial X$, $\partial F/\partial Y$ and $\partial F/\partial Z$ are not all zero at P.

For simplicity, we will use affine coordinates $x = X/Z$ and $y = Y/Z$, and denote an elliptic curve by the *(affine) Weierstrass equation*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{2.1}$$

Then the elliptic curve is the set of points $(x,y) \in \mathbb{A}^2(\overline{K}) = \overline{K} \times \overline{K}$ that satisfy equation (2.1), together with the extra point at infinity $\mathcal{O}$. E is said to be defined over K, denoted E/K, if $a_1, a_2, a_3, a_4, a_6 \in K$. Then K is called the definition field. We denote by $E(K)$ the set of K-rational points, i.e. the points with both coordinates in K, together with $\mathcal{O}$. Observe that by definition, we can write $E = E(\overline{K})$.

Two elliptic curves E_1/K, E_2/K are isomorphic over K, denoted $E_1/K \cong E_2/K$, if there exist $u, r, s, t \in K$, $u \neq 0$, such that the *admissible change of variables*

$$(x,y) \longrightarrow (u^2x + r, u^3y + u^2sx + t)$$

transforms the equation of E_1 into the equation of E_2. We will generally be working with elliptic curves over a finite field $K = \mathbb{F}_q$ consisting of q elements, where $q = p^m$ is a prime power. In this case, the algebraic closure of K is given by $\overline{K} = \bigcup_{i \geq 1} \mathbb{F}_{q^i}$. If the characteristic p of K is greater

than 3, then any curve defined over K is isomorphic with a curve of particularly simple form, namely:

$$E: \ y^2 = x^3 + ax + b, \quad a, b \in K. \tag{2.2}$$

Similarly, one can simplify the Weierstrass equation for curves over finite fields of characteristic $p = 2$ and 3; we will not deal with these cases here.

There exists a natural operation that makes the set of points on an elliptic curve into a group. This operation, known as the 'tangent and chord method', is written additively and has the point at infinity $\mathcal{O}$ as zero element. We next give the explicit formula for the addition of two points on a curve defined over a field of characteristic > 3. Let $P = (x_1, y_1), Q = (x_2, y_2) \in E \setminus \{\mathcal{O}\}$. The point $-P$ is given by $(x_1, -y_1)$. Suppose $Q \neq -P$, then $P + Q = (x_3, y_3)$, where

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned}$$

and

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q, \\ \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } P = Q. \end{cases}$$

Similar formulae exist for curves over characteristic 2 and 3.

Since there exists an addition on elliptic curves, we can also define scalar multiplication. Namely, for $m \in \mathbb{Z}$ and $P \in E$, the *multiplication by m* is given by:

$$[m]P = P + \ldots + P \quad (m \text{ terms}) \text{ for } m > 0,$$

$$[0]P = \mathcal{O}, \quad \text{and} \quad [-m]P = [m](-P) \quad \text{for } m < 0.$$

Now we can define the *order* of a point $P \in E$ as the smallest positive integer k such that $[k]P = \mathcal{O}$. If no such integer exists, then the order is said to be infinite. If $[n]P = \mathcal{O}$ for $P \in E$, then P is called an *n-torsion point.* Remark that a point P is an n-torsion point if and only if its order divides n. The subgroup of n-torsion points in E is denoted by $E[n]$, so

$$E[n] = \{P \in E : [n]P = \mathcal{O}\}.$$

Further, we write $E(K)[n]$ for the subgroup of n-torsion points in $E(K)$, so

$$E(K)[n] = \{P \in E(K) : [n]P = \mathcal{O}\}.$$

Let E be an elliptic curve defined over $\mathbb{F}_q$. The number of points in $E(\mathbb{F}_q)$, called the order of the elliptic curve, is denoted by $\#E(\mathbb{F}_q)$. The *trace of Frobenius* or simply *trace* of a curve is the value t satisfying $\#E(\mathbb{F}_q) = q + 1 - t$. The elliptic curve $E/\mathbb{F}_q$ is said to be *supersingular* if the characteristic p of $\mathbb{F}_q$ divides t, and *non-supersingular* otherwise. In other words, the curve $E/\mathbb{F}_q$ is supersingular if $t \equiv 0 \pmod{p}$. The following is an important result by Hasse.

Theorem 2.1 (Hasse's bound; Silverman [76], Theorem 5.1.1) *The trace t of a curve satisfies $|t| \leq 2\sqrt{q}$.*

We now give a result on the group structure of E that we need later on. Throughout the report, we will write $\mathbb{Z}_n$ for $\mathbb{Z}/n\mathbb{Z}$.

Lemma 2.2 (Silverman [76], Corollary 3.6.4 and Theorem 5.3.1) *If n and q are relatively prime, then $E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n$. If $n = p^e$, then either $E[p^e] = \{\mathcal{O}\}$ if E is supersingular, or $E[p^e] \cong \mathbb{Z}_{p^e}$ if E is non-supersingular.*

Note that from Lemma 2.2 follows that for n coprime to q, the number of n-torsion points $\#E[n]$ is equal to n^2; a fact that will frequently be used. Moreover, if n is prime, then $E[n]$ is generated by any two linearly independent n-torsion points.

A rational mapping from E to E that is also a group homomorphism is called an *endomorphism*. The set of endomorphisms on E forms a group and is denoted by $\mathrm{End}(E)$. The multiplication-by-m map, sending a point P to $[m]P$, is an endomorphism. Further, on every elliptic curve defined over $\mathbb{F}_q$ there exists another endomorphism, known as the *Frobenius endomorphism* or *Frobenius map*. The Frobenius endomorphism $\Phi : E \to E$ sends a point (x, y) to (x^q, y^q) and fixes $\mathcal{O}$. Then $P \in E(\mathbb{F}_q)$ if and only if $\Phi(P) = P$.

2.2 Functions on an Elliptic Curve

Let $K = \mathbb{F}_q$ and let the curve E/K be given by the Weierstrass equation

$$E: \; y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{2.3}$$

Define the function $r \in K[x, y]$ by

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

then the *coordinate ring* $K[E]$ of E over K is defined as the integral domain

$$K[E] = K[x, y]/(r),$$

where (r) is the ideal in $K[x, y]$ generated by r. Similarly, $\overline{K}[E]$ is defined as

$$\overline{K}[E] = \overline{K}[x, y]/(r).$$

We can write $l \in \overline{K}[E]$ in *canonical form*

$$l(x, y) = v(x) + yw(x), \quad v(x), w(x) \in \overline{K}[x],$$

by using the relation (2.3) to knock down exponents of y that are greater than or equal to 2.

By taking the field of fractions of $K[E]$, we obtain the function field $K(E)$. Here, two elements f_1/g_1 and $f_2/g_2 \in K(E)$ are identified if $f_1g_2 = f_2g_1$. Again, we can do the same for $\overline{K}$, so $\overline{K}(E)$ is the field of fractions of $\overline{K}[E]$. An element of $\overline{K}(E)$ is called a *rational function*.

A non-zero rational function $f \in \overline{K}(E)^*$ is said to be defined at a point $P \in E \setminus \{\mathcal{O}\}$ if f can be written as $f = g/h$, for $g, h \in \overline{K}[E]$, with $h(P) \neq 0$. In that case, the evaluation of f at P, given by $f(P) = g(P)/h(P)$, is well-defined. Further, f is said to have a *zero* at P if $f(P) = 0$, and f is said to have a *pole* at P (denoted by $f(P) = \infty$) if f is not defined at P. We will deal with the multiplicity of the zeros and poles in the next section. Note that a non-zero rational function can only have a finite number of zeros and poles.

To determine the value of a rational function at the point $\mathcal{O}$, it seems natural to compare the degrees of the denominator and the enumerator. However, when trying to find the degree of a function, we cannot simply assign a weight of 1 to both variables x and y, as this would violate the relation $y^2 = x^3 + ax + b$. Therefore, we define x and y to have weight 2 and 3, respectively. Then, for nonzero $g(x, y) = v(x) + yw(x) \in \overline{K}[E]$, the degree of g is given by

$$\deg(g) = \max\{2 \cdot \deg_x(v), 3 + 2 \cdot \deg_x(w)\},$$

where $\deg_x(p)$ denotes the usual degree of a polynomial p in the variable x. Now let $f = g/h \in \overline{K}(E)$. If $\deg(g) < \deg(h)$, then $f(\mathcal{O}) = 0$. If $\deg(g) > \deg(h)$, then f is said to have a pole at $\mathcal{O}$, denoted $f(\mathcal{O}) = \infty$. If $\deg(g) = \deg(h)$, then $f(\mathcal{O}) = a/b$, where a and b are the coefficients of the leading terms of g and h, written in canonical form, respectively.

Example 2.3 Consider the curve $E/K : y^2 = x^3 + 3x$, where $K = \mathbb{F}_{11}$, and the rational function $f = \frac{y+x+1}{x+8} \in K(E)$. Then f can be written as g/h with $g = y + x + 1$ and $h = x + 8$. Now $\deg(g) = 3$ and $\deg(h) = 2$, so f has a pole at $\mathcal{O}$ and we write $f(\mathcal{O}) = \infty$.

2.3 Multiplicity of Zeros and Poles

Recall that a rational function $f \in \overline{K}(E)$ on an elliptic curve E is said to have a zero at a point $P \in E$ if $f(P) = 0$ and a pole at P if $f(P) = \infty$. We next describe how to find the multiplicity of the zeros and poles of a function.

For every point $P \in E$, there exists a rational function u with $u(P) = 0$, satisfying the following property: every non-zero rational function $f \in \overline{K}(E)^*$ can be written as

$$f = u^d s,$$

where $s \in \overline{K}(E)$, $s(P) \neq 0, \infty$, for some integer d. A function u with this property is called a *uniformizing parameter* for P. The value of d does not depend on the choice of u. We can find a uniformizing parameter using the following theorem:

Theorem 2.4 (Menezes [54], Theorem 2.22) *Let $P \in E$. If $l : ax + by + c = 0$ is any line through P that is not tangent to E at P, then l is a uniformizing parameter for P.*

Searching for a uniformizing parameter, we can distinguish between three general cases, namely

$$P \notin E[2], \quad P \in E[2] \setminus \{\mathcal{O}\}, \quad \text{and} \quad P = \mathcal{O}.$$

Let $P = (c, d) \notin E[2]$, then $u = x - c$ is a line through P not tangent to E, so $x - c$ is a uniformizing parameter for P. If $P = (c, 0) \in E[2] \setminus \{\mathcal{O}\}$, then a line through P not tangent to E, and thus a uniformizing parameter for P, is given by $u = y$. Finally, after switching to a different set of parameters, one can see that $u = x/y$ is a uniformizing parameter for $P = \mathcal{O}$ (for details see [54]).

Let u be a uniformizing parameter for $P \in E$. The non-zero rational function $f \in \overline{K}(E)^*$ can be decomposed as $f = u^d s$, where $s \in \overline{K}(E)$ and $s(P) \neq 0, \infty$. Then the *order of f at P*, denoted $\mathrm{ord}_P(f)$, equals d. If P is a zero of f, then $\mathrm{ord}_P(f) > 0$ and the zero is said to have *multiplicity* $\mathrm{ord}_P(f)$. In case P is a pole, then $\mathrm{ord}_P(f) < 0$ and *the multiplicity of the pole* is defined as $-\mathrm{ord}_P(f)$. Remark that if P is neither a zero nor a pole, then $\mathrm{ord}_P(f) = 0$.

Example 2.5 Recall from Example 2.3 that the rational function $f = \frac{y+x+1}{x+8} \in K(E)$ on the curve $E/K : y^2 = x^3 + 3x$ with $K = \mathbb{F}_{11}$ has a pole at the point at infinity $\mathcal{O}$. A uniformizing parameter for $\mathcal{O}$ is $u = x/y$. Then write

$$f = u^d s = (x/y)^d s = \frac{y + x + 1}{x + 8}, \quad \text{so} \quad s = \frac{y^d(y + x + 1)}{x^d(x + 8)}.$$

Then for $d = -1$, we see that $s = \frac{x(y+x+1)}{y(x+8)}$, which is equal to one at $\mathcal{O}$. Hence, $\mathrm{ord}_{\mathcal{O}}(f) = -1$ and we say that f has a pole of multiplicity one at $\mathcal{O}$.

2.4 Divisor Theory

Up to a constant multiple, a rational function is uniquely determined by its zeros and poles. Therefore, it makes sense to construct a tool that can be used to record these special points of a function. As we are dealing with points on an elliptic curve and their multiplicities, finitely many of which are unequal to 0, it seems natural to use a formal sum. So, define a divisor D as a formal sum of points

$$D = \sum_{P \in E} n_P(P),$$

where $n_P \in \mathbb{Z}$ and $n_P = 0$ for all but finitely many $P \in E$. The group of divisors of E, denoted $\mathrm{Div}(E)$, is the free abelian group generated by the points of E, where addition is given by

$$\sum_{P \in E} n_P(P) + \sum_{P \in E} m_P(P) = \sum_{P \in E} (n_P + m_P)(P).$$

The *support of a divisor* $D = \sum_{P \in E} n_P(P) \in \mathrm{Div}(E)$ is given by the set of points

$$\mathrm{supp}(D) = \{P \in E | n_P \neq 0\}.$$

Further, its *degree* $\deg(D)$ is defined by

$$\deg(D) = \sum_{P \in E} n_P.$$

It is easily verified that the divisors of degree 0, denoted $\mathrm{Div}^0(E)$, form a subgroup of $\mathrm{Div}(E)$.

Since the number of zeros and poles of a non-zero rational function $f \in \overline{K}(E)^*$ is finite, we can now define the *divisor of a function* f, denoted $\mathrm{div}(f)$, as

$$\mathrm{div}(f) = \sum_{P \in E} \mathrm{ord}_P(f)(P).$$

Note that $\mathrm{div}(f) = 0$ if and only if f is a constant function.

A divisor $D \in \mathrm{Div}(E)$ is called *principal* if $D = \mathrm{div}(f)$ for some rational function f. Further, two divisors $D_1, D_2 \in \mathrm{Div}(E)$ are said to be *(linearly) equivalent*, denoted $D_1 \sim D_2$, if $D_1 - D_2$ is principal, i.e. we can write $D_1 = D_2 + \mathrm{div}(f)$ for some rational function f. The following is an important theorem in divisor theory.

Theorem 2.6 (Silverman [76], Corollary 3.3.5) *Let $D = \sum_{P \in E} n_P(P)$ be a divisor. Then D is principal if and only if*

$$\sum_{P \in E} n_P = 0 \quad \textit{and} \quad \sum_{P \in E} [n_P]P = \mathcal{O}.$$

Consider the following theorem, regarding the equivalence with divisors of a certain form:

Theorem 2.7 (Lynn [50]) *Let $D \in Div(E)$. Then there exists a unique point $P \in E$ such that*

$$D \sim (P) + (deg(D) - 1)(\mathcal{O}).$$

As a consequence, if D is a degree 0 divisor, then it is equivalent to $(P) - (\mathcal{O})$ for some point P.

The set of all principal divisors is denoted $\mathrm{Prin}(E)$. The quotient group

$$\mathrm{Pic}(E) = \mathrm{Div}(E)/\mathrm{Prin}(E),$$

which represents the set of divisors that are not principal, is called the *Picard group* or *divisor class group*. From the fact that

$$\mathrm{div}(f_1 f_2) = \mathrm{div}(f_1) + \mathrm{div}(f_2)$$

for all $f_1, f_2 \in \overline{K}(E)$ follows that $\mathrm{Prin}(E)$ forms a subgroup of $\mathrm{Div}^0(E)$. Hence, we can define the group called the *degree zero part* of the Picard group, or divisor class group of E, as follows:

$$\mathrm{Pic}^0(E) = \mathrm{Div}^0(E)/\mathrm{Prin}(E).$$

We next describe how to evaluate a rational function $f \in \overline{K}(E)$ in a divisor $D = \sum_{P \in E} n_P(P)$ that satisfies $\mathrm{supp}(\mathrm{div}(f)) \cap \mathrm{supp}(D) = \emptyset$. The *evaluation of f in D* is given by

$$f(D) = \prod_{P \in \mathrm{supp}(D)} f(P)^{n_P}.$$

This evaluation is well-defined because D and $\mathrm{div}(f)$ have disjoint support. For future reference we give the following lemma.

Lemma 2.8 *Let $D \in Div^0(E)$ be a degree zero divisor and $f_1 \in \overline{K}(E)$ a rational function such that $supp(div(f_1)) \cap supp(D) = \emptyset$. Let $c \in \overline{K}^*$, then the rational function $f_2 = cf_1$ satisfies*

$$f_2(D) = f_1(D).$$

Proof. For brevity, we write $\mathcal{D} = \text{supp}(D)$. Observe that $\text{supp}(\text{div}(f_1)) = \text{supp}(\text{div}(f_2))$, so $\text{div}(f_2)$ has support disjoint from $\mathcal{D}$. Write $D = \sum_{P \in E} n_P(P)$, where $\sum_{P \in E} n_P = \sum_{P \in \mathcal{D}} n_P = 0$. Then

$$f_2(D) = \prod_{P \in \mathcal{D}} f_2(P)^{n_P} = \prod_{P \in \mathcal{D}} (cf_1(P))^{n_P} = c^{\sum_{P \in \mathcal{D}} n_P} \prod_{P \in \mathcal{D}} f_1(P)^{n_P} = \prod_{P \in \mathcal{D}} f_1(P)^{n_P} = f_1(D).$$

□

We conclude this section with a result from Weil, which is an important tool in the study of bilinear maps on elliptic curves.

Theorem 2.9 (Weil's reciprocity law) *Let $f, g \in \overline{K}(E)$. Then*

$$f(div(g)) = g(div(f)).$$

See [34] or [35] for a proof for curves that are not necessarily elliptic, or [17] for a proof for elliptic curves specifically.

2.5 Computing the Function of a Principal Divisor

Recall that for any degree zero divisor $D \in \text{Div}^0(E)$, there is a unique point $P \in E$ such that $D \sim (P) - (\mathcal{O})$. In other words, D can be written in what we will call *canonical form*:

$$D = (P) - (\mathcal{O}) + \text{div}(f), \tag{2.4}$$

where $f \in \overline{K}(E)$ is uniquely determined up to a constant multiple. For given degree zero divisor D, we will next show how to compute P and f in (2.4) (see [54, 58]). For this purpose, we will first give an explicit formula for adding two divisors in canonical form, such that the result is in canonical form as well.

Let $D_1, D_2 \in \text{Div}^0(E)$ be given by

$$\begin{aligned} D_1 &= (P_1) - (\mathcal{O}) + \text{div}(f_1), \\ D_2 &= (P_2) - (\mathcal{O}) + \text{div}(f_2). \end{aligned}$$

Let $P_1 + P_2 = P_3$, and let $l : l_1 y + l_2 x + l_3 = 0$ be the equation of the line through P_1 and P_2 and $v : x + v_1 = 0$ the vertical line through P_3. If $P_1 = P_2$ then l is the line tangent to P_1, and if $P_3 = \mathcal{O}$ then take $v = 1$. Then

$$\text{div}(l) = (P_1) + (P_2) + (-P_3) - 3(\mathcal{O}),$$

and

$$\text{div}(v) = (P_3) + (-P_3) - 2(\mathcal{O}).$$

Now we can write the sum of divisors $D_1 + D_2$ as:

$$\begin{aligned} D_1 + D_2 &= (P_1) + (P_2) - 2(\mathcal{O}) + \text{div}(f_1 f_2) \\ &= (P_3) - (\mathcal{O}) + \text{div}(l) - \text{div}(v) + \text{div}(f_1 f_2) \\ &= (P_3) - (\mathcal{O}) + \text{div}(f_1 f_2 f_3), \end{aligned}$$

where $f_3 = l/v$. Note that f_3, regarded as element of $\overline{K}(x, y)$, is defined at all points except for P_3 and $-P_3$, since the function v is zero at P_3 and $-P_3$.

We describe a method for writing the principal divisor $D = \sum_{i=1}^{n} a_i(P_i) \in \mathrm{Prin}(E)$ as the divisor of a function $f \in \overline{K}(E)$. Since $\deg(D) = 0$, we can write

$$D = \sum_{i=1}^{n} a_i(P_i) = \sum_{i=1}^{n} a_i((P_i) - (\mathcal{O})).$$

Before we proceed, we need the notion of an addition chain. Let $a \in \mathbb{N}$, then an *addition chain* for a is a sequence $1 = d_1, d_2, \ldots, d_t = a$, such that each d_j, $2 \leq j \leq t$, can be written as $d_j = d_k + d_l$, where $k, l < j$. Now let $d_1^{(i)}, \ldots, d_{t_i}^{(i)}$ be an addition chain for a_i. Then for each $i = 1, \ldots, n$ we use the method described above to successively express $d_j^{(i)}((P_i) - (\mathcal{O}))$ for $j = 1, \ldots, t_i$ in canonical form. Let the canonical form for the summands $a_i((P_i) - (\mathcal{O}))$ be given by $(P_i') - (\mathcal{O}) + \mathrm{div}(f_i)$. Finally, we add the divisors $(P_i') - (\mathcal{O}) + \mathrm{div}(f_i)$ for $1 \leq i \leq n$.

Menezes [54] argues that if we keep f in factored form, then f will be of polynomial size and its computation takes polynomial time. Provided that f (as element of $\overline{K}(x, y)$) is defined at a point P, also the evaluation of $f(P)$ takes polynomial time. By construction, f is undefined at most at the points $\pm Q_j$, where the intermediate divisors are given by $D_j = (Q_j) - (\mathcal{O}) + \mathrm{div}(g_j)$.

2.5.1 Example

Consider the curve $E/K : y^2 = x^3 + 3x$, where $K = \mathbb{F}_{11}$. Table 2.1 shows the points on $E(K)$, along with their orders. Since the number of points $\#E(\mathbb{F}_{11}) = 12$, we have $t = q + 1 - \#E(\mathbb{F}_{11}) = 0$, so E is supersingular.

Point	Order	Point	Order
$P_0 = \mathcal{O}$	1	$P_6 = (3,5)$	3
$P_1 = (0,0)$	2	$P_7 = (3,6)$	3
$P_2 = (1,2)$	6	$P_8 = (6,5)$	4
$P_3 = (1,9)$	6	$P_9 = (6,6)$	4
$P_4 = (2,5)$	12	$P_{10} = (7,1)$	12
$P_5 = (2,6)$	12	$P_{11} = (7,10)$	12

Table 2.1: $\mathbb{F}_{11}$-rational points on $E : y^2 = x^3 + 3x$

The divisor $D = 6(P_2) - 6(\mathcal{O})$ is clearly principal, because the order of P_2 is 6. An addition chain for 6 is given by $1, 2, 4, 6$. Then the rational function f with $\mathrm{div}(f) = D$ is computed as follows:

$$\begin{aligned}
(P_2) - (\mathcal{O}) &= (P_2) - (\mathcal{O}) + \mathrm{div}(1). \\
2(P_2) - 2(\mathcal{O}) &= \big((P_2) - (\mathcal{O})\big) + \big((P_2) - (\mathcal{O})\big) \\
&= (P_7) - (\mathcal{O}) + \mathrm{div}\Big(\frac{y + 4x + 5}{x + 8}\Big). \\
4(P_2) - 4(\mathcal{O}) &= \big(2(P_2) - 2(\mathcal{O})\big) + \big(2(P_2) - 2(\mathcal{O})\big) \\
&= (P_6) - (\mathcal{O}) + \mathrm{div}\Big(\frac{(y + 4x + 5)^2}{(x + 8)^2}\,\frac{(y + 3x + 7)}{(x + 8)}\Big). \\
6(P_2) - 6(\mathcal{O}) &= \big(2(P_2) - 2(\mathcal{O})\big) + \big(4(P_2) - 4(\mathcal{O})\big) \\
&= \mathrm{div}\Big(\frac{(y + 4x + 5)^3}{(x + 8)^3}\,\frac{(y + 3x + 7)}{(x + 8)}\,\frac{(x + 8)}{1}\Big).
\end{aligned}$$

The rational function f is given by

$$f = \frac{(y + 4x + 5)^3 (y + 3x + 7)}{(x + 8)^3}.$$

Considered as an element of $\overline{K}(x,y)$, f is undefined at the points P_6 and P_7. However, regarded as a rational function (i.e. an element of $\overline{K}(E)$), f can be rewritten as follows:

$$\begin{aligned} f &= \frac{(y+4x+5)^3(y+3x+7)}{(x+8)^3} \cdot \frac{(y-4x-5)^3}{(y-4x-5)^3} \\ &= \frac{(y^2+6x^2+4x+8)^3}{(x+8)^3} \cdot \frac{(y+3x+7)}{(y-4x-5)^3} \\ &= \frac{(x^3+6x^2+7x+8)^3}{(x+8)^3} \cdot \frac{(y+3x+7)}{(y-4x-5)^3} \\ &= \frac{(x+8)^3(x+10)^6}{(x+8)^3} \cdot \frac{(y+3x+7)}{(y-4x-5)^3} \\ &= \frac{(x+10)^6(y+3x+7)}{(y-4x-5)^3}, \end{aligned}$$

which is actually defined at $P_6 = (3,5)$. Similarly,

$$\begin{aligned} f &= \frac{(y+4x+5)^3(y+3x+7)}{(x+8)^3} \cdot \frac{(y-3x-7)}{(y-3x-7)} \\ &= \frac{(y+4x+5)^3}{(x+8)^3} \cdot \frac{(x^3+2x^2+5x+6)}{(y-3x-7)} \\ &= \frac{(y+4x+5)^3}{(x+8)^3} \cdot \frac{(x+8)^3}{(y-3x-7)} \\ &= \frac{(y+4x+5)^3}{(y-3x-7)}, \end{aligned}$$

which is in turn defined at $P_7 = (3,6)$. Thus, being a rational function, f is defined at both P_6 and P_7.

Chapter 3

Weil Pairing

This chapter is dedicated to the Weil pairing. We start in Section 3.1 with the definition of this pairing as given in [76]. We then discuss some properties that are important for the practical use of the Weil pairing in cryptography in Section 3.2. Section 3.3 deals with another, closely related definition of the Weil pairing. This alternative definition makes it possible to efficiently compute the pairing; this is shown in Section 3.4.

3.1 Definition

For the definition of the Weil pairing, we need the notion of the ramification index, which is defined as follows:

Definition 3.1 *Let $F : E \to E$ be a nonconstant rational mapping, $P \in E$ and u a uniformizing variable for $F(P)$. Then the ramification index of F at P is defined by*

$$e_F(P) = ord_P(u \circ F).$$

By the definition of the order of a point, we see that $e_F(P)$ is independent of the choice of u. Further, since u is a uniformizing variable for $F(P)$, it follows that $u \circ F$ has a zero at P, and thus $e_F(P) \geq 1$. A result from [18] is that if F is an endomorphism, then $e_F(P)$ is constant for all P and $e_F(P)$ is denoted by e_F. In particular, if F is the multiplication-by-m map for some integer m, then $e_F = 1$.

Another definition we need is the following:

Definition 3.2 *Let $F : E \to E$ be a nonconstant rational mapping. The homomorphism $F^* : Div(E) \to Div(E)$ is given by*

$$F^*((Q)) = \sum_{F(P)=Q} e_F(P) \cdot (P).$$

Though this definition might seem rather artificial, it is constructed such that the following identity holds:

$$\mathrm{div}(r \circ F) = F^*(\mathrm{div}(r)), \tag{3.1}$$

where r is a nonzero rational function. The proof that equation (3.1) holds, uses the fact that

$$\mathrm{ord}_P(r \circ F) = (\mathrm{ord}_{F(P)} r)(e_F(P))$$

(for details see [18]).

With these definitions out of the way, we are now ready to describe the setting for the Weil pairing. Recall that a divisor $D = \sum_{P \in E} n_P(P)$ is principal if and only if $\deg(D) = 0$ and

$\sum_{P\in E}[n_P]P = \mathcal{O}$. Let $m \geq 2$ be a fixed integer coprime to $p = \mathrm{char}(K)$. Suppose $T \in E[m]$, then it easily follows that the divisor $m(T) - m(\mathcal{O})$ is principal. Let $f \in \overline{K}(E)$ be a function such that

$$\mathrm{div}(f) = m(T) - m(\mathcal{O}).$$

Now consider the divisor $[m]^*(T) - [m]^*(\mathcal{O})$. By Definition 3.2, this divisor can be written as

$$\sum_{[m]P=T} e_{[m]}(P)\cdot(P) - \sum_{[m]P=\mathcal{O}} e_{[m]}(P)\cdot(P). \tag{3.2}$$

Let $T' \in E$ be such that $[m]T' = T$. Such T' always exists, because we are working over an algebraically closed field $\overline{K}$. Since $e_{[m]} = 1$, we can rephrase (3.2) as:

$$\sum_{R\in E[m]} (T'+R) - (R),$$

which clearly has degree 0. Moreover, since $\#E[m] = m^2$ by Lemma 2.2 and $[m^2]T' = \mathcal{O}$, we find that

$$\sum_{R\in E[m]} T' + R - R = [m^2]T' = \mathcal{O}.$$

Hence, the divisor $[m]^*(T) - [m]^*(\mathcal{O})$ is principal and there is a function $g \in \overline{K}(E)$ with

$$\mathrm{div}(g) = [m]^*(T) - [m]^*(\mathcal{O}).$$

Now the definition of the Weil pairing (as given in [76]) is the following:

Definition 3.3 *With the notation as introduced above, the Weil e_m-pairing*

$$e_m : E[m] \times E[m] \to \mu_m$$

is given by:

$$e_m(S,T) = \frac{g(X+S)}{g(X)}, \tag{3.3}$$

where $X \in E$ is any point such that $g(X+S), g(X) \neq 0, \infty$, and μ_m is the group of mth roots of unity in $\overline{K}$.

Below we prove that the Weil pairing is well-defined.

Theorem 3.4 *The Weil pairing e_m is well-defined, i.e. maps to an mth root of unity and does not depend on the choice of function g and point X.*

Proof. Let $S, T \in E[m]$. Note that

$$\mathrm{div}(f\circ[m]) = [m]^*\mathrm{div}(f) = [m]^*(m((T)-(\mathcal{O}))) = m\cdot\mathrm{div}(g) = \mathrm{div}(g^m),$$

so $g^m = c\cdot(f\circ[m])$ for some $c \in \overline{K}^*$. Now, for any $X \in E$:

$$g(X+S)^m = cf([m]X+[m]S) = cf([m]X) = g(X)^m,$$

which shows that

$$e_m(S,T)^m = \frac{g(X+S)^m}{g(X)^m} = 1,$$

and thus $e_m(S,T)$ is indeed an mth root of unity. Moreover, $e_m(S,T)$ does not depend on the choice of g, as g is unique up to a constant multiple.

We next show that the outcome of the pairing is irrespective of the choice of X. Let $\mathcal{T}_P : E \to E$ be the 'translation-by-P' map, which sends the point Q to $Q+P$. Then expression (3.3) can be written as

$$e_m(S,T) = \frac{g \circ \mathcal{T}_S}{g}.$$

Lemma 13.3 in [18] shows that for $S, T \in E[m]$, we have $\mathrm{div}(g \circ \mathcal{T}_S) = \mathrm{div}(g)$. Hence $(g \circ \mathcal{T}_S)/g$ is a constant and thus $e_m(S,T) = g(X+S)/g(X)$ does not depend on the choice for X (provided that g is defined at X and $X+S$). □

3.2 Properties

The next theorem states that the Weil pairing satisfies some important properties ([76, 18]). For comprehensibility, we include the proof as given in [76].

Theorem 3.5 (Silverman [76], Proposition 3.8.1) *Let $S_1, S_2, S, T_1, T_2, T \in E[m]$. The Weil pairing as defined in Definition 3.3 satisfies the following properties:*

1. $e_m(S_1+S_2,T) = e_m(S_1,T)e_m(S_2,T)$ *(linearity in the first factor).*
2. $e_m(S,T_1+T_2) = e_m(S,T_1)e_m(S,T_2)$ *(linearity in the second factor).*
3. $e_m(S,S) = 1$ *(identity).*
4. $e_m(S,T) = e_m(T,S)^{-1}$ *(alternation).*
5. *If $e_m(S,T) = 1$ for all $S \in E[m]$, then $T = \mathcal{O}$ (non-degeneracy).*

Proof.

1. For linearity in the first factor, choose X such that g is defined at X, $X+S_1$ and $X+S_1+S_2$. Then

$$e_m(S_1+S_2,T) = \frac{g(X+S_1+S_2)}{g(X)} = \frac{g(X+S_1+S_2)g(X+S_1)}{g(X+S_1)g(X)} = e_m(S_2,T)e_m(S_1,T).$$

2. For linearity in the second factor, let $f_1, f_2, f_3, g_1, g_2, g_3$ be functions as above for T_1, T_2, and $T_3 = T_1+T_2$. Choose $h \in \overline{K}(E)$ with divisor

$$\mathrm{div}(h) = (T_1+T_2) - (T_1) - (T_2) + (\mathcal{O}).$$

Then

$$\mathrm{div}(f_3/f_1f_2) = m\mathrm{div}(h),$$

so

$$f_3 = cf_1f_2h^m \quad \text{for some } c \in \overline{K}^*.$$

Compose with the multiplication-by-m map, use $f_i \circ [m] = g_i^m$, and take mth roots to find

$$g_3 = c'g_1g_2(h \circ [m]) \quad \text{for some } c' \in \overline{K}^*.$$

Now

$$e_m(S,T_1+T_2) = \frac{g_3(X+S)}{g_3(X)} = \frac{g_1(X+S)g_2(X+S)}{g_1(X)g_2(X)}\frac{h([m]X+[m]S)}{h([m]X)} = e_m(S,T_1)e_m(S,T_2).$$

3. For the identity, observe that:

$$\operatorname{div}\Big(\prod_{i=0}^{m-1} f \circ \mathcal{T}_{[i]T}\Big) = \sum_{i=0}^{m-1} \operatorname{div}(f \circ \mathcal{T}_{[i]T}) = m \sum_{i=0}^{m-1} ([1-i]T) - ([-i]T) = 0.$$

Hence $\prod_{i=0}^{m-1} f \circ \mathcal{T}_{[i]T}$ is constant; and if we choose some $T' \in E$ with $[m]T' = T$, then $\prod_{i=0}^{m-1} g \circ \mathcal{T}_{[i]T'}$ is also constant, because its mth power is the above product of f's. Evaluating the product of g's at X and $X + T'$ yields

$$\prod_{i=0}^{m-1} g(X + [i]T') = \prod_{i=0}^{m-1} g(X + [i+1]T').$$

Now cancelling like terms gives

$$g(X) = g(X + [m]T') = g(X + T),$$

so

$$e_m(T, T) = g(X + T)/g(X) = 1.$$

4. To prove alternation, we combine properties 1.-4.:

$$1 = e_m(S + T, S + T) = e_m(S, S)e_m(S, T)e_m(T, S)e_m(T, T) = e_m(S, T)e_m(T, S),$$

so $e_m(S, T) = e_m(T, S)^{-1}$.

5. Non-degeneracy: if $e_m(S, T) = 1$ for all $S \in E[m]$, so $g(X + S) = g(X)$ for all $S \in E[m]$, then from Theorem 3.4.10 in [76] follows that $g = h \circ [m]$ for some function $h \in \overline{K}(E)$. But then

$$(h \circ [m])^m = g^m = f \circ [m],$$

so $f = ch^m$ for some constant $c \in \overline{K}^*$. Hence

$$m\operatorname{div}(h) = \operatorname{div}(f) = m(T) - m(\mathcal{O}),$$

so

$$\operatorname{div}(h) = (T) - (\mathcal{O}).$$

Therefore $T = \mathcal{O}$. □

Consider the following corollary, concerning the Weil pairing of two independent m-torsion points.

Corollary 3.6 *Let m be a prime and S and T be two linearly independent m-torsion points. Then the Weil pairing of S and T is non-trivial, i.e. $e_m(S, T) \neq 1$.*

Proof. By the non-degeneracy of the Weil pairing, there exists an m-torsion point S', independent from S, such that $e_m(S, S') \neq 1$. Since m is prime, $e_m(S, S')$ is a primitive mth root of unity in $\overline{K}$. Moreover, because S and S' are independent, the group of m-torsion points is generated by S and S'. Hence, we can write T as $T = [a]S + [b]S'$ for some $0 \leq a \leq m-1$ and $1 \leq b \leq m-1$. Then

$$e_m(S, T) = e_m(S, [a]S + [b]S') = e_m(S, S)^a \cdot e_m(S, S')^b = e_m(S, S')^b \neq 1,$$

where the last inequality follows from the fact that $e_m(S, S')$ is a primitive mth root of unity. □

3.3 Alternative Definition

Besides the definition given in Section 3.1, there is another, closely related definition, which makes it possible to efficiently compute the Weil pairing. For the time being, we denote this alternative Weil pairing by e'_m. The definition as stated in [54] is as follows. Fix an integer m coprime to p, and let $S,T \in E[m]$. Let A and B be divisors, such that $A \sim (S) - (\mathcal{O})$ and $B \sim (T) - (\mathcal{O})$, and A and B have disjoint support. Since S and T are m-torsion points, it easily follows that mA and mB are principle divisors. So there are $f_A, f_B \in \overline{K}(E)$ such that

$$\mathrm{div}(f_A) = mA \quad \text{and} \quad \mathrm{div}(f_B) = mB.$$

Definition 3.7 *With the notation as introduced above, the alternative Weil pairing*

$$e'_m : E[m] \times E[m] \to \mu_m$$

is given by:

$$e'_m(S,T) = \frac{f_A(B)}{f_B(A)}.$$

The following theorem asserts that the outcome of the alternative Weil pairing is indeed an mth root of unity and does not depend on the choice of divisors A and B and functions f_A and f_B.

Theorem 3.8 *The Weil pairing e'_m as defined in Definition 3.7 is well-defined, i.e. maps to an mth root of unity and is independent of the choice of divisors A and B and functions f_A and f_B.*

Proof. Observe that by Weil's reciprocity law

$$\Big(\frac{f_A(B)}{f_B(A)}\Big)^m = \frac{f_A(B)^m}{f_B(A)^m} = \frac{f_A(mB)}{f_B(mA)} = \frac{f_A(\mathrm{div}(f_B))}{f_B(mA)} = \frac{f_B(\mathrm{div}(f_A))}{f_B(m_A)} = \frac{f_B(m_A)}{f_B(m_A)} = 1,$$

which shows that $e'_m(S,T)$ is an mth root of unity.

Consider the Weil pairing $e'_m(S,T) = f_A(B)/f_B(A)$. Let $B' \sim (T) - (\mathcal{O})$ be a divisor such that A and B' have disjoint support, and let $f_{B'}$ be a divisor such that $\mathrm{div} f_{B'} = mB'$. Since $B \sim B'$, we can write $B' = B + \mathrm{div}(h)$ and thus $f_{B'} = f_B h^m$ for some $h \in E(\overline{K})$. Then

$$\frac{f_A(B')}{f_{B'}(A)} = \frac{f_A(B) f_A(\mathrm{div}(h))}{f_B(A) h^m(A)} = \frac{f_A(B) f_A(\mathrm{div}(h))}{f_B(A) h(mA)} = \frac{f_A(B)}{f_B(A)} \frac{f_A(\mathrm{div}(h))}{h(\mathrm{div}(f_A))} = \frac{f_A(B)}{f_B(A)},$$

where the last equality holds by Weil's reciprocity law. This proves that the pairing does not depend on the choice of B; an analogous argument can be used to prove the same for A.

Furthermore, functions f_A and f_B are unique up to a constant. Since both f_A and f_B are being evaluated in a degree zero divisor, Lemma 2.8 implies that outcome of the pairing is irrespective of the choice of these constants. □

Although sometimes claimed in the literature ([76, 54]), the two definitions of the Weil pairing are not completely equivalent, as remarked in [17] and [37]. To be precise, the definitions turn out to satisfy the relation $e_m(S,T) = 1/e'_m(S,T) = e'_m(T,S)$, which is proven below. See also [35], or [37] for a sophisticated proof of this relation for the more general version of the Weil pairing on algebraic curves.

Theorem 3.9 *Let $S,T \in E[m]$. Then $e_m(S,T) = 1/e'_m(S,T)$.*

Proof. First of all, note that f in Definition 3.3 equals f_B in Definition 3.7. Recall that T' satisfies $[m]T' = T$; let $S' \in E$ be such that $[m]S' = S$. Similar to g in Definition 3.3, there is a g_A that satisfies

$$\operatorname{div}(g_A) = [m]^*(S) - [m]^*(\mathcal{O}) = \sum_{R \in E[m]} (S' + R) - (R),$$

and thus $g_A^m = f_A \circ [m]$.

Let $X \in E$ be such that the divisor

$$D = (m-1)(S' + X) + (S' - S + X) - m(X)$$

has support disjoint from $\operatorname{supp}(\operatorname{div}(g))$. Clearly,

$$\deg(D) = 0 \quad \text{and} \quad [m-1]S' + [m-1]X + S' - S + X - [m]X = \mathcal{O},$$

so D is a principal divisor. Let $h \in \overline{K}(E)$ such that $\operatorname{div}(h) = D$. Now apply Weil's reciprocity law to g and h:

$$g(\operatorname{div}(h)) = h(\operatorname{div}(g)).$$

Then

$$\begin{aligned}
g(\operatorname{div}(h)) &= \frac{g(S' + X)^{m-1} \cdot g(S' - S + X)}{g(X)^m} \\
&= \frac{g(S' + X)^m}{g(X)^m} \cdot \frac{g(S' - S + X)}{g(S' + X)} \\
&= \frac{f \circ [m](S' + X)}{f \circ [m](X)} \cdot \frac{g(S'')}{g(S'' + S)} \quad (\text{where } S'' = S' - S + X) \\
&= \frac{f(S + [m]X)}{f([m]X)} \cdot \frac{1}{e_m(S, T)},
\end{aligned}$$

where $e_m(S, T) = g(S'' + S)/g(S'')$ is the Weil pairing as in Definition 3.3.

On the other hand,

$$h(\operatorname{div}(g)) = \prod_{R \in E[m]} \frac{h(T' + R)}{h(R)}, \quad \text{since } \operatorname{div}(g) = \sum_{R \in E[m]} (T' + R) - (R).$$

Define the function $H \in \overline{K}(E)$ as

$$H(P) = \prod_{R \in E[m]} h(P + R) = \prod_{R \in E[m]} h \circ \mathcal{T}_R(P).$$

Then

$$\begin{aligned}
\operatorname{div}(H) &= \sum_{R \in E[m]} \operatorname{div}(h \circ \mathcal{T}_R) \\
&= \sum_{R \in E[m]} \big((m-1)(S' + X - R) + (S' - S + X - R) - m(X - R)\big) \\
&= \sum_{R \in E[m]} \big((m-1)(S' + X + R) + (S' + X + R) - m(X + R)\big) \\
&= m \sum_{R \in E[m]} (S' + X + R) - (X + R) \\
&= m \operatorname{div}(g_A \circ \mathcal{T}_{-X}) \\
&= \operatorname{div}(g_A^m \circ \mathcal{T}_{-X}).
\end{aligned}$$

Hence we can write

$$H = g_A^m \circ \mathcal{T}_{-X} = f_A \circ [m] \circ \mathcal{T}_{-X},$$

and thus

$$\begin{aligned} \prod_{R \in E[m]} h(T' + R) &= H(T') \\ &= f_A \circ [m] \circ \mathcal{T}_{-X}(T') \\ &= f_A \circ [m](T' - X) \\ &= f_A(T - [m]X). \end{aligned}$$

Further,

$$\begin{aligned} \prod_{R \in E[m]} h(R) &= H(\mathcal{O}) \\ &= f_A \circ [m] \circ \mathcal{T}_{-X}(\mathcal{O}) \\ &= f_A(-[m]X). \end{aligned}$$

Finally, by Weil's reciprocity law,

$$h(\mathrm{div}(g)) = \prod_{R \in E[m]} \frac{h(T' + R)}{h(R)} = \frac{f_A(T - [m]X)}{f_A(-[m]X)}$$

equals

$$g(\mathrm{div}(h)) = \frac{f(S + [m]X)}{f([m]X)} \cdot \frac{1}{e_m(S, T)}.$$

Then, since $f = f_B$,

$$e_m(S, T) = \frac{f_A(-[m]X)}{f_A(T - [m]X)} \frac{f_B(S + [m]X)}{f_B([m]X)} = \frac{f_B(A)}{f_A(B)} = \frac{1}{e'_m(S, T)},$$

where $A \sim (S) - (\mathcal{O})$ and $B \sim (T) - (\mathcal{O})$. □

The relation $e_m(S, T) = e'_m(T, S)$ implies that also the alternative definition of the Weil pairing $e'_m(S, T)$ (as in Definition 3.7) satisfies the properties listed in Theorem 3.5. Nonetheless, we next prove the bilinearity directly for this definition, in order to get some insight in the construction of the (alternative) Weil pairing. We do not handle the identity and alternation properties, as they are trivial. Remarkably, no proof of the non-degeneracy that directly uses the alternative definition seems to be known.

Theorem 3.10 *The alternative Weil pairing as defined in Definition 3.7 is bilinear.*

Proof. We prove linearity in the first factor; linearity in the second goes analogously. Write $e'_m(S_1 + S_2, T) = f_A(B)/f_B(A)$, where

$$A \sim (S_1 + S_2) - (\mathcal{O}), \quad B \sim (T) - (\mathcal{O}), \quad \mathrm{div}(f_A) = mA, \quad \text{and} \quad \mathrm{div}(f_B) = mB.$$

Further, let divisors A_1, A_2 and functions f_{A_1}, f_{A_2} be such that

$$A_1 \sim (S_1) - (\mathcal{O}), \quad A_2 \sim (S_2) - (\mathcal{O}), \quad \mathrm{div}(f_{A_1}) = mA_1, \quad \text{and} \quad \mathrm{div}(f_{A_2}) = mA_2.$$

Observe that the divisor $(S_1 + S_2) - (S_1) - (S_2) + (\mathcal{O})$ is principal, so $(S_1) + (S_2) - 2(\mathcal{O}) \sim (S_1 + S_2) - (\mathcal{O})$. Hence, we can take $A = A_1 + A_2$. Then since

$$\mathrm{div}(f_A) = mA = mA_1 + mA_2 = \mathrm{div}(f_{A_1}) + \mathrm{div}(f_{A_2}) = \mathrm{div}(f_{A_1} f_{A_2}),$$

we can write $f_A = f_{A_1} f_{A_2}$, and thus

$$e'_m(S_1 + S_2, T) = \frac{f_A(B)}{f_B(A)} = \frac{f_{A_1} \cdot f_{A_2}(B)}{f_B(A_1 + A_2)} = \frac{f_{A_1}(B)}{f_B(A_1)} \frac{f_{A_2}(B)}{f_B(A_2)} = e'_m(S_1, T) e'_m(S_2, T).$$

□

The fact that $e_m(S,T)$ is not fully equivalent to $e'_m(S,T)$ does not have consequences for the use of the Weil pairing in practice. Namely, if the pairing $e_m(S,T)$ is a non-trivial mth root of unity, then so is the pairing $e'_m(S,T) = 1/e_m(S,T)$. Another way to look at it, is to observe that the role of the m-torsion points S and T can indifferently be interchanged, so one might as well work with $e'_m(T,S) = e_m(S,T)$ instead of $e'_m(S,T)$. Therefore, the ambiguity in the definition of the Weil pairing does not lead to confusion. (This probably causes the frequent appearance of the incorrect assertion of equivalence of e'_m and e_m in the literature.) So without loss of generality, we can choose either one of the definitions. From here on, we will work with the latter (Definition 3.7), as this is the definition for which an efficient algorithm for the computation exists. To ease notation, we will drop the addition 'alternative' and simply denote the Weil pairing by e_m (rather than e'_m).

Remark 3.11 We have defined the Weil pairing as a bilinear map that maps to the mth roots of unity in $\overline{K}$, where $K = \mathbb{F}_q$ is a finite field with q elements. In fact, we do not need the whole closure $\overline{K}$ of K; it suffices to consider a subfield $\mathbb{F}_{q^{k_w}}$ of $\overline{K}$, which is an extension field of $\mathbb{F}_q$ of degree k_w. Here k_w is the smallest integer such that $E[m] \subseteq E(\mathbb{F}_{q^{k_w}})$. This parameter, called the *Weil-embedding degree of the elliptic curve with respect to m*, turns out to play an important part in the applications of pairing-based cryptography. We will discuss the Weil-embedding degree of a curve into detail in Chapter 5.

3.4 Miller's Algorithm for the Weil Pairing

As we have seen, there are two closely related definitions of the Weil pairing. The latter (Definition 3.7) has the advantage that there is an efficient algorithm for the evaluation of the pairing. This algorithm, known as Miller's algorithm for the Weil pairing ([58, 54, 9, 55]), proceeds as follows. Suppose we want to compute the Weil pairing of the m-torsion points $S, T \in E[m] \subseteq E(\mathbb{F}_{q^{k_w}})$. For convenience, we write $K' = \mathbb{F}_{q^{k_w}}$. Choose points $S', T' \in E(K')$ such that $S', S + S', T'$ and $T + T'$ are all different. Then set $A = (S + S') - (S')$ and $B = (T + T') - (T')$. Note that

$$A - (S) + (\mathcal{O}) = (S + S') - (S) - (S') + (\mathcal{O}),$$

which is clearly a principal divisor, so $A \sim (S) - (\mathcal{O})$ as required. Similarly, $B \sim (T) - (\mathcal{O})$. Now we can use the method of Section 2.5 to efficiently compute $f_A, f_B \in K'(E)$, such that

$$\operatorname{div}(f_A) = mA = m(S + S') - m(S') \quad \text{and} \quad \operatorname{div}(f_B) = mB = m(T + T') - m(T').$$

As mentioned in Section 2.5, to prevent the rational functions f_A and f_B from getting exponentially large, we keep them in factored form. Finally, we can compute $e_m(S,T)$ as:

$$e_m(S,T) = \frac{f_A(B)}{f_B(A)} = \frac{f_A((T + T') - (T'))}{f_B((S + S') - (S'))} = \frac{f_A(T + T') f_B(S')}{f_A(T') f_B(S + S')}. \tag{3.4}$$

The fact that the points $S + S'$ and S' are distinct from $T + T'$ and T' guarantees that f_A (respectively f_B), considered as a rational function, is defined at $T + T'$ and T (respectively $S + S'$ and S'). However, while carrying out the algorithm, we want to treat the functions f_A and f_B as elements of $K'(x,y)$ rather than $K'(E)$. Therefore, the points S' and T' should be chosen so that, even regarded as an element of $K'(x,y)$, the function f_A (respectively f_B) is defined at $T + T'$ and T (respectively $S + S'$ and S').

Suppose we use a fixed addition chain $1 = a_1, a_2, \ldots, a_t = m$ for m to construct the functions f_A and f_B. Then by construction, f_A, regarded as an element of $K'(x,y)$, is undefined at most at the $4t$ points given by

$$\pm[a_1](S+S'), \pm[a_2](S+S'), \ldots, \pm[a_t](S+S') \quad \text{and} \quad \pm[a_1]S', \pm[a_2]S', \ldots, \pm[a_t]S'. \tag{3.5}$$

Therefore, if $T+T'$ and T' are distinct from the points listed in (3.5), we are sure that the function f_A is defined at these points. Similarly, $f_B \in K'(x,y)$ is undefined at most at the $4t$ points

$$\pm[a_1](T+T'), \pm[a_2](T+T'), \ldots, \pm[a_t](T+T') \quad \text{and} \quad \pm[a_1]T', \pm[a_2]T', \ldots, \pm[a_t]T'. \tag{3.6}$$

As f_B needs to be evaluated in $S+S'$ and S', we want these points to be distinct from those mentioned in (3.6).

We next examine the probability of picking a random pair of points (S', T') satisfying these conditions. For fixed S', both $T+T'$ and T' should be distinct from the $4t$ points listed in (3.5), leading to a maximum of $8t$ possibilities for T' for which this condition is not met. Further, there are $8t$ possibilities for T' for which one of the points in the list (3.6) equals either $S+S'$ or S', resulting in a total of $16t$ possibly unsuitable points. Thus an upper bound for the number of pairs $(S', T') \in E(K') \times E(K')$ that do not satisfy the conditions is given by $16t \cdot \#E(K')$. For every value of m, there exists an addition chain of length $t \leq 2\log_2 m$, hence the probability of randomly picking an unsuitable pair is at most

$$\frac{16t \cdot \#E(K')}{\#E(K')^2} = \frac{16t}{\#E(K')} \leq \frac{32 \cdot \log_2 m}{\#E(K')}.$$

Since $\#E(K') \geq m$, this probability is less than a half for $m \geq 1024$. Note that for cryptographic applications we usually have $m \gg 1024$, so the probability of choosing a good pair will in practice be close to 1.

3.4.1 Example

We return to the curve of our previous example: $E/\mathbb{F}_{11} : y^2 = x^3 + 3x$. We could compute the Weil pairing of two m-torsion points in $E(\mathbb{F}_{11})[m]$ for some integer m. However, from the order of the points, we can see that the group structure of $E(\mathbb{F}_{11})$ is cyclic. In other words, all points are linearly dependent. Then by the bilinearity of the pairing and the fact that $e_m(P,P) = 1$ for all $P \in E[m]$, the Weil pairing will map everything to 1. For instance,

$$e_6(P_3, P_6) = e_6(P_3, [4]P_3) = e_6(P_3, P_3)^4 = 1.$$

Consider instead the points of E over the quadratic extension $\mathbb{F}_{11^2}$ of $\mathbb{F}_{11}$, which is obtained by allowing the primitive fourth root of unity i (i.e. $i^2 = -1$). It is easily verified that if $(x,y) \in E(\mathbb{F}_{11^2})$, then also $(-x, iy) \in E(\mathbb{F}_{11^2})$. Let the endomorphism $\phi : E(\mathbb{F}_{11^2}) \to E(\mathbb{F}_{11^2})$ be given by

$$\phi : (x,y) \to (-x, iy) \quad \text{and} \quad \phi : \mathcal{O} \to \mathcal{O}.$$

Clearly, the image $\phi(P) \in E(\mathbb{F}_{11^2})$ of a point $P \in E(\mathbb{F}_{11})$, $P \neq \mathcal{O}$, is not an $\mathbb{F}_{11}$-rational point (i.e. $\phi(P) \notin E(\mathbb{F}_{11})$). Hence $\phi(P)$ is linearly independent of P. Furthermore, if P is an m-torsion point, i.e. $[m]P = \mathcal{O}$, then

$$[m]\phi(P) = \phi([m]P) = \phi(\mathcal{O}) = \mathcal{O},$$

so also $\phi(P)$ is an m-torsion point. Thus we can calculate the Weil pairing $e_m(P, \phi(P))$, which by Corollary 3.6 is non-trivial because of the linear independency between P and $\phi(P)$.

Take for example the points $S = P_6 = (3,5)$ and $T = \phi(P_6) = (8,5i)$, both of which have order 3. We choose $S' = (6,6)$ and $T' = (9,6i)$, then $S+S' = (7,1)$ and $T+T' = (4,10i)$,

so $S', S+S', T', T+T'$ are indeed all different. To compute the Weil pairing $e_3(S,T)$, we need rational functions f_A and f_B such that

$$\mathrm{div}(f_A) = 3(S+S') - 3(S') \quad \text{and} \quad \mathrm{div}(f_B) = 3(T+T') - 3(T').$$

Using the method described in the beginning of this section, we compute:

$$\begin{aligned} 3(S+S') - 3(\mathcal{O}) &= ((6,5)) - (\mathcal{O}) + \mathrm{div}\Big(\frac{(y+2x+7)(y+5x+8)}{(x+10)(x+5)}\Big), \\ 3(S') - 3(\mathcal{O}) &= ((6,5)) - (\mathcal{O}) + \mathrm{div}\Big(\frac{(y+10x)^2}{x(x+5)}\Big). \end{aligned}$$

Subtracting these equations, we get

$$3(S+S') - 3(S) = \mathrm{div}\Big(\frac{(y+2x+7)(y+5x+8)x}{(x+10)(y+10x)^2}\Big),$$

so f_A is given by:

$$f_A = \frac{(y+2x+7)(y+5x+8)x}{(x+10)(y+10x)^2}.$$

Similarly, we compute

$$\begin{aligned} 3(T+T') - 3(\mathcal{O}) &= ((5,6i)) - (\mathcal{O}) + \mathrm{div}\Big(\frac{(y+2ix+4i)(y+5ix+3i)}{(x+1)(x+6)}\Big), \\ 3(T') - 3(\mathcal{O}) &= ((5,6i)) - (\mathcal{O}) + \mathrm{div}\Big(\frac{(y+4ix+2i)(y+8ix+10i)}{(x+1)(x+6)}\Big). \end{aligned}$$

Then subtracting these equations yields us f_B:

$$f_B = \frac{(y+2ix+4i)(y+5ix+3i)}{(y+4ix+2i)(y+8ix+10i)}.$$

Plugging these results into equation (3.4), we get the Weil pairing of P_6 and $\phi(P_6)$:

$$e_3(P_6, \phi(P_6)) = \frac{f_A(T+T')f_B(S')}{f_A(T')f_B(S+S')} = \frac{f_A((4,10i))f_B(6,6)}{f_A((9,6i))f_B((7,1))} = 5+8i.$$

Indeed $(5+8i)^3 = 1$, so $e_3(P_6, \phi(P_6))$ is a 3rd root of unity in $\mathbb{F}_{11^2}$.

Note that, as observed in Remark 3.11, we do not need to work over the closure of the finite field $\mathbb{F}_{11}$. In fact, a field extension $\mathbb{F}_{11^2}$ of degree 2 suffices for computing the Weil pairing.

Remark 3.12 Such an endomorphism ϕ that maps a point to a linearly independent point of the same order is called a *distortion map*. Distortion maps, which are useful cryptographic tools, are discussed in Chapter 6.

3.4.2 Implementation in *Mathematica*

We implement the Weil pairing e_m for supersingular curves of the form $E/\mathbb{F}_p : y^2 = x^3 + ax$ with prime $p = 3 \pmod 4$ and $E/\mathbb{F}_p : y^2 = x^3 + b$ with prime $p = 2 \pmod 3$ in *Mathematica* (see Appendix A for the code). For this purpose, we define the following auxiliary modules. The modules `Pt` (entering a point), `Neg` (negating a point), `PointAddition` (adding two points), and `PointMultiplication` (multiplying a point by a scalar) follow in a straightforward manner from the formulae in Section 2.1. The modules `DivisorAddition` and `DivisorMultiplication`, for adding and taking scalar-multiples of divisors in canonical form, follow the method outlined in Section 2.5. Picking a random point in the ground field, as is done by the module

`RandomPointGroundField`, is a simple matter under the assumption that $p = 3 \pmod 4$. The module `DistortionMap` defines the distortion map, which is of standard form for these curves (see Chapter 6).

Now the module `PickTUGroundField` uses `RandomPointGroundField` to pick two candidates S', T' for the construction of the divisors $A = (S + S') - (S')$ and $B = (T + T') - (T')$, and checks whether $S + S', S', T + T', T'$ are all distinct. The module `ComputeFunctionWeil` computes a function having a divisor of the form $m(S + S') - m(S')$, which is easily done using the module `DivisorMultiplication`. Finally, the module `WeilPairing` combines `PickTUGroundField` and `ComputeFunctionWeil` to find divisors A, B and functions f_A, f_B. Next, checks are done to ensure that the functions f_A and f_B are defined at the points $S + S', S', T + T', T'$ and the pairing is evaluated.

We give examples for the curves $E_1/\mathbb{F}_{11} : y^2 = x^3 + 3x$ and $E_2/\mathbb{F}_{131} : y^2 = x^3 + 31$. We show that the pairing of two dependent points is indeed the identity. Moreover, in correspondence with the example of Subsection 3.4.1, the outcome of the pairing $e_3(P_6, \phi(P_6))$ on the curve E_1 is shown to be $5 + 8i$.

Chapter 4

Tate Pairing

In addition to the Weil pairing, there exists another bilinear map on the group of points on an elliptic curve, which is known as the Tate pairing ([28, 27]). We give the definition of the Tate pairing in Section 4.1 and discuss some useful properties in Section 4.2. Next, we give an algorithm to efficiently compute the Tate pairing in Section 4.3 and examine its relation with the Weil pairing in Section 4.4. Section 4.5 concludes this chapter with some observations on the optimization of the algorithm.

4.1 Definition

Consider the curve E/K defined over some finite field K. Let m be such that $E(K)$ contains a point of order m. The set of distinct points obtained from multiplying every point on $E(K)$ by m is denoted $mE(K)$. This set is a coset of $E(K)$ and is also denoted by $C_{\mathcal{O}}$, as it contains $\mathcal{O}$. The number of elements in the coset is $\#C_{\mathcal{O}} = \#E(K)/m$. Other cosets C_R can be formed by adding a point R not in the coset $C_{\mathcal{O}}$ to every element in the coset. This way, the group of points $E(K)$ can be split into m distinct cosets. If m is prime, then every coset contains exactly one m-torsion point. The quotient group of all cosets is given by $E(K)/mE(K)$. This is clarified in the example below. For details, see [72].

Example 4.1 Consider our usual example $E/\mathbb{F}_{11} : y^2 = x^3 + 3x$. The points on $E(\mathbb{F}_{11})$ and their orders are given in Table 2.1. We want to split up $E(\mathbb{F}_{11})$ into cosets for $m = 3$. Observe that

$$[3]\mathcal{O} = [3]P_6 = [3]P_7 = \mathcal{O}, \quad [3]P_1 = [3]P_2 = [3]P_3 = P_1,$$
$$[3]P_4 = [3]P_9 = [3]P_{10} = P_8, \quad \text{and} \quad [3]P_5 = [3]P_8 = [3]P_{11} = P_9.$$

Hence, $mE(\mathbb{F}_{11}) = C_{\mathcal{O}} = \{\mathcal{O}, P_1, P_8, P_9\}$. Adding the points P_2, respectively P_3 to all elements in $C_{\mathcal{O}}$ yields the other cosets C_{P_2}, respectively C_{P_3}:

$$C_{P_2} = \{P_2, P_6, P_{10}, P_5\} \quad \text{and} \quad C_{P_3} = \{P_3, P_7, P_4, P_{11}\}.$$

Cosets $C_{\mathcal{O}}, C_{P_2}$ and C_{P_3} all contain exactly one m-torsion point, namely $\mathcal{O}, P_6$ and P_7 respectively. The quotient group $E(\mathbb{F}_{11})/mE(\mathbb{F}_{11})$ is given by $\{C_{\mathcal{O}}, C_{P_2}, C_{P_3}\}$.

Let $E/\mathbb{F}_q$ be an elliptic curve defined over a field $\mathbb{F}_q$. Let m be a positive integer, coprime to q, such that $E(\mathbb{F}_q)$ contains a point of order m. In cryptographic implementations, m is usually taken to be a large prime such that $m|\#E(\mathbb{F}_q)$. Let k be the smallest integer satisfying $m|q^k - 1$. This value is the *(Tate-)embedding degree of the curve with respect to* m. We will elaborate on this in Chapter 5.

Similar to the Weil pairing, the Tate pairing is defined in terms of rational functions evaluated in a certain divisor. Let $P \in E(\mathbb{F}_{q^k})[m]$, then $m(P) - m(\mathcal{O})$ is a principal divisor. So there is a

rational function $g \in \mathbb{F}_{q^k}(E)$ with $\mathrm{div}(g) = m(P) - m(\mathcal{O})$. Now let Q be a point representing a coset in $E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k})$, then we construct a divisor $D \in \mathrm{Div}^0(E)$ such that $D \sim (Q) - (\mathcal{O})$. D should be chosen so that its support is disjoint from the support of the divisor of g. Then the Tate pairing is defined as follows.

Definition 4.2 *With the notation as introduced above, the Tate pairing*

$$\langle \cdot, \cdot \rangle : E(\mathbb{F}_{q^k})[m] \times E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^m,$$

is given by:

$$\langle P, Q \rangle = g(D).$$

Note that the result of the Tate pairing is not a unique value, but an equivalence class in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^m$. That is, two elements $a, b \in \mathbb{F}_{q^k}^*$ are equivalent (denoted $a \equiv b$) if and only if there is a $c \in \mathbb{F}_{q^k}^*$ such that $a = bc^m$. If a unique outcome of the pairing is required, as is often the case in cryptographic applications, we need to eliminate the mth powers. This is done by raising the value to the power $(q^k - 1)/m$, since $a^{q^k-1} = 1$ for all $a \in \mathbb{F}_{q^k}^*$. Consequently, the value after this exponentiation is an mth root of unity. This leads to an alternative definition of the Tate pairing that is sometimes used in the literature, which maps uniquely to the group μ_m of mth roots of unity:

$$\langle P, Q \rangle' = g(D)^{(q^k-1)/m}.$$

However, we will work with the former definition, for which the pairing maps to an equivalence class, and perform an exponentiation afterwards whenever a unique value is required.

The following theorem asserts that the Tate pairing is well-defined in the sense that the choice of g and D does not matter. The proof of the independency of the choice of D makes clear why the result of the pairing is to be considered as an equivalence class in the quotient group $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^m$. Namely, a different choice for D possibly results in a different representative for the equivalence class.

Theorem 4.3 *The outcome of the Tate pairing does not depend on the choice of the rational function g and divisor D.*

Proof. The function g is determined up to a constant multiple. However, since D is a degree zero divisor, Lemma 2.8 implies that this constant has no influence on the outcome of the evaluation of g in D.

Let g be a function satisfying $\mathrm{div}(g) = m(P) - m(\mathcal{O})$ and $D_1 \sim D_2 \sim (Q) - (\mathcal{O})$ such that the support of both D_1 and D_2 is disjoint from $\mathrm{supp}(\mathrm{div}(g))$. Then we can write $D_1 = D_2 + \mathrm{div}(h)$ for some function h with $\mathrm{supp}(\mathrm{div}(h)) \cap \mathrm{supp}(\mathrm{div}(g)) = \emptyset$. Hence

$$\begin{aligned} g(D_1) &= g(D_2 + \mathrm{div}(h)) = g(D_2)g(\mathrm{div}(h)) = g(D_2)h(\mathrm{div}(g)) \\ &= g(D_2)h(m(P) - m(\mathcal{O})) = g(D_2)h((P) - (\mathcal{O}))^m \equiv g(D_2), \end{aligned}$$

which completes the proof. □

4.2 Properties

Similar to the Weil pairing, the Tate pairing satisfies some properties that make the pairing suitable for use in cryptography.

Theorem 4.4 *The Tate pairing satisfies the following properties:*

1. $\langle \mathcal{O}, Q \rangle = 1$ *for all* $Q \in E(\mathbb{F}_{q^k})$ *and* $\langle P, Q \rangle \in (\mathbb{F}_{q^k}^*)^m$ *for all* $P \in E(\mathbb{F}_{q^k})[m]$ *and all* $Q \in mE(\mathbb{F}_{q^k})$ *(well-defined).*

2. *For each point* $P \in E(\mathbb{F}_{q^k})[m] \setminus \{\mathcal{O}\}$ *there is some point* $Q \in E(\mathbb{F}_{q^k})$ *such that* $\langle P, Q\rangle \notin (\mathbb{F}_{q^k}^*)^m$ *(non-degeneracy).*

3. *For all* $P_1, P_2, P \in E(\mathbb{F}_{q^k})[m]$ *and* $Q_1, Q_2, Q \in E(\mathbb{F}_{q^k})$*, we have* $\langle P_1+P_2, Q\rangle \equiv \langle P_1, Q\rangle\langle P_2, Q\rangle$*, and* $\langle P, Q_1 + Q_2\rangle \equiv \langle P, Q_1\rangle\langle P, Q_2\rangle$ *(bilinearity).*

Proof.

1. For the first part, let $g \in \mathbb{F}_{q^k}(E)$ be the constant function that always outputs 1, then $\mathrm{div}(g) = 0$ as required. Then $\langle \mathcal{O}, Q\rangle = g(Q) = 1$ for all $Q \in E(\mathbb{F}_{q^k})$.

   To prove the second part, let $P \in E(\mathbb{F}_{q^k})[m]$, $Q \in mE(\mathbb{F}_{q^k})$ and $g \in \mathbb{F}_{q^k}(E)$ such that $\mathrm{div}(g) = m(P) - m(\mathcal{O})$. Then there exists a $Q' \in E(\mathbb{F}_{q^k})$ such that $[m]Q' = Q$. Let $D \sim (Q) - (\mathcal{O})$ and $A \sim (Q') - (\mathcal{O})$, with support disjoint from the support of $\mathrm{div}(g)$. Then

   $$D - mA \sim (Q) - (\mathcal{O}) - m(Q') + m(\mathcal{O}) = ([m]Q') - m(Q') + (m-1)(\mathcal{O}),$$

   which is principal, so $D \sim mA$. Then

   $$\langle P, Q\rangle = g(D) \equiv g(mA) = g(A)^m \in (\mathbb{F}_{q^k}^*)^m.$$

2. A sophisticated proof of the non-degeneracy is given in [28] (Section 2). For a more elementary proof using Weil's reciprocity law, see [34] (Theorem 3).

3. Let $P_1, P_2, P \in E(\mathbb{F}_{q^k})[m]$ and $Q_1, Q_2, Q \in E(\mathbb{F}_{q^k})$. Write

   $$\langle P_1 + P_2, Q\rangle = g(D), \quad \text{where} \quad D \sim (Q) - (\mathcal{O}) \quad \text{and} \quad \mathrm{div}(g) = m(P_1 + P_2) - m(\mathcal{O}),$$

   such that $P_1, P_2, P_1 + P_2, \mathcal{O} \notin \mathrm{supp}(D)$. Further, let g_1, g_2 be functions such that

   $$\mathrm{div}(g_1) = m(P_1) - m(\mathcal{O}) \quad \text{and} \quad \mathrm{div}(g_2) = m(P_2) - m(\mathcal{O}).$$

   The divisor $A = (P_1 + P_2) - (P_1) - (P_2) + (\mathcal{O})$ is principal, so there is a function, say h, such that $\mathrm{div}(h) = A$. Then

   $$\mathrm{div}(g) - \mathrm{div}(g_1) - \mathrm{div}(g_2) = m(P_1 + P_2) - m(P_1) - m(P_2) + m(\mathcal{O}) = m\mathrm{div}(h) = \mathrm{div}(h^m),$$

   so we can write $g = g_1 g_2 h^m$. Then

   $$\langle P_1 + P_2, Q\rangle = g(D) = g_1(D)g_2(D)h^m(D) \equiv \langle P_1, Q\rangle\langle P_2, Q\rangle.$$

   To prove linearity in the second factor, write

   $$\langle P, Q_1 + Q_2\rangle = g(D), \quad \text{where} \quad D \sim (Q_1 + Q_2) - (\mathcal{O}) \quad \text{and} \quad \mathrm{div}(g) = m(P) - m(\mathcal{O}),$$

   such that D and $\mathrm{div}(g)$ have disjoint support. Further, let divisors D_1, D_2 be such that

   $$D_1 \sim (Q_1) - (\mathcal{O}) \quad \text{and} \quad D_2 \sim (Q_2) - (\mathcal{O}),$$

   where both D_1 and D_2 have support disjoint from $\mathrm{supp}(\mathrm{div}(g))$. Then

   $$D - D_1 - D_2 \sim (Q_1 + Q_2) - (Q_1) - (Q_2) + (\mathcal{O}),$$

   which is principal, so $D_1 + D_2 \sim D$. So we can write

   $$\langle P, Q_1 + Q_2\rangle = g(D) \equiv g(D_1 + D_2) = g(D_1)g(D_2) = \langle P, Q_1\rangle\langle P, Q_2\rangle.$$

□

The second observation of part 1 of the above theorem implies that the outcome of the Tate pairing is irrespective of which element of the particular coset is chosen for the second parameter; namely, any element in $mE(\mathbb{F}_{q^k})$ will be mapped to $(\mathbb{F}_{q^k}^*)^m$, and will therefore vanish in the quotient group $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^m$.

In contrast to the Weil pairing, which always maps $e_m(P,P)$ to 1 for $P \in E[m]$, the outcome of the Tate pairing $\langle P,P\rangle$ for $P \in E(\mathbb{F}_{q^k})[m]$ is not necessarily the identity in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^m$ (i.e. an mth power). In cryptographic applications, it is generally the case that P is an m-torsion point in the ground field, so $P \in E(\mathbb{F}_q)[m]$, and m is coprime to q. Then a result by Galbraith [30] says the following:

Lemma 4.5 (Galbraith [30]) *Let $P \in E(\mathbb{F}_q)[m]$, $P \neq \mathcal{O}$ and m coprime q. Then for the Tate pairing $\langle P,P\rangle$ to be nontrivial, it is necessary that $k = 1$.*

Proof. We write $\langle P,P\rangle = g(D)$, where $\mathrm{div}(g) = m(P) - m(\mathcal{O})$ and $D \sim (P) - (\mathcal{O})$. Since $P \in E(\mathbb{F}_q)[m]$, it follows that $g \in \mathbb{F}_q(E)$, and thus $g(D) \in \mathbb{F}_q^*$. Now suppose $k > 1$, then $m \not| (q-1)$. Further, m is coprime to q, so every element of $\mathbb{F}_q^*$ is an mth power and thus $g(D) \in (\mathbb{F}_{q^k}^*)^m$. Thus to have $\langle P,P\rangle$ nontrivial, a necessary condition is $k = 1$. □

Note that although $k = 1$ is a necessary condition, it is not sufficient. For trace 2 curves, as mentioned in [43], a point P of order m results in a non-trivial pairing $\langle P,P\rangle \not\equiv 1$ if m^2 does not divide $p-1$. However, it is an open problem how to construct these curves.

From Theorem 4.4 and Lemma 4.5 follows the corollary below.

Corollary 4.6 *Let $P \in E(\mathbb{F}_q)[m]$, $P \neq \mathcal{O}$ and m prime. Let $k > 1$ and $Q \in E(\mathbb{F}_{q^k})[m]$ an m-torsion point independent from P. Then $\langle P,Q\rangle$ is non-trivial, i.e. $\langle P,Q\rangle \not\equiv 1$.*

Proof. Suppose that $\langle P,Q\rangle \equiv 1$. Let $R \in E(\mathbb{F}_{q^k})$ and consider the Tate pairing $\langle P,R\rangle$. Every coset in $E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k})$ contains exactly one m-torsion point (see Section 4.1). Let R' be the m-torsion point that is in the same coset as R, so $R = R' + [m]R''$ for some $R'' \in E(\mathbb{F}_{q^k})$. Then $\langle P,R\rangle \equiv \langle P,R'\rangle$. Since the group of m-torsion points is generated by P and Q, we can write the m-torsion point R' as $R' = [a]P + [b]Q$ for some $0 \leq a,b \leq m-1$. Then

$$\langle P,R\rangle \equiv \langle P,R'\rangle = \langle P,[a]P+[b]Q\rangle \equiv \langle P,P\rangle^a \cdot \langle P,Q\rangle^b \equiv 1,$$

where the last equivalence holds by Lemma 4.5. However, this is in contradiction with the non-degeneracy property of the Tate pairing. Hence $\langle P,Q\rangle \not\equiv 1$. □

4.3 Miller's Algorithm for the Tate Pairing

As with the Weil pairing, there is an efficient algorithm for the computation, known as Miller's algorithm for the Tate pairing (see [5]). Let us start with introducing some notation. For each pair of points U,V on the elliptic curve $E(\mathbb{F}_{q^k})$, let $g_{U,V} \in \mathbb{F}_{q^k}(E)$ be the rational function given by the line $g_{U,V} : l_1 y + l_2 x + l_3 = 0$ through U and V. Naturally, if $U = V$, then $g_{U,V}$ is the given by the equation of the tangent line at U, and if either U or V is the point at infinity $\mathcal{O}$, then $g_{U,V}$ represents the vertical line through the other point. Furthermore, for brevity, we write g_U instead of $g_{U,-U}$.

Now we are ready to give Miller's formula, which lies at the heart of Miller's algorithm for computing the Tate pairing.

Theorem 4.7 (Miller's formula) *Let $P \in E(\mathbb{F}_{q^k})$ and f_c a rational function such that $div(f_c) = c(P) - ([c]P) - (c-1)(\mathcal{O})$ for $c \in \mathbb{Z}$. Then for all $a,b \in \mathbb{Z}$, the following holds:*

$$f_{a+b} = f_a \cdot f_b \cdot g_{[a]P,[b]P}/g_{[a+b]P}.$$

The theorem can be proven by simply writing out and comparing the divisors of the functions on both sides of the equation; for details see [5]. It is interesting to note that this is a special case of the method for adding two divisors in canonical form in Section 2.5. Namely, let divisors D_1 and D_2 be given by $a(P) - a(\mathcal{O})$ and $b(P) - b(\mathcal{O})$ respectively. Then their canonical form is given by

$$D_1 = ([a]P) - (\mathcal{O}) + \mathrm{div}(f_a) \quad \text{and} \quad D_2 = ([b]P) - (\mathcal{O}) + \mathrm{div}(f_b),$$

where f_a and f_b are as in Theorem 4.7. Then their sum $D_1 + D_2 = (a+b)(P) - (a+b)(\mathcal{O})$ in canonical form is given by

$$(a+b)(P) - (a+b)(\mathcal{O}) = ([a+b]P) - (\mathcal{O}) + \mathrm{div}(f_a \cdot f_b \cdot g_{[a]P,[b]P}/g_{[a+b]P}),$$

where $g_{[a]P,[b]P}/g_{[a+b]P}$ is denoted by $l/v = f_3$ in Section 2.5. Now bringing the terms $([a+b]P) - (\mathcal{O})$ to the left, we get the relation

$$\mathrm{div}(f_{a+b}) = \mathrm{div}(f_a \cdot f_b \cdot g_{[a]P,[b]P}/g_{[a+b]P}),$$

which underlies Theorem 4.7.

Recall that the computation of the Tate pairing requires a rational function g with $\mathrm{div}(g) = m(P) - m(\mathcal{O})$ for $P \in E(\mathbb{F}_{q^k})[m]$. Since P is an m-torsion point, we can write $g = f_m$ with the notation as in the theorem above. Miller's formula suggests to build this function recursively, using the double-and-add method as follows.

Since $\mathrm{div}(f_1) = 0$, we can take $f_1 = 1$. Then for $a > 0$, we have

$$f_{a+1} = f_a \cdot g_{[a]P,P}/g_{[a+1]P} \quad \text{and} \quad f_{2a} = f_a^2 \cdot g_{[a]P,[a]P}/g_{[2a]P}.$$

Let $(m_t, \ldots, m_1, m_0)_2$ be the binary representation of m, where $m_t = 1$. Starting with $f_{(m_t)_2} = f_1 = 1$, we can successively compute $f_{(m_t,\ldots,m_i)_2}$ for $i = t-1, \ldots, 0$ by doubling and adding $f_{(m_t,\ldots,m_{i+1})_2}$ if $m_i = 1$, and only doubling if $m_i = 0$. The result will be $g = f_m = f_{(m_t,\ldots,m_1,m_0)_2}$, where $\mathrm{div}(g) = m(P) - m(\mathcal{O})$ as required.

The function g needs to be evaluated at a divisor D equivalent to $(Q) - (\mathcal{O})$. Taking for instance $D = (Q+Q') - (Q')$ for some $Q' \in E(\mathbb{F}_{q^k})$, we need to compute $g(Q+Q')/g(Q')$. Instead of first constructing g completely and plugging in D afterwards, it is more efficient to do the evaluation along the way. That is, we keep track of the evaluations at D of the intermediate functions $f_{(m_t,\ldots,m_i)_2}$, rather than the functions themselves. To be precise, we use the formulas

$$f_{a+1}(D) = f_{a+1}(Q+Q')/f_{a+1}(Q') = f_a(D) \cdot \frac{g_{[a]P,P}(Q+Q')g_{[a+1]P}(Q')}{g_{[a+1]P}(Q+Q')g_{[a]P,P}(Q')}, \tag{4.1}$$

and

$$f_{2a}(D) = f_{2a}(Q+Q')/f_{2a}(Q') = f_a(D)^2 \cdot \frac{g_{[a]P,[a]P}(Q+Q')g_{[2a]P}(Q')}{g_{[2a]P}(Q+Q')g_{[a]P,[a]P}(Q')}. \tag{4.2}$$

The above is summarized in Figure 4.1.

Remark 4.8 This method can be applied because the divisor of the function g is equal to $m(P) - m(\mathcal{O})$. Remark that the Weil pairing requires functions f_A, f_B whose divisors are equivalent to $m(S) - m(\mathcal{O})$, respectively $m(T) - m(\mathcal{O})$ for S, T m-torsion points, but not equal. Hence, these functions for the Weil pairing cannot be constructed using this efficient double-and-add method. Instead, one needs to compute the canonical form for the divisors $m(S+S') - m(\mathcal{O})$ and $m(S') - m(\mathcal{O})$ for some $S' \neq \mathcal{O}$ and subtract these divisors to get the desired function f_A (similarly for f_B), as described in Section 3.4.

1. Choose a random point $Q' \in E(\mathbb{F}_{q^k})$ and compute $S = Q + Q' \in E(\mathbb{F}_{q^k})$.
2. Set $t = \lfloor \log_2(m) \rfloor$, and let $(m_t, \ldots, m_0)_2$ be the binary representation of m.
   Set $f = 1$ and $V = P$.
3. For $i = t - 1$ to 0 do
   - Set $f = f^2(g_{V,V}(S)g_{[2]V}(Q'))/(g_{[2]V}(S)g_{V,V}(Q'))$ and $V = [2]V$.
   - If $m_i = 1$ then set $f = f(g_{V,P}(S)g_{V+P}(Q'))/(g_{V+P}(S)g_{V,P}(Q'))$ and $V = V + P$.
4. Return f.

Figure 4.1: Miller's algorithm for the Tate pairing

4.3.1 Example

Again, we consider the elliptic curve $E/\mathbb{F}_{11} : y^2 = x^3 + 3x$. The Tate-embedding degree k of the curve with respect to 6 is equal to 2, because $6|(11^2 - 1)$ but $6 \nmid (11 - 1)$. To compute the Tate pairing $\langle P, Q \rangle$ for $P = (1, 9)$, $Q = \phi(P) = (10, 9i)$ and $m = 6$, we carry out Miller's algorithm.

1. We choose $Q' \in E(\mathbb{F}_{11^2})$ to be $Q' = (6, 6)$. Then $S = Q + Q' = (8 + 7i, 10 + 6i)$.
2. The binary representation of $m = 6$ is given by $(m_2, m_1, m_0)_2 = (1, 1, 0)_2$, so $t = \lfloor \log_2(6) \rfloor = 2$. Further, we set $f = 1$ and $V = P = (1, 9)$.
3. For $i = 1$:
   - We compute $g_{V,V}$ and $g_{[2]V}$:
   $$g_{V,V} = y + 7x + 6 \quad \text{and} \quad g_{[2]V} = x + 8.$$
   Then
   $$g_{V,V}(S) = 6, \quad g_{[2]V}(Q') = 3, \quad g_{[2]V}(S) = 5 + 7i, \quad \text{and} \quad g_{V,V}(Q') = 10,$$
   and thus we set
   $$f = 1^2 \frac{6 \cdot 3}{(5 + 7i) \cdot 10} = 8 + 2i \quad \text{and} \quad V = [2](1, 9) = (3, 5).$$
   - Since $m_1 = 1$, we compute $g_{V,P}$ and g_{V+P}:
   $$g_{V,P} = y + 2x \quad \text{and} \quad g_{V+P} = x.$$
   Then
   $$g_{V,P}(S) = 4 + 9i, \quad g_{V+P}(Q') = 6, \quad g_{V+P}(S) = 8 + 7i, \quad \text{and} \quad g_{V,P}(Q') = 7,$$
   and thus we set
   $$f = (8 + 2i)\frac{(4 + 9i)6}{(8 + 7i)7} = 5 + 4i \quad \text{and} \quad V = (3, 5) + (1, 9) = (0, 0).$$

For $i = 0$:

- We compute $g_{V,V}$ and $g_{[2]V}$:
$$g_{V,V} = x \quad \text{and} \quad g_{2V} = 1.$$
Then
$$g_{V,V}(S) = 8 + 7i, \quad g_{[2]V}(Q') = 1, \quad g_{[2]V}(S) = 1, \quad \text{and} \quad g_{V,V}(Q') = 6,$$
hence we set
$$f = (5 + 4i)^2 \frac{8 + 7i}{6} = 2 + 7i \quad \text{and} \quad V = 2(0, 0) = \mathcal{O}.$$

- Since $m_0 = 0$, end.

4. $i = 0$, so the program terminates and returns $f = 2 + 7i$.

The outcome of the Tate pairing of $P = (1, 9)$ and $Q = (10, 9i)$ is $\langle P, Q \rangle = 2 + 7i \in \mathbb{F}_{11^2}^*/(\mathbb{F}_{11^2}^*)^6$. Recall that the value $2 + 7i$ represents an equivalence class modulo 6th powers. If a unique value of the pairing is required, the result of the pairing should be raised to the power $(11^2 - 1)/6 = 20$: $(2 + 7i)^{20} = 5 + 3i$. Indeed, $(5 + 3i)^6 = 1$, so $5 + 3i$ is a 6th root of unity in $\mathbb{F}_{11^2}^*$. Since 6 is not a prime, it is possible that the outcome of the Tate pairing is not a primitive 6th root. This is in fact the case, since $(5 + 3i)^3 = 1$.

4.3.2 Implementation in *Mathematica*

The implementation of the Tate pairing in *Mathematica* (see Appendix A), also uses the auxiliary modules mentioned in Subsection 3.4.2. Further, the module `ComputeFunctionTate` computes the function g such that $\mathrm{div}(g) = m(P) - m(\mathcal{O})$ for an m-torsion point P. This is done by using the module `DivisorMultiplication` to multiply the divisor $(P) - (\mathcal{O})$ by m and express the result in canonical form. Note that `DivisorMultiplication` uses the double-and-add method described in the beginning of this section.

Finally, the module `TatePairing` combines `RandomPointGroundField` to pick a point Q' to construct the divisor $D = (Q + Q') - (Q')$ and `ComputeFunctionTate` to construct g. Next, the module checks whether g is defined at $Q + Q', Q'$ and the pairing is evaluated. Further, a module `TatePairingUnique` is defined that simply performs an exponentiation on the outcome of `TatePairing` to get a unique value. Since g is constructed using the double-and-add method, the computation of the Tate pairing is done as in Miller's algorithm (Figure 4.1), except that the divisor D is plugged in afterwards, instead of during the construction of g.

As with the Weil pairing, we give examples for the curves $E_1/\mathbb{F}_{11} : y^2 = x^3 + 3x$ and $E_2/\mathbb{F}_{131} : y^2 = x^3 + 31$. We demonstrate that the Tate pairing of two points is not a unique value but an equivalence class. Moreover, it is shown that if the second parameter of the pairing is an element of $mE(\mathbb{F}_{q^k})$, then the result is the identity.

4.4 Comparison with the Weil Pairing

In this section, we compare the Tate pairing with the Weil pairing. We start with examining the relation between these pairings from an algebraic point of view. Essentially, there is no relation between these pairings, as they are defined on different sets. However, when we take Q to be an m-torsion point and regard the outcome of the Weil pairing as element of $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^m$, we can compare the two definitions as in Subsection 4.4.1.

Next we compare the efficiency of the pairings in Subsection 4.4.2. It turns out that the Tate pairing can be computed faster and is thus, from a computational point of view, more suitable for use in cryptographic systems.

4.4.1 Algebraic Relation

Let P and Q be m-torsion points. As we have seen, the Tate pairing is defined as $\langle P, Q \rangle = g(D)$, where $\mathrm{div}(g) = m(P) - m(\mathcal{O})$ and D is a divisor equivalent to $(Q) - (\mathcal{O})$ with support disjoint from $\mathrm{supp}(\mathrm{div}(g))$. By taking $D = (Q + Q') - (Q')$ for $Q' \in E(\mathbb{F}_{q^k})$, with $Q', Q + Q' \neq P, \mathcal{O}$, we meet the conditions and we can write $g(D) = g(Q + Q')/g(Q')$. For sake of clarity, we will denote this function g in the remainder of this subsection as g_A, so

$$\langle P, Q \rangle = g_A(Q + Q')/g_A(Q').$$

Similarly, we define the pairing $\langle Q, P\rangle$ as

$$\langle Q, P\rangle = g_B(P + P')/g_B(P'),$$

where $\mathrm{div}(g_B) = m(Q) - m(\mathcal{O})$ and $P' \in E(\mathbb{F}_{q^k})$, with $P', P + P' \neq Q, \mathcal{O}$.

Recall that the definition of the Weil pairing is given by

$$e_m(P, Q) = \frac{f_A(Q + Q')f_B(P')}{f_A(Q')f_B(P + P')},$$

where $P', Q' \in E(\mathbb{F}_{q^k})$ and f_A, f_B are such that

$$\mathrm{div}(f_A) = m(P + P') - m(P') \quad \text{and} \quad \mathrm{div}(f_B) = m(Q + Q') - m(Q').$$

It is easily verified that $(P + P') - (P') - (P) + (\mathcal{O})$ is a principal divisor and thus there exists a rational function h_A such that

$$\mathrm{div}(h_A) = (P + P') - (P') - (P) + (\mathcal{O}).$$

Then

$$\mathrm{div}(f_A) = m((P + P') - (P')) = m\mathrm{div}(h_A) + \mathrm{div}(g_A) = \mathrm{div}(h_A^m g_A),$$

so h_A can be chosen such that

$$f_A = h_A^m g_A.$$

Similarly, there is a rational function h_B such that

$$\mathrm{div}(h_B) = (Q + Q') - (Q') - (Q) + (\mathcal{O}) \quad \text{and} \quad f_B = h_B^m g_B.$$

Plugging these results into the definition of the Weil pairing, we obtain:

$$\begin{aligned} e_m(P, Q) &= \frac{h_A^m(Q + Q')g_A(Q + Q')}{h_A^m(Q')g_A(Q')} \frac{h_B^m(P')g_B(P')}{h_B^m(P + P')g_B(P + P')} \\ &= \left(\frac{h_A(Q + Q')h_B(P')}{h_A(Q')h_B(P + P')}\right)^m \frac{g_A(Q + Q')g_B(P')}{g_A(Q')g_B(P + P')}. \end{aligned}$$

Observe that the term $\left((h_A(Q + Q')h_B(P'))/(h_A(Q')h_B(P + P'))\right)^m$ is an element of $(\mathbb{F}_{q^k}^*)^m$, and therefore vanishes in the quotient group $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^m$. Hence, regarded as an element of this quotient group, the Weil pairing can be written as

$$e_m(P, Q) \equiv \frac{g_A(Q + Q')g_B(P')}{g_A(Q')g_B(P + P')} = \frac{\langle P, Q\rangle}{\langle Q, P\rangle}.$$

By raising both sides to the power $(q^k - 1)/m$, we can eliminate mth powers and obtain a unique value:

$$e_m(P, Q)^{(q^k-1)/m} = \left(\frac{\langle P, Q\rangle}{\langle Q, P\rangle}\right)^{(q^k-1)/m}.$$

Example 4.9 We return to the example of Subsection 4.3.1, where we computed $\langle P, Q\rangle^{20} = 5 + 3i$. Similarly, we can compute the Tate pairing $\langle Q, P\rangle$, which is the equivalence class of $1 + 2i \in \mathbb{F}_{11^2}^*/(\mathbb{F}_{11^2}^*)^6$. The unique value obtained from raising this outcome to the power 20 is $(1 + 2i)^{20} = 5 + 8i$. To illustrate the relation between the Weil and the Tate pairing, we also compute the Weil pairing $e_6(P, Q) = 5 + 3i$ and verify

$$\frac{\langle P, Q\rangle^{20}}{\langle Q, P\rangle^{20}} = 5 + 8i = e_6(P, Q)^{20}.$$

4.4.2 Efficiency of the Pairings

We next compare the Weil and the Tate pairing from a computational point of view. The relation shown in the previous section indicates that computation of the Weil pairing takes roughly twice as long as the computation of the Tate pairing. This can be verified by looking at the algorithms for computing the pairings. Considering the algorithm for computing the Weil pairing $e_m(P,Q)$, we see that we need to construct two functions f_A and f_B such that $\mathrm{div}(f_A) \sim m(P) - m(\mathcal{O})$ and $\mathrm{div}(f_B) \sim m(Q) - m(\mathcal{O})$. These functions need to be evaluated in divisors $B \sim (Q) - (\mathcal{O})$ and $A \sim (P) - (\mathcal{O})$, respectively. The Tate pairing $\langle P,Q\rangle$, on the other hand, only requires a single function g such that $\mathrm{div}(g) = m(P) - m(\mathcal{O})$, which needs to be evaluated in a divisor $D \sim (Q) - (\mathcal{O})$. Moreover, the function g can be constructed using the double-and-add method of Section 4.3, in contrast to f_A and f_B. This implies that the computation of the Weil pairing takes at least twice the computation time for the Tate pairing. The possible additional cost for the final exponentiation in the Tate pairing does not offset the difference in computation times. Hence, from a computational point of view, the Tate pairing is superior to the Weil pairing.

4.5 Efficient Implementation of the Tate Pairing

In many cryptographic applications, the computation of the pairing is the most costly operation in the algorithm. To make pairing-based cryptography practically feasible, it is therefore important to implement the pairing as efficiently as possible. This section is dedicated to implementation issues of the Tate pairing, which is faster than the Weil pairing. As we have seen, Miller's algorithm is more efficient if we evaluate the intermediate functions right away, rather than first compute the final function and then do the evaluation. In fact, the algorithm can be sped up even more. One idea is to adapt the computations in the algorithm such that they are made more efficient. A second suggestion is to make clever choices for the parameters in order to reduce the workload. We will outline how these two ideas can be exploited to make significant improvements to Miller's algorithm for the Tate pairing in Subsections 4.5.1 and 4.5.2, respectively.

4.5.1 Adaptation of the Algorithm

Along the line of the idea of improving the efficiency of Miller's algorithm lies the observation made in [31], regarding the fact that divisions are more expensive than multiplications. Instead of performing divisions along the way, we can represent the function f as a quotient of functions f_1/f_2, using only multiplications. Then only a single final division is required for the computation of f.

Another suggestion made in [31] is to precompute $[n]P$ for all possible values of n inside a 'window' of a certain number of bits. Then the number of addition operations can be reduced by considering windows in the binary representation of the exponent m, rather than performing additions bit by bit (i.e. using windows of size 1). This method is particularly useful if the binary expansion of m has a large Hamming weight (that is, a large number of ones).

The authors of [5] argue that if P is fixed or used repeatedly, some profit can be made on precomputing the coefficients of the lines $g_{V,P}$ that arise from Miller's algorithm. They also describe how to use the structure of the exponent $(q^k - 1)/m$ to speed up the final powering, in case a unique outcome of the pairing is required.

In [39], the authors propose to switch to a projective coordinate system to improve the efficiency of the elliptic curve addition and doubling operations. Further, they describe a way to directly compute the terms

$$g_{[a]P,P}(Q+Q')g_{[a+1]P}(Q') \quad \text{in (4.1)} \quad \text{and} \quad g_{[a]P,[a]P}(Q+Q')g_{[2a]P}(Q') \quad \text{in (4.2)},$$

thereby increasing the number of operations in the ground field $\mathbb{F}_q$, but reducing the number of operations in the larger field $\mathbb{F}_{q^k}$. Since doing computations in the extension field $\mathbb{F}_{q^k}$ is more expensive than in the smaller field $\mathbb{F}_q$, this method might pay off if k is sufficiently large. Lastly, if Miller's algorithm requires w consecutive doublings, then directly computing $[2^w]P$ can save $w-1$ multiplications.

Finally, [24] suggests to combine the two phases in the 'double-and-add' step, by computing the function f_{2a+1} at once. This is done by directly constructing the parabola that represents the term given by

$$\frac{g_{[a]P,[a]P} \cdot g_{[2a]P,P}}{g_{[2a]P}},$$

which occurs in the function

$$f_{2a+1} = f_{2a} \cdot \frac{g_{[2a]P,P}}{g_{[2a+1]P}} = f_a^2 \cdot \frac{g_{[a]P,[a]P}}{g_{[2a]P}} \cdot \frac{g_{[2a]P,P}}{g_{[2a+1]P}} = \frac{f_a^2}{g_{[2a+1]P}} \cdot \frac{g_{[a]P,[a]P} \cdot g_{[2a]P,P}}{g_{[2a]P}}.$$

As with the 'windows' method described above, the larger the hamming weight of the binary expansion of m is, the better this method works.

4.5.2 Choice of Parameters

The authors of [5] and [31] independently remark that the choice of the prime order m to a large extent influences the complexity of Miller's algorithm. Considering step 3 of the algorithm, we see that for the bits in the binary expansion $(m_t, \ldots, m_0)_2$ of m that equal 0, the addition step can be skipped. Obviously, it is wise to choose a prime m for which the Hamming weight of the binary expansion is as low as possible. For instance, one could use a Solinas prime of the form $m = 2^\beta \pm 2^\alpha \pm 1$. For practical values, [5] argues, such primes always exist.

Further, [31] describes that if we can write $N = mh$, where N is the group order, m a large prime factor and h a small cofactor, then the Tate pairing $\langle P, Q\rangle^{(q^k-1)/m} = g(D)^{(q^k-1)/m}$ (where $\mathrm{div}(g) = m(P) - m(\mathcal{O})$ and $D \sim (Q) - (\mathcal{O})$) can be computed as follows. Let g' be a function with $\mathrm{div}(g') = N(P) - N(\mathcal{O})$, then $\mathrm{div}(g') = h\mathrm{div}(g) = \mathrm{div}(g^h)$. Now we can compute

$$g'(D)^{(q^k-1)/N} = g(D)^{h(q^k-1)/(hm)} = \langle P, Q\rangle^{(q^k-1)/m},$$

using Miller's algorithm. If the binary representation of N has a low Hamming weight and m does not, then it can be quite profitable to compute the pairing with respect to N rather than m.

A result of [5] for supersingular curves, which was generalized to non-supersingular curves in [7], involves the choice of divisor D.

Theorem 4.10 (Barreto et al. [7], Theorem 3) *Let $P \in E(\mathbb{F}_q)[m]$ and $Q \in E(\mathbb{F}_{q^k})$ be linearly independent points on a curve with embedding degree $k > 1$. Then the Tate pairing $\langle P, Q\rangle^{(q^k-1)/m} = g(D)^{(q^k-1)/m}$, where $div(g) = m(P) - m(\mathcal{O})$ and $D \sim (Q) - (\mathcal{O})$ can be computed as*

$$\langle P, Q\rangle^{(q^k-1)/m} = g(Q)^{(q^k-1)/m} \quad \textit{for} \quad Q \neq \mathcal{O}.$$

As a consequence, we can replace formulas (4.1) and (4.2) in Miller's algorithm by

$$f_{a+1}(Q) = f_a(Q) \cdot \frac{g_{[a]P,P}(Q)}{g_{[a+1]P}(Q)}, \quad \text{and} \quad f_{2a}(Q) = f_a(Q)^2 \cdot \frac{g_{[a]P,[a]P}(Q)}{g_{[2a]P}(Q)},$$

thereby reducing the complexity of the algorithm. From the theorem also follows the following corollary:

Corollary 4.11 (Barreto et al. [7], Corollary 1) *Let $d > 1$ such that $d|k$. Then one can freely multiply or divide $g(Q)$ by any nonzero $\mathbb{F}_{q^{k/d}}$ factor without affecting the pairing value.*

In [5], it is described how the above corollary enables one to discard the $g_{[2]V}$ and g_{V+P} denominators in Miller's formula for certain supersingular curves. In [7], this result is generalized to the statement that for any curve the $g_{[2]V}$ and g_{V+P} denominators can be omitted if k is even, Q is such that $\Phi^{k/2}(Q) = -Q$, and m divides the order of Q. (Here Φ is the Frobenius endomorphism given Section 2.1.) Moreover, [7] gives an algorithm for generating these so-called 'pairing-friendly' groups.

Lastly, as noted in [31], one can profit from working with curves in small characteristic. These cases are particularly interesting from a cryptographic point of view, as they provide supersingular curves with large embedding degree ($k = 4, 6$, see Chapter 5). In characteristic two, certain arithmetic operations such as exponentiation and point doubling can be implemented very efficiently, which obviously improves the performance of Miller's algorithm. Similarly, in characteristic three, there are efficient methods for cubing and point tripling. To exploit this fact, we need to adapt Miller's algorithm to use the (signed) base-3 representation of the prime order m, in stead of its binary expansion.

Chapter 5

Embedding Degree

Consider the elliptic curve $E/\mathbb{F}_q$ and suppose there is an m-torsion point on $E(\mathbb{F}_q)$ for integer m. The definitions of the Weil and the Tate pairing on the curve E require some field extension of the ground field $\mathbb{F}_q$. The degree of this field extension is known as the Weil-, respectively Tate-embedding degree. For convenience, we distinguish between these by denoting the Weil-embedding degree by k_w and the Tate-embedding degree by k_t. In the previous chapters we have briefly mentioned how k_w and k_t can be determined. Namely, the Weil-embedding degree k_w is defined to be the smallest integer such that the curve over $\mathbb{F}_{q^{k_w}}$ contains all m-torsion points, i.e. $E[m] \subseteq E(\mathbb{F}_{q^{k_w}})$ (Remark 3.11). For the Tate pairing, k_t is the smallest integer such that $m|q^{k_t} - 1$ (Section 4.1).

In this chapter, we examine why the pairings require field extensions of this particular degree. For this purpose, we first give a lower bound k for k_w and k_t in Section 5.1. Next, we show that in all cases $k_t = k$ and in most cases $k_w = k$ in Section 5.2. Because of these facts, it is generally said that the *embedding degree of the curve with respect to* m is k. Namely, for any curve, the smallest extension degree for which either the Weil or the Tate pairing can be applied is the (Tate-)embedding degree $k_t = k$. (This justifies the use of k instead of k_t in Chapter 4.) Lastly, we discuss which curves have embedding degree sufficiently small for cryptographic purposes in Section 5.3.

At this point, we should remark that we often slightly abuse notation. Namely, if m is clear from the context, we frequently leave out the reference to m and simply talk about the the embedding degree of a curve. In cryptographic applications we usually work with curves for which the prime factorization of the order contains a single large prime m. In this case, when we mention the embedding degree of the curve, we actually mean with respect to that m.

Remark 5.1 Besides the term *embedding degree*, other names for the embedding degree often used in the literature are *extension degree*, *MOV degree*, *security parameter*, and *security multiplier*. We will always denote k by the term *embedding degree*; if we talk about *extension degree*, we mean the degree of some field extension, which is (in our definition) not necessarily equal to the embedding degree.

5.1 A Lower Bound

Recall that the pairings map to the group of (equivalence classes represented by) mth roots of unity μ_m for some integer m. Intuitively, one would expect that for the pairings to work, a necessary condition is that all the mth roots are present in the extension field. (In fact, [35] notes that this condition is often necessary for the non-degeneracy of the pairings.) Hence, this results in the following lower bound for both the Weil- and the Tate-embedding degree.

Definition 5.2 *Let $E/\mathbb{F}_q$ be an elliptic curve and $E[m]$ the group of m-torsion points on E for integer m. A lower bound for k_w and k_t for the curve E with respect to m is given by k, where k is the smallest integer for which all mth roots of unity are contained in the multiplicative group $\mathbb{F}_{q^k}^*$, so*

$$\mu_m \subseteq \mathbb{F}_{q^k}^* \quad \text{and} \quad \mu_m \not\subseteq \mathbb{F}_{q^l}^* \quad \text{for} \quad l < k.$$

Consider the following lemma that tells us something about the value of k.

Lemma 5.3 *Let $m > 0$, then*

$$\mu_m \subseteq \mathbb{F}_{q^k}^* \Leftrightarrow m | q^k - 1.$$

Proof. Suppose $\mu_m \subseteq \mathbb{F}_{q^k}^*$. We can write $q^k - 1 = cm + b$ for some integers c and $0 \leq b < m$. Let $a \in \mu_m$ be a primitive mth root of unity. Then since $a \in \mathbb{F}_{q^k}^*$, we have

$$1 = a^{q^k-1} = a^{cm+b} = (a^m)^c a^b = a^b.$$

Since $b < m$ and a is primitive, it follows that $b = 0$ and thus $m | q^k - 1$. So

$$\mu_m \subseteq \mathbb{F}_{q^k}^* \Rightarrow m | q^k - 1.$$

Now suppose $m | q^k - 1$. Then $q^k - 1 = cm$ for some integer c. Let $a \in \mu_m$, then we have

$$a^{q^k-1} = a^{cm} = 1,$$

so $a \in \mathbb{F}_{q^k}^*$. Therefore, $\mu_m \subseteq \mathbb{F}_{q^k}^*$, so we conclude

$$m | q^k - 1 \Rightarrow \mu_m \subseteq \mathbb{F}_{q^k}^*,$$

which completes the proof of the lemma. □

Due to Lemma 5.3, we can replace Definition 5.2 with the following equivalent definition:

Definition 5.4 *Let $E/\mathbb{F}_q$ be an elliptic curve and $E[m]$ the group of m-torsion points on E for integer m. A lower bound for k_w and k_t for the curve E with respect to m is given by k, where*

$$m | q^k - 1, \quad \text{and} \quad m \nmid q^l - 1 \quad \text{for} \quad 0 < l < k.$$

5.2 Additional Conditions

As we have seen in the previous section, the extension degree k, as defined in Definitions 5.2 and 5.4, is a lower bound for both the Weil-embedding degree k_w and the Tate-embedding degree k_t. Hence, $k_w \geq k$ and $k_t \geq k$ are necessary conditions for the Weil and the Tate pairing, respectively. In this section, we examine whether this condition is also sufficient, i.e. whether k_w and k_t can be as low as the optimal value k.

Recall that the Tate pairing takes as input one point from the group of m-torsion points $E(\mathbb{F}_{q^k})[m]$ and the other from the quotient group $E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k})$. For $k \geq 1$ there always exists an m-torsion point on $E(\mathbb{F}_{q^k})$, as well as a non-trivial (i.e. $\not\equiv \mathcal{O}$) representative in $E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k})$. Hence, the properties of the input parameters do not lead to any additional constraints on the required extension degree. Since also the output of the Tate pairing is well-defined for extension degree k, we find that $m | q^k - 1$ is a sufficient condition for the application of the Tate pairing. Thus the Tate-embedding degree always reaches the optimal value $k_t = k$.

The Weil pairing, on the contrary, does sometimes need a field extension of higher degree than k. This is caused by the fact that the Weil pairing, in contrast to the Tate pairing, takes two m-torsion

points as input. For the pairing to be non-trivial, these points need to be independent. Lemma 2.2 says that the group of points $E[m]$ is generated by two independent points. It follows that the complete group of m-torsion points needs to be on the curve over the extension field $E(\mathbb{F}_{q^{k_w}})$. Thus the stronger condition $E[m] \subseteq E(\mathbb{F}_{q^{k_w}})$, imposed by the independency requirement of the two m-torsion points, is necessary for the Weil pairing. In fact, this condition is equivalent to $m|q^{k_w} - 1$ and two other conditions, according to a result from Schoof [71].

Initially, the gap between the conditions $E[m] \subseteq E(\mathbb{F}_{q^k})$ and $m|q^k - 1$ was generally believed to be quite large, see for instance [28] and [54]. In other words, it was assumed that for a considerable number of cases the Tate pairing could be efficiently computed, whereas the Weil pairing would involve computations over an extension field of insuperably large degree. In 1998, however, Balasubramanian and Koblitz [3] showed that under a weak assumption, which is almost always satisfied in practical applications, the conditions $E[m] \subseteq E(\mathbb{F}_{q^k})$ and $m|q^k - 1$ are equivalent.

Theorem 5.5 (Balasubramanian, Koblitz [3]) *Let E be an elliptic curve defined over $\mathbb{F}_q$, and suppose that m is a prime that divides $N = \#E(\mathbb{F}_q)$ but does not divide $q-1$. Then $E[m] \subseteq E(\mathbb{F}_{q^k})$ if and only if $m|(q^k - 1)$.*

Theorem 5.5 asserts that if $m \nmid q - 1$, or equivalently $k > 1$, then the Weil-embedding degree k_w reaches the optimal value $k_w = k$. So if the condition $k > 1$ is met, as is often the case in practical applications, then both the Weil and the Tate pairing have embedding degree $k_w = k_t = k$. Further, the condition $k > 1$ is not necessary for $k_w = k$. That is, even if $k = 1$, then it is still possible that $E[m] \subseteq E(\mathbb{F}_{q^k})$ (see for instance Example 5.1 in [54]). A necessary condition for $E[m] \subseteq E(\mathbb{F}_{q^k})$ is that $m^2|\#E(\mathbb{F}_{q^k})$, as mentioned in [83]. On the other hand, [3] shows that if $k = 1$ and the group $E(\mathbb{F}_q)[m]$ is cyclic, then we have to go to the extension field $\mathbb{F}_{q^m}$ to get all m-torsion points, hence $k_w = m$ and $k_t = 1$.

In conclusion, we have seen that in all cases $k_t = k$ and $k_w \geq k$. Moreover, if $k > 1$, then $k_w = k_t = k$. For these reasons, it is generally said that the embedding degree of a curve (with respect to m) is k, where k is as in Definitions 5.2 and 5.4. Note that for elliptic curves with trace $t = 2$, we have $\#E(\mathbb{F}_q) = q - 1$ and thus $m|q - 1$ because $m|\#E(\mathbb{F}_q)$. So trace 2 curves have embedding degree $k = 1$ and the Tate pairing can be efficiently computed. Below, we give an example of an elliptic curve (with trace $\neq 2$) for which the Tate pairing reaches the lower bound k, but the Weil pairing does not.

5.2.1 Example

For $q = 31$, consider the curve $E/\mathbb{F}_q$ given by the Weierstrass equation $y^2 = x^3 + x + 17$. The number of points on the curve is $n = \#E(\mathbb{F}_{31}) = 25$, so $t = q + 1 - n = 7$. We set m equal to the largest prime divisor of n, so $m = 5$. We even have $m^2|n$, which is a necessary (but not sufficient) condition for $E[m] \subseteq E(\mathbb{F}_q)$. Then the embedding degree k of the curve with respect to m equals 1, since $m|q - 1$. Thus we can apply the Tate pairing without even having to extend the field $\mathbb{F}_q$.

However, the group of m-torsion points on $E(\mathbb{F}_q)$ is given by

$$E(\mathbb{F}_q)[m] = \{(3,4), (14,4), (3,27), (14,27), \mathcal{O}\} \cong \mathbb{Z}_5.$$

Since the group $E(\mathbb{F}_q)[m]$ is cyclic, one has to go to $\mathbb{F}_{q^m}$ to get all the m-torsion points. So in this case, $E[m] \subseteq E(\mathbb{F}_{31^5})$, but $E[m] \not\subseteq E(\mathbb{F}_{31^l})$ for $l < 5$. In other words, in order to apply the Weil pairing, we need to extend the ground field to a degree of 5. Although the Weil pairing can still be computed in this small toy example, practical values of q and m will render the pairing infeasible. The Tate pairing, on the contrary, can be computed very efficiently since $k = 1$.

5.3 Curves with Small Embedding Degree

In a nutshell, the Tate and the Weil pairings are bilinear maps that map a pair of points on an elliptic curve over a finite field $\mathbb{F}_q$ to the multiplicative group $\mathbb{F}_{q^k}^*$ of an extension field of degree k. Clearly, the larger the embedding degree k, the more computational effort needed to compute the pairing. Hence, in order to be able to efficiently evaluate the pairing, we need k to be sufficiently small. On the other hand, in cryptographic applications the security of the system is based on the hardness of a certain problem in $\mathbb{F}_{q^k}^*$, which is related to the value of k (see Chapter 7). In this case we want k to be small enough to enable efficient computation of the pairing, but simultaneously large enough to guarantee the hardness of the problem. On the whole, we are interested in elliptic curves with (moderately) small embedding degree. For the pairing to be computable, the value of k should in any case satisfy $k < (\log q)^2$. The authors of [3] show that curves satisfying this condition are very sparse, so a randomly chosen curve is with overwhelming probability not suitable for pairings-based cryptography. So far, the only two classes of curves that are known to have suitable embedding degree are supersingular curves and so-called MNT curves.

5.3.1 Supersingular Curves

The following two results come from Menezes [54].

Lemma 5.6 (Menezes [54], Lemma 2.9) *There exists an elliptic curve $E/\mathbb{F}_q$ such that $E(\mathbb{F}_q)$ has order $q+1-t$ over $\mathbb{F}_q$ if and only if one of the following conditions holds:*

1. *$t \not\equiv 0 (\mathrm{mod}\ p)$ and $t^2 \leq 4q$.*
2. *m is odd and one of the following holds:*
   (a) *$t = 0$.*
   (b) *$t^2 = 2q$ and $p = 2$.*
   (c) *$t^2 = 3q$ and $p = 3$.*
3. *m is even and one of the following holds:*
   (a) *$t^2 = 4q$.*
   (b) *$t^2 = q$ and $p \not\equiv 1$ (mod 3).*
   (c) *$t = 0$ and $p \not\equiv 1$ (mod 4).*

Lemma 5.7 (Menezes [54], Lemma 2.13) *Let E be a supersingular curve and $\#E(\mathbb{F}_q) = q + 1 - t$.*

1. *If $t^2 = q$, $2q$, or $3q$, then $E(\mathbb{F}_q)$ is cyclic.*
2. *If $t^2 = 4q$, then either $E(\mathbb{F}_q) \cong \mathbb{Z}_{\sqrt{q}-1} \oplus \mathbb{Z}_{\sqrt{q}-1}$ or $E(\mathbb{F}_q) \cong \mathbb{Z}_{\sqrt{q}+1} \oplus \mathbb{Z}_{\sqrt{q}+1}$, depending on whether $t = 2\sqrt{q}$ or $t = -2\sqrt{q}$ respectively.*
3. *If $t = 0$ and $q \not\equiv 3$ (mod 4), then $E(\mathbb{F}_q)$ is cyclic. If $t = 0$ and $q \equiv 3$ (mod 4), then either $E(\mathbb{F}_q)$ is cyclic, or $E(\mathbb{F}_q) \cong \mathbb{Z}_{(q+1)/2} \oplus \mathbb{Z}_2$.*

Let the character of $\mathbb{F}_q$ be p, i.e. $q = p^l$ for some integer l and prime p. Combining Lemmas 5.6 and 5.7, we can divide the class of supersingular curves into the following subclasses (for details see [54]):

1. $t = 0$ and $E(\mathbb{F}_q) \cong \mathbb{Z}_{q+1}$.
2. $t = 0$ and $E(\mathbb{F}_q) \cong \mathbb{Z}_{(q+1)/2} \oplus \mathbb{Z}_2$ (and $q \equiv 3 \pmod 4$).
3. $t^2 = q$ (and l is even).

4. $t^2 = 2q$ (and $p = 2$ and l is odd).

5. $t^2 = 3q$ (and $p = 3$ and l is odd).

6. $t^2 = 4q$ (and l is even).

The following result tells us the embedding degrees of the various classes of supersingular elliptic curves. In particular, for these curves we have $k \leq 6$, which makes them suitable for pairing-based cryptography.

Theorem 5.8 (Menezes [54], Table 5.2) *For the curves in class (1),(2),(3),(4),(5), and (6), we have embedding degree $k = 2, 2, 3, 4, 6,$ and 1 respectively.*

Proof.

1. Let $E/\mathbb{F}_q$ be a class (1) supersingular curve. Then $n = \#E(\mathbb{F}_q) = q + 1$, so $n \nmid q - 1$. But $n(n-2) = (q+1)(q-1) = q^2 - 1$, so $n | q^2 - 1$ and thus $k = 2$.

2. Similar to class (1) curves.

3. Let $E/\mathbb{F}_q$ be a class (3) supersingular curve. Then $t^2 = q$ and $n = \#E(\mathbb{F}_q) = q + 1 \pm \sqrt{q}$. Notice that $q^3 - 1$ factors as $q^3 - 1 = (q + 1 + \sqrt{q})(q + 1 - \sqrt{q})(q - 1)$, so $n | q^3 - 1$. Further, $q^j - 1$ does not contain $n = q + 1 \pm \sqrt{q}$ as a factor for $0 < j < 3$.

4. Let $E/\mathbb{F}_q$ be a class (4) supersingular curve. Then $t^2 = 2q$ and $n = \#E(\mathbb{F}_q) = q + 1 \pm \sqrt{2q}$. Notice that $q^4 - 1$ factors as $q^4 - 1 = (q + 1 + \sqrt{q})(q + 1 - \sqrt{q})(q + 1)(q - 1)$, so $n | q^4 - 1$. Further, $q^j - 1$ does not contain $n = q + 1 \pm \sqrt{2q}$ as a factor for $0 < j < 4$.

5. Let $E/\mathbb{F}_q$ be a class (5) supersingular curve. Then $t^2 = 3q$ and $n = \#E(\mathbb{F}_q) = q + 1 \pm \sqrt{3q}$. Notice that $q^6 - 1$ factors as $q^6 - 1 = (q + 1 + \sqrt{3q})^2 (q + 1 - \sqrt{3q})^2 (q + 1)(q - 1)$, so $n | q^6 - 1$. Further, $q^j - 1$ does not contain $n = q + 1 \pm \sqrt{3q}$ as a factor for $0 < j < 6$.

6. Let $E/\mathbb{F}_q$ be a class (6) supersingular curve. Then $t^2 = 4q$ and $n = \#E(\mathbb{F}_q) = q + 1 \pm 2\sqrt{q} = (\sqrt{q} \pm 1)^2$. Notice that $q - 1$ factors as $q - 1 = (\sqrt{q} + 1)(\sqrt{q} - 1)$, so $\sqrt{n} | q - 1$. Hence the embedding degree of $E/\mathbb{F}_q$ with respect to $\sqrt{n}$ equals one. □

Remark 5.9 The theorem above deals with the (Tate-)embedding degree k. For the classes (1)-(5) the (Tate-)embedding degree $k > 1$, so by Theorem 5.5 we have $k_w = k$. This is not sure for class (6), since here $k = 1$. Nevertheless, following the proof of Lemma 5.11 in [54], it can easily be verified that $k_w = k = 1$ for class (6) supersingular curves as well.

Example 5.10 Consider the curve $E/\mathbb{F}_{11}$ from previous examples in Subsections 2.5.1 and 3.4.1 given by $E : y^2 = x^3 + 3x$. The trace t of this curve equals 0, because $\#E(\mathbb{F}_{11}) = 12 = q + 1$. Hence the curve is supersingular. As remarked in Subsection 3.4.1, the group structure of $E(\mathbb{F}_{11})$ is cyclic and, in particular, equivalent to $\mathbb{Z}_{12}$. Therefore, E falls into class (1) and has embedding degree 2. This corresponds to the observations made in Subsection 3.4.1.

According to the following theorem, it is easy to construct supersingular curves.

Theorem 5.11 (Ireland, Rosen [38]) *Let p be a prime.*

- *The curve $E/\mathbb{F}_p$ given by $E : y^2 = x^3 + ax$ is supersingular if and only if $p \equiv 3 \pmod 4$.*
- *The curve $E/\mathbb{F}_p$ given by $E : y^2 = x^3 + b$ is supersingular if and only if $p \equiv 2 \pmod 3$.*

Note that there are also supersingular curves of the form $E : x^3 + ax + b$, with $a, b \neq 0$.

Hence, the class of supersingular elliptic curves provides us with a large collection of curves with small embedding degree. We point out that supersingular curves come equipped with a remarkably rich structure, which entails both advantages and disadvantages for their use in cryptography. We will elaborate on this in Chapter 7. One of the advantages of the rich structure is the existence of distortion maps on supersingular curves, see Chapter 6.

Note that there exist supersingular curves of genus > 1 (i.e. not elliptic) that have embedding degree larger than 6. We will not go into detail, as curves of higher genus fall outside the scope of this study. Interested readers are referred to [30], where an upper bound on the embedding degree depending only on the genus of the curve is given, along with examples of supersingular curves with embedding degree up to 12. In [68] these results are extended to the even more general case of supersingular abelian varieties.

5.3.2 MNT-Curves

Until 2001, supersingular curves were the only curves known to have embedding degree small enough to apply the Weil or the Tate pairing. In [61], however, Miyaji, Nakabayashi and Takano describe a method to construct ordinary (i.e. non-supersingular) curves for embedding degree $k = 3, 4$ and 6. We refer to these curves as MNT-curves. Their approach is as follows. Let $E/\mathbb{F}_q$ be an elliptic curve of prime order $n = \#E(\mathbb{F}_q) = q + 1 - t$, where t is the trace of E. Next, fix a k ([61] only deals with $k = 3, 4, 6$) and check which restrictions on the parameters n, q and t are imposed by the condition

$$m | q^k - 1, \quad \text{and} \quad m \not| q^l - 1 \quad \text{for} \quad 0 < l < k.$$

For $k = 3, 4, 6$, these restrictions, also called *MNT-criteria*, are listed in the following theorem.

Theorem 5.12 (Miyaji et al. [61], Theorems 2-4) *Let $E/\mathbb{F}_q$ be an elliptic curve with trace t.*

- *If (q, t) can be represented by $q = 12l^2 - 1$ and $t = -1 \pm 6l$ for some $l \in \mathbb{Z}$, then the embedding degree of E is $k = 3$.*
- *If (q, t) can be represented by $q = l^2 + l + 1$ and $t = -l, l + 1$ for some $l \in \mathbb{Z}$, then the embedding degree of E is $k = 4$.*
- *If (q, t) can be represented by $q = 4l^2 + 1$ and $t = 1 \pm 2l$ for some $l \in \mathbb{Z}$, then the embedding degree of E is $k = 6$.*

The next step is to construct curves that fall into one of the three classes described in Theorem 5.12. This is done by using the complex multiplication (CM-)method (for details see [2]), which is the only known method for building a curve for given t and q. The difficulty, however, is that in order to apply the CM-method, one needs to know the solution to a certain equation, called the CM-equation. Miyaji et al. [61] show that for $k \in \{3, 4, 6\}$ and q prime, the CM-equation can be reduced to an equation of particular form, known as Pell's equation, for which the solution is known.

The independent studies of Barreto et al. [6] and Dupont et al. [23] generalize the MNT-criteria to arbitrary k, i.e. not restricted to $k \in \{3, 4, 6\}$ (q is still assumed to be prime). For general k, however, the CM-equation cannot be reduced to Pell's equation, and therefore the method of [61] does not work. The authors of [6] circumvent this problem by first choosing a value to serve as solution to the CM-equation and next searching for parameters that yield a CM-equation with that particular solution. They further provide examples of curves with embedding degree 7, 11 and 12.

In [23], the problem of finding a solution to the CM-equation is solved differently. Their approach is to choose the parameter t in such a way that the solution to the CM-equation is enforced to be small. (To be precise, they pick $|t|$ to be the largest integer within the Hasse bound, i.e. $|t| = \lfloor 2\sqrt{q} \rfloor$.) They then perform an exhaustive search on the remaining parameters to see if the resulting CM-equation has a suitable solution. They clarify their method with examples of the construction of elliptic curves with embedding degree $k = 5, 7, 10, 11$ and 50.

Downside to both methods is that they cannot be used to find curves over fields of fixed prime order q. Instead, the algorithms take as input the desired embedding degree k and a lower bound on the size of the subgroup m, and output a prime q and an elliptic curve $E/\mathbb{F}_q$ satisfying the prerequisites. Another disadvantage is that for arbitrary k, both methods cannot reach a curve of (near-)prime order, as is desirable in cryptographic applications. In fact, for the general methods, the ratio $\log q / \log m$ can be no less than 2, which boils down to having a subgroup of prime order m of size $O(\sqrt{q})$ at best. On the bright side, [6] also describes a method that can achieve a better ratio for the particular case where k is divisible by 3.

Recently, Brezing and Weng [14] have extended the work of [6] to reach a better relation between the prime order m and the field characteristic q for arbitrary embedding degree k. They corroborate their results with examples of curves with ratio $\log q / \log m$ close to one. In particular, for k prime they achieve a ratio of approximately $\frac{k+2}{k-1}$, which means that m is of size $O(q^{(k-1)/(k+2)})$. Unfortunately, this method also suffers from the fact that q cannot be fixed beforehand.

We next give an example of MNT-curves for $k = 3, 4, 6$.

Example 5.13

- Consider the curve $E/\mathbb{F}_{47}$ given by Weierstrass equation $y^2 = x^3 + 6x + 5$. Then $n = \#E(\mathbb{F}_{47}) = 37$ and $t = q + 1 - n = 11$. For $l = 2$ we have $q = 12l^2 - 1$ and $t = -1 + 6l$, and thus the embedding degree of this curve is $k = 3$.
- Consider the curve $E/\mathbb{F}_{31}$ given by Weierstrass equation $y^2 = x^3 + 2x + 2$. Then $n = \#E(\mathbb{F}_{31}) = 26$ $(= 2 \cdot 13)$ and $t = q + 1 - n = 6$. For $l = 5$ we have $q = l^2 + l + 1$ and $t = l + 1$, and thus the embedding degree of this curve (with respect to 13) is $k = 4$.
- Consider the curve $E/\mathbb{F}_{101}$ given by Weierstrass equation $y^2 = x^3 + 3x + 6$. Then $n = \#E(\mathbb{F}_{101}) = 91$ $(= 7 \cdot 13)$ and $t = q + 1 - n = 11$. For $l = 5$ we have $q = 4l^2 + 1$ and $t = 1 + 2l$, and thus the embedding degree of this curve (with respect to either 7 or 13) is $k = 6$.

Chapter 6

Distortion Maps

Let P be an m-torsion point with coordinates in $\mathbb{F}_q$ for m an integer coprime to q. As we have seen, the outcome of Weil pairing $e_m(P,P)$ (Theorem 3.5) and, for $k > 1$, the Tate pairing $\langle P, P\rangle$ (Lemma 4.5) is the identity. Then, because of the bilinearity, the pairing of two linearly dependent points yields the identity as well. Hence, in applications we often need two linearly independent points. This chapter deals with Verheul's distortion map on supersingular curves [83], which is a useful tool for finding such a pair of independent points.

In the next section we give the definition of the distortion map and show that they do exist on supersingular curves, but not on most non-supersingular curves. In Section 6.2 we describe how the distortion map can be used to modify the Weil and Tate pairings to make them satisfy a stronger non-degeneracy property, which is often desired in cryptographic applications. We deal with the cryptographic use of (modified) pairings in Section 6.3.

6.1 Definition

Consider the following definition of a distortion map.

Definition 6.1 *Let m be an integer coprime to q, $E/\mathbb{F}_q$ an elliptic curve and $P \in E(\mathbb{F}_q)[m]$. A distortion map with respect to P is an endomorphism $\phi \in End(E)$ that maps the point P to a point $\phi(P)$ that is linearly independent from P.*

Note that an endomorphism always maps the point at infinity $\mathcal{O}$ to itself, so there exists no distortion map with respect to $\mathcal{O}$. Moreover, the image of an m-torsion point under an endomorphism is again an m-torsion point. Hence, the distortion map ϕ with respect to $P \in E(\mathbb{F}_q)[m]$, if existing, maps to an m-torsion point $\phi(P) \in E[m]$. Furthermore, if the group of m-torsion points is not contained in the curve over the ground field, i.e. $E[m] \not\subseteq E(\mathbb{F}_q)$, then the mth torsion group over the ground field $E(\mathbb{F}_q)[m]$ is cyclic. So, being independent from P, the image $\phi(P)$ of the point P will have coordinates in an extension field of $\mathbb{F}_q$.

6.1.1 Non-Supersingular Curves

Let $E/\mathbb{F}_q$ be a non-supersingular elliptic curve. Its endomorphism ring $\mathrm{End}(E)$ contains endomorphisms such as the multiplication-by-n map and the Frobenius map. Furthermore, $\mathrm{End}(E)$ is abelian, i.e. $\phi_1(\phi_2(P)) = \phi_2(\phi_1(P))$ for all $\phi_1, \phi_2 \in \mathrm{End}(E)$ and $P \in E$. Due to this fact, non-supersingular curves cannot contain a distortion map if $E[m] \not\subseteq E(\mathbb{F}_q)$, as proven by Verheul [83].

Theorem 6.2 (Verheul [83], Theorem 11) *Let $E/\mathbb{F}_q$ be a non-supersingular curve and let $P \in E(\mathbb{F}_q)[m]$ for integer m. If m is relatively prime to $p = char(q)$ and $E[m] \not\subseteq E(\mathbb{F}_q)$, then there cannot exist a distortion map ϕ with respect to P.*

Proof. Suppose such a distortion map ϕ does exist. Then $Q = \phi(P) \notin E(\mathbb{F}_q)$, because $E[m] \not\subseteq E(\mathbb{F}_q)$. Since the endomorphism ring is abelian, we have

$$Q = \phi(P) = \phi(\Phi(P)) = \Phi(\phi(P)) = \Phi(Q).$$

This implies that $Q \in E(\mathbb{F}_q)$, which is a contradiction. □

The following example concerns the case that $E[m]$ is contained in the ground field, which was left as an open question by Verheul [83]. In fact, the example shows that for $E[m] \subseteq E(\mathbb{F}_q)$, a distortion map can exist on non-supersingular curves.

Example 6.3 Consider the curve $E/\mathbb{F}_{197} : y^2 = x^3 - 4x$. Then $\#E(\mathbb{F}_{197}) = 196 = 2^2 \cdot 7^2$. The torsion group $E[7] \subseteq E(\mathbb{F}_{197})$ can be generated by the two linearly independent points $P_1 = (24, 23)$ and $P_2 = (173, 125)$. Observe that 14 is a fourth root of unity in $\mathbb{F}_{197}$, so $14^2 = -1$. Hence, the map

$$\phi : (x, y) \to (-x, 14 \cdot y) \quad \text{and} \quad \phi : \mathcal{O} \to \mathcal{O},$$

is an endomorphism, i.e. $\phi \in \mathrm{End}(E)$. Moreover, $\phi(P_1) = P_2$, so ϕ is a distortion map on $E[7]$.

We should remark that for cryptographic applications, it is desirable to have m close or equal to $\#E(\mathbb{F}_q)$. However, [83] mentions that a necessary condition for $E[m] \subseteq E(\mathbb{F}_q)$, is that $m^2 | \#E(\mathbb{F}_q)$. Hence, non-supersingular curves containing a distortion map generally do not yield cryptographically interesting cases.

6.1.2 Supersingular Curves

The endomorphism ring of supersingular curves, on the other hand, is not abelian. For this reason, supersingular curves enable the existence of distortion maps even if $E[m]$ is not contained in the ground field. In fact, Verheul [83] remarks that distortion maps always exist on supersingular curves, with only a finite number of exceptions.

We already came across a distortion map in Section 3.4.1, where we used the endomorphism

$$\phi : (x, y) \to (-x, iy) \quad \text{on the curve} \quad E/\mathbb{F}_{11} : y^2 = x^3 + 3x$$

to map the point $(3, 5)$ to the independent point $(8, 5i)$. This and other distortion maps are listed in Table 6.1 (see [42, 43]). Note that these endomorphisms do not need to be distortion maps with respect to all points on the curve. We have remarked already that distortion maps with respect to $\mathcal{O}$ do not exist. Furthermore, returning to our usual example $E/\mathbb{F}_{11} : y^2 = x^3 + 3x$, we see that the 2-torsion point $P_1 = (0, 0)$ is mapped to itself by $\phi : (x, y) \to (-x, iy)$. Since all other points on $E(\mathbb{F}_q)$ have $y \neq 0$, they are mapped to linearly independent points. Hence, ϕ is a distortion map with respect to all points on $E(\mathbb{F}_q)$ except for $\mathcal{O}$ and P_1.

Field	Curve	Distortion map	Conditions	Group order
$\mathbb{F}_p$	$y^2 = x^3 + ax$	$(x, y) \mapsto (-x, iy)$ $i^2 = -1$	$p \equiv 3 (\mathrm{mod} 4)$	$p + 1$
$\mathbb{F}_p$	$y^2 = x^3 + a$	$(x, y) \mapsto (\zeta x, y)$ $\zeta^3 = 1,\ \zeta \neq 1$	$p \equiv 2 (\mathrm{mod} 3)$	$p + 1$
$\mathbb{F}_{p^2}$	$y^2 = x^3 + a$ $a \neq \mathbb{F}_p$	$(x, y) \mapsto (\omega \frac{x^p}{r^{(2p-1)/3}}, \frac{y^p}{r^{p-1}})$ $r^2 = a,\ r \in \mathbb{F}_{p^2}$ $\omega^3 = r,\ \omega \in \mathbb{F}_{p^6}$	$p \equiv 2 (\mathrm{mod} 3)$	$p^2 - p + 1$

Table 6.1: Distortion maps on supersingular curves

6.2 Modified Pairings

When a distortion map is available, we can modify the Weil and Tate pairing such that they satisfy a stronger non-degeneracy property.

Definition 6.4 *Let m be a prime and $E/\mathbb{F}_q$ a supersingular curve, on which a distortion map ϕ exists with respect to all points in $E(\mathbb{F}_q)[m]$ other than $\mathcal{O}$. Let $P, Q \in E[m]$, then the modified Weil pairing of m-torsion points P and Q is defined as*

$$\widehat{e}_m(\cdot,\cdot) : E[m] \times E[m] \to \mu_m \quad \text{with} \quad \widehat{e}_m(P,Q) = e_m(P, \phi(Q)).$$

The modified Weil pairing satisfies the following property.

Theorem 6.5 *Let $\widehat{e}_m$ be the modified Weil pairing as in Definition 6.4 and let $P \in E(\mathbb{F}_q)[m]$, $P \neq \mathcal{O}$. Then $\widehat{e}_m(P,P) \neq 1$. We say that the modified Weil pairing is* strongly non-degenerate.

Proof. If $P \in E(\mathbb{F}_q)[m]$ with $P \neq \mathcal{O}$, then $\phi(P)$ is an m-torsion point independent from P. Then by Corollary 3.6

$$\widehat{e}_m(P,P) = e_m(P, \phi(P)) \neq 1.$$

□

Similarly, we can modify the Tate pairing.

Definition 6.6 *Let m be a prime and $E/\mathbb{F}_q$ a supersingular curve, on which a distortion map ϕ exists with respect to all points in $E(\mathbb{F}_q)[m]$ other than $\mathcal{O}$. Let $P \in E(\mathbb{F}_{q^k})[m]$ and $Q \in E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k})$, then the modified Tate pairing of P and Q is defined as*

$$\widehat{\langle\cdot,\cdot\rangle} : E(\mathbb{F}_{q^k})[m] \times E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k})^m \quad \text{with} \quad \widehat{\langle P,Q\rangle} = \langle P, \phi(Q)\rangle.$$

Let $P \in E(\mathbb{F}_q)[m]$. Recall from Lemma 4.5 that for $k > 1$ the Tate pairing $\langle P,P\rangle$ is trivial (i.e. $\langle P,P\rangle \in (\mathbb{F}_{q^k})^m \equiv 1$). We show that the modified Tate pairing satisfies the strong non-degeneracy property for $k > 1$ (or equivalently $m \nmid q-1$), and thus $\widehat{\langle P,P\rangle}$ is non-trivial for $P \in E(\mathbb{F}_q)[m]$.

Theorem 6.7 *Let $\widehat{\langle\cdot,\cdot\rangle}$ be the modified Tate pairing as in Definition 6.6 and let $P \in E(\mathbb{F}_q)[m]$, $P \neq \mathcal{O}$. If $m \nmid q-1$, then $\widehat{\langle P,P\rangle} \notin (\mathbb{F}_{q^k})^m$. We say that the modified Tate pairing is* strongly non-degenerate.

Proof. If $P \in E(\mathbb{F}_q)[m]$ with $P \neq \mathcal{O}$, then $\phi(P) \in E(\mathbb{F}_{q^k})[m]$ is an m-torsion point independent from P. Then by Corollary 4.6

$$\widehat{\langle P,P\rangle} = \langle P, \phi(P)\rangle \neq 1.$$

□

Observe that the proof of Corollary 4.6 uses the fact that $\langle P,P\rangle \equiv 1$ for $k > 1$. Now consider the case that $k = 1$. Then if $\widehat{\langle P,P\rangle}$ is trivial, the non-degeneracy of the Tate pairing implies that $\langle P,P\rangle$ is non-trivial. Hence for $k = 1$, either the regular Tate pairing $\langle\cdot,\cdot\rangle$ or the modified Tate pairing $\widehat{\langle\cdot,\cdot\rangle}$ (if existing) is strongly non-degenerate.

6.3 Cryptographic Use

In this section, we explain how (modified) pairings can be used in cryptographic applications. The security of cryptosystems is generally based on some hard problem (see Chapter 7). Such a problem is defined on a group G that is cyclic, i.e. generated by a single element. Ideally, the order of the element, and thus the order of the group, is a prime. Hence, we would like to define the pairings on cyclic groups of prime order. Moreover, we would like the outcome of the pairings to be non-trivial for all input pairs with both parameters different from $\mathcal{O}$. We can distinguish between asymmetric and symmetric pairings.

6.3.1 Asymmetric Pairings

Let $E/\mathbb{F}_q$ be an elliptic curve and $n = \#E(\mathbb{F}_q)$ the group order of the curve. Let m be a prime such that $m|n$ and $m \nmid q$. Then the group of m-torsion points $E[m]$ has the structure $E[m] \cong \mathbb{Z}_m \oplus \mathbb{Z}_m$ and is thus generated by two elements, say S and T. Denote the group generated by S by G_1 and the group generated by T by G_2, i.e. $G_1 = \langle S \rangle$ and $G_2 = \langle T \rangle$. Now consider the Weil pairing $e_m(P, Q)$, where $P \in G_1$ and $Q \in G_2$. By Corollary 3.6, $e_m(P, Q) = 1$ if and only if $P = \mathcal{O}$ or $Q = \mathcal{O}$. Moreover, e_m maps to the group μ_m of mth roots of unity, which is a cyclic group of order m as well. Denote this group by G_3. Hence, we can create a pairing with the desired properties as above from the Weil pairing, by restricting the first coordinate to G_1 and the second to G_2:

$$e_m(\cdot, \cdot) : G_1 \times G_2 \to G_3, \quad \text{with} \quad e_m(P, Q) = 1 \Leftrightarrow P = \mathcal{O} \vee Q = \mathcal{O}.$$

Similarly, we can restrict the first and second coordinate of the Tate pairing to G_1, respectively G_2. Clearly, an exponentiation is required to get a unique outcome in G_3. Then by Corollary 4.6, for $k > 1$ we have

$$\langle \cdot, \cdot \rangle^{(q^k-1)/m} : G_1 \times G_2 \to G_3, \quad \text{with} \quad \langle P, Q \rangle^{(q^k-1)/m} = 1 \Leftrightarrow P = \mathcal{O} \vee Q = \mathcal{O}.$$

In cryptography, these pairings are said to be *asymmetric*, as $G_1 \neq G_2$.

If $m^2 \nmid n$, then $E[m] \not\subseteq E(\mathbb{F}_q)$. Hence, the mth torsion group is not contained in the curve over the ground field and the group G_1 is given by $G_1 = E(\mathbb{F}_q)[m]$. In practice, often $P \in G_1$ is given, and we need to find an appropriate Q such that the pairing is non-trivial, i.e. $Q \in G_2, Q \neq \mathcal{O}$. This can be a problem when no distortion map is available. Note that these asymmetric pairings are non-degenerate by definition. The notion of strong non-degeneracy does not apply to these asymmetric pairings, as the input parameters come from different groups.

6.3.2 Symmetric Pairings

A symmetric pairing can be obtained by restricting both parameters of the modified Weil pairing to the group G_1. Then by Theorem 6.5 and the bilinearity of the pairing, the modified Weil pairing $\hat{e}_m(P, Q)$ of $P, Q \in G_1$ is trivial if and only if $P = \mathcal{O}$ or $Q = \mathcal{O}$. Hence

$$\hat{e}_m(\cdot, \cdot) : G_1 \times G_1 \to G_3, \quad \text{with} \quad \hat{e}_m(P, Q) = 1 \Leftrightarrow P = \mathcal{O} \vee Q = \mathcal{O}.$$

Similarly, we can restrict both parameters of the modified Tate pairing to G_1 and perform an exponentiation to get a unique outcome. Then by Theorem 6.7 and the bilinearity of the pairing, we have for $k > 1$:

$$\widehat{\langle \cdot, \cdot \rangle}^{(q^k-1)/m} : G_1 \times G_1 \to G_3, \quad \text{with} \quad \widehat{\langle P, Q \rangle}^{(q^k-1)/m} = 1 \Leftrightarrow P = \mathcal{O} \vee Q = \mathcal{O}.$$

These pairings are called *symmetric*, because both input parameters come from the same cyclic group G_1.

Observe that, by construction, the symmetric pairings are strongly non-degenerate. Hence, when given an $S \in G_1$, we do not have any problem to find an appropriate T. In fact, because of the strong non-degeneracy, we can take S itself, or any multiple of S except for $\mathcal{O}$. Therefore, symmetric pairings are in practice often more convenient than asymmetric pairings.

Chapter 7

Elliptic Curve Cryptography

Elliptic curve cryptography is realized by considering the well-known problem of taking discrete logarithms on the group of points on an elliptic curve. After giving an introduction to the discrete logarithm problem defined on an arbitrary group in Section 7.1, we focus on its analogue on elliptic curves in Section 7.2. Lastly, Section 7.3 describes how pairings can be used to make certain elliptic curves into so-called Gap Diffie-Hellman groups, which lie at the heart of pairing-based systems.

7.1 Introduction to the Discrete Logarithm Problem

We give the general definition of the discrete logarithm problem and related problems, discuss the currently known attacks and describe some standard protocols based on this problem.

7.1.1 Discrete Logarithm and Related Problems

The security of many cryptosystems relies on the hardness of the discrete logarithm problem. In the following definition, we describe when we consider a problem to be *hard*, in a rather informal manner. For a more formal definition, see [64] (which uses term *intractable* instead of the term *hard*).

Definition 7.1 *We consider a problem to be easy when there exists an algorithm that solves the problem with running time that is polynomial in size of the input. We consider a problem to be hard when no such polynomial time algorithm exists.*

An algorithm is called *subexponential* time when it runs asymptotically faster than any exponential time algorithm, but asymptotically slower than any polynomial time algorithm (see [54]). Hence, we can divide the set of hard problems into two classes. We say that problems that can be solved by a subexponential time algorithm are easier than problems solved in exponential time at best, even though both classes of problems are hard.

We are now ready to give the definition of the discrete logarithm problem and related problems. We regard these problems on the finite multiplicative group $(G, *)$ of order n. As explained in the next section, we can, without loss of generality, assume that the order n of G is prime. In other words, G is cyclic and can be generated by a single element, say g.

Definition 7.2 *Let $h \in G$, such that $h = g^x$ for some unknown $x \in \mathbb{Z}_n^*$. Given g and h, the discrete logarithm problem (DL) is to find x. We use the notation $DL_g(h) = x$.*

Closely related to the discrete logarithm problem is the computational Diffie-Hellman problem.

Definition 7.3 *Let $a, b \in \mathbb{Z}_n^*$. Given g, g^a, g^b, the computational Diffie-Hellman problem (CDH) is to find the element $h \in G$ such that $h = g^{ab}$. We use the notation $CDH_g(g^a, g^b) = h$.*

Obviously, CDH is no harder than DL; the ability to compute discrete logarithms directly enables one to solve the computational Diffie-Hellman problem. Thus when the computational Diffie-Hellman problem in a group is hard, then so is the discrete logarithm problem. Conversely, it is not known whether the hardness of DL in general guarantees that CDH is hard as well. For a group of given prime order n, however, [52, 53] show that the two problems are equivalent if one can construct an elliptic curve over F_n with certain properties. But since there exists no provable method for constructing such curves for general prime order n, [52, 53] do not provide proof of the equivalence of DL and CDH in any group. Nevertheless, since their ideas are considered a strong heuristical argument, the discrete logarithm problem and the computational Diffie-Hellman problem are widely believed to be equivalent. Henceforth, when examining the security of cryptosystems based on the computational Diffie-Hellman problem, we will consider the hardness the discrete logarithm problem.

Besides the computational Diffie-Hellman problem, there exists a weaker version, known as the decision Diffie-Hellman problem.

Definition 7.4 *Let $a, b, c \in \mathbb{Z}_n^*$. Given g, g^a, g^b, g^c, the decision Diffie-Hellman problem (DDH) is to decide whether $g^c = g^{ab} \pmod n$. We use the notation $DDH_g(g^a, g^b, g^c) = 1$ if $CDH_g(g^a, g^b) = g^c$ and $DDH_g(g^a, g^b, g^c) = 0$ otherwise.*

It is clear that the computational Diffie-Hellman problem is at least as hard as its decisional version. Indeed, a method to solve DDH would be to compute $h = \mathrm{CDH}_g(g^a, g^b)$, and check whether $g^c = h$. Conversely, for most groups it is not clear whether DDH is easier than CDH. Groups that do satisfy this property, i.e. where CDH is hard but DDH is easy, are named *Gap Diffie-Hellman (GDH) groups*, which will be discussed in Section 7.3. The relation between DL, CDH and DDH can be depicted as follows.

$$\mathrm{DL} \longrightarrow \mathrm{CDH} \longrightarrow \mathrm{DDH}$$

Throughout the report, $A \longrightarrow B$ means that problem A is at least as hard as problem B.

7.1.2 Attacks on the Discrete Logarithm Problem

Several attacks are known on the discrete logarithm problem (see [54, 63, 82]). The methods that do not use the properties of the underlying group, besides multiplication, inversion and unique encoding of the group elements, are called *generic*. Examples of generic methods are Shanks' "Baby-Step Giant-Step" method and Pollard's "Lambda" and "Rho" (or "catching kangaroos") methods. The fact that these algorithms do not require particular properties makes them work in any group.

A result of Shoup [75] states that any generic method takes at least $O(\sqrt{n})$ operations to solve the discrete logarithm problem, where n is the prime order of the group. Therefore, the generic methods mentioned above are referred to as *square root methods*. Note the amount of work that needs to be done to solve the discrete logarithm with a generic method grows exponentially in the size of the input. This makes groups on which no attacks other than generic ones are known suitable for the design of DL-based cryptographic protocols.

Note that so far, we have assumed that the discrete logarithm problem is defined on a group that has prime order. Of course we can just as well work with a group of arbitrary order instead. However, by the Chinese Remainder Theorem, it suffices to solve the problem in the subgroups of prime power order. Moreover, due to Pohlig and Hellman [67], there exists a simplification that reduces DL to the subgroups of prime order in polynomial time. This so-called "Pohlig-Hellman method" does not use any additional properties of the group and is thus a generic method. In other words, DL over a group of non-prime order is no harder than DL over the subgroup of largest prime order. Therefore, without loss of generality, we may assume that the order of the group on which we consider the discrete logarithm problem is of prime order, as in Definitions 7.2-7.4.

On the other hand, methods that do make use of specific properties of the underlying group are known as "Index Calculus" algorithms. Briefly, these algorithms require a small suitable set of elements, called a factor base, and a way to decompose random group elements into elements of the factor base, if possible. Consequently, the Index Calculus method can only be applied to groups for which such a suitable factor base and decomposition method are available. So far, the only known groups that fit the bill are multiplicative groups of finite fields and class groups of imaginary quadratic fields. An extension to the Index Calculus method worth mentioning is the "Number Field Sieve" (NFS) [81]. The NFS uses a somewhat different approach to exploit the ideas of the Index Calculus Method. Since it requires the same tools as the Index Calculus method, the NFS cannot be applied to other groups than those to which the Index Calculus method applies. The NFS and Index Calculus method result in a probabilistic subexponential time algorithm. For details on these methods, the reader is referred to [54, 63, 57, 81].

Most DL-based protocols were originally defined on the multiplicative group of a finite field $\mathbb{F}_q^*$. The Index Calculus and NFS, providing subexponential algorithms, are at present the best known methods for solving DL in this group. Therefore, apart from weak parameter choices, the hardness of the discrete logarithm problem in $\mathbb{F}_q^*$ is generally believed to be subexponential.

7.1.3 Some Standard Protocols

For future reference, we describe some standard protocols based on the computational Diffie-Hellman problem.

Diffie-Hellman key exchange
In order two establish a common secret key over an insecure channel, two parties A and B can carry out the Diffie-Hellman key exchange [22]. (We remark that despite the name *key exchange*, the established secret does not necessarily need to be used as a key. Moreover, since the secret is not known beforehand but agreed upon in the protocol, *secret agreement* would perhaps have been a better name. Nonetheless, we stick with the term key exchange, as this is the common name for the protocol in the literature.) Let G of prime order n and g be as in Definitions 7.2-7.4. The common secret shared by A and B will be an element of G.

1. A randomly selects an integer $a \in \mathbb{Z}_n^*$, computes g^a, and transmits g^a to B.
2. B randomly selects an integer $b \in \mathbb{Z}_n^*$, computes g^b, and transmits g^b to A.
3. A computes $(g^b)^a$ and B computes $(g^a)^b$; the shared secret is $h = g^{ab}$.

It is clear that the security of the Diffie-Hellman key exchange is based on the hardness of the computational Diffie-Hellman problem. Namely, an eavesdropper who intercepts the transmitted messages g^a and g^b needs to solve $\mathrm{CDH}_g(g^a, g^b)$ to retrieve the secret g^{ab}, where g is publicly available. Note that this protocol suffers from a man-in-the-middle attack, as the parties A and B do not authenticate themselves.

ElGamal encryption
Again, let G and g be as in Definitions 7.2-7.4. In the ElGamal encryption scheme, user A has secret key $s \in \mathbb{Z}_n^*$ and a corresponding public key $h = g^s$. B can encrypt a message M, which is represented as an element of G, with A's public key h as follows.

1. B picks a random integer $r \in \mathbb{Z}_n^*$ and computes $a = g^r$ and $b = h^r M$.
2. B sends cipher text (a, b) to A.
3. A computes $b/a^s = (h^r M)/g^{rs} = (h^r M)/h^r = M$.

The message M is hidden by the mask h^r, which is uniformly distributed in G^* as r is chosen at random. Intended recipient A can compute the mask $a^s = g^{rs} = h^r$ because of his knowledge of the secret key s. An eavesdropper not having this knowledge, needs to solve $\mathrm{CDH}_g(a, h) = g^{rs} = h^r$ to find the mask and be able to decrypt the message. Therefore, we say that the ElGamal encryption scheme is based on the hardness of the computational Diffie-Hellman problem.

7.2 Discrete Logarithm on Elliptic Curves

Although most DL-based protocols were originally defined on the multiplicative group of a finite field $\mathbb{F}_q^*$, the discrete logarithm problem (and Diffie-Hellman problems) can of course be defined on any group. The protocols can thus be translated in terms of groups that possibly allow better security or more efficient arithmetic. In some groups, however, taking discrete logarithms is easy, for example in the additive group of a finite field. Obviously, such groups are not suitable for cryptographic purposes, as security is based on the hardness of DL. Fortunately, there also exist groups in which solving the discrete logarithm problem is believed to be harder than in $\mathbb{F}_q^*$, where the Index Calculus method provides a subexponential algorithm. The group of points on an elliptic curve is an example of such a group.

In this section we first explore the use of elliptic curves in cryptography. Next we explain that some care should be taken in selecting the curve, as for some classes of curves there exist reductions to groups for which DL is easier. At the end of this section, we focus on the consequences of these observations for the use of elliptic curve cryptosystems in practice.

7.2.1 Use of Elliptic Curves

In 1985, Koblitz [46] and Miller [59] independently proposed to use the group of points on an elliptic curve for the definition of the discrete logarithm problem. The main reason for their choice was that the Index Calculus method does not have a natural analogue in this particular group. A number of researchers have tried to extend the Index Calculus method to make it suitable for elliptic curves, but without succes (see [63]). For example, the Xedni Calculus method by Silverman [77] is shown to be unlikely to yield a practical solution in [40]. Hence, there seems to be no general subexponential time algorithm for solving the discrete logarithm problem. Consequently, elliptic curve cryptosystems can achieve the same level of security as their analogues in $\mathbb{F}_q^*$, using shorter keys. The fact that there exist many different curves of about the same size can be considered an additional advantage of elliptic curve cryptography.

Since its introduction in 1985, there has been a lot of research into elliptic curve cryptography and numerous cryptosystems have been proposed. However, there is some concern among certain cryptographers about the use of elliptic curves. The fact that Index Calculus method cannot be applied does not guarantee that no other algorithm exists that takes less than exponential time. In fact, elliptic curves come equipped with a certain structure, some more than others, which can potentially be exploited to design an efficient algorithm for solving discrete logarithms. Moreover, unlike for instance DL-based systems on the multiplicative group of a finite field, elliptic curve cryptosystems have not yet existed long enough to provide empirical evidence of their security. On the contrary, many curves that were initially proposed for use in cryptosystems, such as supersingular curves and curves of trace one, have been shown to be insecure in the sense that there exists an efficiently computable reduction to a group where solving DL takes subexponential or, even worse, linear time. Here the pairings discussed in Part I turn out to play an important part. We focus on this in the next section.

7.2.2 Reductions to Other Groups

Arbitrary elliptic curves suffer from the problem that it is difficult to compute the number of points on the curve, i.e. the order of the group. Especially in the early years of elliptic curve

cryptography, when Schoof's point counting algorithm [71] had not yet matured, determining the order of the group was a time-consuming task. It was for this reason that the designers of elliptic curve cryptosystems tended to choose curves known to have a particular structure, enabling them to immediately compute the number of points. Examples of such curves are supersingular curves (proposed in [46, 59]) and curves of trace one (proposed in [60]). However, this additional structure also provides cryptanalysts with handles to attack the system. For instance, the MOV- and FR-attack described below reduce DL on supersingular elliptic curves to its analogue on the multiplicative group of some finite field, where a subexponential time algorithm is available due to the Index Calculus method. Even more drastic are the attacks on curves of trace one described at the end of this subsection, which reduce the discrete logarithm problem to its analogue on the additive group of a finite field, which is solvable in polynomial time. As a consequence, supersingular curves could be used for DL-based systems but with parameters lengths large enough to provide security under a subexponential time attack, whereas trace one curves should be avoided altogether. More on this in Subsection 7.2.3.

MOV-reduction
Recall that the discrete logarithm problem on the multiplicative group of a finite field can be solved in probabilistic subexponential time by the Index Calculus method. In 1993, Menezes, Okamoto and Vanstone [55] presented an attack that reduces the discrete logarithm problem on the group of points on a supersingular curve over $\mathbb{F}_q$ to its analogue in the multiplicative group of some extension field $\mathbb{F}^*_{q^{k_w}}$ of degree k_w, which can be solved in subexponential time if $k_w \leq \log(q)$.

Let $E/\mathbb{F}_q$ be an elliptic curve and $P, Q \in E(\mathbb{F}_q)$ m-torsion points on the curve such that $Q = lP$. The MOV algorithm for reducing the discrete logarithm of P and Q uses the Weil pairing $e_m : E[m] \times E[m] \to \mathbb{F}^*_{q^{k_w}}$ as follows (see [45]).

1. Determine the smallest integer k_w such that $E[m] \subseteq E(\mathbb{F}_{q^{k_w}})$.
2. Find $S \in E[m]$ such that $\alpha = e_m(P, S)$ has order m.
3. Compute $\beta = e_m(Q, S)$.
4. Compute $l = \mathrm{DL}_\alpha(\beta)$, the discrete logarithm of β to the base α in $\mathbb{F}^*_{q^{k_w}}$.

The original paper [55] (see also [54]) only deals with the case of supersingular curves. For these curves, the Weil-embedding degree k_w is less than or equal to six, thus sufficiently small to allow a subexponential attack on the elliptic curve DL. Moreover, since the structure of the supersingular curve $E(\mathbb{F}_{q^{k_w}})$ is of the form $\mathbb{Z}_{cm_1} \oplus \mathbb{Z}_{cm_1}$ for certain c and m_1 (see [54, 76]), there exists an efficient method to find a suitable point S in step 2 of the algorithm above. In particular, set $S = [cm_1/m]S'$ for a random point $S' \in E(\mathbb{F}_{q^{k_w}})$, then S satisfies the condition with probability $1 - 1/m$, which is close to one for large m. Verheul's invention of distortion maps ϕ on supersingular curves [83] simplifies the search for a suitable S, as one can simply take $S = \phi(P)$ (see Chapter 6).

Cryptographically interesting non-supersingular curves do not have a distortion map or convenient structure as above. Hence, a generalization of the MOV-attack to non-supersingular curves requires a different method for finding a suitable point S. Such methods are given in [33] for non-supersingular curves having structure $E(\mathbb{F}_{q^{k_w}}) \cong \mathbb{Z}_{c_1 m_1} \oplus \mathbb{Z}_{c_2 m_1}$ with $c_2 m \nmid c_1$, and in [45] for general non-supersingular curves.

FR-reduction
Similar to the MOV-reduction, the FR-reduction [28] proceeds by reducing the discrete logarithm problem on the elliptic curve $E/\mathbb{F}_q$ to the multiplicative group of the extension field $\mathbb{F}^*_{q^k}$ of embedding degree k. This reduction, due to Frey and Rück, employs the Tate pairing instead of the Weil pairing. Let E, P, Q be as above, and let $\langle \cdot, \cdot \rangle : E(\mathbb{F}_{q^k})[m] \times E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k}) \to \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^m$ be the Tate pairing.

1. Determine the smallest integer k such that $m|q^k - 1$.
2. Pick $S \in E(\mathbb{F}_{q^k})$ different from P, Q such that $S \notin mE(\mathbb{F}_{q^k})$.
3. Compute $\alpha = \langle P, S \rangle^{(q^k-1)/m}$ and $\beta = \langle Q, S \rangle^{(q^k-1)/m}$.
4. Compute $l = \mathrm{DL}_\alpha(\beta)$, the discrete logarithm of β to the base α in $\mathbb{F}_{q^k}^*$.

Note that the condition in Step 2 is easily verified by checking whether $[(q^k - 1)/m]S \neq \mathcal{O}$. Since the Tate pairing can be computed faster and more often than the Weil pairing, the FR-reduction is in all aspects superior to the MOV-reduction. Although the MOV- and FR-reduction can be efficiently applied for supersingular curves, Balasubramanian and Koblitz [3] show that, with overwhelming probability, the embedding degree k for randomly generated curves is too large (i.e. $k > \log(q)$).

Reduction on anomalous curves
An elliptic curve $E/\mathbb{F}_q$ is said to be *anomalous* when its trace equals 1, i.e. when $\#E(\mathbb{F}_q) = q$. Miyaji [60] proposes the class of anomalous curves for use in elliptic curve cryptosystems, because of its immunity against the MOV- and FR-reduction. However, there exist even more efficient reductions on anomalous curves, which enable one to solve DL in polynomial time. Because of these reductions, the discrete logarithm problem on anomalous curves is easy and thus unsuitable for cryptography.

One of these reductions was independently found by both Smart [80], and Satoh and Araki [70]. It proceeds by lifting points on the curve $E(\mathbb{F}_q)$ on which the discrete logarithm problem is defined, to points on the curve $E(\mathbb{Q}_p)$ defined over the p-adic numbers $\mathbb{Q}_p$, where p is the characteristic of $\mathbb{F}_q$ (see [76] for details on curves over p-adic numbers). The discrete logarithm problem can then be reduced to its analogue in the group $\mathbb{F}_q^+$, for which it is known to be solvable in polynomial time. This reduction only works for curves satisfying $\#E(\mathbb{F}_q) = |\mathbb{F}_q^+| = q$, i.e. having trace one.

A similar reduction, given by Semaev [73], makes use of a map that embeds the group generated by the base of the logarithm into the additive group $\mathbb{F}_q^+$. Since this map is a homomorphism that can be evaluated in $O(\log(p))$ operations, this reduction leads to a polynomial time method for solving the DL problem on anomalous elliptic curves.

7.2.3 Security Issues

As described in the previous subsection, some precautions should be taken in selecting the elliptic curve. Experience has taught that one should not simply select curves with rich structure to simplify the arithmetic and ease point counting. Namely, this structure can potentially be used to reduce the discrete logarithm problem on that curve to its analogue in a group where taking discrete logarithms takes less than exponential time. An example of such curves is the class of supersingular curves, on which the discrete logarithm problem is reducible to an instance of DL on the multiplicative group of a finite field.

To solve this problem, one could take the parameter lengths of the elliptic curve system long enough to guarantee security not only in the elliptic curve instance, but in the corresponding reduction group as well. This solution might appear awkward, since it takes away the virtue of having shorter keys – initially the main reason for using elliptic curves (see Subsection 7.2.1). In some cases, however, the structure that makes a reduction possible and therefore forces one to use longer keys, can also be used to design gap Diffie-Hellman groups (see Section 7.3).

A more natural solution is to avoid curves that have more structure than average altogether. Remark that we should not just discard curves for which a reduction is known, such as supersingular and anomalous curves, but also curves that are built in a special way, for instance curves built by

complex multiplication (see [2]). Basically, the only way to do this is by selecting the elliptic curve completely at random, and optionally doing some checks for susceptibility to known reductions afterwards. In [3], it is shown that, with overwhelming probability, the FR- (and MOV-) reduction cannot be applied to a randomly selected elliptic curve, so additional checks might not even be necessary. As remarked above, a disadvantage of randomly selected curves is that it is difficult to determine the number of points on these curves.

We next discuss how long (or better: how short) the keys for elliptic curve systems should be, in order to have cryptographic strength similar to traditional DL-based systems over the multiplicative group of a finite field. When the elliptic curve is chosen properly, the complexity of the best known algorithm to solve DL takes time exponential in size $N_1 = \lceil \log_2 q \rceil$. On the other hand, we consider the discrete logarithm problem on the multiplicative group $\mathbb{F}_p^*$ (p prime) having complexity subexponential in the size $N_2 = \lceil \log_2 p \rceil$ due to the Index Calculus method. The field sizes in bits, given by N_1 for the elliptic curve and N_2 for the multiplicative group of the finite field, can be regarded as the key sizes for the cryptosystem. In [8] the following relation between N_1 and N_2 for similar security is derived:

$$N_1 \approx 4.91 \; N_2^{1/3} (\log(N_2 \log 2))^{2/3}.$$

It follows that in order to achieve security equivalent to conventional systems with key lengths of 1024 and 4096 bits, key sizes of 173 and 313 bits respectively suffice in the elliptic curve system. We remark that these values are considerably larger than those proposed by Lenstra and Verheul [47], who accompany their values with well founded arguments. Lenstra and Verheul claim elliptic curve key sizes in a range from 135 to 139 bits are sufficient to provide security equivalent to 1028 bits in a conventional system and elliptic curve key sizes ranging from 206 to 272 bits for a conventional key size of 4047 bits. Further, [47] gives an extensive comparison with other cryptosystems and a prediction of required key sizes in the future.

7.3 Gap Diffie-Hellman Groups

In this section, we give the definition of gap Diffie-Hellman groups, explain how they can be constructed using pairings, and describe how the bilinearity of the pairings gives rise to the definition of the bilinear Diffie-Hellman problem. Lastly, we discuss some security issues that arise from the use of gap Diffie-Hellman groups in cryptography.

7.3.1 Definition

A variant of the computational and decision Diffie-Hellman problems is the gap Diffie-Hellman problem, which is defined as follows. Again, let $(G, *)$ be a cyclic group of prime order n, with generator g.

Definition 7.5 *Let $a, b \in \mathbb{Z}_n^*$. Given g, g^a, g^b, the gap Diffie-Hellman problem (GDH) is to solve the computational Diffie-Hellman problem $CDH_g(g^a, g^b)$, possibly with help of a decision Diffie-Hellman oracle.*

For arbitrary $g_1, g_2, g_3 \in G$, the decision Diffie-Hellman oracle (DDH-oracle) gives the answer to the problem $DDH_g(g_1, g_2, g_3)$. Since the gap Diffie-Hellman problem asks to solve an instance of a computational Diffie-Hellman problem, it is obvious that GDH is no harder than CDH. For most groups, it is not clear whether GDH is strictly easier than CDH since no polynomial time reduction from DDH to CDH is known. See [64] for a formal description of gap problems.

The gap Diffie-Hellman problem naturally occurs in groups where CDH is hard but DDH is easy, i.e. where a gap is present between the hardness of the two problems. Groups that satisfy this property are called *gap Diffie-Hellman groups (GDH groups)*. In these groups, the problems CDH

and GDH are equivalent, as the DDH oracle is readily available. Figure 7.1 depicts the relation between DL, CDH and DDH in a GDH group. Here, the dotted line represents the separation between hard and easy problems; problems on the left are hard, while those on the right are easy. Until the discovery of the constructive use of pairings in 2000, no examples of GDH groups were known. In the next section, we describe how pairings as the Weil and the Tate pairing can be used to create GDH groups.

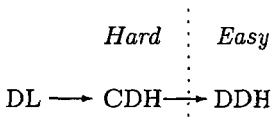

Figure 7.1: GDH group

Though the definition of a gap problem might seem rather artificial at first sight, it does occur naturally in cryptographic applications - in particular in signature schemes. Namely, the security of a signature scheme is based on a hard computational problem, whereas signature verification should involve solving some easy decisional problem. Therefore, in gap Diffie-Hellman groups a basic and comprehensible DL-based signature scheme exists (see [11]). We describe this so-called GDH signature scheme below. Remark that this scheme is similar to Chaum's undeniable signature scheme [19], except for the latter being defined on a group where DDH is hard. The verification procedure in Chaum's scheme consists of an interactive protocol in which some confirmer (often the signer himself) proves the validity of the signature - hence the term 'undeniable'. Okamoto and Pointcheval [64] point out that the confirmer can be seen as DDH-oracle. Therefore, [64] shows that the security of Chaum's undeniable signature scheme, which was an open problem for many years, is based on GDH.

GDH signature scheme
Let G and g be as in Definition 7.5 and let $H : \{0,1\}^* \to G^*$ be a hash function. As in ElGamal encryption, user A has a secret key $s \in \mathbb{Z}_n^*$ and a corresponding public key $h = g^s$. Let $M \in \{0,1\}^*$ be the message to be signed.

1. A computes $m = H(M) \in G^*$ and $\sigma = m^s$. The signature is $\sigma \in G^*$.
2. B computes $m = H(M)$ and accepts the signature if $\mathrm{DDH}_g(h, m, \sigma) = 1$ and disavows otherwise.

Since g is a generator for G, we can write $m = g^y$ for some y. If σ is a valid signature, then $(g, h, m, \sigma) = (g, g^s, g^y, g^{sy})$ is indeed a Diffie-Hellman tuple. Obviously, anyone able solve $\mathrm{CDH}_g(h, m) = \mathrm{CDH}_g(g^s, g^y) = g^{sy} = \sigma$ can forge A's signature, so CDH needs to be a hard problem. On the other hand, the signature verification procedure consists of solving a decisional Diffie-Hellman problem, so DDH should be easy. This is exactly the definition of a GDH group.

7.3.2 Realization with Pairings

We next give the definition of a symmetric pairing on a group G_1 and show that the existence of such a pairing enables one to solve DDH in G_1. Let G_1 be a cyclic group of prime order n. Since we will define G_1 to be a subgroup of the group of points on an elliptic curve, we write the group operation additively and write $G_1 = \langle P \rangle$ for some point P of order n. Denote by G_2 a cyclic group of order n, where the group operation is written multiplicatively. Then a pairing is defined as follows.

Definition 7.6 *A map $e : G_1 \times G_1 \to G_2$ is called a (symmetric) pairing, if it satisfies the following properties:*

1. *Bilinear: $e(P_1 + P_2, P_3) = e(P_1, P_3) \cdot e(P_2, P_3)$ and $e(P_1, P_2 + P_3) = (P_1, P_2) \cdot e(P_1, P_3)$ for all $P_1, P_2, P_3 \in G_1$.*

2. *Non-degenerate in the sense that* $e(P,P) \neq 1$ *(which we call* strongly non-degenerate*).*

3. *Efficiently computable.*

Since the pairing is strongly non-degenerate and G_2 is cyclic, it follows that $e(P,P)$ generates the group G_2. Now the pairing can be used to solve the decision Diffie-Hellman problem in G_1 given by $\text{DDH}_P(aP, bP, cP)$ (i.e. decide whether $abP = cP$). Namely, by the bilinearity of the pairing, we have

$$e(aP, bP) = e(P, bP)^a = e(P, abP) = e(P, cP) \quad \text{if and only if} \quad ab = c.$$

Hence, if a symmetric pairing is available, then DDH in G_1 is easy. Further, since the computational Diffie-Hellman problem is believed to be equivalent to the discrete logarithm problem, which is assumed to be hard on elliptic curves, the group G_1 is a GDH group.

Joux [41] discovers that a pairing as described above can be used to implement an efficient protocol for tripartite Diffie-Hellman key exchange (see Section 9.1). Furthermore, he proposes to use the Weil or the Tate pairing for this purpose. However, for the Weil pairing and in most cases the Tate pairing we have that $e(P,P) = 1$, as described in Chapters 3 and 4. Hence, they do not satisfy the property of strong non-degeneracy. Fortunately, they do satisfy a somewhat weaker notion of non-degeneracy, namely $e(P,Q) \neq 1$ for P and Q linearly independent. Joux remarks that, using either the Weil or the Tate pairing, one can efficiently solve a decision problem slightly different from regular DDH. Briefly, this so-called *decision Diffie-Hellman co-problem* (denoted *Co-DDH*) is given (P, aP, Q, bQ), with P and Q linearly independent, to decide whether $a = b$. Thus, [41] gives the first concrete example of a GDH group, be it with a slightly adapted version of the decision problem. Barreto et al. [7] show how to find a point Q independent from P. The Weil and the Tate pairing that do not satisfy the strong non-degeneracy property are denoted by the term *asymmetric pairings*, as explained in Chapter 6.

Verheul [83] improves the tripartite Diffie-Hellman key exchange protocol for supersingular curves by introducing distortion maps (see Section 9.1). As described in Chapter 6, a distortion map is a linear endomorphism ϕ that maps a point on a supersingular curve to a linearly independent point of the same order. This map can be used to modify the Weil or the Tate pairing such that they do satisfy the strong non-degeneracy property $e(P,P) \neq 1$. Thus these modified Weil and Tate pairings on supersingular curves are symmetric pairings as in Definition 7.6 and can therefore be used to create GDH groups. Another possibility is to exploit the ordinary (i.e. non-supersingular) curves for which the Tate pairing does sometimes yield a symmetric pairing (i.e. $e(P,P) \neq 1$), namely curves of trace 2 (see Section 4.2). Joux and Nguyen [43] describe how to construct elliptic curves for which CDH is provably equivalent to DL but DDH is easy, thus representing GDH groups.

7.3.3 Bilinear Diffie-Hellman Problem

The realization of GDH groups with pairings enables one to implement protocols such as the GDH signature scheme. Moreover, the bilinearity of these pairings offer even a lot more possibilities for building cryptographic applications. Namely, the bilinearity $e(aP, Q) = e(P, Q)^a = e(P, aQ)$ can be used to transport the (possibly secret) value a from one coordinate to the other, without the need of having a 'in the clear'. It may be clear that this feature makes pairings into useful cryptographic building blocks. Consequently, the security of most pairing-based protocols depends on a new type of Diffie-Hellman problem, known as the bilinear Diffie-Hellman problem (see [20]). Let $(G_1, +)$ be an additive cyclic group of order n with generator P, $(G_2, *)$ a multiplicative cyclic group also of order n, and $e : G_1 \times G_1 \to G_2$ a symmetric pairing. For convenience we take n prime, as is done in almost all practical applications. Then $h = e(P,P)$ is a generator of G_2, and the bilinear Diffie-Hellman problem is defined as follows.

Definition 7.7 *Let $a, b, c \in \mathbb{Z}_n^*$. Given P, aP, bP, cP, the bilinear Diffie-Hellman problem (BDH) is to compute $e(P,P)^{abc}$. We use the notation $BDH_P(aP, bP, cP) = e(P,P)^{abc}$.*

It is obvious that BDH is no harder than CDH in G_1; namely having the solution to $Q = \text{CDH}_P(aP, bP)$, one can easily compute $e(Q, cP) = e(P,P)^{abc}$ (similarly for $Q = \text{CDH}_P(aP, cP)$ and $Q = \text{CDH}_P(bP, cP)$). Further, BDH also depends on the hardness of CDH in G_2. For instance, the computational Diffie-Hellman problem on $e(P, aP) = h^a$ and $e(bP, cP) = h^{bc}$ asks to find $\text{CDH}_h(h^a, h^{bc}) = h^{abc} = e(P,P)^{abc}$, which is exactly the solution to the bilinear Diffie-Hellman problem. So the hardness of BDH depends on the hardness of CDH in both G_1 and G_2.

To say something about the converse of above implications (i.e. whether hardness of CDH in G_1 and G_2 also implies hardness of BDH), Cheon and Lee [20] investigate certain properties of the pairing such as weak-invertibility. A pairing $e : G_1 \times G_1 \to G_2$ is said to be *weak-invertible* if for any $h_1 \in G_2$, an inverse image (g_1, g_2) such that $e(g_1, g_2) = h_1$ is efficiently computable. If e satisfies this property, then the bilinear Diffie-Hellman problem is equivalent to the computational Diffie-Hellman problem in both G_1 and G_2. Unfortunately, it is not known whether the modified Weil and Tate pairings are weak-invertible. Therefore, it is uncertain if the hardness of BDH is equivalent to the hardness of CDH in the corresponding groups G_1 and G_2. Nevertheless, this is often assumed to be the case, as no method is known to solve BDH without first solving CDH in either G_1 or G_2. The relations between the several Diffie-Hellman problems when a pairing $e : G_1 \times G_1 \to G_2$ is available, are depicted in Figure 7.2. Here the subscript G_i denotes the group on which the problem is defined. Observe that G_1 is a GDH group.

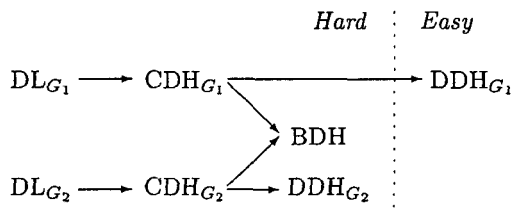

Figure 7.2: GDH group with pairing

7.3.4 Security Issues

Recall from Subsection 7.2.1 that elliptic curves were initially proposed for use in DL-based cryptosystems, because they can achieve security similar to DL-based systems on other groups, while using shorter keys. It turned out, however, that efficient reductions to less secure groups exist on certain curves, as described in Subsection 7.2.2. Hence, in order to avoid these weak cases, elliptic curves should be selected with great care. This can be done by choosing the curves completely at random and possibly perform some optional checks (see Subsection 7.2.3).

On the other hand, curves to which the reductions apply can be used for different purposes. As described above, the bilinear maps that make the MOV- and FR-reduction possible can be used to create GDH groups. Moreover, the bilinearity of these maps can be exploited to build new cryptosystems with nice properties. We have to keep in mind, though, that we cannot use shorter keys in this situation, since we are working with curves vulnerable to the MOV- and FR-reduction. More in particular, to ensure the hardness of the bilinear Diffie-Hellman problem, we have to make sure that CDH is hard in both G_1 and G_2. Chapters 5 and 6 explain that G_1 is a subgroup of the curve $E(\mathbb{F}_q)$ over a finite field $\mathbb{F}_q$, and G_2 is a subgroup of the multiplicative group of an extension field $\mathbb{F}_{q^k}$ of embedding degree k. The desired value for k is a trade-off between security and efficiency; a larger k enhances the hardness of CDH in G_2, whereas a smaller k renders the computation of the pairing more efficient.

A concern among some cryptographers regards the hardness of the bilinear Diffie-Hellman problem, on which the security of pairing-based cryptography relies. Though BDH is often assumed to be equivalent to CDH in the groups involved, there is at present no proof of this assumption. As with DL on elliptic curves, there might be certain cases for which BDH is less secure than in general. Moreover, the extra structure making curves suitable for pairing-based cryptography potentially provides cryptanalysts with tools to attack the system. Since the constructive use of pairings is a very recent discovery, there has not been enough time for a thorough scrutiny, which takes many years. Only the passage of time and the absence of cryptanalytic breakthroughs will take away the scepsis present among some cryptographers. Further, if particular cases are found that prove to be weaker than general, then the understanding of these cases and the ability to avoid them will improve the trust in pairing-based cryptography.

Chapter 8

Identity-Based Cryptography

Undoubtedly the most striking application of pairings is the realization of identity-based public-key cryptography (ID-PKC), the concept of which was already proposed by Shamir [74] in 1984. Shamir's aim was to ease the bother with the certificates that are essential in conventional public-key settings (public-key infrastructure) for guaranteeing the authenticity of the public keys. After explaining the concept of ID-PKC in Section 8.1, we compare it with the traditional, certificate-based public-key cryptography in Section 8.2.

Although the idea of ID-PKC dates back to 1984, its full realization remained an open problem for many years. In his original article [74], Shamir did come up with a concrete example of an identity-based signature scheme. Identity-based encryption, however, turned out to be a more difficult task. In 2001, Boneh and Franklin proposed the first fully functional implementation of an identity-based encryption scheme, using the bilinearity of pairings. We focus on this scheme and other identity-based applications in Section 8.3.

8.1 Definition

In this section, we discuss public-key cryptography in general, give the definition of identity-based cryptography, and define some notions of security for identity-based cryptography.

8.1.1 Public-Key Cryptography

In public-key cryptography each user U has a key pair (p_U, s_U) consisting of a public key and a secret key. This key pair is either generated by the user himself (or herself in the case of a feminine user), or by some central authority. In the latter case, a secure channel is needed to transport the key pair to the user. Key pair generation is done by choosing a secret key s_U at random and applying some one-way function to s_U to obtain p_U. Since the function is one-way, other parties cannot retrieve the secret key s_U from the public key p_U. For example, the secret key s in the ElGamal encryption scheme is chosen at random and the corresponding public key is set to be $h = g^s$, where g is some publicly known generator for the group. Computing s from h and g boils down to calculating a discrete logarithm.

Other parties use U's public key to encrypt messages intended for U or to verify his signatures. This can be done by anyone, since the key p_U is public knowledge. On the other hand, to decrypt a received cipher text or sign a message, U uses his secret key s_U. The secrecy of the cipher text and the authenticity of the signature then follow from the fact that user U is the only one knowing s_U.

The main concern in a public-key setting is the authenticity of the public key. Clearly, if a malevolent person can convince other participants that U's public key is some key of his choice

instead of the correct p_U, he can decrypt messages intended for U only and forge signatures under U's name. Therefore, it is of great importance that participants in a PKC system can verify the authenticity of other users' public keys.

The conventional solution to the authentication problem is the use of a public-key infrastructure (PKI). A PKI often works with a party trusted by all users, called *Certification Authority* (*CA*), which can guarantee the correctness of the public keys as follows. User U identifies himself to the CA (following some authentication procedure) and presents his public key p_U. Next, U provides a 'proof of possession' of the corresponding secret key; this is usually done by signing the certification request with the secret key. If the CA is convinced that user U indeed owns the secret key corresponding to the public key p_U, he publishes a certificate containing U's identity, U's public key and possibly some other information, and signs all this information with his secret key. Participants who want to communicate securely with U then need to look up the certificate issued by the CA. A valid signature from the CA will convince them of the authenticity of the public key p_U.

In addition to a CA, PKI often makes use of several other authorities, each with a different task. For instance, there can exist a Registration Authority (RA) that takes care of the authentication procedure, a Validation Authority (VA) that guarantees the validity of certificates, a Time Stamping Authority (TSA) that time stamps electronic documents, etcetera. For simplicity, we lump together all these authorities and assume that the body that we call Certification Authority takes care of all these tasks.

8.1.2 Identity-Based Cryptography

In 1984, Shamir [74] invented the concept of *identity-based cryptography*, which addresses the authenticity problem of public keys in a different way (see also [56]). His idea was to avoid the need for authentication altogether, by making sure that the actual value of a user's public key is inherently linked to his identity. More precisely, the public key of a user is derived directly from publicly available information that uniquely and undeniably identifies that user. This information is denoted by a user's *(digital) identity*. Depending on the application, the identity can range from (combinations of) the user's name, social security number, phone number, email address, and possibly other personal information. In this setting, a user's public key is readily available to anyone who knows his identity, so there is no need to look up the key in some database. Moreover, the fact that there is no doubt about the authenticity of the public key takes away the need for certificates as in a PKI setting. We should note, however, that the realization of the link between users and their digital identities is far from trivial, as discussed in Remark 8.1.

Recall that in conventional public-key systems, key pairs are generated by randomly choosing a secret key and applying some one-way function to compute the public key. In ID-based cryptography, key pairs are obtained differently. First of all, the public key is uniquely determined by the identity of the user, instead of computed from the secret key. The secret key thus needs to be derived from the public key, instead of the other way around. From this follows that key generation cannot be performed by the users themselves. Namely, if a user knew how to generate the secret key corresponding to his particular public key, he could also derive the secret keys of others. Therefore, we need a third party, called *Key Generation Center* (*KGC*). After some authentication procedure (similar to authentication to the CA in a PKI setting), the KGC generates the secret key of the user. The KGC is able to do this, as he has the privilege of knowing some secret information, called the *master key*. The publicly available information that corresponds to the master key is denoted by the *system parameters*. A user's secret key is then computed as some one-way function of the public key and the master key. Observe that this setting always requires a secure channel to transport the secret key from the KGC to the correct user, in contrast to PKI where users possibly generate their own key pairs. Shamir [74] proposes to do this by storing the secret keys on smart cards.

Along with the concept of ID-based cryptography, Shamir [74] outlines an implementation of an ID-based signature scheme. Similar to RSA, this signature scheme is based on the hardness of factoring large numbers and root extracting modulo a large number. However, Shamir observes that the RSA encryption scheme cannot be converted to an identity-based encryption scheme. In fact, the design of a fully satisfactory encryption scheme remained an open problem until 2001, when Boneh and Franklin [9] discovered that the bilinearity of pairings can be used to turn the ElGamal encryption scheme into an identity-based variant. We will discuss this in Section 8.3. At the same time of [9], another identity-based encryption scheme appeared due to Cocks [21]. Although this non pairing-based scheme is quite elegant, it suffers from a large bandwidth requirement. We will not discuss it in further detail.

Remark 8.1 Identity-based cryptography is founded on the implicit assumption that there exists an unambiguous link between a physical user and the unique piece of information that is denoted by 'digital identity'. In practice, the realization of such a link is a more complex task than one would initially expect. Within a company, for instance, there can be several employees with the same name. To distinguish between them, there often needs to be a trusted party (which we can call a *Registration Authority* or *RA*) that issues the 'digital identities' to the users. For example, if the users' identities are derived from their email addresses, then the department that issues these addresses functions as the RA. It may be clear that with this function comes a great deal of responsibility. For simplicity, we often consider the RA as part of the KGC. Note that a link between users and their digital identities is also needed in PKI-based settings, where a user's public key and identity are bound together in a certificate. More on this in Section 8.2.

8.1.3 Security of Identity-Based Cryptosystems

In public-key encryption, there are several concepts of security. The ones we will consider here are semantic security and chosen ciphertext security. The latter is sufficiently strong for most applications and is therefore an acceptable notion of security for conventional (i.e. non identity-based) public-key encryption schemes. These notions of semantic security and chosen ciphertext security are described below in a rather informal manner. For a more formal description, see [9, 13].

Semantic security: Intuitively, a standard (i.e. non identity-based) public-key encryption scheme is semantically secure, if an adversary who is given a ciphertext encrypted with some known public key cannot learn anything about the corresponding plaintext. More in particular, consider the following game between a challenger and an adversary. Suppose the challenger gives an adversary a random public key. The adversary then comes up with two messages, one of which is randomly chosen and encrypted by the challenger to form the challenge ciphertext. If the adversary correctly guesses which one, he wins the game. The system is said to be semantically secure if no polynomial time adversary can win the game with probability non-negligibly more than a half. (See [9] for the definition of negligible.) We adopt the notation from [9] and say the system is IND-CPA secure.

Chosen ciphertext security: For the definition of chosen ciphertext security, a similar game is played. Only this time, the adversary is given the additional opportunity to issue a number of decryption queries to the challenger. That is, the adversary presents a ciphertext of his choice and the challenger responds with the decryption of that ciphertext under the secret key corresponding to the public key given to the adversary. The only restriction is that the adversary cannot issue a decryption query for the challenge ciphertext outputted by the challenger. A standard public-key encryption system is said to be chosen ciphertext secure (denoted IND-CCA) if no polynomial time adversary can win this game with probability non-negligibly more than a half. Chosen ciphertext security implies that given a ciphertext and a public key, an adversary cannot learn anything about the plaintext, even if he has access to the decryptions of a number of ciphertexts of his choice.

These notions of security have to be strengthened to meet the needs of identity-based encryption, where any value can serve as public key. In particular, public keys are not random. Hence, the adversary may already have access to a number of secret keys corresponding to public keys of his choice. Moreover, an identity-based scheme has to be secure against an attack on an encryption under a public key of the adversary's choice rather than a random public key. Below, we describe how the notions of semantic security and chosen ciphertext security are enhanced, such that they fit the new situation. We need these notions of security in Subsections 8.3.1 and 8.3.2, where we discuss the security of the ID-based encryption scheme by Boneh and Franklin [9].

Semantic security for identity-based schemes: To define semantic security for identity-based encryption schemes, we consider an adversary and a challenger who play a game similar to the IND-CPA game. In this setting, however, the adversary can choose the public key he wishes to be challenged on, instead of getting a random public key from the challenger. Furthermore, the adversary is given the additional opportunity to issue a number of key extraction queries to the challenger. That is, the adversary presents a public key of his choice and the challenger responds with the corresponding secret key. An obvious constraint is that the adversary cannot ask to be challenged on a public key that appeared in a previous key extraction query. A second limitation is that, after choosing a public key to be challenged on, the adversary cannot issue a key extraction query on that particular key. An identity-based scheme is said to be semantically secure (denoted IND-ID-CPA) if no polynomial time adversary can win the game with probability non-negligibly more than a half.

Chosen ciphertext security for identity-based schemes: For chosen ciphertext security in an identity-based encryption scheme, the adversary and challenger again play a game like in the IND-ID-CPA situation. Again, the adversary can choose the public key he wants to be challenged on and issue private key extraction queries. Moreover, the adversary is also given the opportunity to issue decryption queries as in the IND-CCA game. In this situation, a decryption query consists of a ciphertext together with a public key. The challenger responds with the decryption of that ciphertext under the corresponding secret key. Obviously, after choosing a public key and receiving a challenge ciphertext, the adversary is not allowed to issue a decryption query consisting of that particular combination. An identity-based scheme is said to be chosen ciphertext secure (denoted IND-ID-CCA) if no polynomial time adversary can win the game with probability non-negligibly more than a half.

Similar to the security of encryption schemes, there are several notions of security for signature schemes in public-key cryptography. The most general notion for non identity-based signature schemes is security against existential forgery on adaptively chosen message attacks. We give the definition of this security concept and describe how it is strengthened to provide security in an identity-based setting (see [16, 13]). In Subsection 8.3.4, the ID-based signature scheme by Cha and Cheon [16] is shown to be secure against adaptively chosen message attacks in the identity-based setting.

Chosen message security: Consider the following game where the challenger gives the adversary a random public key. The adversary is allowed to issue a number of sign queries to the challenger. Here a sign query consists of a message of the adversary's choice, to which the challenger responds with a signature under the secret key corresponding to the given public key. The adversary wins the game if he can output a valid message-signature pair under that secret key, where the message has not previously appeared in a sign query. A non identity-based signature scheme is said to be secure against existential forgery on adaptively chosen message attacks (denoted EUF-CMA) if no polynomial time adversary can win the game with non-negligible probability.

Chosen message security for identity-based schemes: The challenger and adversary play a game similar to the non identity-based situation, only this time the adversary can also issue

a number of private key extraction queries for identities ID of his choice. Sign queries are now of the form (ID, m), where the challenger responds with a signature for the message m under the secret key corresponding to the public key derived from identity ID. The adversary eventually outputs (ID, m, σ), such that ID did not occur as key extraction query or (ID, m) as sign query. The adversary wins the game if σ is a valid signature for the message m under the secret key corresponding to the public key derived from ID. An identity-based signature scheme is said to be secure against existential forgery on adaptively chosen message attacks (denoted EUF-ID-CMA) if no polynomial time adversary can win the game with non-negligible probability.

8.2 Comparison to PKI

Recall that both ID-based and PKI-based cryptosystems are asymmetric. Hence, the protocols for encryption, decryption, signing and signature verification have similar functionality in both systems. The main difference, however, is key management. Identity-based cryptography was initially proposed to avoid the need for certificates for public key authentication. In fact, an ideal ID-based system would satisfy the following properties (see [56]):

- Users only need to know the identity of the user they want to communicate with.
- There is no need for keeping public directories such as files with public keys or certificates.
- The services of the KGC are needed only during the system set-up phase.

However, in practice such an ideal scheme seems to be infeasible. Depending on the application, there are several practical issues that stand in the way of the realization of an ideal ID-based system. We compare ID-based cryptography to the traditional PKI setting, considering the following practical aspects: authenticity of system parameters, registration at the authority, key escrow, key revocation and key rollover, key distribution, master key security, scalability, commercial maturity, and additional possibilities (see [74, 56, 65, 9]).

Authenticity of system parameters
Suppose an attacker in an ID-based system generates his own master key and corresponding system parameters, and fools users into believing that these forged system parameters are correct. Then for any public key, he can derive the corresponding secret key under his master key. Hence, he can decrypt any message encrypted under his forged parameters. Further, he can create signatures under any name, which will be accepted by users who believe his parameters. The attacker might even impersonate the KGC and issue secret keys to users on request. Note that previously existing secret keys of users, generated by the KGC, are not compromised. In conclusion, it is of great importance that the KGC somehow guarantees the authenticity of the correct system parameters.

A similar problem occurs in a PKI situation, where the users need to be sure of the authenticity of the public key of the CA. Namely, if an attacker can make users believe that some public key of his choice is the public key of the CA, then he can create certificates containing forged public keys for which he owns the secret key. Consequently, the attacker can read the messages encrypted under the forged public keys and create signatures that appear to be valid under those keys. The only difference with the ID-based setting is that the attacker cannot impersonate the CA, as users would notice that their requested certificate contains an incorrect public key.

Registration at the authority
In both systems, a user who wants to participate needs to register at a Registration Authority (RA), which we often consider part of the CA, respectively KGC. After some authentication procedure, the RA issues a unique digital identity to the user, for instance in the form of an email address. In a PKI-based system, a user can now present his digital identity and public key to the CA, along with a proof of possession of the corresponding secret key. The CA then issues a

certificate that binds together the digital identity and the public key. Similarly, in an ID-based setting, the user presents his digital identity to the KGC. The RA is responsible for the uniqueness of the digital identity and the link between the identity and the physical user. The KGC then computes the secret key corresponding to the public key derived from the digital identity.

Identity-based systems have the additional disadvantage that the secret key needs to be transported from the KGC to the user. Thus, a secure channel that guarantees both confidentiality and authenticity is required. Therefore, ID-based systems seem to work best in applications where it is easy to achieve a secure channel (for instance relatively small systems), or where users request secret keys not very often (for instance only at the set-up of the system).

Key escrow
Identity-based cryptosystems have inherent key escrow. Namely, since the KGC owns the master key, he can generate any private key at any moment. Depending on the application, key escrow is not necessarily a bad thing. For instance, within a company, the director who functions as the KGC might want to be able to keep track of the communications between the employees. Moreover, key escrow enables recovery of lost keys. For signatures schemes, however, it is often highly undesirable to have key escrow, as it prevents non-repudiation. Note that this escrow capability is also present in PKI settings where key pairs are generated by some central authority, in case this authority stores the issued keys. On the other hand, if users generate the key pairs themselves or the central authority does not store the keys, there is no way that keys can be retrieved and there is no key escrow. This in contrast to an ID-based setting, where the KGC always has the ability to regenerate keys.

The ID-based encryption scheme by Boneh and Franklin [9] provides a way to hamper key escrow by introducing multiple KGCs, say $n \geq 2$ (see Subsection 8.3.3). Briefly, each of these n KGCs has its own master key. A user presents his public key to each KGC and gets from each of them a partial secret key in return. Then the correct secret key is obtained by combining the n partial secret keys. This way, the master key, and thus the ability to retrieve any user's secret key, is distributed among n KGCs. A secret key can only be recovered when all n KGCs collude. Remark that in practice, setting up a system with multiple KGCs can be a complex task. Moreover, the overall workload gets n times as large, as each separate KGC needs to perform a same amount of work as required in a system with a single KGC. Hence, an ID-based system works best in an application where key escrow is not an objection, or where the group of users is small enough to allow multiple KGCs.

Key revocation and key rollover
When a certificate is revoked in a PKI-based scheme, other users are notified by means of a public *Certificate Revocation List* or *CRL*. This occurs for instance when a user leaves the user group or a secret key is compromised. In the latter case, or when a key pair needs to be replaced after expiration of the certificate (key rollover), a user can simply generate a new key pair and obtain a certificate for it. Revocation lists, enabling users to check the validity of public keys, can be used in an ID-based setting as well. However, since a user's public key is derived from his identity, he cannot simply obtain a new key pair after revocation as in a PKI-based scheme. Namely, it can be very inconvenient or simply impossible to change identity every time a new key pair is needed.

A partial solution to this problem could be to derive the public key not solely from the identity, but to concatenate the identity with some other general information. For instance, if the current year is added to the identity, users can use their secret key during that year only. Hence, secret keys expire annually and each user has to request a new key every year. Unlike PKI, users do not have to obtain new certificates from other users, as the public key is still uniquely determined in a straightforward manner (since the current year is common knowledge). But what if a user's secret key is compromised halfway during the year? Then that user has to wait until the end of the year before he can get a new key. This situation can be improved by making the system more

granular; for instance by concatenating the identity with the current date instead of the current year. The big disadvantage, though, is that each user then has to obtain a new secret key from the KGC every day. This results in a large increase in communications and a computational overhead for the KGC, which has to calculate all private keys. Hence, the length of the validity period is a trade-off between granularity on one hand and efficiency on the other.

Another (unsatisfactory) solution would be to concatenate the identity with some particular key information, instead of general information such as current date or year. For example, this information could be as simple as an index number, or more particularly, the date the key was issued. This way, when a key is compromised, the user can request a new key with an increased index or an updated issuance date. This would destroy, however, the unique property of identity-based cryptography that public keys can be derived from the identity (in combination with publicly known information) solely. Namely, other users would have to look up the index number or date of issuance of the public key in some directory. But the problem of the authentication of that value would put us back in the situation where we started when we described public-key cryptography.

Key distribution
As described in Subsection 8.1.2, the great simplification of key distribution (from the users' point of view) was the main reason to introduce identity-based cryptography. Namely, all public keys can be derived from the identity of the users. So obtaining someone's public key, for encryption or signature verification, becomes a simple and transparent procedure. This in contrast to PKI, where one has to look up the corresponding certificate, verify the CA's signature and check the expiration date of the certificate. Moreover, one has to check the validity of the certificate, which can be done by either using an up-to-date CRL or performing an online check at a Verification Authority.

Master key security
The KGC in an identity-based scheme forms a single point of weakness. An attacker who is able to retrieve the KGC's master key can derive all secret keys, and is thus able to read all messages and forge signatures under everyone's name. Therefore, it is very important for a KGC to keep his master key secret. To prevent the master key from being stored in one place, it can be distributed among several KGCs, as with the prevention of key escrow. To a smaller extent, the CA in a PKI-based setting is a single point of weakness as well. If the CA's secret key is compromised, an attacker can create certificates in the CA's name for new public keys of his own choice, thereby fooling other users into believing in the authenticity of those public keys. However, knowledge of the CA's secret key does not enable an attacker to retrieve previously existing secret keys. So the attacker cannot read messages encrypted under previously existing public keys, or forge signature under corresponding secret keys.

Scalability
In an ID-based scheme, the KGC has to perform an authenticity check and compute a secret key for each key request. Especially when the number of key requests is large, for instance in a system where keys expire frequently (see *key revocation*), this may result in a heavy workload for the KGC. A solution could be to spread the work across several KGCs, possibly at different levels of authority (see [32]). Observe that these multiple KGCs cannot also be used to prevent key escrow or increase security by distributing the master key. In this case, namely, each one of the KGCs has to do a same amount of work as in a setting with a single KGC. Again, ID-based systems seem to work best in a small-scale deployment.

Commercial maturity
Being introduced as recently as 2001, the realization of ID-based cryptography by pairings is a very new technology. Although the potentials of ID-based cryptography are promising, there turn out to be several drawbacks when it comes to practical applications. Moreover, at present the technology mainly exists in theory and is hardly tested in practice. This in contrast to PKI-based

cryptography, which has been an established and widespread technology for many years already. Moreover, unlike ID-based cryptography, PKI has been standardized to a large extent, which is an important practical aspect. Considering the conservative attitude of companies and organizations that deploy public-key cryptography, it is hard to imagine that ID-based cryptography will replace PKI at short notice.

Additional possibilities
The fact that any value can be a public key offers some additional possibilities. For instance, we can concatenate the identity with other information to accomplish certain nice properties that do not exist in a PKI setting. We have touched on this when dealing with key revocation, where adding the current date or year to the identities results in key pairs that expire after a certain period. In fact, in this setting, users can send encrypted messages into the future. This is done by using the public key derived from the concatenation of the identity with a future date instead of the current date. Now the recipient cannot decrypt the message until the specified date, when the KGC issues the corresponding secret key. Here we assume that the KGC is honest and does not issue keys before the specified date.

We can extend this idea by including even more information in the public key, such as some confidentiality specification. For instance, one can encrypt a message using as public key the identity concatenated with some date and the specification "secret". Then the recipient can only obtain the corresponding secret key and thus decrypt the message, if he has secret clearance on the specified date. Note that the KGC, which issues the secret keys, has the responsibility of deciding whether that particular user should have secret clearance or not.

Another application lies a bit further from the original idea of identity-based cryptography. Again, we use the fact that any value can serve as a public key, but this time we use information different from identities of the users. Therefore, this type of cryptography is called *identifier*-based cryptography (see also Subsection 8.3.5) and we denote public keys by *identifier keys.* An example can be found in [9], where identifier-based cryptography is used to delegate decryption capabilities. Here identifier-based cryptography is combined with PKI as follows. User U plays the role of KGC and generates his own system parameters and master key for an ID-based encryption scheme. U keeps his master key secret and publishes the system parameters. These parameters function as U's public key in a PKI encryption scheme (which we will call U's public PKI-key). As in a regular PKI setting, the authenticity of U's public PKI-key (the system parameters) is guaranteed by a certificate from a CA. Now other users can encrypt messages for user U as in an ID-based scheme with system parameters that are given by U's public PKI-key, using an arbitrary value as identifier key. Since U (and in fact only U) owns the master key for the ID-based system, he can derive the secret key for any identifier key and is thus able to read the messages encrypted under his system parameters.

Applications of the scheme described above lie in the delegation of decryption keys. For instance, U can instruct other users to use the current date as identifier key. When U goes on a trip for a number of days, he installs on his laptop only the secret keys for those particular days and not the master key. If the laptop is stolen, the keys for the duration of the trip are compromised, but the master key remains secret. For another application, U can tell other users to use the subject line of an email as identifier key for the encryption of that email. Suppose U has several employees, each responsible for a different task. U then provides each employee with the secret key corresponding to the identifier key that specifies a subject within that employee's responsibilities. This way, U achieves that his employees can read only mail that concerns them, while other users only need a single public PKI-key to encrypt messages for U. See Subsection 8.3.5 for more examples of identifier-based cryptography.

8.3 Realization with Pairings

Now we have dealt with the concept of identity-based cryptography, we are ready to describe its realization through pairings. We start with the identity-based encryption scheme due to Boneh and Franklin [9], denoted *IBE*. For comprehensibility, they give two versions of IBE, *BasicIdent* and *FullIdent*. We adopt this approach and show in Subsection 8.3.1 that BasicIdent is closely related to the ElGamal encryption scheme and that it is IND-ID-CPA secure. In Subsection 8.3.2, we discuss how BasicIdent can be transformed to FullIdent, which is shown to provide IND-ID-CCA security. The BasicIdent and FullIdent schemes are described in general terms, without using properties of the particular pairing. In Subsection 8.3.3, we deal with the concrete pairing as proposed by Boneh and Franklin [9] and make some observations on IBE. Identity-based signature schemes and other identity-based applications are discussed in Subsections 8.3.4 and 8.3.5, respectively.

8.3.1 The BasicIdent IBE Scheme

The IBE scheme of Boneh and Franklin [9] consists of four algorithms: *Setup*, *Extract*, *Encrypt*, and *Decrypt*. *Setup* is run by the KGC to generate the master key and the system parameters. This is done on input of a security parameter k, which specifies the bit length of the groups. The algorithm *Extract* is carried out by the KGC to generate a secret key corresponding to the identity of a user. As with regular public-key cryptography, the algorithm *Encrypt* takes as input a message and a public key to produce a ciphertext. Similarly, *Decrypt* enables the owner of the corresponding secret key to decrypt the ciphertext. The version BasicIdent of IBE is as follows.

Setup: On input of a security parameter k, the algorithm works as follows:

1. Generate a random k-bit prime q, two groups $(G_1, +), (G_2, *)$ of order q, and a symmetric pairing $e : G_1 \times G_1 \to G_2$. Choose an arbitrary generator $P \in G_1$.
2. Pick a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$.
3. Choose cryptographic hash functions $H_1 : \{0,1\}^* \to G_1^*$ and $H_2 : G_2 \to \{0,1\}^n$ for some n.

The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = G_1^* \times \{0,1\}^n$. The system parameters are $params = \langle q, G_1, G_2, e, n, P, P_{pub}, H_1, H_2 \rangle$. The master key is $s \in \mathbb{Z}_q^*$.

Extract: For a given string $ID \in \{0,1\}^*$ the algorithm works as follows:

1. Compute $Q_{ID} = H_1(ID) \in G_1^*$.
2. Set the secret key d_{ID} to be $d_{ID} = sQ_{ID}$, where s is the master key.

Encrypt: To encrypt a message $M \in \mathcal{M}$ under the public key ID, the algorithm works as follows:

1. Compute $Q_{ID} = H_1(ID) \in G_1^*$.
2. Choose a random $r \in \mathbb{Z}_q^*$.
3. Set the ciphertext to be

$$C = \langle rP, M \oplus H_2(g_{ID}^r) \rangle, \quad \text{where} \quad g_{ID} = e(Q_{ID}, P_{pub}) \in G_2^*.$$

Decrypt: Let $C = \langle U, V \rangle \in \mathcal{C}$ be a ciphertext encrypted using the public key ID. To decrypt C using the private key $d_{ID} \in G_1^*$ compute:

$$V \oplus H_2(e(d_{ID}, U)) = M.$$

Observe that at the encryption phase, the message M is bitwise exclusive-ored with the mask $H_2(g_{ID}^r)$ to form V. At the decryption phase, this V is bitwise exclusive-ored with $H_2(e(d_{ID}, U))$. Then by the identity

$$e(d_{ID}, U) = e(sQ_{ID}, rP) = e(Q_{ID}, P)^{sr} = e(Q_{ID}, P_{pub})^r = g_{ID}^r$$

we find that the masks are equal. So when the message M is encrypted with public key Q_{ID}, then decryption with the corresponding secret key $d_{ID} = sQ_{ID}$ does indeed yield the original message M.

Remark 8.2 The choice for the value of n does not affect the security of the scheme. Hence, n can be chosen such that it is easy to construct the hash function H_2, or such that the messages and ciphertexts have convenient lengths.

We compare the scheme BasicIdent with the ElGamal encryption scheme of Subsection 7.1.3. Here user A has a public key $h = g^s$ corresponding to his secret key s for some generator g. The message M is concealed by the mask h^r for some randomly chosen r. A can compute the mask from g^r (which is included in the ciphertext) and the knowledge of his secret key s. Note that the group operation is written multiplicatively. Now consider user A in the BasicIdent scheme, with public key $Q_A = H_1(A)$ and secret key $d_A = sQ_A$. Set $g = e(Q_A, P)$ and $h = g_A = e(Q_A, P_{pub})$, then indeed g is a generator of G_2, h can be regarded as A's public key, and $g^s = h$ as in the ElGamal encryption scheme. Similar to ElGamal, a message M is concealed by the the mask given by (the hash of) h^r for some random r. Note that this time the group operation is written additively. A (and besides A only the KGC) can compute the mask

$$e(d_A, rP) = e(Q_A, P)^{sr} = g^{sr} = h^r \tag{8.1}$$

from his knowledge of his secret key d_A and the value rP (which is included in the ciphertext).

Observe that the scheme BasicIdent bears much resemblance with the ElGamal encryption scheme. Both schemes hide the message M behind the mask h^r, where h is A's public key and r a random value. The value of r is then communicated to A by including in the ciphertext rP (with additive notation), respectively g^r (with multiplicative notation), where P and g are publicly known generators. Although A cannot get his hands on r in the clear, he can use rP, respectively g^r, in combination with his secret key to compute the mask. Once A knows the mask used for encryption, he can simply remove it by inverting the operation.

The main difference of BasicIdent with ElGamal is that A does not have the secret s in the clear, since s is the master key known only to the KGC. Instead, A's secret key $d_A = sQ_A$ represents s in disguise. In fact, the unique bilinearity property of the pairing enables A to compute the mask, without the need of knowing s explicitly. Namely, when we write out expression (8.1), we get

$$e(d_A, rP) = e(sQ_A, P)^r = e(Q_A, P)^{sr} = e(Q_A, sP)^r = e(Q_A, P_{pub})^r = h^r.$$

As remarked in Subsection 7.3.3, the bilinearity is used to transport the secret value s from the first coordinate to the second, without knowing the actual value of s. This property is essential in this identity-based scheme, as it allows the master key s to be kept secret, while secret keys $d_{ID} = sQ_{ID}$ are sufficient for the users to decrypt their messages.

The security of BasicIdent is considered in the random oracle model, i.e. the security analysis will regard the hash functions H_1 and H_2 as random oracles. The security of BasicIdent is based on the bilinear Diffie-Hellman problem. Namely, [9] proves that if BDH is hard, then the scheme BasicIdent is a semantically secure identity-based encryption scheme (IND-ID-CPA). This is done by showing that an adversary who wins the IND-ID-CPA game with probability non-negligibly more than a half can be used to construct an algorithm that solves BDH with non-negligible probability. This is made precise in the following theorem.

Theorem 8.3 (Boneh and Franklin [9], Theorem 4.1) *Suppose the hash functions H_1 and H_2 are random oracles. Suppose an adversary $\mathcal{A}$ wins the IND-CA-CPA game in the BasicIdent scheme with probability $1/2+\epsilon$ for non-negligible ϵ. Suppose $\mathcal{A}$ makes at most $q_E > 0$ key extraction queries and $q_{H_2} > 0$ hash queries to H_2. Then there is an algorithm $\mathcal{B}$ that solves BDH with probability at least $\frac{2\epsilon}{e(1+q_E)\cdot q_{H_2}}$. Here $e \approx 2.71$ is the base of the natural logarithm. The running time of $\mathcal{B}$ is $O(time(\mathcal{A}))$.*

8.3.2 The FullIdent IBE Scheme

For many applications of identity-based encryption, IND-ID-CPA security is not sufficient. For instance, when it might be possible for an adversary to get his hands on the decryption of some known ciphertexts, we need a stronger notion of security such as IND-ID-CCA. Therefore, Boneh and Franklin [9] convert the BasicIdent scheme into the FullIdent scheme, which they prove to be IND-ID-CCA secure. They use the following result by Fujisaki and Okamoto [29], which shows how to create an IND-CCA secure hybrid scheme from an originally IND-CPA secure scheme.

Theorem 8.4 (Fujisaki and Okamoto [29], Theorem 14) *Let $\mathcal{E}$ be an IND-CPA secure probabilistic public-key encryption scheme and let $\mathcal{E}_{pk}(M;r)$ denote the encryption of M using the random bits r under the public key pk. Then the hybrid scheme $\mathcal{E}^{hy}$ defined as*

$$\mathcal{E}^{hy}_{pk}(M) = \langle \mathcal{E}_{pk}(\sigma; H(\sigma, M)),\ G(\sigma) \oplus M \rangle$$

is IND-CCA secure. Here σ is a random string chosen from an appropriate domain and G and H are appropriate hash functions.

Boneh and Franklin apply this transformation to the BasicIdent scheme to get the following scheme, called FullIdent.

Setup: As in the BasicIdent scheme. In addition, we pick a hash function $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and a hash function $H_4 : \{0,1\}^n \to \{0,1\}^n$.

The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = G_1^* \times \{0,1\}^n \times \{0,1\}^n$. The system parameters are $params = \langle q, G_1, G_2, e, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$. The master key is $s \in \mathbb{Z}_q^*$.

Extract: As in the BasicIdent scheme.

Encrypt: To encrypt the message $M \in \mathcal{M}$ under the the public key ID, the algorithm works as follows:

1. Compute $Q_{ID} = H_1(ID) \in G_1^*$.
2. Choose a random $\sigma \in \{0,1\}^n$.
3. Set $r = H_3(\sigma, M)$ and set the ciphertext to be

$$C = \langle rP, \sigma \oplus H_2(g_{ID}^r), M \oplus H_4(\sigma) \rangle, \quad \text{where} \quad g_{ID} = e(Q_{ID}, P_{pub}) \in G_2.$$

Decrypt: Let $C = \langle U, V, W \rangle$ be a ciphertext encrypted using the private key ID. To decrypt C using the private key $d_{ID} \in G_1^*$, the algorithm works as follows:

1. Compute $V \oplus H_2(e(d_{ID}, U)) = \sigma$.
2. Compute $W \oplus H_4(\sigma) = M$.
3. Set $r = H_3(\sigma, M)$. Check whether $U = rP$. If not, reject the ciphertext.
4. Output M as the decryption of C.

At the encryption phase, the message M is concealed by the hash $H_4(\sigma)$ of the random value σ. Further, σ is bitwise exclusive-ored with the mask $H_2(g_{ID}^r)$ to form V. At the decryption phase, this V is bitwise exclusive-ored with $H_2(e(d_{ID}, U))$. This yields the correct value for σ, because of the identity $g_{ID}^r = e(d_{ID}, U)$ as in the BasicIdent scheme. Now the mask $H_4(\sigma)$ can be removed from $W = M \oplus H_4(\sigma)$ to retrieve the message M. This shows the consistency of the FullIdent scheme.

Observe that the first part of the ciphertext $\langle rP, \sigma \oplus H_2(g_{ID}^r)\rangle$ corresponds to the BasicIdent encryption of σ using the random bits $r = H_3(\sigma, M)$ (which are random by the randomness of σ). Further, the last part $M \oplus H_4(\sigma)$ is a one-time padding of the message M with the random bits $H_4(\sigma)$. Now if we let H_3 and H_4 denote the hash functions H and G respectively, we see that FullIdent is a hybrid scheme as in Theorem 8.4.

The security of the FullIdent scheme is considered in the random oracle model, i.e. hash functions H_1, H_2, H_3, H_4 are viewed as random oracles. Boneh and Franklin [9] show that under the assumption that BDH is a hard problem, FullIdent is a chosen ciphertext secure identity-based encryption scheme (IND-ID-CCA). Namely, if an adversary wins the IND-ID-CCA game with probability non-negligibly more than a half, then an algorithm can be constructed that solves BDH with non-negligible probability. This is made precise in the following theorem.

Theorem 8.5 (Boneh and Franklin [9], Theorem 4.4 and 4.5) *Regard the hash functions H_1, H_2, H_3, H_4 as random oracles. Suppose an adversary $\mathcal{A}$ wins the IND-ID-CCA game in the FullIdent scheme with probability $1/2+\epsilon$ for non-negligible ϵ and $\mathcal{A}$ runs in time at most t. Suppose $\mathcal{A}$ makes at most $q_E > 0$ key extraction queries and $q_{H_2}, q_{H_3}, q_{H_4}$ queries to the hash functions H_2, H_3, H_4, respectively. Then there is an algorithm $\mathcal{B}$ that solves BDH with probability at least*

$$\frac{1}{q_{H_2}(q_{H_3} + q_{H_4})}\left[\left(\frac{\epsilon}{e(1 + q_E + q_D)} + 1\right)(1 - 2/q)^{q_D} - 1\right].$$

The running time of $\mathcal{B}$ is at most $t + O((q_{H_4} + q_{H_3}) \cdot n)$. Here $e \approx 2.71$ is the base of the natural logarithm, n the length of σ and q the size of the groups G_1 and G_2.

From here on, when we talk about the IBE scheme by Boneh and Franklin, we mean the FullIdent version of IBE.

8.3.3 Observations on IBE

For a concrete implementation of their IBE scheme, Boneh and Franklin [9] propose to use the modified Weil pairing on the supersingular elliptic curve $y^2 = x^3 + 1$ over $\mathbb{F}_p$ for some suitable prime p. Concretely, they pick a random k-bit prime for q and set p to be the smallest prime satisfying (1) $p = 2 \pmod 3$, (2) $q \not| p + 1$, and (3) $q^2 | p + 1$. Then G_1 is the subgroup $E(\mathbb{F}_p)[q]$ of q-torsion points in $E(\mathbb{F}_p)$ and G_2 is the group μ_q of qth roots of unity in the multiplicative group of the finite field $\mathbb{F}_{p^2}^*$. Now the modified Weil pairing $\hat{e}_q : G_1 \times G_1 \to G_2$ is a symmetric pairing (see Section 6.3) and thus suitable for use in the IBE scheme.

Recall that hash function H_1 hashes onto the group G_1^*. In this concrete implementation, however, G_1 is a subgroup of the group of points on an elliptic curve, for which it is hard to build a hash function in practice. Therefore, Boneh and Franklin describe how to relax the hashing requirements. Instead of hashing directly onto G_1^*, one can hash onto some set $A \subseteq \{0,1\}^*$ and use an *admissible encoding function* L (for the definition, see [9]) to map A onto G_1^*. This relaxation does not affect the security (see [9], Theorem 4.7). It is not hard to find a suitable admissible encoding function L and set A for the above example, as shown in [9].

Boneh and Franklin observe that also the modified Tate pairing can be used for the implementation of the IBE scheme. In fact, since the Tate pairing can be evaluated faster than the Weil pairing,

it will improve the efficiency of the scheme. Furthermore, it is possible to use other elliptic curves. Note that the pairing needs to be strongly non-degenerate, so we have to use symmetric pairings. In [9], it is described that the IBE scheme can be adapted to suit asymmetric pairings, such as the regular Weil and Tate pairing, as well. In this case, we can also use non-supersingular curves with sufficiently small embedding degree, such as MNT-curves. However, security is then based on a computational problem that is slightly different from BDH, namely the bilinear Diffie-Hellman co-problem (Co-BDH). In a nutshell, given (P, aP, bP, Q, aQ, cQ) with P and Q linearly independent, Co-BDH asks to compute $e(P,Q)^{abc}$ (compare to Co-DDH in Subsection 7.3.2). We will not go into further detail.

As described in Section 8.2, in some applications it is desired to distribute the master key among several KGCs. An interesting property of Boneh and Franklin's IBE scheme is that it provides a robust way to achieve this using threshold cryptography. Suppose there are n KGCs. According to Shamir secret sharing, the master key can be distributed such that each t out of n KGCs can retrieve the master key s. Here, each KGC is given one share s_i of the master key. A user who wants to request a secret key, presents its public key Q_{ID} to t KGCs of his choice. Each of them responds with the partial secret key $d_{ID}^{(i)} = s_i Q_{ID}$. The user then constructs its secret key d_{ID} as $d_{ID} = \sum \lambda_i d_{ID}^{(i)}$, where the λ_i's are the appropriate Lagrange coefficients. Moreover, the fact that DDH is easy in G_1 implies that it is easy to catch a dishonest KGC. Namely, after receiving their share s_i, each KGC publishes the value $P_{pub}^{(i)} = s_i P$. The user can test whether the KGC's response $d_{ID}^{(i)}$ is valid by performing the following check:

$$e(d_{ID}^{(i)}, P) = e(Q_{id}, P_{pub}^{(i)}).$$

If this is not the case, it is immediately clear that the KGC cheated.

8.3.4 Identity-Based Signature Scheme

Recall that the design of a satisfactory ID-based encryption scheme was an open problem until 2001. For ID-based signature schemes, on the other hand, several proposals appeared even before the discovery of the constructive use of pairings. In fact, along with the introduction of identity-based cryptography, Shamir already gave an example of a signature scheme [74]. This scheme did not come with a security proof and had a primarily illustrative role. Examples of satisfactory ID-based signature schemes (not making use of pairings) that have appeared since are [26, 25].

When Boneh and Franklin [9] proposed their IBE scheme, the demand arose for an ID-based signature scheme sharing the same system parameters and keys. Namely, such a compatible signature scheme together with the IBE scheme would result in a complete solution to identity-based cryptography. Since [9], a number of such ID-based signature schemes have been proposed. The first pairing-based signature scheme (actually even prior to [9]) is due to Sakai et al. [69], though they do not include a proof of security. A rather comprehensible scheme resembling the ElGamal signature scheme is given by Paterson [66]. Although he gives some security arguments, Paterson omits a formal security proof. Hess' scheme [36], which is computationally the most efficient, does come with a formal proof of security. However, since Hess assumes that the identity is fixed, this notion of security is not as strong as EUF-ID-CMA. We will describe the identity-based signature scheme by Cha and Cheon [16], which is provably EUF-ID-CMA secure.

The identity-based signature scheme by Cha and Cheon consists of four algorithms: *Setup*, *Extract*, *Sign*, *Verify*. Similar to IBE, the algorithms *Setup* and *Extract* are carried out by the KGC. *Sign* and *Verify* are performed by the users to compute and verify a signature on a message, using the secret and public key, respectively. The algorithms are as follows.

Setup: Similar to IBE. In addition, we pick a hash function $H_5 : \{0,1\}^* \times G \to \mathbb{Z}_q^*$.

Extract: Similar to IBE.

Sign: To sign a message M under secret key d_{ID}, the algorithm works as follows:

- Pick a random $r \in \mathbb{Z}_q^*$.
- Compute $U = rQ_{ID}$, $h = H_5(M, U)$, and $V = (r + h)D_{ID}$.
- Set the signature to be $\sigma = (U, V)$.

Verify: To verify a signature $\sigma = (U, V)$ on a message M, check whether

$$e(P, V) = e(P_{pub}, U + hQ_{ID}), \quad \text{where} \quad h = H_5(M, U).$$

This completes the description of the scheme. Consistency follows from the bilinearity of the pairing and is easily verified:

$$e(P, V) = e(P, (r + h)D_{ID}) = e(P, s(r + h)Q_{ID}) = e(sP, (r + h)Q_{ID}) = e(P_{pub}, U + hQ_{ID}).$$

As with IBE, the security of this scheme is considered in the random oracle model, so the hash functions are viewed as random oracles. Cha and Cheon [16] show that this scheme is EUF-ID-CMA secure under the assumption that CDH is hard in G_1. Note that this assumption is weaker than the assumption needed for the security of IBE, namely that BDH is hard. The following theorem shows that an adversary that wins the EUF-ID-CMA game with non-negligible probability can be used to construct an algorithm that solves CDH with non-negligible probability.

Theorem 8.6 (Cha and Cheon [16], Theorem 3) *Suppose the hash functions H_1, H_5 are random oracles. Suppose an adversary $\mathcal{A}$ runs in time at most t and makes at most q_E key extraction queries, q_S sign queries and q_{H_1}, q_{H_5} queries to the hash functions H_1, H_5, respectively. Suppose $\mathcal{A}$ wins the EUF-ID-CMA game with probability $\epsilon \geq 10(q_S + 1)(q_S + q_{H_5})q_{H_1}/(q - 1)$, then there is an algorithm $\mathcal{B}$ that solves CDH in G_1 with probability $\geq 1/9$ and running time $\leq \frac{23 q_{H_5} q_{H_1} t}{\epsilon(1 - \frac{1}{q})}$.*

8.3.5 Other Identity-Based Applications

Besides encryption and signature schemes, there are a number of other schemes that arise from identity-based cryptography. Some of them are truly identity-based in the sense that the identity of the users is used to derive their public key. Examples of such schemes are ID-based signcryption schemes and ID-based (authenticated) key agreement schemes, which we describe below. Other applications lie in the closely related area of so-called *identifier*-based cryptography, which we already mentioned in Section 8.2. Identifier-based cryptography makes use of the same machinery as identity-based cryptography – in particular it exploits the fact that any value can serve as public key. The subtle difference is that in the case of identifier-based schemes, public keys are derived from data other than identities of users. This way an efficient solution can be found for an access control system, with which we deal later on in this section.

Identity-based signcryption scheme

In some applications, communication needs to provide both secrecy and authenticity. This can be achieved by first signing the message and subsequently encrypting the message-signature pair, as is done in the identity-based scheme by Boyen [13]. Another possibility is to perform signing and encryption simultaneously in order to reduce computational costs. This concept is known as *signcryption*.

A first implementation of an identity-based signcryption scheme is given by Malone-Lee in [51]. Nalla and Reddy [62] propose an improved scheme that is computationally more efficient. Both schemes come with heuristic arguments for security rather than a formal proof. On the contrary, Libert and Quisquater [48] do accompany their identity-based signcryption scheme with a rigorous security proof. The latter scheme is computationally slightly more efficient than the one by Malone-Lee. A variant of signcryption that deserves a mention is Lynn's authenticated identity-based

encryption scheme [49], which provides authentication but does not support non-repudiation (in contrast to signature and signcryption schemes). Libert and Quisquater [48] also describe how their signcryption scheme can be transformed to an authenticated encryption scheme, thereby disabling non-repudiation but improving security.

Identity-based (authenticated) key agreement
Sakai et al. [69] propose a pairing-based protocol that enables two parties that have never met to share a common key, as in the classic Diffie-Hellman key exchange in Section 7.1.3. Unlike Diffie-Hellman key exchange, this key agreement scheme has the nice property that it is fully non-interactive. This feature follows from the identity-based nature of the protocol, which proceeds as follows. Consider a system with algorithms *Setup* and *Extract* as in the IBE scheme. Each user U is assumed to have a public key $Q_U = H_1(ID_U)$ derived from U's identity ID_U and a corresponding secret key $d_U = sQ_U$, where s is the KGC's master key. Suppose users A and B want to share a secret key, then they can both compute $e(d_A, Q_B) = e(Q_A, Q_B)^s = e(Q_A, d_B)$ without the need to communicate even once. Moreover, the fact that the scheme is non-interactive rules out a man-in-the-middle attack. Note that this scheme has the disadvantages that the key is static and also the KGC is capable of computing this key (which can also be an advantage in some applications).

Smart [79] describes how the scheme by Sakai et al. can be adapted such that the common key is dynamic rather than static. This is done by translating the MQV-protocol, which is a variant of the Diffie-Hellman key exchange secure against the man-in-the-middle attack, to the identity-based setting. This identity-based interactive authenticated key exchange works as follows. Users A and B choose an ephemeral (i.e. short-term) secret key a and b respectively, and send the corresponding ephemeral public key aP and bP respectively to the other user. Both users can now compute the common key

$$e(aQ_B, P_{pub}) \cdot e(d_A, bP) = e(aQ_B + bQ_A, P_{pub}) = e(bQ_A, P_{pub}) \cdot e(d_B, aP).$$

The message flow is identical to that of Diffie-Hellman key exchange.

Access control
An application of identifier-based cryptography lies in the encryption of sensitive information in an access control system, as described by Smart [78]. In such systems, users are granted access to information if they have the required privileges, which in this case means knowledge of particular pieces of information. These required pieces of information, or secret keys, are represented in a logical formula involving conjunctions and disjunctions. For example, let d_A, d_B, d_C, d_D be secret keys corresponding to the identifiers (or public keys) A, B, C, D respectively. Then the formula $A \wedge (B \vee (C \wedge D))$ means that only users knowing either the secret keys d_A and d_B, or the secret keys d_A, d_C and d_D have access to the information. In other words, the information is encrypted in such a way that only a correct combination of secret keys (either (d_A, d_B) or (d_A, d_C, d_D)) enables a user to decrypt. Smart achieves this by exploiting the bilinearity of the pairings and the fact that any value can serve as identifier. Clearly, public keys are not identities of users, hence we use the term identifier-based rather than identity-based cryptography.

Chapter 9

Other Applications of Pairings

Besides identity-based cryptography, there are numerous other cryptosystems with convenient properties that can be built with pairings. We will discuss two of these in more detail. The first is Joux's tripartite key exchange [41], which initiated the elaborate research into pairing-based cryptography. The second is the efficient signature scheme by Boneh et al. [11], which provides short signatures and elegantly exploits the properties of GDH groups. Other examples of pairing-based (non identity-based) protocols are an ElGamal encryption scheme with escrow property [9], escrowable encryption in combination with non-repudiative signatures [83], a credential pseudonymous certificate system [84], aggregate signatures [10], an encryption scheme secure in the standard model [15], a signature scheme secure in the standard model [12], and a forward-secure encryption scheme [44].

9.1 Joux's Tripartite Diffie-Hellman Key Exchange

In 2000, Joux [41] discovers that pairings can be used for the construction of cryptographic protocols. Concretely, he uses them to implement a quite simple tripartite variant of the classic Diffie-Hellman key exchange (see Subsection 7.1.3). By the bilinearity of the pairings, it is possible for three parties to share a common key in only a single round of communication, where all previously known protocols take at least two rounds. Joux further remarks that in this setting DDH (or rather Co-DDH, as described in Subsection 7.3.2) becomes easy, thereby indirectly giving the first example of a GDH group.

Joux argues that if there exists a symmetric pairing $e : G_1 \times G_1 \to G_2$ where P is a generator for G_1, then the following protocol can be constructed. Suppose three parties A, B, and C wish to share a secret key over an insecure channel. They randomly choose a number a, b, and c respectively, which serve as ephemeral secret keys, and publish aP, bP, and cP respectively, the ephemeral public keys. All three parties can now compute the common key

$$e(bP, cP)^a = e(aP, cP)^b = e(aP, bP)^c = e(P, P)^{abc}.$$

An eavesdropper who intercepts communications aP, bP, and cP, can only retrieve the common key $e(P, P)^{abc}$ if he is able to solve the bilinear Diffie-Hellman problem $\mathrm{BDH}_P(aP, bP, cP)$. Note that, like the original bipartite Diffie-Hellman key exchange, this scheme is vulnerable to a man-in-the-middle attack.

In the above protocol, it is essential that the map is strongly non-degenerate; otherwise the pairing would map everything to the identity element in G_2 and the common key would not exactly be secret. However, since the notion of distortion maps had not yet been discovered by the time of [41], Joux did not have a strongly non-degenerate pairing available. Therefore, he was forced to adapt the protocol so that it would fit for asymmetric pairings such as the Weil and the Tate

pairing. The adapted version requires two independent points P and Q. The involved parties publish (aP, aQ), (bP, bQ), and (cP, cQ) respectively. They can now compute the common key

$$e(bP, cQ)^a = e(aP, cQ)^b = e(aP, bQ)^c = e(P, Q)^{abc},$$

which is non-trivial by the independency of P and Q and the non-degeneracy of the pairing. However, the bandwidth requirement is twice as large as in the previous protocol, as all parties have to publish two points in stead of one. Security is now based on the Co-BDH problem (see Subsection 8.3.3) instead of the BDH problem. When Verheul [83] discovers the existence of distortion maps for supersingular curves, he remarks that these can be used to create strongly non-degenerate (i.e. symmetric) pairings such as the modified Weil and Tate pairing. Verheul thus improves the tripartite key exchange by realizing Joux's original protocol.

What remains is the vulnerability to the man-in-the-middle attack. Al-Riyami and Paterson [1] show how this problem can be fixed in a traditional PKI-setting. As usual in PKI, user A has a secret key x and a public key $\mu_A = xP$. We will denote these as long-term keys, as opposed to the ephemeral (or short-term) secret key a and public key aP. Further, A has a certificate Cert_A from the CA, where his identity and his long-term public key are bound together by the CA's signature. Similarly, users B and C have certificates Cert_B, respectively Cert_C, for their long-term public keys $\mu_B = yP$ and $\mu_C = zP$. Now the users publish $aP||\text{Cert}_A$, $bP||\text{Cert}_B$, and $cP||\text{Cert}_C$ respectively, where a, b, c are the ephemeral secret keys as before and $||$ denotes concatenation. (Note that the messages are similar to the original protocol, except for the addition of the certificates containing the users's long-term public keys. In particular, this protocol still consists of just a single round of communication.) Next all three users compute the common key as a function of their own secret keys (both short- and long-term), and the (short- and long-term) public keys of the others. Al-Riyami and Paterson describe four ways to do this, all of which use the bilinearity of the pairings to guarantee consistency of the shared key. Note that the certificates guarantee the authenticity of the long-term keys, so the users can no longer be impersonated through a man-in-the-middle attack.

9.2 Short Signatures

As observed by Okamoto and Pointcheval [64], the definition of a GDH group naturally suggests an elegant signature scheme (see Subsection 7.3.1). For convenience, we repeat the protocol of this appropriately named GDH signature scheme, this time writing the group operation additively rather than multiplicatively.

GDH signature scheme
Let $(G_1, +)$ be a cyclic group of prime order n with generator P and let $H : \{0,1\}^* \to G_1^*$ be a hash function. User A has a secret key $s \in \mathbb{Z}_n^*$ and a corresponding public key $P_{pub} = sP$. Let $M \in \{0,1\}^*$ be the message to be signed.

1. A computes $Q = H(M) \in G_1^*$ and $\sigma = sQ$. The signature is $\sigma \in G_1^*$.
2. B computes $Q = H(M)$ and accepts the signature if $\text{DDH}_P(P_{pub}, Q, \sigma) = 1$ and disavows otherwise.

The main idea is that the signer uses his secret key s to turn the hashed message Q into a signature σ such that (P, P_{pub}, Q, σ) is a Diffie-Hellman tuple. Computation of σ without knowledge of s boils down to solving an instance of CDH, whereas signature verification involves solving DDH. Thus, to simultaneously provide security and efficiency, we need to define this scheme on a group where CDH is hard, yet DDH is easy – a GDH group. As described in Subsection 7.3.2, the only known way to create a GDH group is by using a symmetric pairing $e : G_1 \times G_1 \to G_2$, where G_2 is a cyclic group of order n. Then the verification is done by checking whether $e(P, \sigma) = e(P_{pub}, Q)$.

Boneh, Lynn and Shacham [11] observe that a GDH signature is a single element of G_1^*. Hence, they look for a GDH group in which elements have a short representation. Boneh et al. propose to use the GDH group built from the modified Weil pairing $\widehat{e}_n : E(\mathbb{F}_q) \times E(\mathbb{F}_q) \rightarrow \mathbb{F}_{q^k}$ on a supersingular elliptic curve with embedding degree $k = 6$. This way, they achieve signatures that are half the size of for instance DSA or ECDSA, while providing similar security. Moreover, they remark that when curves with larger embedding degree are used, the signature size can be reduced even further, while maintaining the same level of security. However, for supersingular curves the maximum embedding degree is six. Therefore, one would have to use non-supersingular curves for which no distortion maps and consequently no strongly non-degenerate pairings exist. Hence, the algorithm needs to be slightly adapted such that the point P and hashed message Q come from different groups and are thus linearly independent. Signature verification then involves a decision Diffie-Hellman co-problem (Co-DDH, see Subsection 7.3.2), whereas security is based on the computational Diffie-Hellman co-problem (Co-CDH, see Subsection 8.3.3).

Security of the GDH signature scheme by Boneh et.al. [11] is proven in the random oracle model. The following theorem shows that the scheme is EUF-CMA secure under the assumption that CDH is hard in G_1.

Theorem 9.1 (Boneh, Lynn and Shacham [11]) *Suppose the hash function H is a random oracle. Suppose an adversary $\mathcal{A}$ runs in time at most t and makes at most q_S sign queries and at most q_H hash queries. Suppose $\mathcal{A}$ wins the EUF-CMA game with non-negligible probability ϵ. Then there is an algorithm $\mathcal{B}$ that solves CDH in G_1 with probability at least $\epsilon/(2e \cdot q_s)$ in running time at most $t + 2c_{\mathcal{A}}(logp)(q_H + q_S)$, where $c_{\mathcal{A}}$ is a small constant and e the base of the natural logarithm.*

Chapter 10

Concluding Remarks

This last chapter is devoted to some concluding remarks on the topics of this study. Section 10.1 deals with the mathematical aspects that were covered in Part 1 (Chapters 3 - 6). Section 10.2 discusses some application-related issues that emerged in Part 2 (Chapters 7 - 9).

10.1 Theory of the Pairings

We compare the Weil and the Tate pairing, observe which curves are suitable for pairing-based applications, and suggest some topics for further research.

10.1.1 The Weil versus the Tate Pairing

From a practical point of view, the two main criteria in comparing the pairings are performance and applicability. The Tate pairing turns out to be superior in both aspects.

Performance: The computation of the Tate pairing can be done approximately twice as fast as the computation of the Weil pairing. Since pairings are often the most costly operation in practical applications, this aspect is an important advantage of the Tate pairing.

Applicability: The applicability of both the Weil and the Tate pairing depends on the embedding degree of the elliptic curve. Moreover, an additional requirement exists for the application of the Weil pairing. Hence, the Tate pairing can be applied whenever the Weil pairing can. We should remark, however, that for practical applications this additional requirement for the Weil pairing is almost always met.

A disadvantage of the Tate pairing is that the outcome is not a unique value, which is often required in applications. This problem can be solved by performing an exponentiation on the outcome of the Tate pairing. Even after this exponentiation, the Tate pairing is considerably faster than the Weil pairing.

An advantage of the Weil pairing is that its definition is more comprehensible than that of the Tate pairing, which involves equivalence classes of quotient groups. Further, for the Weil pairing another definition exists that can be used to prove the properties of the pairing. In particular, the non-degeneracy property is easier to understand for this definition of the Weil pairing than for the Tate pairing.

Despite these two slight disadvantages, the Tate pairing seems to be preferable to the Weil pairing for use in practical applications, because of the better performance and applicability. The advantages of the Weil and the Tate pairing are listed in Table 10.1.

Weil pairing	Tate pairing
Easier to understand	Better performance
Unique outcome	Applicable more often

Table 10.1: Advantages of the Weil and the Tate pairing

10.1.2 Suitable Curves

The Weil and the Tate pairing can be applied if the embedding degree of the elliptic curve is sufficiently small. However, with overwhelming probability, the embedding degree of a randomly chosen curve is too large. Hence, if we want to find curves suitable for pairing-based applications, we need to consider special classes of curves.

One of these special classes is the class of supersingular elliptic curves. The embedding degree of supersingular curves is less than or equal to 6, which is sufficiently small for efficient computation of the pairings. Moreover, supersingular curves have a rich structure, which makes the existence of distortion maps possible. These distortion maps map a point on the curve to a linearly independent point of the same order. Hence, they can be used to modify the pairings such that they satisfy the strong non-degeneracy property, which often comes in handy in cryptographic applications. On the other hand, the rich structure potentially provides cryptanalysts with tools to attack cryptosystems defined on these curves.

For some applications an embedding degree slightly larger than 6 is desired. Since this is not possible for supersingular curves, one needs to consider non-supersingular curves that are constructed in a special way. These curves, known as MNT-curves, can have arbitrary embedding degree. Downside to these curves, however, is that they are often not of (near) prime-order, which is desired in cryptographic applications. Further, the definition field cannot be chosen beforehand, but follows from the construction. Moreover, since MNT-curves are non-supersingular, no distortion maps exists on curves that are cryptographically interesting.

10.1.3 Further Research

Some topics for further research that have cryptographic relevance could be:

MNT-curves: It would be interesting to have a generalization that makes it possible to construct MNT-curves having arbitrary embedding degree, (near) prime-order and arbitrary definition field.

Curves of higher genus: Pairings are not restricted to curves that are elliptic, i.e. having genus one. Hence, research into curves of higher genus, which are beyond the scope of this study, could yield classes of curves with nice properties.

Implementation issues: General improvements of the algorithm or discovery of specific efficient cases are of cryptographic importance. However, we doubt if there is much room for improvement, because a lot of effort has been put in the optimization of the pairing evaluation already.

Further, we point out some questions that seem to have little cryptographic relevance (other than enhancing the insight into the theory of pairings).

Open questions: The following questions appear to be left open in the literature.

- Can non-degeneracy of the Weil pairing be proven directly for the alternative definition (Definition 3.7)?

- Lemma 4.5 proves the triviality of the Tate pairing of a point in the ground field with itself for embedding degree greater than one. What can be said in general when the embedding degree is equal to one?
- Which supersingular curves do not have a distortion map? In particular, can supersingular curves with embedding degree one have a distortion map?
- Consider an endomorphism on a supersingular curve that is a distortion map with respect to a certain point. Can a (general) lower bound for m be given, such that the endomorphism is a distortion map for all m-torsion points over the ground field, except for the point at infinity?

10.2 Pairing-Based Cryptography

We discuss the status of identity-based cryptography, make some observations on the security of pairing-based crypto, and suggest some topics for further research.

10.2.1 Identity-Based Cryptography

Pairings on elliptic curves have proven to be useful building blocks for many cryptosystems. Yet the most remarkable application of pairings is the realization of identity-based cryptography. In this type of public-key cryptography, the public keys are derived from the identity of the users. Ideally, identity-based cryptography promises to provide an alternative to public-key infrastructures, which are the conventional solution to the public key authentication problem. Namely, no certificates are needed because of the identity-based nature of the public keys. Hence, public key distribution is greatly simplified compared to settings based on public-key infrastructures. Moreover, the freedom in choosing the public key enables the design of cryptosystems with interesting properties that cannot be realized with conventional public-key cryptography. This variation is called identifier-based cryptography.

However, real life turns out not to be ideal. Several practical issues stand in the way of the realization of an ideal identity-based system. For instance, problems arise from the inherent key escrow, the difficulty of key revocation, and the commercial immaturity. Most of these problems get more severe as the system gets larger. Hence, identity-based systems seem to work best in a small-scale deployment. In conclusion, identity-based cryptography is not likely to be suitable for large-scale, autonomous application. Still, because of the simple key distribution and additional possibilities, identity-based cryptography could develop into a valuable supplement to traditional public-key infrastructures. The advantages and disadvantages of identity-based cryptography in relation to PKI-based cryptography are summarized below.

Advantages:

- Simplification key distribution.
- Additional possibilities identifier-based crypto.

Disadvantages:

- Problems with key escrow.
- Problems with key revocation.
- Commercial immaturity.

10.2.2 Security of Pairing-Based Cryptography

Conventional elliptic curve cryptosystems are based on the hardness of either the computational or the decision Diffie-Hellman problem on the group of points on an elliptic curve. These problems have been studied extensively and are understood well enough to provide confidence in their hardness. Pairing-based cryptography often relies on the bilinear Diffie-Hellman problem (or related problems). The bilinear Diffie-Hellman problem is often assumed to be equivalent to the computational version; however, at present no proof of this equivalence exists. Moreover, it has not been around long enough for a thorough scrutiny. Therefore, there exists some scepsis among cryptographers when it comes to the security of pairing-based systems. Further research into the hardness of the bilinear Diffie-Hellman problem is needed to reduce this scepsis and increase the trust in pairing-based systems.

10.2.3 Further Research

We recommend the following topics for further research.

Bilinear Diffie-Hellman problem: The security of many pairing-based cryptosystems is based on the bilinear Diffie-Hellman problem or related problems. It is therefore important to get a thorough understanding of these problems, examine their relation to other problems, and find specific weak cases that are to be avoided.

Applications pairing-based crypto: A lot of effort has been put in finding pairing-based applications already. Nevertheless, we believe that pairing-based cryptography still has some interesting applications in store.

Security in standard model: The security of the identity-based encryption scheme using pairings is proven in the random oracle model. It would be interesting to find a way to adapt the scheme such that it provides security in the standard model.

Practical experience identity-based crypto: A systematic research into the practical experience of the application of identity-based cryptosystems in different environments is needed to make clear what the potentials of identity-based crypto really are.

Bibliography

[1] Al-Riyami, S.S. and Paterson, K.G. - Tripartite Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2003/167, 2003. `http://eprint.iacr.org`.

[2] Atkin, A.O.L. and Morain F. - Elliptic Curves and Primality Proving. *Mathematics of Computation*, 61(203), 1993, pp. 29–68.

[3] Balasubramanian, R. and Koblitz, N. - The Improbability That an Elliptic Curve Has Subexponential Discrete Log Problem under the Menezes-Okamoto-Vanstone Algorithm. *Journal of Cryptology*, Vol. 2, 1998, pp. 141–145.

[4] Barreto, P.S.L.M. - Pairing-Based Crypto Lounge. Available from `http://planeta.terra.com.br/informatica/paulobarreto/pblounge.html`.

[5] Baretto, P.S.L.M., Kim, H.Y., Lynn, B. and Scott, M. - Efficient Algorithms for Pairing-Based Cryptosystems. *Advances in Cryptology - Crypto 2002*, LNCS 2442, Springer-Verlag (2002), pp. 354–368.

[6] Barreto, P.S.M.L., Lynn, B. and Scott, M. - Constructing Elliptic Curves with Prescribed Embedding Degrees. *Proc. of the Third Workshop on Security in Communications Networks (SCN'2002)*, LNCS 2576, Springer-Verlag (2003), pp. 257–267.

[7] Barreto, P.S.M.L., Lynn, B. and Scott, M. - On the Selection of Pairing-Friendly Groups. Cryptology ePrint Archive, Report 2003/086, 2003. `http://eprint.iacr.org`.

[8] Blake, I., Seroussi, G. and Smart, N. - *Elliptic Curves in Cryptography*. Camebridge University Press, 1999.

[9] Boneh, D. and Franklin, M. - Identity-Based Encryption from the Weil Pairing. *Advances in Cryptology - Crypto 2001*, LNCS 2139, Springer-Verlag (2001), pp. 213–229.

[10] Boneh, D., Gentry, C., Lynn, B. and Shacham, H. - Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. *Advances in Cryptology - Eurocrypt 2003*, LNCS 2652, Springer-Verlag (2003), pp. 416–432.

[11] Boneh, D., Lynn, B. and Shacham, H. - Short Signatures from the Weil Pairing. *Advances in Cryptology - Asiacrypt 2001*, LNCS 2248, Springer-Verlag (2002), pp. 514–532.

[12] Boneh, D., Mironov, I. and Shoup, V. - A Secure Signature Scheme from Bilinear Maps. *Topics in Cryptology - CT-RSA 2003*, LNCS 2612, Springer-Verlag (2003), pp. 98–110.

[13] Boyen, X. - Multipurpose Identity-Based Signcryption: A Swiss Army Knife for Identity-Based Cryptography. *Advances in Cryptology - Crypto 2003*, LNCS 2729, Springer-Verlag (2003), pp. 382–398.

[14] Brezing, F. and Weng, A. - Elliptic Curves Suitable for Pairing Based Cryptography. Cryptology ePrint Archive, Report 2003/143, 2003. `http://eprint.iacr.org`.

[15] Canetti, R., Halevi, S. and Katz, J. - Chosen-Ciphertext Security from Identity-Based Encryption. *Advances in Cryptology - Eurocrypt 2003*, LNCS 2656, Springer-Verlag (2003), pp. 255–271.

[16] Cha, J.C. and Cheon, J.H. - An Identity-Based Signature from Gap Diffie-Hellman Groups. *Practice and Theory in Public Key Cryptography - PKC 2003*, LNCS 2567, Springer-Verlag (2003), pp. 18–30.

[17] Charlap, L.S. and Coley, R. - An Elementary Introduction to Elliptic Curves II. *CCR Expository Report No. 34*, 1990.

[18] Charlap, L.S. and Robbins, D.P. - An Elementary Introduction to Elliptic Curves. *CRD Expository Report No. 31*, 1988.

[19] Chaum, D. - Zero-Knowledge Undeniable Signatures. *Advances of Cryptology - Eurocrypt '90*, LNCS 473, Springer-Verlag (1991), pp. 458–464.

[20] Cheon, J.H. and Lee, D.H. - Diffie-Hellman Problems and Bilinear Maps. Cryptology ePrint Archive, Report 2002/117, 2002. `http://eprint.iacr.org`.

[21] Cocks, C. - An Identity Based Encryption Scheme Based on Quadratic Residues. *Proc. of 8th IMA International Conference on Cryptography and Coding*, LNCS 2260, Springer-Verlag (2001), pp. 360–363.

[22] Diffie, W. and Hellman, M. - New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6) (1976), pp. 644–654.

[23] Dupont, R., Enge, A. and Morain, F. - Building Curves with Arbitrary Small MOV Degree over Finite Fields. Cryptology ePrint Archive, Report 2002/094, 2002. `http://eprint.iacr.org`.

[24] Eisenträger, K., Lauter, K. and Montgomery, P.L. - Fast Elliptic Curve Arithmetic and Improved Weil Pairing Evaluation. *Topics in Cryptology - CT-RSA 2003*, LNCS 2612, Springer-Verlag (2003), pp. 343–354.

[25] Feige, U., Fiat, A. and Shamir, A. - Zero-knowledge proofs of identity, *Journal of Cryptology*, Vol. 1, 1988, pp. 77–94.

[26] Fiat, A. and Shamir, A. - How to prove yourself: Practical solutions to identification and signature problems. *Advances in Cryptology - Crypto '86*, LNCS 263, Springer-Verlag (1986), pp. 186–194.

[27] Frey, G., Müller, M. and Rück, H.G. - The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. *IEEE Transactions on Information Theory* 45(5) (1999), pp. 1717–1719.

[28] Frey, G. and Rück, H.G. - A remark concerning the m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62 No.206 (1994), pp. 865–874.

[29] Fujiaki, E. and Okamoto, T. - Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Advances of Cryptology - Crypto '99*, LNCS 1666, Springer-Verlag (1999), pp. 537-554.

[30] Galbraith, S.D. - Supersingular Curves in Cryptography. *Advances in Cryptology - Asiacrypt 2001*, LNCS 2248, Springer-Verlag (2001), pp. 495–513.

[31] Galbraith, S.D., Harrison, K. and Soldera, D. - Implementing the Tate Pairing. *Algorithmic Number Theory 5th International Symposium, ANTS-V*, LNCS 2369, Springer-Verlag (2002), pp. 324–337.

[32] Gentry, C. and Silverberg, A. - Hierarchical ID-based cryptography. Cryptology ePrint Archive, Report 2002/056, 2002. http://eprint.iacr.org.

[33] Harasawa, R., Shikata, J., Suzuki, J. and Imai, H. - Comparing the MOV and FR Reductions in Elliptic Curve Cryptography. *Advances in Cryptology - Eurocrypt '99*, LNCS 1592, Springer-Verlag (1999), pp. 190–205.

[34] Hess, F. - A Note on the Tate Pairing of Curves over Finite Fields. To appear in *Arch. Math.*, 2002.

[35] Hess, F. - Some Remarks on the Weil and Tate Pairings of Curves over Finite Fields. Unpublished manuscript.

[36] Hess, F. - Exponent Group Signature Schemes and Efficient Identity Based Signature Schemes Based on Pairings. Cryptology ePrint Archive, Report 2002/012, 2002. http://eprint.iacr.org.

[37] Howe, E.W. - The Weil pairing and the Hilbert symbol. *Mathematische Annalen* 305, Springer Verlag (1996), pp. 387–392.

[38] Ireland, K. and Rosen, M. - *A Classical Introduction to Modern Number Theory*, Springer Verlag, 1990.

[39] Izu, T. and Takagi,T - Efficient Computations of the Tate Pairing for the Large MOV Degrees. *5th International Conference on Information Security and Cryptology - ICISC 2002*, LNCS 2587, Springer-Verlag (2003), pp. 283–297.

[40] Jacobsen Jr., M.J., Koblitz, N., Silverman, J.H., Stein, A. and Teske, E. - Analysis of the Xedni calculus attack. *Designs, Codes and Cryptography*, Vol 19 (2000). Available from http://www.cacr.math.uwaterloo.ca.

[41] Joux, A. - A One Round Protocol for Tripartite Diffie-Hellman. *Algorithmic Number Theory Symposium - ANTS-IV*, LNCS 1838, Springer-Verlag (2000), pp. 385–394.

[42] Joux, A. - The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. *Algorithmic Number Theory Symposium - Ants-V*, LNSC 2369, Springer-Verlag (2002), pp. 20–32.

[43] Joux, A. and Nguyen, K. - Seperating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, 2001. http://eprint.iacr.org.

[44] Katz, J. - A Forward-Secure Public-Key Encryption Scheme. Cryptology ePrint Archive, Report 2002/060, 2002. http://eprint.iacr.org.

[45] Kanayama, N., Kobayashi, T., Saito, T. and Uchiyama, S. - Remarks on the Elliptic Curve Discrete Logarithm Problems. *Journal of IEICE Transactions of Fundamentals of Electronics*, Vol. E83-A, no. 1, 2000.

[46] Koblitz, N. - Elliptic curve cryptosystems. *Mathematics of Computation*, 48 (1987), pp. 203–209.

[47] Lenstra, A.K. and Verheul, E.R. - Selecting Cryptographic Key Sizes. *Journal of Cryptology*, Vol. 14, No. 4, 2001, pp. 255–293.

[48] Libert, B. and Quisquater, J.-J. - New identity based signcryption schemes from pairings. Cryptology ePrint Archive, Report 2003/023, 2003. http://eprint.iacr.org.

[49] Lynn, B. - Authenticated Identity-Based Encryption. Cryptology ePrint Archive, Report 2002/072, 2002. http://eprint.iacr.org.

[50] Lynn, B. - Elliptic curves. Available from
http://rooster.stanford.edu/~ben/notes/elliptic.

[51] Malone-Lee, J. - Identity-Based Signcryption. Cryptology ePrint Archive, Report 2002/098, 2002. http://eprint.iacr.org.

[52] Maurer, U.M. - Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms. *Advances in Cryptology - Crypto '94*, LNCS 839, Springer-Verlag (1994), pp. 271–281.

[53] Maurer, U.M. and Wolf, S. - The Relationship between Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms. *Advances in Cryptology - Crypto '96*, LNCS 1109, Springer-Verlag (1999), pp. 268–282.

[54] Menezes, A.J. - *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

[55] Menezes, A.J. Okamoto, T. and Vanstone, S.A. - Reducing Elliptic Curve Logarithms in a Finite Field. *IEEE Transactions on Information Theory* 39 (1993), pp. 1639–1646.

[56] Menezes, A.J., Van Oorschot, P.C. and Vanstone, S.A. - *Handbook of Applied Cryptography.* CRC Press, 1996.

[57] Miller, V. - Elliptic Curves and their Use in Cryptography. Available from
http://www.cacr.math.uwaterloo.ca/conferences/1998/ecc98/miller-paper.ps.

[58] Miller, V. - Short Programs for Functions on Curves. Unpublished manuscript, 1986.

[59] Miller, V. - Use of Elliptic Curves in Cryptography. *Advances in Cryptology - Crypto '85*, LNCS 218, Springer-Verlag (1986), pp. 417–426.

[60] Miyaji, A. - Elliptic Curves over $\mathbb{F}_p$ Suitable for Cryptosystems. *Advances in Cryptology - Auscrypt '92*, LNCS 718, Springer-Verlag (1993) pp. 479–491.

[61] Miyaji, A., Nakabayashi, M. and Takano, S. - New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *IEICE Transformations on Fundamentals* E84-A(5) (2001), pp. 1234–1243.

[62] Nalla, D. and Reddy, K.C. - Signcryption Scheme for Idenity-Based Cryptosystems. Cryptology ePrint Archive, Report 2003/066, 2003. http://eprint.iacr.org.

[63] Odlyzko, A. - Discrete Logarithms: The Past and the Future. *Designs, Codes, and Cryptography*, Vol. 19, No. 2–3, 2000, pp. 129–145.

[64] Okamoto, T. and Pointcheval, D. - The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. *Practice and Theory in Public Key Cryptography - PKC 2001*, LNCS 1992, Springer-Verlag (2001), pp. 104–118.

[65] Paterson, K.G. - Cryptography from Pairings: A Snapshot of Current Research. *Information Security Technical Report* 7(3) (2002), pp. 41–54.

[66] Paterson, K.G. - ID-based Signatures from Pairings on Elliptic Curves. Cryptology ePrint Archive, Report 2002/004, 2002. http://eprint.iacr.org.

[67] Pohlig, G.C. and Hellman, M.E. - An Improved Algorithm for Computing Logarithms over $GF(p)$ and its Cryptographic Significance. *IEEE Transactions on Information Theory* 24 (1978), pp. 106–110.

[68] Rubin, K. and Silverberg, A. - Supersingular Abelian Varieties in Cryptology. *Advances in Cryptology - Crypto 2002*, LNCS 2442, Springer-Verlag (2002), pp. 336–353.

[69] Sakai, R., Ohgishi, K. and Kasahara, K. - Cryptosystems based on Pairing. *2000 Symposium on Cryptography and Information Security (SCIS2000)*, Okinawa, Japan, Jan. 26–28, 2000.

[70] Satoh, T. and Araki, K. - Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves. *Comm. Math. Univ. Sancti Pauli*, 47, 1998, pp. 81–92.

[71] Schoof, R. - Nonsingular Plane Cubic Curves over Finite Fields. *Journal of Combinatorical Theory*, A 46 (1987), pp. 183–211.

[72] Scott, M. - The Tate Pairing. Available from `www.computing.dcu.ie/~mike/tate.html`.

[73] Semaev, I.A. - Evalutation of Discrete Logarithms in a Group of p-Torsion Points of an Elliptic Curve in Characteristic p. *Mathematics of Computation*, 67 (1998), pp. 353–356.

[74] Shamir, A. - Identity-Based Cryptosystems and Signature Schemes. *Advances in Cryptology - Crypto '84*, LNCS 196, Springer-Verlag (1984), pp. 47–53.

[75] Shoup, V - Lower Bounds for Discrete Logarithms and Related Problems. *Advances of Cryptology - Eurocrypt '97*, LNCS 1233, Springer-Verlag (1997), pp. 256–266.

[76] Silverman, J.H. - *The Arithmetic of Elliptic Curves*, Springer-Verlag, New York, 1986.

[77] Silverman, J.H. - The Xedni calculus and the elliptic curve discrete logarithm problem. *Designs, Codes and Cryptography*, Vol. 20 (2000), pp. 5–40.

[78] Smart, N.P. - Access Control Using Pairing Based Cryptography. *Topics in Cryptology - CT-RSA 2003*, LNCS 2612, Springer-Verlag (2003), pp. 111–121.

[79] Smart, N.P. - An Identity Based Authenticated Key Agreement Protocol Based on the Weil Pairing. Cryptology ePrint Archive, Report 2001/111, 2001. `http://eprint.iacr.org`.

[80] Smart, N.P. - The Discrete Logarithm Problem on Elliptic Curves of Trace One. *Journal of Cryptology*, Vol. 12, No. 3, 1999, pp. 193–196.

[81] Studholme, C. - *The Discrete Log Problem*. PhD. Thesis, University of Toronto, 2001.

[82] Van Tilborg, H.C.A. - *Fundamentals of cryptology: a professional reference and interactive tutorial*, Kluwer Academics Publishers, 2000.

[83] Verheul, E. - Evidence that XTR Is More Secure than Supersingular Elliptic Curve Cryptosystems. *Advances in Cryptology - Eurocrypt 2001*, LNCS 2045, Springer-Verlag (2001), pp. 195–210.

[84] Verheul, E. - Self-Blindable Credential Certificates from the Weil Pairing. *Advances in Cryptology – Asiacrypt 2001*, LNCS 2248, Springer-Verlag (2002), pp. 533–551.

Appendix A

Mathematica Code

We give algorithms for computing the Weil and the Tate pairing on the elliptic curves $E/\mathbb{F}_p: y^2 = x^3 + ax$ ($p = 3 \pmod 4$) and $E/\mathbb{F}_p: y^2 = x^3 + b$ ($p = 2 \pmod 3$). Since these curves are class (1) supersingular curves, they have embedding degree $k = 2$. Therefore, we consider the field extension $\mathbb{F}_{p^2}$ of degree 2. We apply the algorithms to the curves $E_1/\mathbb{F}_{11}: y^2 = x^3 + 3x$ and $E_2/\mathbb{F}_{131}: y^2 = x^3 + 31$.

```
<< Algebra`FiniteFields`;

fld = GF[p, {1, 0, 1}];
```

We start with defining some functions and modules that make our life easier, such as the function `Pt[a,b,c,d]` that gives us the point $(a + b\,i,\ c + d\,i)$, and the module `Neg[P]` that returns the point on the curve given by $-P$.

```
Pt[a_, b_, c_, d_] := {fld[{a, b}], fld[{c, d}]}

Neg[P_] := Module[

  (* Input: a point P on E (Fp^2);
   Output: the point -P on E (Fp^2). *)

  {nP}, If[P == O, nP = O, ,
   If[P[[2]] == fld[{0, 0}], nP = P, , nP = {P[[1]], -P[[2]]}]];
  nP]
```

On input of two points $P, Q \in E(\mathbb{F}_{p^2})$, the module `PointAddition[P,Q]` computes the sum of P and Q.

```
PointAddition[P_, Q_] := Module[

  (* Input: two points P and Q on E (Fp^2);
   Output: R, the sum of P and Q on E (Fp^2). *)

  {λ, R1, R2, R},
  If[P == O,
   If[Q == O, R = O, , R = Q], ,
   If[Q == O, R = P, ,
    If[P == Neg[Q],
     R = O, ,
     If[P == Q, λ = (3 P[[1]]^2 + a)/(2 P[[2]]), , λ = (Q[[2]] - P[[2]])/(Q[[1]] - P[[1]])];
     If[R1 = λ^2 - P[[1]] - Q[[1]]; R1 == 0, R1 = fld[{0, 0}]];
     If[R2 = λ (P[[1]] - R1) - P[[2]]; R2 == 0, R2 = fld[{0, 0}]];
     R = {R1, R2}]]];
  R]
```

We use the module above to define the module `PointMultiplication[P,k]`, which enables us to multiply a point $P \in E(\mathbb{F}_{p^2})$ with a scalar k.

```
PointMultiplication[P_, k_] := Module[

  (* Input: a point P on E (F_p^2) and an integer k ≥ 1;
   Output: the point [k] P on E (F_p^2). *)

  {binexp, l, H},
  binexp = IntegerDigits[k, 2]; l = Length[binexp]; H = P;
  For[j = 2, j ≤ l, H = PointAddition[H, H];
   If[binexp[[j]] == 1, H = PointAddition[H, P]]; j++];
  H]
```

We rewrite the module `PointAddition[P,Q]` to the module `DivisorAddition[D1,D2,x,y]`, which adds two divisors in canonical form and expresses the sum of those divisors in canonical form. So the module takes as input a pair of divisors $D_1 = \{P_1, f_1\} = (P_1) - (O) + \mathrm{div}(f_1)$ and $D_2 = \{P_2, f_2\} = (P_2) - (O) + \mathrm{div}(f_2)$, and the variables x, y for the functions, and outputs $D_1 + D_2 = D_3 = \{P_3, f_3\} = (P_3) - (O) + \mathrm{div}(f_3)$.

```
DivisorAddition[D1_, D2_, x_, y_] := Module[

  (* Input: two divisors in canonical form D1 =
    {P1,f1} and D2 = {P2,f2} and the variables x and y for the function;
   Output: the divisor D3 = {P3,f3},
   where D3 is the sum of D1 and D2 in canonical form. *)

  {λ, R, R1, R2, l, v},
  If[D1[[1]] == O,
   If[D2[[1]] == O, R = O; l = 1; v = 1, ,
    R = D2[[1]]; l = x - D2[[1, 1]]; v = x - D2[[1, 1]]], ,
   If[D2[[1]] == O, R = D1[[1]]; l = x - D1[[1, 1]]; v = x - D1[[1, 1]], ,
    If[D1[[1]] == Neg[D2[[1]]],
     R = O; l = x - D1[[1, 1]]; v = 1, ,
     If[D1[[1]] == D2[[1]],
      λ = (3 D1[[1, 1]]^2 + a)/(2 D1[[1, 2]]), , λ = (D2[[1, 2]] - D1[[1, 2]])/(D2[[1, 1]] - D1[[1, 1]])];
     If[R1 = λ^2 - D1[[1, 1]] - D2[[1, 1]]; R1 == 0, R1 = fld[{0, 0}]];
     If[R2 = λ (D1[[1, 1]] - R1) - D1[[1, 2]]; R2 == 0, R2 = fld[{0, 0}]];
     R = {R1, R2}; l = y - λ x + (λ D1[[1, 1]] - D1[[1, 2]]); v = x - R1]]];
  {R, D1[[2]] D2[[2]] l / v}]
```

Similarly, the module `DivisorMultiplication[D1,k,x,y]` multiplies a divisor D_1 in canonical form with an integer k, and expresses the resulting divisor in canonical form as well.

```
DivisorMultiplication[D1_, k_, x_, y_] := Module[

  (* Input: a divisor D1 = {P1,f1} in canonical form,
   an integer k ≥ 1, and the variables x and y for the function;
   Output: the divisor k D1 = D2 = {P2,f2} in canonical form. *)

  {binexp, l, D2},
  binexp = IntegerDigits[k, 2]; l = Length[binexp]; D2 = D1;
  For[j = 2, j ≤ l, D2 = DivisorAddition[D2, D2, x, y];
   If[binexp[[j]] == 1, D2 = DivisorAddition[D2, D1, x, y]]; j++];
  D2]
```

The following module randomly picks a point $P \neq O$ on the curve over the ground field $E(\mathbb{F}_p)$. This is done by choosing an x-coordinate at random, and solving the resulting quadratic equation for y, if possible. We assume that the prime p satisfies $p = 3 \pmod 4$, because this greatly simplifies the extraction of square roots in the finite field $\mathbb{F}_p$.

```
RandomPointGroundField := Module[

  (* Input: - ;
   Output: a random point P ∈ E (Fp), P ≠ O. *)

  {P1, P2},
  While[P1 = Random[Integer, {0, p - 1}]; JacobiSymbol[P1^3 + a P1 + b, p] == -1];
  If[Random[Integer] == 1, P2 = Mod[(P1^3 + a P1 + b)^((p+1)/4), p],
   P2 = Mod[-(P1^3 + a P1 + b)^((p+1)/4), p]];
  Pt[P1, 0, P2, 0]]
```

Let P and Q be two m-torsion points on $E(\mathbb{F}_{11^2})$. The Weil pairing requires two points $T \neq -P$ and $U \neq -Q$ such that $T \neq U$, $Q+U$ and $P+T \neq U$, $Q+U$. We use the module `RandomPointGroundField` to pick random points on the curve over the ground field, and check whether they satisfy the conditions. Note that taking points on the curve over the ground field rather than over the extension field does not affect the outcome of the pairing, but it does speed up the computations.

```
PickTUGroundField[P_, Q_] := Module[

  (* Input: two points P,Q ∈ E (Fp²);
   Output: points {T,U,P+T,Q+U}, where T,U ∈ E (Fp), T ≠ O,
   -P and U ≠ O,-Q such that P+T ≠ U,Q+U and T ≠ U,Q+U. *)

  {T, U, PT, QU},
  While[T = RandomPointGroundField; PT = PointAddition[P, T];
   U = RandomPointGroundField; QU = PointAddition[Q, U];
   P == Neg[T] ∨ Q == Neg[U] ∨ PT == U ∨ PT == QU ∨ T == U ∨ T == QU];
  {T, U, PT, QU}]
```

For the Weil pairing $e_m(P, Q)$ to be non-trivial, the two m-torsion points P and Q need to be independent. The module `DistortionMap[P]` computes a point Q of the same order as, but independent from P. Note that $\phi : (x, y) \to (-x, i\,y)$ is a distortion map for the curve $y^2 = x^3 + a\,x$, whereas a distortion map for $y^2 = x^3 + b$ is given by $\phi : (x, y) \to (\omega\,x, y)$, with ω a primitive third root of unity.

```
DistortionMap[P_] := Module[

  (* Input: point P;
   Outup: point ϕ (P), where ϕ is the distortion map. *)

  {Q, ω},
  If[P == O, Q = O, , If[b == 0, Q = {If[P[[1]] == fld[{0, 0}], fld[{0, 0}], , -P[[1]]],
      If[P[[2]] == fld[{0, 0}], fld[{0, 0}], , fld[{0, 1}] P[[2]]]},
    While[ω = fld[{Random[Integer, {1, p - 1}],
        Random[Integer, {0, p - 1}]}]^((p^2-1)/3); ω == fld[{1, 0}]];
    Q = {If[P[[1]] == fld[{0, 0}], fld[{0, 0}], , ω P[[1]]], P[[2]]}];
  Q]
```

On input of an m-torsion point P, an arbitrary T, the value m, and variables x, y, the module `ComputeFunctionWeil[P,T,m,x,y]` computes the function $f(x, y)$ such that $\mathrm{div}(f) = m(P + T) - m(T)$.

```
ComputeFunctionWeil[P_, T_, m_, x_, y_] := Module[

  (* Input: an m-torsion point P, a point T,
   the value m, and the variables x and y for the function;
   Output: function f such that div (f) = m (P+T)-m (T) *)

  {mD1, mD2},
  mD1 = DivisorMultiplication[{PointAddition[P, T], fld[{1, 0}]}, m, x, y];
  mD2 = DivisorMultiplication[{T, 1}, m, x, y];
  mD1[[2]]/mD2[[2]]]
```

We are now ready to give the algorithm for the computation of the Weil pairing $e_m(P, Q)$ of two m-torsion points. The module `WeilPairing[P,Q,m]` first picks points T and U, such that the resulting functions f_A and f_B are defined at $Q + U$, U and $P + T$, T respectively, and then computes $e_m(P, S) = \frac{f_A(Q+U)\, f_B(T)}{f_A(U)\, f_B(P+T)}$.

```
WeilPairing[P_, Q_, m_] := Module[

  (* Input: two m-torsion points P ≠ O and Q ≠ O, and integer m;
   Output: The outcome of the Weil pairing e_m (P,Q). *)

  {T, U, PT, QU, fA, fB, x, y, DenfA, DenfB},
  While[{T, U, PT, QU} = PickTUGroundField[P, Q];
   fA[x_, y_] = ComputeFunctionWeil[P, T, m, x, y];
   fB[x_, y_] = ComputeFunctionWeil[Q, U, m, x, y];
   DenfA[x_, y_] = Denominator[fA[x, y]]; DenfB[x_, y_] = Denominator[fB[x, y]];
   DenfA[U[[1]], U[[2]]] == 0 ∨ DenfA[QU[[1]], QU[[2]]] == 0 ∨
    DenfB[T[[1]], T[[2]]] == 0 ∨ DenfB[PT[[1]], PT[[2]]] == 0];
  (fA[QU[[1]], QU[[2]]] fB[T[[1]], T[[2]]]) /
   (fA[U[[1]], U[[2]]] fB[PT[[1]], PT[[2]]])]
```

We next deal with the Tate pairing. On input of an m-torsion point P, the value m, and variables x, y, the following module computes the function $g(x, y)$ such that $\mathrm{div}(g) = m(P) - m(O)$. Note that `ComputeFunctionWeil[P,O,m,x,y] = ComputeFunctionTate[P,m,x,y]`.

```
ComputeFunctionTate[P_, m_, x_, y_] := Module[

  (* Input: an m-torsion point P,
   the value m, and the variables x and y for the function;
   Output: function g such that div (g) = m (P)-m (O). *)

  {mD1},
  mD1 = DivisorMultiplication[{P, fld[{1, 0}]}, m, x, y];
  mD1[[2]]]
```

Now the module `TatePairing[P,Q,m]` proceeds as follows. After computing a function $g(x, y)$ with $\mathrm{div}(g) = m(P) - m(O)$, the algorithm picks a point Q' such that g is defined at $Q + Q'$ and Q', and computes $\langle P, Q\rangle_m = g(Q + Q')/g(Q')$.

```
TatePairing[P_, Q_, m_] := Module[

  (* Input: An m-torsion point P, a point Q, and the value m;
   Output: the outcome of the Tate pairing ⟨P,Q⟩_m. *)

  {S, QP, g, Deng, Numg, x, y},
  g[x_, y_] = ComputeFunctionTate[P, m, x, y];
  Deng[x_, y_] = Denominator[g[x, y]];
  Numg[x_, y_] = Numerator[g[x, y]];
  While[QP = RandomPointGroundField; S = PointAddition[Q, QP];
   S == O ∨ Deng[S[[1]], S[[2]]] == 0 ∨ Numg[QP[[1]], QP[[2]]] == 0 ∨
    Numg[S[[1]], S[[2]]] == 0 ∨ Deng[QP[[1]], QP[[2]]] == 0];
  (Numg[S[[1]], S[[2]]] Deng[QP[[1]], QP[[2]]]) /
   (Deng[S[[1]], S[[2]]] Numg[QP[[1]], QP[[2]]])]
```

The outcome of the Tate pairing is an equivalence class in $\mathbb{F}_{p^2}^* / (\mathbb{F}_{p^2}^*)^m$. Whenever a unique value is required, one can raise the outcome to the power $(p^2 - 1)/m$, to eliminate m-th powers.

```
TatePairingUnique[P_, Q_, m_] := TatePairing[P, Q, m]^((p^2-1)/m)
```

Time for some numerical examples. We apply the Weil and the Tate pairing to the elliptic curve $E_1/\mathbb{F}_{11} : y^2 = x^3 + 3x$.

```
Clear[fld, p, a, b];
a = 3; b = 0; p = 11;
fld = GF[p, {1, 0, 1}];
PrimeQ[p]
Mod[p, 4] == 3
FactorInteger[p + 1]
```

```
True
```

```
True
```

```
{{2, 2}, {3, 1}}
```

The point $P = (3, 5)$ has order 3:

```
P = Pt[3, 0, 5, 0]
P2 = PointMultiplication[P, 2]
P3 = PointMultiplication[P, 3]
```

```
{{3, 0}11, {5, 0}11}
```

```
{{3, 0}11, {6, 0}11}
```

```
O
```

When we apply the Weil pairing to two dependent m-torsion points, the outcome is always 1:

```
WeilPairing[P, P, 3]
WeilPairing[P, P2, 3]
WeilPairing[P, P3, 3]
```

$\{1, 0\}_{11}$

$\{1, 0\}_{11}$

$\{1, 0\}_{11}$

We apply the distortion map to P to find an independent 3-torsion point:

```
Q = DistortionMap[P]
```

$\{\{8, 0\}_{11}, \{0, 5\}_{11}\}$

We now compute the Weil pairing $e_3(P, Q)$, which is indeed a third root of unity:

```
e = WeilPairing[P, Q, 3]
e^3
```

$\{5, 8\}_{11}$

$\{1, 0\}_{11}$

Similarly, we can compute the Tate pairing $\langle P, Q\rangle_3$. The outcome of this pairing is an equivalence class, so the evaluation does not always give the same result. However, if we eliminate the third powers, as is done in the module `TatePairingUnique[P,Q,m]`, we get the unique representative for that particular equivalence class:

```
TatePairing[P, Q, 3]
TatePairing[P, Q, 3]
```

$\{7, 6\}_{11}$

$\{5, 9\}_{11}$

```
TatePairingUnique[P, Q, 3]
TatePairingUnique[P, Q, 3]
```

$\{5, 3\}_{11}$

$\{5, 3\}_{11}$

For the Tate pairing, the second point is not required to be an m-torsion point. In fact, the Tate pairing can be evaluated for any point on the curve. If $R \in m\,E(\mathbb{F}_{p^2})$, then the outcome of the pairing $\langle P, R\rangle_m$ is the equivalence class representing 1.

```
arb = PointAddition[RandomPointGroundField,
  DistortionMap[RandomPointGroundField]]
R = PointMultiplication[arb, 3]
TatePairing[P, R, 3]
TatePairingUnique[P, R, 3]
```

$\{\{3, 4\}_{11}, \{5, 10\}_{11}\}$

$\{\{5, 0\}_{11}, \{0, 6\}_{11}\}$

$\{2, 2\}_{11}$

$\{1, 0\}_{11}$

Furthermore, for $R \in m\,E(\mathbb{F}_{p^2})$ the result of the pairing $\langle P, Q+R\rangle_m$ is the equivalence class of $\langle P, Q\rangle_m$.

```
TatePairing[P, PointAddition[Q, R], 3]
TatePairingUnique[P, PointAddition[Q, R], 3]
```

$\{8, 7\}_{11}$

$\{5, 3\}_{11}$

Our next example is the curve $E_2/\mathbb{F}_{131} : y^2 = x^3 + 31$.

```
Clear[fld, p, a, b, P, Q];
a = 0; b = 31; p = 131;
fld = GF[p, {1, 0, 1}];
PrimeQ[p]
Mod[p, 4] == 3
Mod[p, 3] == 2
FactorInteger[p + 1]
```

True

True

True

$\{\{2, 2\}, \{3, 1\}, \{11, 1\}\}$

Consider the point $P = (78, 25)$, which has order 11:

```
P = Pt[78, 0, 25, 0]
PointMultiplication[P, 11]
```

$\{\{78, 0\}_{131}, \{25, 0\}_{131}\}$

O

Again, the Weil pairing of two dependent m-torsion points is trivial.

```
P7 = PointMultiplication[P, 7]
WeilPairing[P, P7, 11]
```

$\{\{60, 0\}_{131}, \{76, 0\}_{131}\}$

$\{1, 0\}_{131}$

So we apply the distortion map to find an independent 11-torsion point, compute the Weil pairing $e_{11}(P, Q)$ and show that the result is an 11-th root of unity

```
Q = DistortionMap[P]
e = WeilPairing[P, Q, 11]
e^11
```

$\{\{92, 41\}_{131}, \{25, 0\}_{131}\}$

$\{126, 32\}_{131}$

$\{1, 0\}_{131}$

For the Tate pairing, we perform the same computations as in the previous example.

```
TatePairing[P, Q, 11]
TatePairing[P, Q, 11]
```

$\{84, 22\}_{131}$

$\{120, 88\}_{131}$

```
TatePairingUnique[P, Q, 11]
TatePairingUnique[P, Q, 11]
```

$\{126, 99\}_{131}$

$\{126, 99\}_{131}$

```
arb = PointAddition[RandomPointGroundField,
  DistortionMap[RandomPointGroundField]]
R = PointMultiplication[arb, 11]
TatePairing[P, R, 11]
TatePairingUnique[P, R, 11]
```

$\{\{79, 28\}_{131}, \{114, 22\}_{131}\}$

$\{\{95, 123\}_{131}, \{90, 28\}_{131}\}$

$\{32, 99\}_{131}$

$\{1, 0\}_{131}$

```
TatePairing[P, PointAddition[Q, R], 11]
TatePairingUnique[P, PointAddition[Q, R], 11]
```

$\{109, 78\}_{131}$

$\{126, 99\}_{131}$