Bilinear Pairings in Cryptography

Master thesis
#603

by

Dennis Meffert
<d.meffert@student.science.ru.nl>

at

Radboud Universiteit Nijmegen
Computing Science Department
Toernooiveld 1
6525 ED Nijmegen
The Netherlands

supervised by

dr. W. Bosma
dr. D.C. van Leijenhorst

May 24, 2009

Abstract

This thesis consists of two sections. The first section is devoted to the mathematics of pairings. Elliptic and hyperelliptic curves over finite fields are described, as well as rational functions and divisors on such curves. The Weil pairing on elliptic curves, and the Tate pairing on elliptic and hyperelliptic curves are explained with a computational approach in mind.

The second section introduces bilinear pairings in a cryptographic setting. First their relation to the discrete logarithm problem in finite fields is investigated. Then we illustrate how bilinear pairings give rise to new mathematical problems which can be used as a base for secure cryptosystems. We conclude with a description of several cryptosystems based on bilinear pairings.

Acknowledgements

First of all I would like to thank my supervisors, Wieb Bosma and Dick van Leijenhorst, for their guidance during the writing of this thesis and for their helpful comments on the text. Thanks is also due to Ken Madlener for the many fruitful discussions we have had about cryptography. On a personal note I would like to thank my parents for their support.

Contents

Chapter 1

Introduction

Bilinear pairings were originally brought to the cryptographic community by Menezes, Okamoto and Vanstone with their MOV attack [38]. As it turns out pairings can be used to transport the discrete logarithm problem on a certain class of elliptic curves over a finite field to the discrete logarithm problem on a smaller finite field, where a sub-exponential index calculus attack can be used to attack the problem. However, it was the publication of an identity based encryption scheme (by Boneh and Franklin [10]) based on bilinear pairings that triggered a real upsurge in the popularity of pairings among cryptographers. Following Boneh and Franklin, a lot of cryptosystems based on pairings have been proposed which would be hard to construct using more conventional cryptographic primitives. At this moment, pairing-based cryptography is a highly active field of research, with several hundreds of publications.

The goal of this thesis is to provide an overview of the most active topics of research in pairings. The material is presented in two parts. In the first part we will look at the mathematical foundations of bilinear pairings. In the second part we will describe how pairings can be applied in cryptographic settings. The reader is assumed to have knowledge of basic abstract algebra, but the theory of elliptic and hyperelliptic curves necessary to understand pairings is covered in this thesis.

1.1 Thesis outline

The outline of the thesis is as follows.

- *Chapter 2.* In this chapter we provide a short introduction to elliptic and hyperelliptic curves, as well as some concepts from algebraic geometry such as divisors and rational functions.
- *Chapter 3.* Here we introduce bilinear pairings mathematically. The Weil pairing is defined for elliptic curves and the Tate pairing is defined in both the elliptic and the hyperelliptic curve setting.
- *Chapter 4.* This chapter describes the discrete logarithm problem for (hyper)elliptic curve and how bilinear pairings are related to this.
- *Chapter 5.* Here the mathematical problems that arise from bilinear pairings are outlined.

- *Chapter 6.* In this chapter we give an overview of several cryptographic protocols that use bilinear pairings.
- *Chapter 7.* This chapter concludes the thesis with some final remarks.

Chapter 2

Elliptic and Hyperelliptic Curves

In this chapter we first define elliptic curves, describe the group structure on the set of points on an elliptic curve and look at some important theorems. Then we generalize to hyperelliptic curves and see how to define a similar group structure on the Jacobian. We also describe rational functions on curves and divisor theory. Most of the results in this chapter come from [5], [4], [48], [37] and [39].

2.1 Projective Space

Let K be a field. Consider the equivalence relation $\sim$ defined on $K^3 \setminus \{(0,0,0)\}$ given by

$$(X, Y, Z) \sim (\lambda X, \lambda Y, \lambda Z)$$

for every $\lambda \in K^*$. This means two points are considered equal if one is a scalar multiple of the other. We denote the equivalence class of (X, Y, Z) by $[X : Y : Z]$.

Definition 2.1. The set of all equivalence classes is called the *projective plane* over K

$$\mathbb{P}^2_K := \{\, [X : Y : Z] \;\mid\; X, Y, Z \in K \text{ but not } X = Y = Z = 0 \,\}.$$

The set of equivalence classes where $Z \neq 0$ is a copy of K^2 because $[X : Y : Z]$ is equivalent to $[\frac{X}{Z} : \frac{Y}{Z} : 1]$. If $Z = 0$, then either $X \neq 0$ or $Y \neq 0$. If $X \neq 0$ then $[X : Y : 0]$ is equivalent to $[1 : \frac{Y}{X} : 0]$. Now suppose $X = 0$. Then $Y \neq 0$ and $[0 : Y : 0]$ is equivalent to $[0 : 1 : 0]$. In other words, the set of equivalence classes where $Z = 0$ is the union of a copy of K with a single point. If you think of the projective plane as the set of lines passing through the origin in K^3, the latter set of equivalence classes is usually referred to as the *line at infinity*.

2.2 Algebraic Curves

Definition 2.2. Let K be a field. An *affine algebraic curve* (or simply *affine curve*) over K is an equation $f(x, y) = 0$, where f is a polynomial in x and y with coefficients in K. Solutions of this equation are *points* on the curve. A *K-rational point* is a point (x, y) on the curve where x and y are elements of K.

We will make extensive use of the following notation. If $\mathcal{C} : f(x, y) = 0$ is an affine curve, then $\mathcal{C}$ refers to the equation $f(x, y) = 0$. We write $\mathcal{C}/K$ to indicate that $\mathcal{C}$ is defined over the field K. We denote the set of K-rational points with $\mathcal{C}(K)$.

Definition 2.3. A *singular point* or *smooth point* on the curve is a point (x, y) on the curve where both partial derivatives of f vanish. In other words, a point (x, y) where $f(x, y) = \frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = 0$. If a curve has no singular points, then we say the curve is *non-singular*.

Definition 2.4. A polynomial is *homogeneous* of *degree* d if every term has degree d in the variables. For example, $xy^2 - 2x^2z + 8z^3$ is homogeneous of degree 3 in the variables x, y, z. We can make any polynomial in two variables x, y homogeneous in three variables x, y, z by multiplying each term with powers of z. This is known as *homogenization*. For example, homogenizing $xy + 4x - y^3 + 1$ yields $xyz + 4xz^2 + y^3 + z^3$.

Let K be a field and $\mathcal{C} : f(x, y) = 0$ an affine curve defined over K and let $F(x, y, z)$ be the homogenization of f. Suppose F is homogeneous of degree d. The polynomial F cannot be evaluated in a projective point $[x_0 : y_0 : z_0] \in \mathbb{P}^2_K$ since $[x_0 : y_0 : z_0] = [\lambda x_0 : \lambda y_0 : \lambda z_0]$ but $F(x_0, y_0, z_0)$ does not necessarily equal $F(\lambda x_0, \lambda y_0, \lambda z_0)$. In fact

$$F(\lambda x_0, \lambda y_0, \lambda z_0) = \lambda^d F(x_0, y_0, z_0).$$

However, this equality also shows it does make sense to verify if F is zero when evaluated in (x_0, y_0, z_0). Thus, the set of projective points on the curve defined by the equation $F(x, y, z) = 0$:

$$\{[x : y : z] \in \mathbb{P}^2_K \big| F(x, y, z) = 0\}$$

is well defined. This way, given an affine curve we can obtain a *projective curve* by homogenizing the corresponding polynomial. The projective curves we are going to use always have one single point at infinity, so for convenience we will be working with affine coordinates from now on and simply add the point at infinity to the set of points on the curve. We go back to the projective model when we need to investigate the exact behaviour of certain functions in the point at infinity.

2.3 Rational Functions

Let $\mathcal{C}$ be a curve defined over a finite field $K = \mathbb{F}_q$ by the following equation

$$f_{\mathcal{C}}(x, y) = 0.$$

When we are interested in the way polynomial functions behave on the points of $\mathcal{C}$, it makes sense to consider functions g_1 and g_2 "equivalent" if and only if $g_1(x, y) = g_2(x, y)$ for every $(x, y) \in \mathcal{C}(K)$.

Definition 2.5. Following this line of thought, we define the coordinate ring

$$K[\mathcal{C}] := K[x, y]/\langle f_{\mathcal{C}} \rangle$$

where $\langle f_{\mathcal{C}} \rangle$ is the ideal generated by the polynomial $f_{\mathcal{C}}(x, y)$. If $f_{\mathcal{C}}$ is irreducible over K (which will be the case for the curves we are interested in) then $\langle f_{\mathcal{C}} \rangle$ is a prime ideal and hence $K[\mathcal{C}]$ is an integral domain. Consequently, there is a corresponding field of fractions, which is called the *field of rational functions on* $\mathcal{C}$, notated $K(\mathcal{C})$.

Zeroes and Poles

In this section we introduce zeroes and poles of rational functions and their multiplicity. Several facts are stated without proof or even some of the intuition behind it, but a rigorous treatment of these concepts can be found in [31] and [43].

Definition 2.6. A rational function $f \in K(\mathcal{C})$ is *regular* at a point $P \in \mathcal{C}(K)$ if it can be represented as a quotient g/h, where $g, h \in K[\mathcal{C}]$ and $h(P) \neq 0$. A rational function f has a *pole* at the point P if it is not regular at P. It has a *zero* at the point P if it is regular at P and if $g(P) = 0$, so $f(P) = 0$.

The set of functions that are regular in a point $P \in \mathcal{C}(K)$ is a subring of $K(\mathcal{C})$ and is known as the *local ring of* $\mathcal{C}$ *at* P. It is usually denoted $\mathcal{O}_{\mathcal{C},P}$. The local ring $\mathcal{O}_{\mathcal{C},P}$ is also obtained by localizing the ring $K[\mathcal{C}]$ at $\bar{M}_P$, where $\bar{M}_P = \{f \in K[\mathcal{C}] \mid f(P) = 0\}$, which is a maximal ideal. Since $\mathcal{O}_{\mathcal{C},P}$ is a local ring, it has a unique maximal ideal, given by $m_P = \{f \in \mathcal{O}_{\mathcal{C},P} \mid f(P) = 0\}$. A generator for this ideal, traditionally denoted π, is said to be a *uniformizing parameter for* P.

Definition 2.7. For a point $P \in \mathcal{C}(K)$, every regular function $f \in \mathcal{O}_{\mathcal{C},P}$ can be written as $f = \pi^n u$, where π is a uniformizing parameter for P, n is an integer and u is a unit and $u(P) \neq 0, \infty$. This value n is the *multiplicity* or *order* of f at P, and it does not depend on the choice of π. It is also given by $n = \max\{k \in \mathbb{Z} \mid f(P) \in m_p^k\}$. The multiplicity of f at P is denoted $\mathrm{ord}_P(f)$. If $\mathrm{ord}_P(f) > 0$ then f has a zero of order n at P. If $\mathrm{ord}_P(f) < 0$ then f has a pole of order n at P. We can extend this concept to all rational functions. If $f \in K(\mathcal{C})$ and $f \neq 0$ then $f = \frac{g}{h}$ for some $g, h \in \mathcal{O}_{\mathcal{C},P}$. The multiplicity of f in a point P is then defined to be $\mathrm{ord}_P(f) = \mathrm{ord}_P(g) - \mathrm{ord}_P(h)$. If $f = 0$ then $\mathrm{ord}_P(f) = \infty$.

Example 2.1. Let $\mathcal{C} : y^2 - x^3 + x = 0$ be an affine curve over $\mathbb{R}$ and $P = (0, 0) \in \mathcal{C}(\mathbb{R})$. A uniformizing parameter for P is y. Let $f(x, y) = x \in \mathbb{R}(\mathcal{C})$. Then f has a zero at P and $\mathrm{ord}_P(f) = 2$, because $x = (x^3 + x)\frac{x}{x^3+x} = y^2\frac{1}{x^2+1}$.

These definitions do not provide any real insight in how to actually calculate multiplicities or uniformizing parameters, but in a later section we will give explicit formulae for both of these in the case of (hyper)elliptic curves.

2.4 Divisors

This section describes divisors on curves [48]. Divisors will be used in the construction of a group law on hyperelliptic curves, which is covered in the next section. Moreover, they are essential in the definition of bilinear pairings in the next chapter.

A *divisor* $\mathcal{A}$ is a formal sum of points on $\mathcal{C}$

$$\mathcal{A} := \sum_{P \in \mathcal{C}} n_P(P)$$

where $n_P \in \mathbb{Z}$ and $n_P \neq 0$ for only a finite number of points. The set of divisors of an affine curve forms a free abelian group generated by the points on $\mathcal{C}$, known as the *divisor group* of $\mathcal{C}$ which is notated $\mathrm{Div}(\mathcal{C})$.

Degree of a divisor

The *degree* of a divisor $\mathcal{A}$ is given by

$$\deg \mathcal{A} := \sum_{P \in \mathcal{C})} n_P.$$

The set of divisors of degree zero is a subgroup of $\mathrm{Div}(\mathcal{C})$ and is defined as follows:

$$\mathrm{Div}^0(\mathcal{C}) := \{\, \mathcal{A} \in \mathrm{Div}(\mathcal{C}) \mid \deg \mathcal{A} = 0 \,\}.$$

Divisor of a function

We can associate a divisor with a rational function of a curve by counting the multiplicities of the zeros and poles of the points in the function. Let $f \in \overline{K}(\mathcal{C})^*$. The divisor of the function f is defined by

$$(f) := \sum_{P \in \mathcal{C}} \mathrm{ord}_P(f)(P)$$

It follows that $(fg) = (f) + (g)$ and $(f/g) = (f) - (g)$ for every $f, g \in \overline{K}(\mathcal{C})^*$

Theorem 2.1. *Let $f \in \overline{K}(\mathcal{C})^*$. Then $\deg(f) = 0$.*

A proof of this theorem can be found in [31].

Evaluating rational functions in a divisor

Let $f \in K(\mathcal{C})$ and $\mathcal{A} = \sum_{P \in \mathcal{C}(K)} n_P(P) \in \mathrm{Div}(\mathcal{C})$. We define the *evaluation* of f in $\mathcal{A}$ as follows

$$f(\mathcal{A}) := \prod_{P \in \mathcal{C}(K)} f(P)^{n_P}$$

Principal divisors

A divisor $\mathcal{A} \in \mathrm{Div}(\mathcal{C})$ is called a *principal divisor* if there is a non-zero rational function $f \in K(\mathcal{C})^*$ such that $\mathcal{A} = (f)$. It is easy to show that the set of principal divisors forms a subgroup of $\mathrm{Div}(\mathcal{C})$. Construct a mapping

$$\begin{array}{rcl} K(\mathcal{C})^* & \longrightarrow & \mathrm{Div}(\mathcal{C}) \\ f & \longmapsto & (f) \end{array}$$

This is clearly a group homomorphism, as $(fg) = (f) + (g)$ and $(\frac{1}{f}) = -(f)$. It follows that the image of this mapping, which is the set of principal divisors, is a subgroup of $\mathrm{Div}(\mathcal{C})^0$. We denote this subgroup $\mathrm{Prin}(\mathcal{C})$.

Equivalence of divisors

We can define an equivalence relation on divisors by considering divisors $\mathcal{A}$ and $\mathcal{B}$ *linearly equivalent* if their difference is a principal divisor:

$$\mathcal{A} \sim \mathcal{B} \iff \mathcal{A} - \mathcal{B} = (f) \text{ for some } f \in K(\mathcal{C})^*.$$

It follows that $\mathcal{A}$ is a principal divisor if and only if $\mathcal{A} \sim (0)$.

The *Picard group* or *divisor class group* is defined as follows:

$$\mathrm{Pic}(\mathcal{C}) := \mathrm{Div}(\mathcal{C})/\mathrm{Prin}(\mathcal{C}).$$

Thus, two divisors are in the same divisor class if they are equivalent up to addition with a principal divisor.

Support of a divisor

Let $\mathcal{A} = \sum_{P \in \mathcal{C}} n_P(P)$ be a divisor. The *support* of $\mathcal{A}$ is the set of points on the curve appearing in the sum. Formally:

$$\mathrm{supp}(\mathcal{A}) := \{\, P \in \mathcal{C} \mid n_P \neq 0 \,\}.$$

Effective divisors

A divisor $\mathcal{A} = \sum_{P \in \mathcal{C}} n_P(P)$ is said to be *effective* if $n_P \geq 0$ for all $P \in \mathcal{C}$.

2.5 Elliptic Curves

In this section we will define elliptic curves. These results come from [**?**].

2.5.1 Definition

Let K be an algebraically closed field, and consider the following equation defined over $\mathbb{P}^2_K$, known as the *projective Weierstrass equation*

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

with coefficients $a_1, a_2, a_3, a_4, a_6 \in K$.
In order to find the solutions of this equation in $\mathbb{P}^2_K$ we first take a look at the points where $Z = 0$ (the line at infinity). Substituting $Z = 0$ in the equation yields $X^3 = 0$, which means the curve only intersects the line at infinity in the point $[0 : 1 : 0]$. We will refer to this point as *the point at infinity*, denoted ∞. Now suppose $Z \neq 0$. Any solution is of the form $[\frac{X}{Z} : \frac{Y}{Z} : 1]$, which means we might as well use only two variables to describe such a point. First we divide the Weierstrass equation by Z^3

$$\frac{Y^2}{Z^2} + a_1\frac{XY}{Z^2} + a_3\frac{Y}{Z} = \frac{X^3}{Z^3} + a_2\frac{X^2}{Z^2} + a_4\frac{X}{Z} + a_6.$$

Then we substitute $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$ to obtain the *affine Weierstrass equation*

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

This process is known as *dehomogenization*. Using the affine form we can refer to points satisfying this equation without having to use projective coordinates. A point satisfying the equation is now either the point at infinity ∞ or a point of the form (x, y) with $x, y \in K$.

If the curve given by an affine Weierstrass equation

$$\mathcal{E} : y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0$$

is non-singular, then $\mathcal{E}$ is an *elliptic curve*.

The notation described in the previous section on algebraic curves also applies here. If $\mathcal{E}$ is an elliptic curve we use the notation $\mathcal{E}/K$ to indicate that it is defined over the field K. When we write $\mathcal{E}(K)$ we refer to the set of solutions (x, y) with $x, y \in K$ together with the point of infinity ∞.

If $\text{char}(K) \neq 2, 3$ the Weierstrass equation can be simplified through a change of variables into an equation of the form:

$$y^2 = x^3 + ax + b$$

When $\text{char}(K) = 2$ not all terms can be eliminated resulting in an equation in one of the following forms:

$$y^2 + xy = x^3 + ax^2 + b \quad \text{or} \quad y^2 + xy = x^3 + ax + b$$

Finally, if $\text{char}(K) = 3$, the simplified equation is in the following form:

$$y^2 = x^3 + ax^2 + b \quad \text{or} \quad y^2 = x^3 + ax + b$$

The resulting form in the cases where $\text{char}(K) = 2$ or 3 depends on a quantity known as the *j-invariant* which we will not define here. For a complete overview of how to do these substitutions we refer to [48].

2.5.2 Computing the group law

Through a process known as *chord-and-tangent composition*, points on an elliptic curve can be added, giving again another point on the curve. This method is best illustrated geometrically over $\mathbb{R}$.

Adding two distinct points P and Q on this curve is done by "drawing" a line through them. This line will intersect the curve in a third point $-R$, which can be reflected in the x-axis to get the point R. Addition on the curve is defined by $P + Q = R$. If the line through P and Q is vertical (i.e. if $Q = -P$) it will intersect the curve at infinity and thus $P + (-P) = \infty$. Adding a point P to itself is done in a similar fashion. Instead of drawing a line through two distinct points, we now draw the line tangent to the point P. This line again intersects the curve in a third point $-R$ which is reflected to give $P + P = R$, or the line intersects the curve at infinity in which case $P + P = \infty$. The process of adding two points is visualized in figure 2.1 on an elliptic curve $\mathcal{E}$ defined over $\mathbb{R}$.

This can also be described algebraically. Reflecting a point $P = (x, y)$ is done by simply inverting the y coordinate: $-P = (x, -y)$. Addition of two distinct points works as follows. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be distinct points such that $P \neq -Q$. The slope of the line through P and Q is given by $\lambda = \frac{y_1 - y_2}{x_1 - x_2}$. Now we can calculate the coordinates of $P + Q = R = (x_3, y_3)$ by reflecting third point of intersection:

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = -y_1 + \lambda(x_1 - x_3)$$

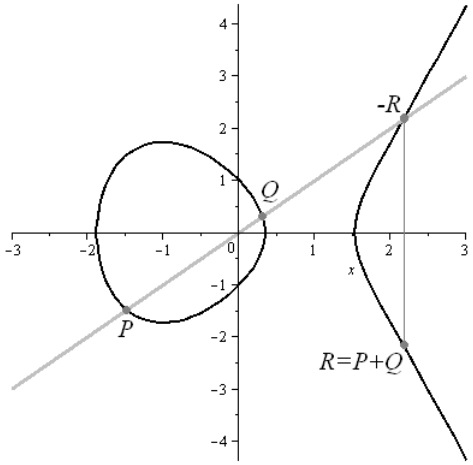

Figure 2.1: Adding two points on the elliptic curve $y^2 = x^3 - 3x + 1$.

Adding $P = (x_1, y_1)$ to itself (also known as *doubling the point*) is done as follows. If $y_1 = 0$ then $2P = \infty$. If $y_1 \neq 0$ then we first need to calculate the slope of the line tangent to P, which is given by $\lambda = \frac{3x_1^2+a}{2y_1}$. The coordinates of $2P = R = (x_3, y_3)$, the reflected third point of intersection are in this case given by:

$$x_3 = \lambda^2 - 2x_1$$
$$y_3 = -y_1 + \lambda(x_1 - x_3)$$

When working with a curve defined over a field of characteristic 2 or 3 these formulas need to be adjusted.

Point addition as defined above naturally provides the set of points on an elliptic curve with the structure of an Abelian group with identity element ∞. For a proof we refer to [48], [34].

2.5.3 Group structure

Group order

For curves over finite fields one can wonder how many rational points it contains. Obviously for every $x \in \mathbb{F}_q$ there are at most two values of $y \in \mathbb{F}_q$ for which (x, y) is a point on the curve, so including the point at infinity there can never be more than $2q + 1$ points on the curve. However, we can give a tighter bound based on the fact that half of the non-zero elements in a finite field are squares. For every $x \in \mathbb{F}_q$ we know that (x, y) is either a solution (in which case $(x, -y)$ is also a solution, unless $y = -y$) or (x, y) is not a solution. Since roughly half of the field elements are squares, both have an equal probability of occuring. This leads us to believe that together with the point at infinity there are $q + 1 + \varepsilon$ points on the curve, where ε is supposed to be relatively small. This turns out to be correct and is formulated more precisely in a famous theorem by Hasse:

Theorem 2.2 (Hasse)**.** *Let $\mathcal{E}$ be an elliptic curve over a finite field $\mathbb{F}_q$. Then the following equation holds:*

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

A proof is given in [48].

The value $\#\mathcal{E}(\mathbb{F}_q) - (q+1)$ is known as the *trace of Frobenius* and is usually denoted t.

Torsion subgroups

Like in every additive group, the addition of points on the curve induces a scalar multiplication of points. For $n \in \mathbb{Z}$ and P a point on an elliptic curve, we define:

$$nP = \begin{cases} \underbrace{P + P + \ldots + P}_{n \text{ times}}, & \text{if } n > 0 \\ \underbrace{(-P) + (-P) + \ldots + (-P)}_{n \text{ times}}, & \text{if } n < 0 \\ \infty, & \text{if } n = 0 \end{cases}$$

The *order* of a point P is the smallest positive integer n such that $nP = \infty$. If the order of a point P divides n (i.e. if $nP = \infty$), then P is called an *n-torsion point*. The set of K-rational n-torsion points forms a subgroup, the *n-torsion group*, denoted $\mathcal{E}(K)[n]$. The following theorem gives us some valuable information about the structure of an n-torsion group.

Theorem 2.3. *Let $\mathcal{E}/\mathbb{F}_{p^m}$ be an elliptic curve and let $n \in \mathbb{Z}$ be a prime coprime to p. Then the order of $\mathcal{E}(\mathbb{F}_{p^m})[n]$ is n^2 and $\mathcal{E}(\mathbb{F}_{p^m})[n] \cong \mathbb{Z}/n\mathbb{Z} \oplus \mathbb{Z}/n\mathbb{Z}$.*

For a special class of curves the n-torsion group is trivial when n is a power of the characteristic of the field over which the curve is defined. We will first define this class of curves. Let $\mathcal{E}/K$ be an elliptic curve, where $K = \mathbb{F}_{p^m}$. If $p \,|\, t$, where t is the trace of Frobenius of $\mathcal{E}$, then $\mathcal{E}$ is called *supersingular*.

2.5.4 Rational functions on elliptic curves

The following is explained in detail in [39]. Suppose $\mathcal{E}/K : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ is an elliptic curve. Observe that every polynomial $f(x, y) \in \overline{K}[\mathcal{E}]$ can be written in the form $a(x) + b(x)y$, by rearranging terms and replacing any occurrence of y^2 by $x^3 + a_2x^2 + a_4x + a_6 - a_1xy - a_3y$. This form can be used to define the degree of f, which we will be needed in determining the orders of zeroes and poles of f as well as the behaviour of f in ∞.

Definition 2.8. Let $f(x, y) = a(x) + b(x)y \neq 0$ be polynomial function in $\overline{K}[\mathcal{E}]$. Then the *degree* of f is defined to be

$$\deg(f) = \max\{2\deg_x(a), 3 + 2\deg_x(b)\}$$

where $\deg_x$ denotes the usual degree of a polynomial in x, taking $\deg_x(0) = \infty$.

The behaviour of rational functions on $\mathcal{E}$ in the point at infinity is defined as follows.

Definition 2.9. Let $f = g/h \in \overline{K}(\mathcal{E})$. If $\deg(g) < \deg(h)$ then we define $f(\infty) = 0$. If $\deg(g) > \deg(h)$ we say f is not defined at ∞. If $\deg(g) = \deg(h)$ then $f(\infty)$ is defined to be the ratio of the leading coefficients of g and h.

A uniformizing parameter π for a point $P \in \mathcal{E}(K)$ can be found as follows. If $P = \infty$ then choose $\pi = x/y$. If $P = (x_P, 0)$ then choose $\pi = y$. If $P = (x_P, y_P)$, $y_P \neq 0$ then choose $\pi = x - x_P$. The order of a rational function f at P can now be determined by writing $f = \pi^n u$ with $n \in \mathbb{Z}$ and u a unit, $u(P) \neq 0, \infty$, as described earlier.

2.6 Hyperelliptic Curves

In this section we define hyperelliptic curves. These results come from [39].

2.6.1 Definition

A hyperelliptic curve $\mathcal{C}$ over a field K is a curve given by the equation:

$$y^2 + h(x)y = f(x)$$

where f is a monic polynomial of degree $2g+1$ and h is a polynomial of degree $\leq g$, for a certain $g \geq 1$. This number g is called the *genus* of the curve. We require the curve to be non-singular, which means the partial derivatives given by

$$\frac{\partial \mathcal{C}}{\partial x} = h'(x)y - f'(x) \text{ and } \frac{\partial \mathcal{C}}{\partial y} = 2y + h(x)$$

should not vanish simultaneously. The curves with genus 1 are exactly the elliptic curves, which means hyperelliptic curves are a generalization of elliptic curves. As a consequence, the theorems and definitions that follow in the next sections also hold for elliptic curves.

As with elliptic curves, we use the notation $\mathcal{C}/K$ to indicate that $\mathcal{C}$ is defined over a field K. We will write $\mathcal{C}$ to refer to the set of solutions together with a point of infinity ∞ and use $\mathcal{C}(K)$ to denote the set of solutions with coordinates in K (i.e. the K-rational points). If $\text{char}(K) = 2$, then $h(x) \neq 0$. If $\text{char}(K) \neq 2$, then we can assume $h(x) = 0$ without loss of generality. If $h(x) = 0$ then $f(x)$ has no repeated roots in $\overline{K}$. For a proof of these facts we refer to [39].

If $P = (x, y)$ is a point on $\mathcal{C}$, then $\iota(P) = (x, -y - h(x))$ is also a point on $\mathcal{C}$, sometimes denoted $\tilde{P}$ and referred to as the *opposite* of P. The map ι is called the *hyperelliptic involution*, and a point for which $\iota(P) = P$ is called a *ramification point* or *special point*. If $\iota(P) \neq P$ then P is said to be an *ordinary point*.

In the section on elliptic curves we gave a bound on the number of points on the curve by stating Hasse's theorem. This theorem can be reformulated so it also applies to hyperelliptic curves.

Theorem 2.4 (Hasse-Weil)**.** *Let $\mathcal{C}$ be a hyperelliptic curve of genus g over a finite field $\mathbb{F}_q$. Then the following equation holds:*

$$|\#\mathcal{C}(\mathbb{F}_q) - (q+1)| \leq 2g\sqrt{q}.$$

Unlike in elliptic curves, the points on a genus > 1 curve do not have the structure of a group. However, in what follows we will see there is a group law defined over another object that we can associate with the curve.

2.6.2 Jacobian

The *(degree 0 part of the) divisor class group* is defined as the set of degree zero divisors modulo the principal divisors:

$$\text{Pic}^0(\mathcal{C}) := \text{Div}^0(\mathcal{C})/\text{Prin}(\mathcal{C}).$$

If $\mathcal{A}$ is a degree zero divisor then we denote the divisor class of $\mathcal{A}$ by $\overline{\mathcal{A}}$. The zero degree divisor class group is isomorphic to an object known as the *Jacobian variety* on $\mathcal{C}$, which is why the divisor class group is sometimes simply referred to as the Jacobian and denoted $\mathrm{Jac}(\mathcal{C})$.

A divisor $\mathcal{A}$ is said to be *semi-reduced* if it is of the form

$$\mathcal{A} = \sum_{P \in \mathcal{C}} n_P(P) - (\sum_{P \in \mathcal{C}} n_P)(\infty)$$

and if $\mathcal{A}$ satisfies for all P in the sum:

1. $P \neq \infty$
2. $n_P \geq 0$
3. if $P = \iota(P)$ and $n_P > 0$ then $n_P = 1$
4. if $P \neq \iota(P)$ and $n_P > 0$ then $n_{\tilde{P}} = 0$

If in addition $\sum_{P \in \mathcal{C}} nP \leq g$, where g is the genus of the curve, then $\mathcal{A}$ is said to be *reduced*.

Example 2.2. Let $\mathcal{C}$ be a curve of genus 2 such that $P_1, P_2, P_3 \in \mathcal{C}$ and $P_i \neq \tilde{P}_j$ for any $i, j \in \{1, 2, 3\}$. Then $(P_1) + 2(P_2) + (P_3) - 4(\infty)$ is a semi-reduced divisor and $(P_1) + (P_2) - 2(\infty)$ is a reduced divisor.

It turns out that every zero degree divisor is linearly equivalent to exactly one reduced divisor, which means every divisor class can be uniquely represented by a reduced divisor. For a given divisor class $\overline{\mathcal{A}}$ we define $\rho(\overline{\mathcal{A}})$ to be the unique reduced divisor of the class. In addition we define $\epsilon(\overline{\mathcal{A}})$ to be the effective part of this divisor (i.e. $\rho(\overline{\mathcal{A}}) = \epsilon(\overline{\mathcal{A}}) - d(\infty)$, where d is the degree of $\epsilon(\overline{\mathcal{A}})$).

Remark 2.1. In the elliptic curve case the Jacobian is isomorphic to the group of points on the curve. The isomorphism $\mathcal{E} \to \mathrm{Pic}^0(\mathcal{E})$ is given by $P \mapsto \overline{(P) - (\infty)}$.

Storing reduced divisors is done by representing them as a pair of polynomials. This representation is known as the *Mumford representation*. Let $\mathcal{C}/K$ be a hyperelliptic curve and $\mathcal{A} = \sum_{P \in \mathcal{C}} n_P(P) - (\sum_{P \in \mathcal{C}} n_P)(\infty)$ a reduced divisor. We use the notation x_P and y_P for the coordinates of a point $P \in \mathcal{C}$. Now we can uniquely represent $\mathcal{A}$ by a pair of polynomials $a, b \in K[x]$. The first polynomial a is defined as follows:

$$a(x) = \prod_{P \in \mathcal{C}} (x - x_P)^{n_P}.$$

The second polynomial b is defined as the unique polynomial satisfying the following properties:

1. $\deg b < \deg a \leq g$,
2. for all $P \in \mathcal{C}$ such that $n_P > 0$: $b(x_P) = y_P$,
3. b is a solution of the congruence $b^2 + hb \equiv f \pmod{a}$.

A reduced divisor represented by polynomials a and b will be denoted $\mathrm{div}(a, b)$.

2.6.3 Computing the group law

The elements of the divisor class group have the structure of an Abelian group. The identity element of this group is the divisor class represented by the reduced divisor with Mumford representation $\mathrm{div}(1,0)$. The inverse of an element $\mathrm{div}(a,b)$ is $\mathrm{div}(a,-b-h)$. We describe an algorithm to add divisor classes, due to Cantor. Assuming Mumford representation it adds two divisors in reduced form and returns the sum of the divisors in reduced form. It is far from efficient, but works for curves of arbitrary genus. For fixed genus it is possible to derive much more efficient algorithms that do not need to invoke the extended Euclidian algorithm.

Algorithm 1: Cantor's Algorithm

Input: Reduced divisors $\mathrm{div}(a_1,b_1)$ and $\mathrm{div}(a_2,b_2)$.
Output: Reduced divisor $\mathrm{div}(a_3,b_3) = \mathrm{div}(a_1,b_1) + \mathrm{div}(a_2,b_2)$.

`// Calculate the sum of the divisors.`
$d_1 \leftarrow \gcd(a_1,a_2) = e_1a_1 + e_2a_2$
$d \leftarrow \gcd(d_1, b_1+b_2+h) = c_1d_1 + c_2(b_1+b_2+h)$
$s_1 \leftarrow c_1e_1$
$s_2 \leftarrow c_1e_2$
$s_3 \leftarrow c_2$
$a_3 \leftarrow a_1a_2/d^2$
$b_3 \leftarrow (s_1a_1b_2 + s_2a_2b_1 + s_3(b_1b_2+f))/d \pmod{a_3}$
`// Make sure the result is in reduced form.`
while $\deg a_3 > g$ **do**
  $a_3 \leftarrow (f - hb_3 - b_3^2)/a_3$
  $b_3 \leftarrow -h - b_3 \pmod{a_3}$
end

Consult [37] for a proof of correctness of this algorithm.

2.6.4 Group structure

Group order

For elliptic curves the order of the group is simply the number of points on the curve. For hyperelliptic curves the points do not have the structure of a group, so we need to determine the order of the divisor class group instead. Obviously when a hyperelliptic curve is defined over a finite field the Jacobian has finite order, since there is only a finite amount of polynomials usable as a representation of a divisor class. For an elliptic curve $\mathcal{E}/K$, the group of points contains a subgroup of K-rational points: $\mathcal{E}(K) \subseteq \mathcal{E}(\overline{K})$. Similarly, the divisor class group of a hyperelliptic curve $\mathcal{C}/K$ contains a subgroup with K-rational divisor classes, but the concept of a rational divisor is a bit more complicated.

Formally, a divisor $\mathcal{A} = \sum_{P\in\mathcal{C}} n_P(x_P,y_P)$ is *K-rational* if $\mathcal{A} = \sum_{P\in\mathcal{C}} n_P(\sigma(x_P),\sigma(y_P))$ for every $\sigma \in \mathrm{Gal}(\overline{K},K)$. It is important to realize that this is not the same as requiring each point in the support of $\mathcal{A}$ to have coordinates in K. However, a principal divisor is K-rational if and only if it is

the divisor of a rational function with coefficients in K. Also, if a divisor class with representative $\mathrm{div}(a,b)$ is K-rational, then the polynomials a, b have coefficients in K. The group of K-rational divisor classes in the divisor class group is denoted $\mathrm{Pic}^0_K(\mathcal{C})$ and is a subgroup of $\mathrm{Pic}^0(\mathcal{C})$.

The following theorem gives a bound on the order of the divisor class group of a curve defined over a finite field with q elements.

Theorem 2.5 (Weil)**.** *Let $\mathcal{C}/\mathbb{F}_q$ be a hyperelliptic curve of genus g. Then*

$$(\sqrt{q}-1)^{2g} \leq \#Pic^0_{\mathbb{F}_q}(\mathcal{C}) \leq (\sqrt{q}+1)^{2g}.$$

Torsion subgroups

The following definitions are essentially the same as the definitions of the torsion subgroups of the group of points on an elliptic curve. Using the group law on the divisor class group a reduced divisor $\mathcal{A}$ can be repeatedly added to itself. For $n \in \mathbb{Z}$ we define:

$$n\mathcal{A} = \begin{cases} \underbrace{\mathcal{A}+\mathcal{A}+\ldots+\mathcal{A}}_{n \text{ times}}, & \text{if } n > 0 \\ \underbrace{(-\mathcal{A})+(-\mathcal{A})+\ldots+(-\mathcal{A})}_{n \text{ times}}, & \text{if } n < 0 \\ \infty, & \text{if } n = 0 \end{cases}$$

The order of a reduced divisor $\mathcal{A}$ is the smallest positive integer n such that $n\mathcal{A} = 0$, where 0 is the divisor with empty support. The divisors with order dividing n form a subgroup denoted $\mathrm{Pic}^0(\mathcal{C})[n]$. The K-rational divisors with order dividing n form a subgroup denoted $\mathrm{Pic}^0_K(\mathcal{C})[n]$.

2.6.5 Example

Let C be a hyperelliptic curve defined by the equation $y^2 = x^5+x^3+1$. Then $\mathcal{C}(\mathbb{F}_3) = \{\infty, (0,1), (0,2), (1,0)\}$. Now let $\mathbb{F}_{3^2} = \mathbb{F}_3/(x^2-x-1)$ and let α be a root of the polynomial x^2-x-1 in $\mathbb{F}_{3^2}$. Then $\mathcal{C}(\mathbb{F}_{3^2}) = \{\infty, (0,1), (0,2), (1,0), (\alpha+1,1), (\alpha+1,2), (2,\alpha+1), (2,2\alpha+2), (2\alpha+2,1), (2\alpha+2,2)\}$.

Table 2.1: $\mathrm{Pic}^0_{\mathbb{F}_3}(\mathcal{C})$

Reduced divisor	Mumford representation
0	$\mathrm{div}(1,0)$
$(0,2)-(\infty)$	$\mathrm{div}(x,2)$
$2(0,2)-2(\infty)$	$\mathrm{div}(x^2,2)$
$(\alpha+1,1)+(2\alpha+1,1)-2(\infty)$	$\mathrm{div}(x^2+1,1)$
$(0,1)+(1,0)-2(\infty)$	$\mathrm{div}(x^2+2x, 2x+10)$
$(1,0)-(\infty)$	$\mathrm{div}(x+2,0)$
$(0,1)+(0,2)-2(\infty)$	$\mathrm{div}(x^2+2x, x+2)$
$(\alpha+1,2)+(2\alpha+1,2)-2(\infty)$	$\mathrm{div}(x^2+1,2)$
$2(0,1)-2(\infty)$	$\mathrm{div}(x^2,1)$
$(0,1)-(\infty)$	$\mathrm{div}(x,1)$

The divisor class group over $\mathbb{F}_3$ is shown in 2.1. It contains two reduced divisors as representatives that have non-$\mathbb{F}_3$-rational points in their support. However, they are $\mathbb{F}_3$-rational divisors. Also notice the polynomials in the Mumford representation all have coordinates in $\mathbb{F}_3$.

Let $\mathcal{A}_1 = (0,2) - (\infty) = \mathrm{div}(x, 2)$ and $\mathcal{A}_2 = 2(0,1) - 2(\infty) = \mathrm{div}(x^2, 1)$. We are going to compute the reduced divisor in the class $\overline{\mathcal{A}}_1 + \overline{\mathcal{A}}_2$ using Cantor's algorithm. First we use the extended Euclidian algorithm to calculate $d_1 = \gcd(a_1, a_2) = \gcd(x, x^2) = 1 \cdot x + 0 \cdot x^2 = e_1 a_1 + e_2 a_2$. Similarly, we then calculate $d = \gcd(d_1, b_1 + b_2 + h) = \gcd(x, 1 + 2 + 0) = \gcd(x, 0) = 1 \cdot x + 0 \cdot 0 = c_1 d_1 + c_2(b_1 + b_2 + h)$. Now $s_1 = c_1 e_1 = 1$, $s_2 = c_1 e_2 = 0$ and $s_3 = c_2 = 0$. Then $a_3 = a_1 a_2 / d^2 = x^3/x^2 = x$ and $b_3 = (s_1 a_1 b_2 + s_2 a_2 b_1 + s_3(b_1 b_2 + f))/d \pmod{a}_3 = (x + 0 + 0)/x = 1 \pmod{a}_3$. Since $\deg a_3 = 1 \leq 2$ we are done and the reduced divisor in the class $\overline{\mathcal{A}_1} + \overline{\mathcal{A}_2} = (0,1) - (\infty) = \mathrm{div}(x, 1)$.

2.6.6 Rational functions on hyperelliptic curves

As with elliptic curves, the following can assist [39]. Suppose $\mathcal{C}/K : y^2 + h(x)y = f(x)$ is a hyperelliptic curve of genus g. Similar to the elliptic curve case, every polynomial $f(x, y) \in \overline{K}[\mathcal{C}]$ can be written in the form $a(x) + b(x)y$, by rearranging terms and replacing any occurrence of y^2 by $f(x) - h(x)y$.

Definition 2.10. Let $f(x, y) = a(x) + b(x)y \neq 0$ be polynomial function in $\overline{K}[\mathcal{C}]$. Then the *degree* of f is defined to be

$$\deg(f) = \max\{2 \deg_x(a), 2g + 1 + 2 \deg_x(b)\}$$

where again $\deg_x$ denotes the usual degree of a polynomial in x, taking $\deg_x(0) = \infty$.

The behaviour of rational functions on $\mathcal{C}$ in the point at infinity is now defined exactly the same as in the elliptic curve case.

A uniformizing parameter π for a point $P \in \mathcal{C}(K)$ can be found as follows. If $P = \infty$ then choose $\pi = x^g/y$. If $P = \tilde{P}$ then choose $\pi = y$. If $P = (x_P, y_P)$, $y_P \neq 0$ then choose $\pi = x - x_P$.

Chapter 3

Pairings

A pairing is map $e : G_1 \times G_2 \rightarrow G_T$ where G_1, G_2 and G_T are groups (usually cyclic of the same prime order). In some instances the pairing is *symmetric*: $G_1 = G_2$. When $G_1 \neq G_2$ the pairing is said to be *asymmetric*. In this chapter we describe how to construct pairings on elliptic curves and the divisor class group of hyperelliptic curves.

3.1 Weil Pairing

The Weil pairing was introduced by Andre Weil in 1940 [52] and has proven to be a useful tool in the study of elliptic curves. There are several definitions of the Weil pairing which are all closely related but not exactly equivalent. We will not focus on these differences here and stick to the definition that allows for straightforward extraction of an algorithm to compute it.

3.1.1 Definition

The Weil pairing is a function that maps a pair of points in an n-torsion group of an elliptic curve to an n^{th} root of unity in some extension field of the field the curve was defined over. More specifically, let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve and let n be a prime divisor of $\#\mathcal{E}(\mathbb{F}_q)$ coprime to $\text{char}(\mathbb{F}_q)$. The *Weil pairing* is a map

$$e : \mathcal{E}(\overline{\mathbb{F}}_q)[n] \times \mathcal{E}(\overline{\mathbb{F}}_q)[n] \rightarrow \overline{\mathbb{F}}_q^*.$$

Sometimes we will use the notation e_n instead of e to specify directly the order of the torsion group on which the pairing is defined.

The Weil pairing maps points to a certain extension field $\mathbb{F}_{q^k}$, which should be large enough to make sure $\mathcal{E}(\mathbb{F}_{q^k})$ contains all n-torsion points (so in reality we do not need the full algebraic closure of $\mathbb{F}_q$). The *embedding degree* is the degree k of this field extension. When $n \nmid q-1$, then k is the smallest integer such that $n \mid q^k - 1$, or equivalently, the smallest integer such that $\mathbb{F}_{q^k}$ contains all n^{th} roots of unity. If $n \mid q-1$ then in some instances $k = 1$, but in other instances $k = n$. From now on we will assume $n \nmid q-1$, since this is the case in most practical situations.

Let $P, Q \in \mathcal{E}(\overline{\mathbb{F}}_{q^k})[n]$. To define $e(P,Q)$, we first need a divisor $\mathcal{A}_P \sim (P) - (\infty)$ and a divisor $\mathcal{A}_Q \sim (Q) - (\infty)$. We require that $\mathcal{A}_P$ and $\mathcal{A}_Q$ have disjoint support. First note that since $nP = nQ = \infty$ it follows that $n\mathcal{A}_P \sim n(P) - n(\infty)$ and $n\mathcal{A}_Q \sim n(Q) - n(\infty)$ are principal divisors. As a consequence of this, there exist rational functions f_P and f_Q such that $(f_P) = n\mathcal{A}_P$ and $(f_Q) = n\mathcal{A}_Q$. We can now define the Weil pairing as:

$$e(P,Q) := \frac{f_P(\mathcal{A}_Q)}{f_Q(\mathcal{A}_P)}$$

The result of the pairing is independent of the divisors chosen and thus the Weil pairing is well-defined.

3.1.2 Properties

The following theorems state that the Weil pairing is a bilinear, alternating and non-degenerate mapping. A proof of these properties is given in [48].

Theorem 3.1. *The Weil pairing is linear in both arguments:*

$$e(P+Q,R) = e(P,R)e(Q,R) \text{ for all points } P,Q,R$$

and

$$e(P,R+S) = e(P,R)e(P,S) \text{ for all points } P,R,S.$$

Theorem 3.2. *The Weil pairing is alternating:*

$$e(P,Q) = e(Q,P)^{-1} \text{ for all points } P,Q.$$

As a consequence, the self-pairing of a point P is always 1:

$$e(P,P) = e(P,P)^{-1} = 1 \text{ for all points } P.$$

Theorem 3.3. *The Weil pairing is non-degenerate:*

$$e(P,Q) = 1 \text{ for all points } Q \text{ if and only if } P = \infty.$$

3.1.3 Computation

We describe how to compute the Weil pairing using an algorithm by Victor Miller [40]. The paper in which the algorithm is described was never published but is cited very frequently. We will use the method as described in [10] and employ the same notation.

The first step is to find divisors $\mathcal{A}_P \sim (P) - (\infty)$ and $\mathcal{A}_Q \sim (Q) - (\infty)$. To do so, choose random $R_1, R_2 \in \mathcal{E}(\mathbb{F}_{q^k})[n]$ and let $\mathcal{A}_P = (P + R_1) - (R_1)$ and $\mathcal{A}_Q = (Q + R_2) - (R_2)$. These are suitable choices, since the difference of $\mathcal{A}_P$ and $(P) - (\infty)$ is indeed a principal divisor:

$$\mathcal{A}_P - \big((P) - (\infty)\big) = \big((P + R_1) - (R_1)\big) - \big((P) - (\infty)\big) = (P + R_1) - (R_1) - (P) + (\infty)$$

and similarly, the difference of $\mathcal{A}_Q$ and $(Q) - (\infty)$ is a principal divisor.

To evaluate $e(P,Q)$ we now have to compute

$$\frac{f_P(\mathcal{A}_Q)}{f_Q(\mathcal{A}_P)} = \frac{f_P\big((Q+R_2)-(R_2)\big)}{f_Q\big((P+R_1)-(R_1)\big)} = \frac{f_P(Q+R_2)}{f_Q(P+R_1)} \cdot \frac{f_Q(R_1)}{f_P(R_2)}.$$

The main problem in computing the Weil Pairing is finding functions f_P and f_Q such that $(f_P) = n\mathcal{A}_P$ and $(f_Q) = n\mathcal{A}_Q$. We show how to construct f_P iteratively. The same method is used to construct f_Q. First define the function f_i for $i \geq 1$ such that

$$(f_i) = i(P+R_1) - i(R_1) - (iP) + (\infty).$$

Since P is a n-torsion point, it follows that

$$(f_n) = n(P+R_1) - n(R_1) - (nP) + (\infty) = n(P+R_1) - n(R_1) + (\infty) = n\mathcal{A}.$$

Lemma 3.1. *Let $P, R_1 \in \mathcal{E}(\mathbb{F}_{q^k})[n]$ be the points as described above, and let i and j be positive integers such that $i \neq j$. Suppose l is a function such that $l(x,y) = 0$ is the equation of the line joining the points iP and jP and suppose v is a function such that $v(x,y) = 0$ is the equation of a vertical line through the point $(i+j)P$. Then*

$$f_{i+j} = f_i \cdot f_j \cdot \frac{l}{v}$$

Proof. We use the divisors of the functions to prove the equality. First note that since l intersects the curve at the points iP, jP and $-(i+j)P$ the divisor of l is given by

$$(l) = (iP) + (jP) + (-(i+j)P) - 3(\infty).$$

The line v intersects the curve at the points $(i+j)P$ and $-(i+j)P$, which means the divisor of v is given by

$$(v) = ((i+j)P) + (-(i+j)P) - 2(\infty).$$

Writing down the divisor of $f_i f_j \frac{l}{v}$ yields the equality needed to prove the lemma:

$$\begin{aligned}
(f_i f_j \tfrac{l}{v}) &= (f_i) + (f_j) + (l) - (v) \\
&= \big(i(P+R_1) - i(R_1) - (iP) + (\infty)\big) + \\
&\quad \big(j(P+R_1) - j(R_1) - (jP) + (\infty)\big) + \\
&\quad \big((iP) + (jP) + (-(i+j)P) - 3(\infty)\big) - \\
&\quad \big(((i+j)P) + (-(i+j)P) - 2(\infty)\big) \\
&= (i+j)(P+R_1) - (i+j)(R_1) - (iP) - (jP) + 2(\infty) + \\
&\quad (iP) + (jP) + (-(i+j)P) - 3(\infty) - ((i+j)P) - (-(i+j)P) + 2(\infty) \\
&= (i+j)(P+R_1) - (i+j)(R_1) - ((i+j)(P)) + (\infty) \\
&= (f_{i+j}).
\end{aligned}$$

□

Constructing f_P this way and then evaluating it in $\mathcal{A}_Q$ would be unwieldy, because f_n becomes very complex for high n. Therefore when computing the Weil Pairing we do not store the actual expressions of the functions, but immediately evaluate them at each step. To arrive at f_n we use the binary representation of $n = [n_m, n_{m-1}, \ldots, n_0]_2$ and use the *double-and-add* method. We initialize f_i to f_1 and iterate through the bits starting at $m-1$ going down to 0. At each iteration we either double i

(if the corresponding bit is 0) or we double i and add 1 (if the corresponding bit is 1). Meanwhile we calculate the corresponding f_i by using the recursion $f_{i+j} = f_i f_j \frac{l}{v}$. After the final iteration i will be equal to n and we will have obtained f_n.

For this method to work, we need to be able to directly calculate the evaluation of f_1 in $\mathcal{A}_Q$. Fortunately, the function f_1 can be explicitly constructed. First note that the divisor of f_1 is given by

$$(f_1) = (P + R_1) - (R_1) - (P) + (\infty).$$

Now let $l(x, y) = 0$ be the equation of the line going through the points P and R_1. Then the divisor of l is given by

$$(l) = (P) + (R_1) + (-(P + R_1)) - 3(\infty).$$

Let $v(x, y) = 0$ be the vertical line going through the point $P + R_1$. Then the divisor of v is given by

$$(v) = (P + R_1) + (-(P + R_1)) - 2(\infty).$$

It follows that

$$\begin{aligned}(v) - (l) &= (P + R_1) + (-(P + R_1)) - 2(\infty) - (P) - (R_1) - (-(P + R_1)) + 3(\infty) \\ &= (P) + (R_1) - (P + R_1) + (\infty) \\ &= (f_1).\end{aligned}$$

As a consequence

$$f_1(\mathcal{A}_Q) = \frac{v(Q + R_2)}{l(Q + R_2)} \cdot \frac{l(R_2)}{v(R_2)}.$$

Combining these ideas leads to an efficient algorithm for computing $f_P(\mathcal{A}_Q)$, which is outlined as follows.

Algorithm 2: Miller's Algorithm [Weil pairing]

Input: $P, Q, R_1, R_2, m, [n_m, \ldots, n_1, n_0]_2 = n$.
Output: $f_P(\mathcal{A}_Q)$.
$l \leftarrow$ line through P and R_1
$v \leftarrow$ vertical line through $P + R_1$
$f \leftarrow \frac{v(Q+R_2)}{l(Q+R_2)} \cdot \frac{l(R_2)}{v(R_2)}$
$T \leftarrow P$
for $i = m - 1$ *to* 0 **do**
  $l \leftarrow$ tangent line at T
  $v \leftarrow$ vertical line through $2T$
  $f \leftarrow f^2 \cdot \frac{l(Q+R_2)}{v(Q+R_2)} \cdot \frac{v(R_2)}{l(R_2)}$
  $T \leftarrow 2T$
  if $n_i = 1$ **then**
    $l \leftarrow$ line through T and P
    $v \leftarrow$ vertical line through $T + P$
    $f \leftarrow f \cdot f_1 \cdot \frac{l(Q+R_2)}{v(Q+R_2)} \cdot \frac{v(R_2)}{l(R_2)}$
    $T \leftarrow T + P$
  end
end

Computing $f_Q(\mathcal{A}_P)$ is done by modifying the algorithm in the obvious way. The Weil pairing is evaluated by running Miller's algorithm twice. The first time to compute $f_P(\mathcal{A}_Q)$ and the second time to compute $f_Q(\mathcal{A}_P)$. Dividing them gives the final result.

3.1.4 Modifications

In order for the Weil pairing to be useful in cryptography it needs to be slightly adapted. First of all we want the Weil Pairing to satisfy a stronger notion of non-degeneracy. In its current form non-degeneracy means that for every $P \neq \infty$ there is at least one Q such that $e(P,Q) \neq 1$ and vice versa. We would like the pairing to be *strongly non-degenerate*, which means the self-pairing should not be trivial (i.e. $e(P,P) \neq 1$ for $P \neq \infty$). Unfortunately, from the alternating property we know that $e(P,P) = 1$. Then, from bilinearity it follows $e(P,Q) = 1$ when P and Q are linearly dependent.

We also would like the pairing to be defined on cyclic groups instead of the product of two cyclic groups such as the n-torsion group of an elliptic curve. We can restrict the inputs to a cyclic subgroup of the n-torsion group, but then the pairing would be trivial on all inputs as they are all linearly dependent. The obvious solution is to take the inputs from separate cyclic subgroups of the n-torsion group. As we shall see, this also solves the problem of strong non-degeneracy.

Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve, n a prime coprime to $\text{char}(\mathbb{F}_q)$ such that $\mathcal{E}(\mathbb{F}_q)$ contains a point of order n, and k the embedding degree. If P_g and Q_g are linearly independent points, they generate the full n-torsion group of $\mathcal{E}$ (i.e. $\langle P_g, Q_g \rangle = \mathcal{E}(\mathbb{F}_q)[n]$). Let $G_1 = \langle P_g \rangle$ and $G_2 = \langle Q_g \rangle$. If we restrict the inputs of the Weil pairing as follows

$$e : G_1 \times G_2 \to \mathbb{F}_{q^k}^*$$

then the pairing satisfies the strong non-degeneracy property.

A practical problem here is to actually find points in G_1 and G_2, but this can be solved if there exists a *distortion map* on $\mathcal{E}$ [4]. A distortion map $\phi : \mathcal{E} \to \mathcal{E}$ maps a point P to a point $\phi(P)$ such that P and $\phi(P)$ are linearly independent. For supersingular curves these distortion maps can always be computed. We can use this property to obtain a symmetric strongly non-degenerate pairing, known as the the *modified Weil Pairing* $\hat{e}(P,Q) : G_1 \times G_1 \to \mathbb{F}_{q^k}^*$. It is defined as follows:

$$\hat{e}(P,Q) := e(P, \phi(Q))$$

3.2 Tate-Lichtenbaum Pairing

The Tate pairing, or Tate-Lichtenbaum pairing was introduced to cryptography by Frey and Ruck [44] for the purpose of transporting the discrete logarithm problem on the divisor class group of a curve $\mathcal{C}/\mathbb{F}_q$ to the multiplicative group of some extension of its field of definition. However, after the arrival of pairing based cryptography, it was quickly proposed as an alternative to the Weil pairing, as its computation is less costly. In this section we will define the Tate pairing for elliptic curves and give an efficient way to compute it. In the next section we will generalize the pairing to hyperelliptic

curves. Obviously the hyperelliptic version of the pairing can be used on elliptic curves as well, but the simplification of the computation in the elliptic case is significant enough to justify separate treatment.

3.2.1 Definition

Like the Weil pairing, the Tate pairing can be defined in terms of evaluations of rational functions in divisors. Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve defined over a finite field and fix a prime n coprime to char $\mathbb{F}_q$ such that $n \mid \#\mathcal{E}(\mathbb{F}_q)$. Let $\mathbb{F}_{q^k}$ be the smallest extension field of $\mathbb{F}_q$ containing the set of n^{th} roots of unity. Then k is the smallest integer such that $n \mid q^k - 1$ (the embedding degree). We first define

$$n\mathcal{E}(\mathbb{F}_q) := \{nP \mid P \in \mathcal{E}(\mathbb{F}_q)\}.$$

The quotient group $\mathcal{E}(\mathbb{F}_q)/n\mathcal{E}(\mathbb{F}_q)$ is the set of equivalence classes of points in $\mathcal{E}(\mathbb{F}_q)$ where points are considered equivalent if their difference is a point of order n. The quotient group $\mathbb{F}_q^*/(\mathbb{F}_q^*)^n$ is the set of equivalence classes of field elements where two elements are considered equivalent if they are the same up to multiplication with an n^{th} power. This quotient group is isomorphic to μ_n (the group of $n^{th} roots$.
The *Tate pairing* is a map

$$t : \mathcal{E}(\mathbb{F}_{q^k})[n] \times \mathcal{E}(\mathbb{F}_{q^k})/n\mathcal{E}(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^n.$$

Similar to the Weil pairing, sometimes we will use the notation t_n to indicate the order of the torsion group.

Let $P \in \mathcal{E}(\mathbb{F}_{q^k})[n]$ and $Q \in \mathcal{E}(\mathbb{F}_{q^k})$, where Q is a representative of an equivalence class in $\mathcal{E}(\mathbb{F}_{q^k})/n\mathcal{E}(\mathbb{F}_{q^k})$. To define $t(P,Q)$ we first need a divisor $\mathcal{A}_Q \sim (Q) - (\infty)$ and a function f_P such that $(f_P) = n(P) - n(\infty)$. We require $\mathcal{A}_Q$ and (f_P) to have disjoint support to ensure the evaluation of f_P in $\mathcal{A}_Q$ is not equal to zero. We can now define the Tate Pairing of P and Q as:

$$t(P,Q) := f_P(\mathcal{A}_Q).$$

3.2.2 Properties

We prove that the Tate pairing is a bilinear and non-degenerate mapping. Then we show that we need an embedding degree larger than 1 to get non-trivial self-pairings.

Theorem 3.4. *The Tate pairing is bilinear:*

$$t(P+Q,R) = t(P,R)t(Q,R) \text{ for all points } P,Q \in \mathcal{E}(\mathbb{F}_{q^k})[n] \text{ and } R \in \mathcal{E}(\mathbb{F}_{q^k})/n\mathcal{E}(\mathbb{F}_{q^k}).$$

and

$$t(P,R+S) = t(P,R)t(P,St) \text{ for all points } P \in \mathcal{E}(\mathbb{F}_{q^k})[n] \text{ and } R,S \in \mathcal{E}(\mathbb{F}_{q^k})/n\mathcal{E}(\mathbb{F}_{q^k}).$$

Proof. We give the proof as described in [4]. Let f_P and f_Q be functions such that $(f_P) = n(P) - n(\infty)$ and $(f_Q) = n(Q) - n(\infty)$. Let $\mathcal{A}_R \sim (R) - (\infty)$ be a divisor with support disjoint to $\{\infty, P, Q, P+Q\}$. Then

$$t(P,R) = f_P(\mathcal{A}_R) \text{ and } t(Q,R) = f_Q(\mathcal{A}_R).$$

Now let g be a function such that $(g) = (P+Q) - (P) - (Q) + (\infty)$. Note that

$$\begin{aligned}(f_P f_Q g^n) &= (f_P) + (f_Q) + n(g) \\ &= n(P) + n(Q) - 2n(\infty) + n(P+Q) - n(P) - n(Q) + n(\infty) \\ &= n(P+Q) - n(\infty).\end{aligned}$$

This means the function $f_P f_Q g^n$ is suitable for use in computation of $t(P+Q, R)$:

$$t(P+Q, Rt) = f_P f_Q g^n(\mathcal{A}_R) = f_P(\mathcal{A}_R) f_Q(\mathcal{A}_R) g(\mathcal{A}_R)^n = t(P,R)t(Q,R)g(\mathcal{A}_R)^n.$$

In other words, $t(P,R)t(Q,R)$ and $t(P+Q,R)$ are equivalent up to a multiplication with a n^{th} power. This means they are equivalent in the quotient group $K^*/(K^*)^n$, proving linearity in the first factor.

Now choose divisors $\mathcal{A}_R \sim (R) - (\infty)$ and $\mathcal{A}_S \sim (S) - (\infty)$. Let f_P be a function such that $(f_P) = n(P) - n(\infty)$. Then

$$t(P,R) = f_P(\mathcal{A}_R) \text{ and } t(P,S) = f_P(\mathcal{A}_S).$$

Note that $\mathcal{A}_R + \mathcal{A}_S \sim (R+S) - (\infty)$, since

$$\begin{aligned}(R+S) - (\infty) - \mathcal{A}_R - \mathcal{A}_S &= (R+S) - (\infty) - (R) + (\infty) - (S) + (\infty) \\ &= (R+S) - (R) - (S) + (\infty).\end{aligned}$$

which clearly is a principal divisor. This means $\mathcal{A}_R + \mathcal{A}_S$ is suitable for use in computation of $t(P, R+S)$, leading us to the equation

$$t(P, R+S) = f_P(\mathcal{A}_R + \mathcal{A}_S) = f_P(\mathcal{A}_R) f_P(\mathcal{A}_S) = t(P,R)t(P,S)$$

proving linearity in the second factor. □

Theorem 3.5. *The Tate Pairing is non-degenerate. If $P \in \mathcal{E}(\mathbb{F}_{q^k})[n]$ then*

$$t(P,Q) = 1 \text{ for all } Q \text{ if and only if } P = \infty.$$

Similarly, if $Q \in \mathcal{E}(\mathbb{F}_{q^k})/n\mathcal{E}(\mathbb{F}_{q^k})$ then

$$t(P,Q) = 1 \text{ for all } P \text{ if and only if } Q = \infty.$$

For a proof, see [29].

Theorem 3.6. *If $k > 1$ (where k is the embedding degree), then self-pairings of points in $\mathcal{E}(\mathbb{F}_q)[n]$ are trivial. As a result of bilinearity it follows that if $k > 1$, then the pairing of any two linearly dependent points in $\mathcal{E}(\mathbb{F}_q)[n]$ is trivial.*

Proof. Let $P \in \mathcal{E}(\mathbb{F}_q)[n]$, $f_P \in \mathbb{F}_q(\mathcal{E})$ such that $(f_P) = n(P) - n(\infty)$, and $\mathcal{A}_P \in \mathrm{Div}(\mathcal{E})$ such that $\mathcal{A}_P \sim (P) - (\infty)$. Then $t(P,P) = f_P(\mathcal{A}_P) \in \mathbb{F}_q^*$. Since $\mathbb{F}_q^*$ is a group of order $q-1$ and $\gcd(n, q-1) = 1$ (because $k > 1$) it follows that $\mathbb{F}_q^* = (\mathbb{F}_q^*)^n$. As a consequence, $t(P,P)$ is an n^{th} power and vanishes in the quotient group $\mathbb{F}_q^*/(\mathbb{F}_q^*)^n$. □

3.2.3 Computation

The same approach as used in Miller's algorithm for computing the Weil Pairing can be used for efficiently computing the Tate Pairing. In computing the Tate Pairing the main problem is to construct a function f such that $(f) = n(P) - n(\infty)$. Again, we can build this function iteratively using the double-and-add method. First we define a function f_i for $i \geq 1$ such that

$$(f_i) = i(P) - (iP) - (i-1)(\infty).$$

Since P is a n-torsion point, it follows that

$$(f_n) = n(P) - (nP) - (n-1)(\infty) = n(P) - n(\infty) = (f).$$

Note that

$$(f_1) = (P) - (P) - 0(\infty) = 0$$

which means $f_1 = 1$.

Lemma 3.2. *Let $P \in \mathcal{E}(\mathbb{F}_{q^k})[n]$, and let i and j be positive integers. Suppose l is a function such that $l(x,y) = 0$ is the equation of the line between the points iP and jP and suppose v is a function such that $v(x,y) = 0$ is the equation of vertical line through the point $(i+j)P$. Then*

$$f_{i+j} = f_i \cdot f_j \cdot \frac{l}{v}$$

Proof. Recall from the lemma used in computing the Weil Pairing the divisors of l and v:

$$\begin{aligned} (l) &= (iP) + (jP) + (-(i+j)P) - 3(\infty) \\ (v) &= ((i+j)P) + (-(i+j)P) - 2(\infty) \end{aligned}$$

Writing down the sum of the divisors results in the equality:

$$\begin{aligned} (f_i f_j \tfrac{l}{v}) &= (f_i) + (f_j) + (l) - (v) \\ &= \big(i(P) - (iP) - (i-1)(\infty)\big) + \\ &\quad \big(j(P) - (jP) - (j-1)(\infty)\big) + \\ &\quad \big((iP) + (jP) + (-(i+j)P) - 3(\infty)\big) - \\ &\quad \big(((i+j)P) + (-(i+j)P) - 2(\infty)\big) \\ &= (i+j)(P) - (iP) - (jP) - (i+j-2)(\infty) + \\ &\quad (iP) + (jP) + (-(i+j)P) - 3(\infty) - \\ &\quad ((i+j)P) - (-(i+j)P) + 2(\infty)) \\ &= (i+j)(P) - ((i+j)P) - (i+j-1)(\infty) \\ &= (f_{i+j}). \end{aligned}$$

□

Recall from the definition of the Tate Pairing that we also need a divisor $\mathcal{A}_Q \sim (Q) - (\infty)$ with support disjoint from (f_P). Let $R \in \mathcal{E}(K)[n]$ such that $R \neq \infty, P, -Q, P-Q$. Then a valid choice for $\mathcal{A}_Q$ is $(Q+R) - (R)$. We can now compute $t(P,Q) = f_P(\mathcal{A}_Q)$ using Miller's algorithm. Note that it is slightly simpler than the version for the Weil Pairing due to the fact that $f_1 = 1$.

Algorithm 3: Miller's Algorithm [Tate Pairing]

Input: $P, Q, R, m, [n_m, \ldots, n_1, n_0]_2 = n$.
Output: $f_P(\mathcal{A}_Q)$.
$f \leftarrow 1$
$T \leftarrow P$
for $i = m - 1$ *to* 0 **do**
  $l \leftarrow$ tangent line at T
  $v \leftarrow$ vertical line through $2T$
  $f \leftarrow f^2 \cdot \frac{l(Q+R)}{v(Q+R)} \cdot \frac{v(R)}{l(R)}$
  $T \leftarrow 2T$
  if $n_i = 1$ **then**
    $l \leftarrow$ line through T and P
    $v \leftarrow$ vertical line through $T + P$
    $f \leftarrow f \cdot \frac{l(Q+R)}{v(Q+R)} \cdot \frac{v(R)}{l(R)}$
    $T \leftarrow T + P$
  end
end

3.2.4 Modifications

Throughout this section, let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve, n a prime coprime to $\text{char}(\mathbb{F}_q)$ such that $\mathcal{E}(\mathbb{F}_q)$ contains a point of order n, and k the embedding degree.

For practical applications of the Tate pairing, it is desirable to have the output be an element of the multiplicative group of a finite field instead of a coset in the quotient group $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^n$. To achieve this we can raise the result to the power $(q^k - 1)/n$. In the literature this process is referred to as the *final exponentiation*. The exponentiated pairing which maps directly into $\mathbb{F}^*_{q^k}$ is called the *reduced Tate pairing*:

$$t'(P, Q) := t(P, Q)^{(q^k-1)/n}$$

To obtain a symmetric bilinear map from the Tate pairing we have to make sure both inputs come from the same group. At first this may seem like a problem since the second input of the Tate pairing is not a n-torsion point on the curve, but a coset in the quotient group $\mathcal{E}(\mathbb{F}_{q^k})/n\mathcal{E}(\mathbb{F}_{q^k})$. Fortunately, if we assume that $\mathcal{E}(\mathbb{F}_{q^k})$ contains no elements of order n^2, then $\mathcal{E}(\mathbb{F}_{q^k})[n]$ can be identified with $\mathcal{E}(\mathbb{F}_{q^k})/n\mathcal{E}(\mathbb{F}_{q^k})$.

Finally, assuming the existence of a distortion map ϕ on $\mathcal{E}$, we can apply the same modification to the Tate pairing as we did to the Weil pairing (assuming $k > 1$) to obtain a symmetric strongly non-degenerate map defined on cyclic groups. Let $P_g \in \mathcal{E}(\mathbb{F}_q)[n]$ and $G_1 = \langle P_g \rangle$. Then the *modified Tate pairing* is a map $\hat{t}(P, Q) : G_1 \times G_1 \to \mathbb{F}^*_{q^k}$, defined as follows:

$$\hat{t}(P, Q) := t'(P, \phi(Q))$$

3.3 Hyperelliptic Tate-Lichtenbaum Pairing

In this section we generalize the Tate-Lichtenbaum pairing to the divisor class group of a hyperelliptic curve. Most of the results are derived from [28], [21] and [16].

3.3.1 Definition

Let $\mathcal{C}/\mathbb{F}_q$ be a hyperelliptic curve, n a prime coprime to $\text{char}(\mathbb{F}_q)$ with $n \mid \#\text{Pic}^0_{\mathbb{F}_q}(\mathcal{C})$, and again k the smallest integer such that $n \mid q^k - 1$ (the embedding degree).

The hyperelliptic Tate pairing is a map

$$t : \text{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})[n] \times \text{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})/n\text{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C}) \to \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^n.$$

Let $\overline{\mathcal{A}}_P$ and $\overline{\mathcal{A}}_Q$ be divisor classes with respective representatives $\mathcal{A}_P$ and $\mathcal{A}_Q$ such that $\text{supp}(\mathcal{A}_P) \cap \text{supp}(\mathcal{A}_Q) = \emptyset$. Let $f_{\mathcal{A}_P}$ be a function such that $(f_{\mathcal{A}_P}) = n\mathcal{A}_P$. Then we define the Tate pairing of $\overline{\mathcal{A}}_P$ and $\overline{\mathcal{A}}_Q$ as follows:

$$t(\overline{\mathcal{A}}_P, \overline{\mathcal{A}}_Q) = f_{\mathcal{A}_P}(\mathcal{A}_Q)$$

The requirement of $\mathcal{A}_P$ and $\mathcal{A}_Q$ having disjoint support is to make sure the evaluation of $f_{\mathcal{A}_P}$ in $\mathcal{A}_Q$ is not zero. Notice how this is a straightforward generalization of the Tate pairing on elliptic curves, where we take $\mathcal{A}_P = (P) - (\infty)$ and $\mathcal{A}_Q \sim (Q) - (\infty)$ to pair two points P and Q.

It can be shown that this pairing is also well-defined.

3.3.2 Properties

The hyperelliptic Tate pairing satisfies the same properties as the elliptic Tate pairing (i.e. bilinearity and non-degeneracy).

3.3.3 Computation

Again, let $\mathcal{C}/\mathbb{F}_q$ be a hyperelliptic curve, n a prime coprime to $\text{char}(\mathbb{F}_q)$ such that $n \mid \#\text{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})$, where k is the embedding degree. For $\overline{\mathcal{A}}_P \in \text{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})[n]$ and $\overline{\mathcal{A}}_Q \in \text{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})/n\text{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})$, we will describe how to compute $t(\overline{\mathcal{A}}_P, \overline{\mathcal{A}}_Q)$.

The first difficulty in the pairing computation is the choice of divisors as representative of the divisor classes. Computing the group operation of the divisor class is done with reduced divisors in Mumford representation, but we cannot use the reduced representative for both inputs in the pairing evaluation because two reduced divisors do not have disjoint support. Indeed, for any two divisor classes $\overline{\mathcal{A}}_P$ and $\overline{\mathcal{A}}_Q$ we have $\infty \in \text{supp}(\rho(\overline{\mathcal{A}}_P)) \cap \text{supp}(\rho(\overline{\mathcal{A}}_Q))$ (recall $\rho(\overline{\mathcal{A}})$ denotes the reduced divisor in $\overline{\mathcal{A}}$). Fortunately, if we impose certain constraints on the function $f_{\mathcal{A}_P}$ we can evaluate $f_{\mathcal{A}_P}$ in $\epsilon(\overline{\mathcal{A}}_Q)$ (the effective part of the reduced divisor in $\overline{\mathcal{A}}_Q$). For details of these constraints we refer to [28] and [21]. However, if we assume that $k > 1$ and $\overline{\mathcal{A}_P} \in \text{Pic}^0_{\mathbb{F}_q}(\mathcal{C})[n]$, then this constraint is not necessary. From now on we will assume this is the case.

Let $\mathcal{A} = \rho(\overline{\mathcal{A}}_P)$. Again, we build $f_{\mathcal{A}}$ iteratively. Let f_i be a function such that

$$(f_i) = i\mathcal{A} - \rho(i\overline{\mathcal{A}}).$$

It is important to realize that $i\mathcal{A}$ is obtained using multiplication by i in the group $\mathrm{Div}(\mathcal{C})$, and $i\overline{\mathcal{A}}$ is obtained using multiplication by i in the group $\mathrm{Pic}^0(\mathcal{C})[n]$. Since $\overline{\mathcal{A}}$ is an n-torsion element of $\mathrm{Pic}^0(\mathcal{C})$ we have:

$$(f_n) = n\mathcal{A} - \rho(n\overline{\mathcal{A}}) = n\mathcal{A}.$$

Also:

$$(f_1) = \mathcal{A} - \rho(\overline{\mathcal{A}}) = 0$$

which means $f_1 = 1$.

Lemma 3.3. *Let $\mathcal{A} = \rho(\overline{\mathcal{A}}_P)$, and let i and j be positive integers. Furthermore, suppose $g_{i\mathcal{A},j\mathcal{A}}$ is a function such that $(g_{i\mathcal{A},j\mathcal{A}}) = \rho(i\overline{\mathcal{A}}) + \rho(j\overline{\mathcal{A}}) - \rho(i\overline{\mathcal{A}} + j\overline{\mathcal{A}})$. Then*

$$f_{i+j} = f_i \cdot f_j \cdot g_{i\mathcal{A},j\mathcal{A}}.$$

Proof.

$$\begin{aligned}(f_i f_j g) &= (f_i) + (f_j) + (g_{i\mathcal{A},j\mathcal{A}}) \\ &= i\mathcal{A} - \rho(i\overline{\mathcal{A}}) \ + \ j\mathcal{A} - \rho(j\overline{\mathcal{A}}) \ + \ \rho(i\overline{\mathcal{A}}) + \rho(j\overline{\mathcal{A}}) - \rho(i\overline{\mathcal{A}} + j\overline{\mathcal{A}}) \\ &= (i+j)\mathcal{A} - \rho((i+j)\overline{\mathcal{A}}). \\ &= (f_{i+j}).\end{aligned}$$

□

The function g can be obtained from Cantor's algorithm as follows. Assume we are using Cantor's algorithm to calculate the sum of two reduced divisors $\mathcal{A}_i$ and $\mathcal{A}_j$ represented by $[a_1, b_1]$ and $[a_2, b_2]$, respectively. The result of the algorithm is a reduced divisor $\mathcal{A}_{i+j} = \rho(\overline{\mathcal{A}}_i + \overline{\mathcal{A}}_j)$. After the first part of the algorithm we have obtained a semi-reduced divisor $\mathcal{A}'_{i+j}$ represented by $[a'_3, b'_3]$, satisfying

$$\mathcal{A}'_{i+j} = \mathcal{A}_i + \mathcal{A}_j - (d).$$

In the second part of the algorithm the divisor is reduced to a divisor $\mathcal{A}_{i+j}$ represented by $[a_3, b_3]$, satisfying

$$\mathcal{A}_{i+j} = \mathcal{A}'_{i+j} - ((y - b'_3)/a_3). \tag{3.1}$$

Substituting 3.1 in 3.2 gives us

$$\mathcal{A}_{i+j} = \mathcal{A}_i + \mathcal{A}_j - (d) - ((y - b'_3)/a_3) = \mathcal{A}_i + \mathcal{A}_j - (d(y - b'_3)/a_3). \tag{3.2}$$

This shows we can take $g = \frac{d(y-b'_3)}{a_3}$.

The full algorithm is thus as follows. Note that the inputs $\mathcal{A}_P$ and $\mathcal{A}_Q$ are the reduced representatives of the input divisor classes.

Algorithm 4: Miller's Algorithm [Hyperelliptic Tate Pairing]

Input: $\mathcal{A}_P, \mathcal{A}_Q, m, [n_m, \ldots, n_1, n_0]_2 = n$.
Output: $f_{\mathcal{A}_P}(\epsilon(\mathcal{A}_Q))$.
$f \leftarrow 1$
$T \leftarrow \mathcal{A}_P$
for $i = m - 1$ *to* 0 **do**
  $T \leftarrow 2T$ extracting $g_{T,T}$ (using Cantor's algorithm)
  $f \leftarrow f^2 \cdot g(\epsilon(\mathcal{A}_Q))$
  if $n_i = 1$ **then**
    $T \leftarrow T + \mathcal{A}_1$ extracting $g_{T,\mathcal{A}_P}$ (using Cantor's algoritm)
    $f \leftarrow f \cdot g(\epsilon(\mathcal{A}_Q))$
  end
end

Just as with the elliptic version, we can raise the result to the power $(q^k - 1)/n$ to obtain the truncated Tate pairing which maps to the group $\mathbb{F}_{q^k}$.

3.3.4 Elliptic versus Hyperelliptic Pairings

One of the main reasons of considering pairings for hyperelliptic curves is the fact that it is then possible to choose from a larger variety of embedding degrees [23]. Also, when working out special cases of addition formulae in genus 2 it is possible to outperform the group arithmetic on elliptic curves. However, it seems the speed of a hyperelliptic pairing computation is more costly and can in most instances be matched by the speed of an elliptic pairing computation [21]. Regarding the size of group elements, for non-supersingular curves it seems the elements in G_2 are larger in the hyperelliptic curve case, as the size of the elements depends on the embedding degree. The conclusion is that in practice, elliptic curves are more efficient then hyperelliptic curves for general pairing applications. [21]

Chapter 4

Discrete Logarithm Problem

This chapter is dedicated to the discrete logarithm problem and closely related problems like Diffie-Hellman. These problems and the protocols arising from them, require a cyclic Abelian group. Traditionally, in these problems the group of choice is the multiplicative group of a finite field $\mathbb{F}_q$, so in the first sections we will use a multiplicative notation for the groups. When we will move these problems to the group of an elliptic curve or hyperelliptic curve we will switch to additive notation. It is shown how pairings can be used in certain instances to attack protocols based on the discrete logarithm problem in (hyper)elliptic curves.

4.1 Problems

From now on, let G be an Abelian group. In G we can define the following closely related problems.

Definition 4.1. The *discrete logarithm problem* (DLP) is the problem of finding the least positive integer a such that the equation $h = g^a$ holds, when the elements $g, h \in G$ are given, provided this integer exists (i.e. when $h \in \langle g \rangle$).

This problem is believed to be computationally hard. We use the notation $\texttt{DLP}_g$ to refer to the problem of computing discrete logarithms with respect to a base $g \in G$, and we write $\texttt{DLP}_G$ to refer to the problem of computing $\texttt{DLP}_g$ for *any* $g \in G$. The same convention is used for the rest of the problems to be defined in this chapter.

Definition 4.2. Closely related to the discrete logarithm problem is the *Diffie-Hellman problem* (DHP). This is the problem of finding g^{ab} when $g, g^a, g^b \in G$ are given. This problem is sometimes also referred to as the *computational Diffie-Hellman problem* (CDH).

Definition 4.3. There is also an easier decisional version, called the *decision Diffie-Hellman problem* (DDH), which is the problem of deciding if $h = g^{ab}$ when $g, g^a, g^b, h \in G$ are given.

4.1.1 Relating the problems

Obviously if you can solve CDH then you can also solve DDH. This relation is denoted $\texttt{CDH}_G \longrightarrow \texttt{DDH}_G$. More specifically, it means that if there is an algorithm $\mathcal{A}$ that solves $\texttt{CDH}_G$ in polynomial time, then it is possible to construct an algorithm with access to $\mathcal{A}$ that solves $\texttt{DDH}_G$ in polynomial time. We also

say $\texttt{CDH}_G$ is a *stronger* problem than $\texttt{DDH}_G$. This notation will be used throughout the entire chapter to describe polynomially bounded reductions of one problem to another.

Another relation that obviously holds is $\texttt{DLP}_G \longrightarrow \texttt{CDH}_G$ (given g, g^a, g^b you can take discrete logarithms to obtain a and b which allows you to compute g^{ab}). It is not known if the converse is true – nobody really knows if it is possible to obtain g^{ab} from g^a and g^b without first determining either a or b.

As shown in [15], if you can efficiently compute the discrete logarithm with respect to a single element h in a cyclic group G, then you can efficiently compute the discrete logarithm with respect to any $h \in G$. The same holds for the computational Diffie-Hellman problem:

Proposition 4.1. *Let G be a cyclic group of prime order p. For any generator $g \in G$ we have:* $\texttt{DLP}_G \longleftrightarrow \texttt{DLP}_g$ *and* $\texttt{CDH}_G \longleftrightarrow \texttt{CDH}_g$.

Proof. Suppose we can solve $\texttt{DLP}_g$ for a certain $g \in G$. Let $h \in G$. We show we can solve $\texttt{DLP}_h$, which proves we can solve $\texttt{DLP}_G$. Suppose we are given $x = h^a \in G$ and we would like to find a. Let $h = g^b$. Then $x = h^a = (g^b)^a = g^{ba}$. We can solve $\texttt{DLP}_g$ so we can obtain ba as well as b. Then a is easily obtained by computing $b^{-1} \pmod p$. This proves $\texttt{DLP}_G \longleftarrow \texttt{DLP}_g$.

Now suppose we have an algorithm $\mathcal{A}$ which solves $\texttt{CDH}_g$ for a certain $g \in G$. We use the notation $\mathcal{A}(g, g^c, g^d) = g^{cd}$. Let $h \in G$. We show we can solve $\texttt{CDH}_h$, which proves we can solve $\texttt{CDH}_G$. Suppose we are given h, h^c and $h^d \in G$. We need to compute h^{cd}. Since g is also a generator for G we know that $h = g^x$ for a certain x. Then $h^c = (g^x)^c = g^{xc}$ and $h^d = (g^x)^d = g^{xd}$. We now compute $\mathcal{A}(g, g^{xc}, g^{xd}) = g^{x^2dc}$. Also, using approximately $\log p$ calls to $\mathcal{A}$ we can calculate $g^{x^{p-2}} = g^{x^{-1}}$. Then $\mathcal{A}(g, g^{x^{-1}}, g^{x^2dc}) = g^{x^{-1}x^2dc} = g^{xdc} = h^{cd}$. This proves $\texttt{CDH}_G \longleftarrow \texttt{CDH}_g$.

The relations $\texttt{DLP}_G \longrightarrow \texttt{DLP}_g$ and $\texttt{CDH}_G \longrightarrow \texttt{CDH}_g$ obviously hold. □

4.2 Protocols

The Diffie-Hellman problem can be used as the basis for several cryptographic protocols. In this section we will shortly describe two of these protocols [5].

4.2.1 Diffie-Hellman key exchange

One application of CDH is the *Diffie-Hellman key exchange protocol*. Suppose two people, traditionally named Alice and Bob, want to share a secret key (which is a random element in some group). Sharing this secret needs to be done by communicating over an insecure channel and should not require any prior interaction between the two parties. Assuming the agreement between the two parties on a group G of large prime order with generator g, and also the hardness of CDH in G, the sharing of a secret key can be done in one round using the following steps:

1. Alice generates a random positive integer a, which should be less than the group order. The information she sends to Bob is:

   $$g^a.$$

   The integer a is kept private.

2. Bob also generates a random positive integer b, which should be less than the group order. The information he sends to Alice is:

$$g^b.$$

The integer b is kept private.

After these two steps Alices computes $(g^b)^a = g^{ab}$ and Bob computes $(g^a)^b = g^{ab}$. This shared secret g^{ab} cannot be recovered without solving `CDH` in G, because any eavesdropper watching the insecure channel only has the following information:

$$G, g, g^a \text{ and } g^b$$

4.2.2 Digital Signature Algorithm

Another famous application of `CDH` is the *Digital Signature Algorithm*. This protocol requires a group G of large prime order p with generator g and a bijective mapping $f : G \to \mathbb{Z}/p\mathbb{Z}$. Suppose Bob wants to sign a message $m \in \mathbb{Z}/p\mathbb{Z}$ with his private key x, which is a positive integer less than p, the order of the group G. Using the following scheme, the authentication of the message can be established using Bob's public key $h = g^x$:

1. Bob generates a random positive integer k, such that $k \leq p$ and computes $a = g^k$. He then computes the solution b to the congruence

$$m = -xf(a) + kb \pmod{p}.$$

The information he sends to Alice is:

$$m, a \text{ and } b.$$

2. Alice first computes

$$u = mb^{-1} \pmod{p} \quad \text{and} \quad v = f(a)b^{-1} \pmod{p}$$

To establish the authenticity of the message, she needs to verify that $g^u g^b = a$:

$$g^u h^v = g^{mb^{-1}} g^{vx} = g^{mb^{-1}+xf(a)b^{-1}} = g^{(m+xf(a))b^{-1}} = g^{kbb^{-1}} = g^k = a.$$

4.3 Attacks

There exist several algorithms to compute the discrete logarithm. Some of them are designed to work only in the multiplicative group of a finite field. Others are more general and can be used in arbitrary group settings. The fastest general attacks are the *Pollard-ρ* and *baby-step giant-step* methods. The running time of both methods is $O(\sqrt{n})$ (where n is the size of the group), which is exponential in the input length (i.e. the number of digits in n). These methods are often also referred to as *square root methods*. The fastest attack specifically tailored to the multiplicative group of a finite field is the *index-calculus* attack. Its running time is sub-exponential in the input length.

Pohlig-Hellman

In addition to the aforementioned attacks, Pohlig and Hellman observed that in order to solve DLP in a finite group G, it is only necessary to solve DLP in the subgroups of prime power order in G. This simplification is based on the chinese remainder theorem. In other words, the security of a protocol based on DLP in a group does not depend on the order of the group but on the largest prime factor of the order of the group.

4.4 Elliptic Curve Discrete Logarithm Problem

The discrete logarithm can be stated in any Abelian group, including the group of points on an elliptic curve. Its use was first suggested independently by Koblitz [35] and Miller [41].

Definition 4.4. Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve, $P \in \mathcal{E}(\mathbb{F}_q)$ and let n be the order of the group of points $\mathcal{E}(\mathbb{F}_q)$. Note that $\mathcal{E}(\mathbb{F}_q)$ is generally not a cyclic group, but we can still compute the discrete logarithm (with respect to P) of points in the cyclic subgroup generated by P. More specifically, given $Q \in \langle P \rangle$, the *elliptic curve discrete logarithm problem* (ECDLP) is the problem of finding the integer a such that

$$Q = aP.$$

The main advantage of the group of points on an elliptic curve, compared to the multiplicative group of a finite field, is that there is no known sub-exponential algorithm (like the index-calculus method) to solve the discrete logarithm problem. This means we can use smaller groups that offer the same level of security. In practice, for this to be true we need to avoid certain types of elliptic curves as we will see in the next section.

4.4.1 Attacks

Obviously, all the general square root methods can be applied to attack the elliptic curve discrete logarithm problem as well. However, there is no known analogue of the index-calculus method for elliptic curves. In what follows, we will see there are two types of curves where the group of points carry additional structure which allows for a more efficient computation of the discrete logarithm.

MOV attack

The discrete logarithm problem on an elliptic curve can be reduced to the discrete logarithm problem on the multiplicative group of some extension field of the finite field over which the curve is defined. If the extension field is reasonably small, it could be more efficient to solve the discrete logarithm problem there using the sub-exponential index-calculus method. The transportation of the discrete logarithm problem is realized with the Weil pairing, and the degree of the required field extension is the embedding degree. As a consequence, supersingular elliptic curves are especially vulnerable to this attack (recall that supersingular curves have an embedding degree no larger than 6). This method is due to Menezes, Okamoto and Vanstone and is usually referred to as the *MOV attack* [38]. It works as follows.

Let $\mathcal{E}/\mathbb{F}_q$ be a supersingular elliptic curve and let P be a point of order n. Let $\hat{e} : \langle P \rangle \times \langle P \rangle \to \mathbb{F}^*_{q^k}$ denote the modified Weil pairing, where k is the embedding degree. We can obtain the discrete logarithm a of $Q \in \langle P \rangle$ as follows. First compute $\hat{e}(P, Q) = \hat{e}(P, aP) = \hat{e}(P, P)^a$. The integer a can then be found by calculating the discrete logarithm in $\mathbb{F}^*_{q^k}$ with respect to $\hat{e}(P, P)$. Note that we are using the modified Weil pairing which makes use of a distortion map ensuring that $\hat{e}(P, P) \neq 1$.

If $\mathcal{E}/\mathbb{F}_q$ is not supersingular we have to use the unmodified Weil pairing $e : \mathcal{E}(\mathbb{F}_{q^k})[n] \times \mathcal{E}(\mathbb{F}_{q^k})[n] \to \mathbb{F}^*_{q^k}$. As a consequence, $e(P, P) = 1$ is not a generator of $\mathbb{F}^*_{q^k}$. As an alternative, we need to find an $S \in \mathcal{E}(\mathbb{F}_{q^k})[n]$ such that $e(Q, S)$ is a generator of $\mathbb{F}^*_{q^k}$. A method to find such a point S is given in [33], but for a random elliptic curve the embedding degree is too large for the MOV-attack to be useful.

It is also interesting to note that the existence of a bilinear map $\hat{e} : G_1 \times G_1 \to G_T$ implies that DDH is easy in G_1. Suppose we are given $P, aP, bP, Q \in G_1$ where $\langle P \rangle = G_1$. To solve DDH we need to decide if $Q = abP$. In order to do so, we evaluate $\hat{e}(aP, bP) = \hat{e}(P, P)^{ab}$. If $Q = abP$ then also $\hat{e}(P, Q) = \hat{e}(P, abP) = \hat{e}(P, Q)^{ab}$. Thus, solving DDH is simply done by comparing the two pairing evaluations $\hat{e}(aP, bP)$ and $\hat{e}(P, Q)$. A group where CDH is hard but DDH is easy is known as a *gap Diffie-Hellman group* and such a group allows for some interesting cryptographic protocols [42], one of which we will be covered in a later chapter.

Anomalous attack

When an elliptic curve $\mathcal{E}/\mathbb{F}_{p^m}$ has a trace of Frobenius equal to 1 the number of points on the curve equals p^m. In this case the elliptic curve is said to be *anomalous* and there exists an efficient reduction which enables an attacker to solve the discrete logarithm problem on $\mathcal{E}$ in polynomial time [49]. This method works by transporting the discrete logarithm problem on $\mathcal{E}$ to the discrete logarithm problem on the additive group of $\mathbb{F}_p$, where the discrete logarithm problem can be solved in linear time. The actual attack will not be described here, as it requires theory beyond the scope of this thesis.

GHS attack

This attack works for some elliptic curves defined over a finite field of characteristic two. It was shown by Gaudry, Hess and Smart [25] that the elliptic curve discrete logarithm problem on an elliptic curve $\mathcal{E}/\mathbb{F}_{2^n}$ can be reduced to the discrete logarithm problem on an associated hyperelliptic curve $\mathcal{C}/\mathbb{F}_{2^m}$ where $m < n$ (i.e. the associated hyperelliptic curve is defined over a smaller field). This is done through a process known as the *Weil descent*. In some instances, where the genus of $\mathcal{C}$ is large enough, there exists a sub-exponential algorithm to solve the discrete logarithm in $\mathcal{C}$. This will be described in the next section.

4.5 Hyperelliptic Curve Discrete Logarithm Problem

When elliptic curve groups were begun to be used in cryptography, a natural question that arose is if it can be generalized to hyperelliptic curves. Koblitz proposed this in 1989 [36]. The discrete logarithm problem on hyperelliptic curves is defined in the divisor class group of the curve.

Definition 4.5. Let $\mathcal{C}/\mathbb{F}_q$ be a hyperelliptic curve and let $\mathcal{A}_1 \in \mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C})$ and $\mathcal{A}_2 \in \langle \mathcal{A}_1 \rangle$. The *hyperelliptic curve discrete logarithm problem* (HCDLP) is the problem of finding the integer a such that

$$\mathcal{A}_2 = a\mathcal{A}_1.$$

Recall that for a hyperelliptic curve $\mathcal{C}/\mathbb{F}_q$ of genus g the following relation holds: $|\mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C})| \sim q^g$. As a consequence, the use of higher genus curves can be advantageous, because the complexity of the generic attacks on the discrete logarithm problem is then $O(\sqrt{q^g})$. In other words, the divisor class group of a higher genus curve has a large group order, while arithmetic is kept in the relatively small finite field. However, we shall see that the discrete logarithm problem on curves with genus > 4 is vulnerable to a non-generic attack.

4.5.1 Attacks

Just as with the elliptic curve case, there are several attacks specifically tailored to operate in the divisor class group of a hyperelliptic curve as well.

Frey-Rück attack

The Frey-Rück attack [44] is a generalization of the MOV attack on ECDLP. Instead of using the Weil pairing we use the hyperelliptic Tate pairing to transport an instance of the discrete logarithm problem in the divisor class group to an instance of the discrete logarithm problem in the multiplicative group of the finite field over which the curve is defined. Again, solving the discrete logarithm problem in the latter group is only feasible if the embedding degree of the curve is small enough.

Rück attack

The Rück attack [**?**] is a generalization of the anomalous attack on ECDLP. Let $\mathcal{C}/\mathbb{F}_{p^n}$ be a hyperelliptic curve. If the divisor class group of C has a subgroup of order p, then the discrete logarithm problem can be solved in this subgroup using at most $O(\log p)$ operations in $\mathbb{F}_{p^n}$. The actual attack will not be described here, as it requires the theory beyond the scope of this thesis.

Index-Calculus

Contrary to elliptic curves, there is an analogue of the index-calculus attack for hyperelliptic curves [4]. The effectiveness of this attack depends on the size of the genus compared to the size of the finite field. A description of this attack will be omitted here, but it is important to note that it is significantly faster than the square root methods on hyperelliptic curves of genus ≥ 4. For this reason, only hyperelliptic curves of genus 2 and 3 should be used for cryptographic purposes.

4.6 Practical Considerations

In this section we will compare the security of elliptic and hyperelliptic curve cryptosystems with the security of conventional cryptosystems when both are based on the discrete logarithm problem. With conventional we mean the group of choice is the multiplicative group of a finite field, though it also provides an indication of the security of other cryptosystems such as RSA, since the best known

algorithms for integer factorization and the best known algorithms for computing discrete logarithms in a finite field are of the same complexity.

From the theorem of Weil we know the divisor class group of an hyperelliptic curve $\mathcal{C}/\mathbb{F}_q$ of genus g has at most $(\sqrt{q}+1)^{2g}$ elements. Using this upper bound the running time of the fastest discrete logarithm methods is

$$O\big((\sqrt{q}+1)^g\big).$$

The running time of the index-calculus attack in a finite field $\mathbb{F}_p$ is [46]

$$O\big(\exp(c(\ln p)^{1/3}(\ln\ln p)^{2/3})\big).$$

Comparing these running times allows us to derive the size of the finite field to use in both situations that yield the same level of security. Since the group elements are represented using finite field elements, this also indicates the key size. For example, if we use a 1024-bit key in conventional methods, we take $p = 2^{1024}$ and solve the equation for q and $g = 1$ to obtain a key size of 174 bits for use in elliptic curve cryptography, and for $g = 2$ to obtain a key size of 80 bits for use in hyperelliptic (genus 2) curve cryptography. A comparison of key sizes for genus $1, 2$ and 3 curves needed to match security levels for the corresponding key size used in conventional cryptography is listed in 4.1. These numbers show that (hyper)elliptic curve based cryptography becomes relatively more advantageous as the key size increases.

Table 4.1: Key Sizes (in bits)

Conventional	Elliptic Curve	Genus 2 Curve	Genus 3 Curve
256	94	47	32
512	128	64	43
1024	174	87	58
2048	234	117	78
4096	313	157	105
8192	417	209	139
16384	554	277	185

Chapter 5

Pairing-Based Cryptography

In this chapter we describe how pairings give rise to a plethora of mathematically hard problems which can be used as the basis of a cryptographic scheme.

5.1 Abstract Pairings

It is possible to create cryptographic primitives based purely on the properties of an abstract pairing, without thinking about the actual instantiations of the groups and the mapping. The advantage of this is that one can focus solely on the cryptographic aspects of the systems and deal with the mathematical implementation later. However, in doing so it is also easy to make invalid assumptions about certain properties of pairings. Examples include: hashing onto the associated groups can be done efficiently, elements of the groups can be represented efficiently or the group operation itself can be implemented efficiently. This issue is discussed in [24]. In the first sections of this chapter we will consider abstract pairings and the problems arising from them. In a later section we discuss suitable choices of groups and pairings.

Let G_1, G_2, G_T be cyclic groups of large prime order q. We write G_1 and G_2 additively and G_T multiplicatively. Recall that a pairing is a mapping

$$\hat{e} : G_1 \times G_2 \to G_T$$

satisfying the property of bilinearity, which means the following should hold:

$$\hat{e}(aP, bR) = \hat{e}(P, R)^{ab}, \text{ for all } P \in G_1, Q \in G_2 \text{ and all } a, b \in \mathbb{Z}$$

A pairing is *admissible* if the mapping is also *non-degenerate* and *computable*. Non-degeneracy means the mapping cannot be the trivial map which sends every pair of elements of G_1 and G_2 to the identity element of G_T. Because all are groups of prime order, it follows that if P is a generator of G_1 and Q is a generator of G_2, then $e(P, Q)$ is a generator of G_T. A mapping is said to be *computable* if an algorithm exists which can efficiently compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$. If $G_1 = G_2$ then the pairing is said to be *symmetric*. Otherwise it is said to be *asymmetric*.

5.2 Complexity Assumptions

From now on, let G_1, G_2 and G_T be cyclic groups of prime order n, and $\hat{e} : G_1 \times G_2 \to G_T$ a bilinear pairing. We will always assume it is non-degenerate and efficiently computable.

5.2.1 Bilinear Diffie-Hellman

The bilinear version of the Diffie-Hellman problem comes in a variety of flavours [50].

Originally, the *Bilinear Diffie-Hellman Problem* (BDH) is defined in the setting of a symmetric pairing of cyclic groups (i.e. $G_1 = G_2$). It is the problem of computing $\hat{e}(P,P)^{abc}$ when $P, aP, bP, cP \in G_1$ are given. This problem is sometimes also referred to as the *computational bilinear Diffie-Hellman problem* (CBDH).

Just as with the regular Diffie-Hellman Problem there is also an easier version, known as the *decision bilinear Diffie-Hellman problem* (DBDH). This is the problem of deciding if $\hat{e}(P,P)^{abc} = r$ when $P, aP, bP, cP \in G_1$ and $r \in G_T$ are given. In a modified version the problem is to decide if $\hat{e}(P,P)^{ab/c} = r$ instead.

These problems can be generalized to the setting of asymmetric pairings. Assume $G_1 \neq G_2$ and let G_1 be generated by P_1 and G_2 be generated by P_2. Given values $i, j, k \in \{1,2\}$, the most general version ($\mathtt{BDH}_{i,j,k}$) is defined as follows. Given $P_1, P_2, aP_i, bP_j, cP_k$, compute $\hat{e}(P_1,P_2)^{abc}$. Some articles mention the BDH-1 and BDH-2 problems, which are equivalent to the $\mathtt{BDH}_{1,1,2}$ and the $\mathtt{BDH}_{1,2,2}$ problems, respectively.

Some pairing based protocols require the existence of an efficiently computable group homomorphism $\varphi : G_2 \to G_1$ for the security proof to hold. The $\mathtt{BDH}^{\varphi}_{i,j,k}$ problem is defined the same as the $\mathtt{BDH}_{i,j,k}$ problem, except that the adversary now has access to an oracle which computes values under the isomorphism φ.

Given $j, k \in \{1,2\}$, the *(bilinear) co-Diffie-Hellman Problem* ($\mathtt{coBDH}_{j,k}$) is the problem of computing $e(P_1,P_2)^{abc}$ when aP_1, aP_2, bP_j and cP_k are given.

The following reductions between these problems exist [50] (where $A \longrightarrow B$ means: if there exists an efficient algorithm to solve A, there exists an efficient algorithm to solve B):

- $\mathtt{BDH}_{i,j,k} \longleftrightarrow \mathtt{BDH}_{i',j',k'}$ and $\mathtt{BDH}^{\varphi}_{i,j,k} \longleftrightarrow \mathtt{BDH}^{\varphi}_{i',j',k'}$ if $i+j+k = i'+j'+k'$.
- $\mathtt{BDH}^{\varphi}_{i,j,k} \longrightarrow \mathtt{BDH}^{\varphi}_{i',j',k'}$ if $i+j+k \leq i'+j'+k'$
- $\mathtt{BDH}_{i,j,k} \longrightarrow \mathtt{BDH}^{\varphi}_{i,j,k}$
- $\mathtt{BDH}_{i,j,k} \longleftarrow \mathtt{BDH}^{\varphi}_{i,j,k}$ if φ exists and is efficiently computable
- $\mathtt{BDH}_{i,j,k} \longrightarrow \mathtt{coBDH}_{j,k} \longrightarrow \mathtt{BDH}^{\varphi}_{2,j,k}$
- $\mathtt{CDH} \longrightarrow \mathtt{BDH}_{i,j,k}$ for every $i, j, k \in \{1,2\}$

5.2.2 Gap Diffie-Hellman Groups

A *gap Diffie-Hellman group* is a group where the CDH problem is hard, but where the DDH problem is easy [12]. Such a group can be used to construct a signature scheme (see the next chapter). It is possible to realize a gap Diffie-Hellman group using a bilinear pairing. If $\hat{e} : G_1 \times G_1 \rightarrow G_T$ is a bilinear pairing, then $\texttt{DDH}_{G_1}$ is easy. Indeed, let $P, aP, bP, cP \in G_1$. Then due to bilinearity $\hat{e}(aP, bP) = \hat{e}(cP, P) \iff \hat{e}(P, P)^{ab} = \hat{e}(P, P)^{c} \iff ab = c$. The CDH problem in G_1 is still assumed to be hard, making G_1 a gap Diffie-Hellman group.

5.2.3 Pairing Inversion

The following can be found in [22]. It is important to investigate the hardness of the problem of inverting a pairing, as this has consequences for the security of cryptosystems based on pairings, and even on cryptosystems based on exponentiation in finite fields. In this section we describe some of the consequences of being able to invert pairings.

The *fixed argument pairing inversion 1 problem* (FAPI-1) is the problem of finding a $Q \in G_1$ such that $e(P, Q) = r$ when $P \in G_1$ and $r \in G_T$ are given. Similarly, the *fixed argument pairing inversion 2 problem* (FAPI-2) is the problem of finding a $P \in G_1$ such that $e(P, Q) = r$ when $Q \in \mathbb{G}_1$ and $r \in G_T$ are given. A pairing e is said to be *strong-invertible* with respect to P if we can efficiently solve FAPI-1 for any $r \in G_T$, when $P \in G_1$ is given. Similarly, e is said to be strong-invertible with respect to Q if we can efficiently solve FAPI-2 for any $r \in G_T$, when $Q \in G_2$ is given.

The *generalized pairing inversion problem* (GPI) is the problem of finding P and $Q \in G_1$ such that $e(P, Q) = r$ when $r \in G_T$ is given. A pairing e is said to be *weak-invertible* if we can efficiently solve GPI for any $r \in G_T$.

To see why it is important to investigate the difficulty of the pairing inversion problems, consider the following reductions:

- FAPI-1 $\longrightarrow$ GPI
- FAPI-2 $\longrightarrow$ GPI
- FAPI-1 $\longrightarrow \texttt{DDH}_{G_1}$
- FAPI-2 $\longrightarrow \texttt{DDH}_{G_2}$
- FAPI-1 $\longrightarrow$ BDH-1
- FAPI-2 $\longrightarrow$ BDH-2
- FAPI-1 and FAPI-2 $\longrightarrow \texttt{CDH}_{G_1}$ and $\texttt{CDH}_{G_2}$ and $\texttt{CDH}_{G_T}$

Computing pairings is done in two stages. In the first stage a divisor is evaluated in a function using Miller's Algorithm. In the second stage the result is exponentiated. However, some pairings can be computed without the exponentiation, and in this case the hardness of the pairing inversion problem depends entirely on the hardness of inverting Miller's algorithm. In these cases it is shown

that inverting Miller's algorithm is hard. On the other hand there are cases where inverting the exponentiation is hard but where inverting Miller's algorithm is easy (see [22]). In other words, it seems that inverting pairings in two stages is infeasible. It is also shown in [22] that methods inverting a pairing in a single step are infeasible. Thus, so far pairing based protocols seem to be secure.

5.3 Choosing groups and mappings

When a cryptographic protocol requires a symmetric pairing, one should choose a supersingular curve with distortion map. In [10] an example of such a curve and distortion map is given. The curve $\mathcal{E} : y^2 = x^3 + 1$ with $p \equiv 2 \pmod 3$ is a supersingular curve. To obtain a distortion map, let $1 \neq \zeta \in \mathbb{F}_{p^2}$ be a solution of $x^3 - 1 = 0 \in \mathbb{F}_{p^2}$. The distortion map $\varphi : \mathcal{E} \to \mathcal{E}$ is the mapping $(x, y) \mapsto (\zeta x, y)$. It takes a point $P \in \mathcal{E}(\mathbb{F}_p)$ to a point $\varphi(P) \in \mathcal{E}(\mathbb{F}_{p^2})$ such that P and $\varphi(P)$ are linearly independent. The modified Weil or Tate pairing as described in chapter 3 can be used as a symmetric pairing.

On the other hand, when a cryptographic protocol requires an asymmetric pairing, then one should choose an ordinary curve. There exist several methods of constructing ordinary curves with prescribed embedding degree. These methods are summarized in [19].

Chapter 6

Pairing-Based Protocols

In this chapter we describe some of the more important cryptographic protocols based on bilinear pairings. In the first section we describe a protocol by Joux which allows three parties to share a secret key in one round. In the second section we focus on identity based encryption, which is probably the most well known application of pairings. In the last section we describe two short signature schemes based on pairings. This is however just a tiny selection of the available pairing based cryptographic protocols. Pairings have proven to be applicable far beyond fundamental cryptographic primitives such as encryption and signatures.

6.1 Three-Party Key Exchange

The Diffie-Hellman Key Exchange protocol can be extended to three parties, but this would require multiple rounds. Joux [32] described a protocol based using a bilinear pairing where only one round is needed to establish a secret key between three parties. The protocol assumes all parties agreed in advance on two groups G_1 and G_T, an element $P \in G_1$ and a bilinear map $e : G_1 \times G_1 \to G_T$. Assuming the hardness of the Bilinear Diffie-Hellman problem in these groups, key agreement is done using the following steps:

1. Alice generates a random positive integer a. The information she broadcasts is:

$$aP.$$

2. Now Bob also generates a random positive integer b. The information he broadcasts is:

$$bP$$

3. And finally, Carol generates a random positive integer c. The information she broadcasts is:

$$cP$$

After these steps everyone can compute $e(P, P)^{abc}$, because by bilinearity we have

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

This shared secret $e(P,P)^{abc}$ cannot be recovered without solving BDH, because any eavesdropper watching the insecure channel only has the following information, which is an instance of BDH:

$$G_1, G_T, e, P, aP, bP \text{ and } cP$$

6.2 Identity Based Encryption

6.2.1 Introduction

Identity-based encryption (IBE) is probably the most celebrated application of pairings in cryptography. In 1984 Shamir [47] envisioned a cryptographic scheme in which the public key is the identity of the user (e.g. his e-mail address) instead of some random generated number. Any string can be used as the public key, as long as it undeniably identifies the user. The obvious advantage of this is that it eliminates the need for users to look up public keys in a directory and the use of certificates binding the public key to an identity. Since the public key cannot be generated, it follows that the corresponding private key needs to be derived from the public key. Because of this property there is a need for a trusted third party that extracts private keys using another secret which nobody else knows. This third party is known as the *key generation center*. Users can obtain the private key corresponding to their identity from the key generation center over a secure channel, once the authenticity of the user is established.

Even though the concept of IBE is now over two decades old, the first working implementation was not realized until 2001, by Boneh and Franklin [10]. Since it is based on pairings and single-handedly responsible for the sudden enormous rise of interest in pairings in cryptography, we will be focusing on this system in this section.

Definition 6.1. Formally, an identity based encryption scheme IBE is specified by four randomized algorithms $(\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D})$ [1]:

- $\mathcal{S}$ (setup). This algorithm takes a security multiplier k as input and returns params (the mathematical objects used in encryption, decryption and key extraction including the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$) and master-key (which will be used by the key generation center to extract private keys). The system parameters are public as they are required for encryption, and the master key is only known to the key generation center.

  Notation: $\langle \mathsf{params}, \mathsf{master\text{-}key} \rangle \leftarrow \mathcal{S}(1^k)$.

- $\mathcal{X}$ (extract). This algorithm takes an identity $\mathsf{ID} \in \{0,1\}^*$, params and master-key as input. It returns the corresponding private key d_{ID}.

  Notation: $d_{\mathsf{ID}} \leftarrow \mathcal{X}(\mathsf{params}, \mathsf{master\text{-}key}, \mathsf{ID})$.

- $\mathcal{E}$ (encrypt). This algorithm takes as input a message $M \in \mathcal{M}$, an identity $\mathsf{ID} \in \{0,1\}^*$ and params. It returns the ciphertext $C \in \mathcal{C}$ which is the encryption of M with the public key ID.

  Notation: $C \leftarrow \mathcal{E}(\mathsf{params}, \mathsf{ID}, M)$.

- $\mathcal{D}$ (decrypt). This algorithm takes as input a ciphertext $C \in \mathcal{C}$, a private key d_{ID} (which belongs to an identity $\mathsf{ID} \in \{0,1\}^*$) and params. It returns the plaintext $M \in \mathcal{M}$ which is the decryption of C with the private key d_{ID}.

  Notation: $M \leftarrow \mathcal{D}(\mathsf{params}, d_{\mathsf{ID}}, C)$.

We also require the scheme to be *consistent* [1]:

$$\forall M \in \mathcal{M} : \mathcal{D}(\mathsf{params}, d_{\mathsf{ID}}, \mathcal{E}(\mathsf{params}, \mathsf{ID}, M)) = M \iff d_{\mathsf{ID}} = \mathcal{X}(\mathsf{params}, \mathsf{master\text{-}key}, \mathsf{ID}).$$

Applications of identity based encryption

As noted in [10], one advantage of identity based encryption over classical public key cryptography is that key revocation is much easier. For example, if daily key expiration is desired the system could be setup in such a way that identities are padded with the current date (i.e. Alice sends an encrypted message to Bob using the public key "bob@company.com — [current-date]"). The private key generator would then only allow Bob on that specific date to extract the corresponding private key. This would also allow Alice to send a message to Bob he can only open on a future date, possibly before he even joined the network. In a traditional public key infrastructure this would be impossible since the public and private key are generated simultaneously.

The same mechanism can be used to easily manage user credentials in a corporate setting. If Alice sends a message encrypted using the public key "bob@company.com — [date] — clearance=secret", then Bob would only be able to decrypt the message if he has the required security clearance on the specified date. The credentials can be easily granted and revoked by the private key generator.

Another interesting application of IBE noted in [10] is delegation of decryption keys. In this application the setting is a bit different. Suppose Bob runs his own IBE infrastructure and controls the private key generation center. Instead of using Bob's identity as a public key, Alice now uses the current date as a public key when sending an encrypted message. Since Bob runs the private key generator he can extract the private key needed to decrypt messages encrypted by any public key. If Bob goes on a trip and takes his laptop, he can extract the private key for every day on the trip and install these on his laptop instead of the master key. If the laptop is then stolen, only the keys usable for the duration of the trip are compromised.

6.2.2 Security models

The security proofs of the cryptographic schemes that are covered in this chapter make use of several different notions of security. In this section we will describe the security notions common in public key cryptography, and show how they can be adapted to suit identity based encryption.

Notions of security in public key cryptography

In defining notions of security [2] it is useful to distinguish between *security goals* and *attack models*. The security goal we will consider is *indistinguishability of encryptions* (IND), which intuitively means

an adversary should not be able to retrieve any information about a plaintext M when he is given the corresponding ciphertext C. The attack models we will consider are:

1. *Chosen-plaintext attack* (CPA). In this attack model the adversary can encrypt arbitrary plaintexts at will.

2. *Chosen-ciphertext attack* (CCA1). In this attack model the adversary not only is able to encrypt arbitrary plaintexts at will, he also gets access to a decryption oracle allowing him to obtain the decryption of any ciphertext, but only before being given the challenge ciphertext.

3. *Adaptive chosen-ciphertext attack* (CCA2). In this attack model the adversary is again able to encrypt plaintexts and is given access to a decryption oracle, but in this attack model the adversary has access to the oracle even after being challenged on the ciphertext.

These security notions can be formally defined with a game simulating an attack against a public key encryption scheme. We will not give the definitions here, but instead intuitively describe how to extend these notions to the setting of identity based encryption and formally define those.

Notions of security in identity based encryption

The security goals in identity based encryption remain the same. However, to keep security as tight as possible, in the attack models we will in addition allow the adversary access to a key extraction oracle which returns the private key corresponding to any identity. Also, the adversary is challenged on an identity of his choice, as opposed to some randomly generated public key. Obviously, the identity he wishes to be challenged on may not have been used in any key extraction oracle query. Similar to public key cryptography, IND-ID-CCA2 is the strongest and most desirable notion of security. This is justified in [1].

Definition 6.2. We can define IND-ID-CCA2 formally using the following game between a challenger and an adversary which is used to simulate an attack against a scheme IBE, which is a straightforward extension of the formal definition for IND-CCA2.

- **Setup.** The challenger runs $\mathcal{S}(1^k)$ for a given security multiplier k, gives the adversary params and keeps master-key to itself.

- **Phase 1.** The adversary is allowed to do decryption and key extraction queries. In a key extraction query the adversary sends the challenger a public key ID_i. The challenger computes $d_{\mathsf{ID}_i} \leftarrow \mathcal{X}(\mathsf{params}, \mathsf{master\text{-}key}, \mathsf{ID}_i)$ and sends it to the adversary. In a decryption query the adversary sends the challenger a public key ID_i and a ciphertext C_i. The challenger first computes $d_{\mathsf{ID}_i} \leftarrow \mathcal{X}(\mathsf{params}, \mathsf{master\text{-}key}, \mathsf{ID}_i)$ and then $M_i \leftarrow \mathcal{D}(\mathsf{params}, d_{\mathsf{ID}_i}, C_i)$, which is sent to the adversary. The adversary does not have to commit to the list of queries in advance; each query can depend on the result of former queries. The adversary decides when phase 1 is over.

- **Challenge.** The adversary picks a public key ID and two messages $M_0, M_1 \in \mathcal{M}$ on which he wants to be challenged. The public key may not have appeared in one of the key extraction queries of phase 1. The challenger picks a random bit $b \in \{0, 1\}$ and obtains $C \leftarrow \mathcal{E}(\mathsf{params}, \mathsf{ID}, M_b)$. C is sent to the adversary.

- **Phase 2.** The same as Phase 1, but in the key extraction queries ID is not allowed and in the decryption queries ID and C are not allowed simultaneously. The adversary decides when phase 2 is over.
- **Guess.** The adversary outputs his guess $b' \in \{0,1\}$ and wins the game if $b = b'$.

If an adversary $\mathcal{A}$ takes pure random guesses then $\Pr[b = b'] = \frac{1}{2}$. Thus, the *advantage* of $\mathcal{A}$ in attacking the scheme is defined as:

$$\mathrm{Adv}_{\mathcal{A}}(k) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

where the probability is over the random bits used by both the challenger and the adversary.

A function $\varepsilon : \mathbb{N} \to \mathbb{R}$ is said to be *negligible* if:

$$\forall c \geq 0 \;\; \exists k_c \in \mathbb{N} \;\; \forall k \geq k_c : \;\; \varepsilon(k) \leq k^{-c}.$$

We consider IBE to be *secure* in the sense of $\mathsf{IND\text{-}ID\text{-}CCA2}$ if $\mathrm{Adv}_{\mathcal{A}}$ is negligible as a function of k for every polynomially bounded adversary $\mathcal{A}$.

Definition 6.3. $\mathsf{IND\text{-}ID\text{-}CPA}$ is defined using a similar game. This time the adversary does not have access to a decryption oracle in phase 1 and phase 2.

- **Setup.** The challenger runs $\mathcal{S}(1^k)$ for a given security multiplier k, gives the adversary params and keeps $\mathsf{master\text{-}key}$ to itself.
- **Phase 1.** The adversary is allowed to do key extraction queries. In a key extraction query the adversary sends the challenger a public key ID_i. The challenger computes $d_{\mathsf{ID}_i} \leftarrow \mathcal{X}(\mathsf{params}, \mathsf{master\text{-}key}, \mathsf{ID}_i)$ and sends it to the adversary. The adversary does not have to commit to the list of queries in advance; each query can depend on the result of former queries. The adversary decides when phase 1 is over.
- **Challenge.** The adversary picks a public key ID and two messages $M_0, M_1 \in \mathcal{M}$ on which he wants to be challenged. The public key may not have appeared in one of the key extraction queries of phase 1. The challenger picks a random bit $b \in \{0,1\}$ and obtains $C \leftarrow \mathcal{E}(\mathsf{params}, \mathsf{ID}, M_b)$. C is sent to the adversary.
- **Phase 2.** The same as Phase 1, but in the key extraction queries ID is not allowed. The adversary decides when phase 2 is over.
- **Guess.** The adversary outputs his guess $b' \in \{0,1\}$ and wins the game if $b = b'$.

Again, the *advantage* of $\mathcal{A}$ in attacking the scheme is defined as:

$$\mathrm{Adv}_{\mathcal{A}}(k) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

where the probability is over the random bits used by both the challenger and the adversary.

We consider IBE to be *secure* in the sense of $\mathsf{IND\text{-}ID\text{-}CPA}$ if $\mathrm{Adv}_{\mathcal{A}}$ is negligible as a function of k for every polynomially bounded adversary $\mathcal{A}$. This is also known as *semantic security*.

Complexity assumptions

As with any cryptographic scheme, the security of identity based encryption systems depends on the hardness of certain mathematical problems. One of the problems very often associated with pairing based protocols is the BDH problem, which was discussed in the previous chapter. However, we still have to define precisely what it means when we assume BDH is "hard". We use the following definitions, as given in [10].

Definition 6.4. An algorithm $\mathcal{A}$ is said to have *advantage* ε in solving BDH in $\langle G_1, G_T, \hat{e}\rangle$ if

$$\Pr\left[\mathcal{A}(P, aP, bP, cP) = \hat{e}(P,P)^{abc} \,\middle|\, a, b, c \leftarrow (\mathbb{Z}/q\mathbb{Z})^*, P \leftarrow G_1^*\right] \geq \varepsilon.$$

The notation $x \leftarrow X$ means that x is randomly sampled from a probability distribution X. If X is a finite set then x is sampled uniformly at random from X.

If no polynomially bounded algorithm has non-negligible advantage in solving BDH in $\langle G_1, G_T, \hat{e}\rangle$ then we say BDH is hard in $\langle G_1, G_T, \hat{e}\rangle$. For the asymptotic formulation we require an algorithm that generates the groups.

Definition 6.5. A *bilinear group generator* is a randomized algorithm with the following properties:

1. It takes a security parameter $k \in \mathbb{Z}^+$.
2. It runs in polynomial time in k.
3. It outputs a prime number q, the description of three groups G_1, G_2, G_T of order q and the description of an admissible bilinear map $\hat{e} : G_1 \times G_2 \to G_T$.

The description of the groups G_1, G_2 and G_T should include generators P_1, P_2 and g respectively, and a polynomial time (in k) algorithm to compute the group action. The size of the groups should depend on the size of k (i.e. q should be a random prime of k bits). The description of the bilinear map $\hat{e}$ should include a polynomial time algorithm to compute $\hat{e}$. If $\mathcal{G}$ is a bilinear group generator, then we denote the output of $\mathcal{G}$ by $\mathcal{G}(1^k) = \langle q, G_1, G_2, G_T, \hat{e}\rangle$.

Definition 6.6. Assuming the existence of a bilinear group generator $\mathcal{G}$, we can formalize the assumption of the hardness of BDH. The advantage of an algorithm $\mathcal{A}$ in solving BDH for $\mathcal{G}$ is defined as follows:

$$\mathrm{Adv}^{\mathtt{BDH}}_{\mathcal{G},\mathcal{A}}(k) = \Pr\left[\mathcal{A}(q, G_1, G_2, G_T, \hat{e}, P, aP, bP, cP) = \hat{e}(P,P)^{abc} \,\middle|\, \begin{array}{l}\langle q, G_1, G_2, G_T, \hat{e}\rangle \leftarrow \mathcal{G}(1^k),\\ P \leftarrow G_1^*, a, b, c \leftarrow (\mathbb{Z}/q\mathbb{Z})^*\end{array}\right]$$

However, it is usually said an algorithm has advantage $\varepsilon(k)$ in solving BDH for $\mathcal{G}$ if $\mathrm{Adv}^{\mathtt{BDH}}_{\mathcal{G},\mathcal{A}}(k) \geq \varepsilon(k)$. If for any randomized polynomial time (in k) algorithm $\mathcal{A}$ we have that $\mathrm{Adv}^{\mathtt{BDH}}_{\mathcal{G},\mathcal{A}}$ is a negligible function, then $\mathcal{G}$ is said to satisfy the BDH-*assumption* and BDH is said to be *hard* in groups generated by $\mathcal{G}$.

Similarly, we can define the $\mathtt{BDH}_{i,j,k}$ assumption.

Random oracles

The identity based encryption scheme we will be focusing on uses random oracles as hash functions. A *random oracle* is a function which maps inputs to truly uniformly chosen outputs, but always maps the same input to the same output every time it is invoked. Random oracles are very often used to represent an ideal hash function $H : \{0,1\}^* \rightarrow \{0,1\}^k$, where $\{0,1\}^*$ is the space of finite binary strings. Of course, random oracles do not really exist in practice, but they can be very useful in security proofs of cryptographic schemes. The idea is to design a cryptosystem which uses random oracles for hash functions, prove its security and then replace the random oracles with real hash functions. Obviously, the proof of security does not hold formally for the system with real hash functions, but the instantiation of the oracle by a real hash function *should* merely be a heuristic, trusted to be secure based on experience with the actual hash function used, because breaking the protocol is then thought to reveal previously unknown flaws of these well studied hash functions. When a cryptographic system using random oracles is proven to be secure in this sense it is said to be secure in the *random oracle model*. This paradigm is developed by Bellare and Rogaway [3] and has led to the design of several protocols that are both efficient and provably secure.

Even though the random oracle methodology seems to be very useful by allowing the designer to focus on the structure of the protocol without having to get lost in details, it seems it also allows the designer to arrive at wrong conclusions regarding the security of the protocol. In [14] it is shown there exist cryptographic systems that are secure in the random oracle model but fail to have any real secure implementation at all. In fact, they show that any protocol secure in the random oracle model can be slightly adapted to have no secure implementation at all, while retaining the level of security in the random oracle model. They arrive at the conclusion that the random oracle methodology is not a sound method of abstracting protocols for use in security analysis. For this reason the authors of the first identity based encryption scheme encouraged the search for a scheme which can be proven to be secure without random oracles.

6.2.3 The Boneh-Franklin identity based encryption scheme

The first fully functional identity-based encryption scheme was given by Boneh and Franklin [10]. In this section we will describe this system. In the original paper the authors construct the system in stages. They first describe a simpler version of the scheme, BasicIdent, which is secure against chosen plaintext attacks in the random oracle model. The system is then transformed using a technique of Fujisaki and Okamoto [20] to a system FullIdent which is shown to be secure against adaptive chosen ciphertext attacks in the random oracle model, assuming the hardness of BDH in the groups and pairing involved.

Definition

The BasicIdent scheme is defined by the following algorithms:

$\mathcal{S}$ (setup). Let k be the desired security parameter.

- Generate two cyclic groups G_1 and G_T of prime order q and a pairing $\hat{e} : G_1 \times G_1 \to G_T$. The size of these groups should depend on the security parameter k. In addition, pick a random generator $P \in G_1$, a random element $s \in (\mathbb{Z}/q\mathbb{Z})^*$ and let $P_{pub} = sP$.
- Pick cryptographic hash functions $H_1 : \{0,1\}^* \to G_1^*$ and $H_2 : G_T \to \{0,1\}^n$ for some $n \in \mathbb{Z}$. For the security proof to work, these hash functions need to be random oracles.
- Let $\mathsf{params} = \langle q, G_1, G_T, \hat{e}, n, P, P_{pub}, H_1, H_2 \rangle$ and $\mathsf{master\text{-}key} = s$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = G_1^* \times \{0,1\}^n$.
- Return $\langle \mathsf{params}, \mathsf{master\text{-}key} \rangle$.

$\mathcal{X}$ (extract). Let $\mathsf{ID} \in \{0,1\}^*$ be the public key to extract the corresponding private key from and let s be master-key.

- Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$ and $d_{\mathsf{ID}} = sQ_{\mathsf{ID}} \in G_1^*$.
- Return d_{ID}.

$\mathcal{E}$ (encrypt). Let $\mathsf{ID} \in \{0,1\}^*$ be the public key to use for encryption and let $M \in \mathcal{M}$ be the plaintext to encrypt.

- Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$.
- Pick a random $r \in (\mathbb{Z}/q\mathbb{Z})^*$.
- Compute $g_{\mathsf{ID}} = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in G_T$.
- Compute $C = \langle rP, M \oplus H_2(g_{\mathsf{ID}}^r) \rangle$.
- Return C.

$\mathcal{D}$ (decrypt). Let $C = \langle U, V \rangle \in \mathcal{C}$ be encrypted with public key ID and let d_{ID} be the corresponding private key.

- Compute $V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U))) = M$.
- Return M.

The key element here is that both the sender and receiver are able to compute $\hat{e}(Q_{\mathsf{ID}}, P)^{rs}$, which is used to mask the plaintext. The sender knows sP and generates r so he can compute $\hat{e}(Q_{\mathsf{ID}}, rsP) = \hat{e}(Q_{\mathsf{ID}}, P)^{rs}$. The receiver on the other hand knows sQ_{ID} and rP so he is able to compute $\hat{e}(sQ_{\mathsf{ID}}, rP) = \hat{e}(Q_{\mathsf{ID}}, P)^{rs}$. Since $Q_{\mathsf{ID}} = tP$ for some $t \in (\mathbb{Z}/q\mathbb{Z})^*$ it follows that anyone who is able to recover the plaintext can compute $\hat{e}(P, P)^{rst}$ from P, rP, sP and tP, which is equivalent to solving an instance of the BDH problem.

The hash function H_1 is used to generate $Q_{\mathsf{ID}} = tP$, where t should be a secret, so it follows that it should be possible to hash onto G_1. This means it should be possible to take a random element from G_1 without having computed the discrete logarithm of this element with respect to P.

Theorem 6.1. BasicIdent *is consistent.*

Proof. Let $\mathsf{ID} \in \{0,1\}^*$, $M \in \mathcal{M}$ be a message, $d_{\mathsf{ID}} = \mathcal{X}(\mathsf{params}, s, \mathsf{ID}) = sQ_{\mathsf{ID}}$ a private key and $\langle U, V \rangle = \mathcal{E}(\mathsf{params}, \mathsf{ID}, M)$ the encryption of M. Then

$$U = rP \text{ and } V = M \oplus H_2(\hat{e}(Q_{\mathsf{ID}}, P_{pub})^r) = M \oplus H_2(\hat{e}(Q_{\mathsf{ID}}, sP)^r$$

Consequently,

$$\begin{aligned}\mathcal{D}(\mathsf{params}, d_{\mathsf{ID}}, \langle U, V\rangle) &= V \oplus H_2\big(\hat{e}(d_{\mathsf{ID}}, U)\big)\\ &= M \oplus H_2\big(\hat{e}(Q_{\mathsf{ID}}, sP)^r\big) \oplus H_2\big(\hat{e}(sQ_{\mathsf{ID}}, rP)\big)\\ &= M.\end{aligned}$$

□

Using a technique due to Fujisaki-Okamoto [20], BasicIdent is transformed into a system with a stronger level of security. This system, named FullIdent, is described by the following algorithms:

$\mathcal{S}$ (setup). Let k be the desired security parameter.

- Generate two cyclic groups G_1 and G_T of prime order q and a pairing $\hat{e} : G_1 \times G_1 \to G_T$. The size of these groups should depend on the security parameter k. In addition, pick a random generator $P \in G_1$, a random element $s \in (\mathbb{Z}/q\mathbb{Z})^*$ and let $P_{pub} = sP$.
- Pick cryptographic hash functions $H_1 : \{0,1\}^* \to G_1^*$, $H_2 : G_T \to \{0,1\}^n$ for some $n \in \mathbb{Z}$, $H_3 : \{0,1\}^n \times \{0,1\}^n \to (\mathbb{Z}/q\mathbb{Z})^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$. For the security proof to work, these hash functions need to be random oracles.
- Let $\mathsf{params} = \langle q, G_1, G_T, \hat{e}, n, P, P_{pub}, H_1, H_2, H_3, H_4\rangle$ and $\mathsf{master\text{-}key} = s$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = G_1^* \times \{0,1\}^n \times \{0,1\}^n$.
- Return $\langle \mathsf{params}, \mathsf{master\text{-}key}\rangle$.

$\mathcal{X}$ (extract). Let $\mathsf{ID} \in \{0,1\}^*$ be the public key to extract the corresponding private key from and let s be master-key.

- Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$ and $d_{\mathsf{ID}} = sQ_{\mathsf{ID}} \in G_1^*$.
- Return d_{ID}.

$\mathcal{E}$ (encrypt). Let $\mathsf{ID} \in \{0,1\}^*$ be the public key to use for encryption and let $M \in \mathcal{M}$ be the plaintext to encrypt.

- Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$.
- Pick a random $\sigma \in \mathcal{M}$ and let $r = H_3(\sigma, M)$.
- Compute $g_{\mathsf{ID}} = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in G_T$.
- Compute $C = \langle rP, \sigma \oplus H_2(g_{\mathsf{ID}}^r), M \oplus H_4(\sigma)\rangle$.
- Return C.

$\mathcal{D}$ (decrypt). Let $C = \langle U, V, W\rangle \in \mathcal{C}$ be encrypted with public key ID and let d_{ID} be the corresponding private key.

- Reject the ciphertext if $P \notin G_1^*$.
- Compute $V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = \sigma \oplus H_2(g_{\mathsf{ID}}^r) \oplus H_2(\hat{e}(d_{\mathsf{ID}}, rP)) = \sigma$.
- Compute $W \oplus H_4(\sigma) = M$.
- Let $r = H_3(\sigma, M)$. If $U \neq rP$ then the ciphertext is forged and should be rejected, otherwise M is the decryption of C.

- Return M.

The encryption is similar to the encryption in BasicIdent. Instead of using the pairing computation g_{ID}^r to generate a bit string that masks the plaintext, the pairing computation is used in the same way to mask another random bit string σ, which is in turn used to mask the plaintext. This construction allows the system to verify if any given ciphertext is really the result of applying the encryption algorithm to a bit string. Forged ciphertexts are thus rejected, preventing an adversary to learn any information about the plaintext he could obtain by creating slightly modified versions of a ciphertext (i.e. by flipping small amounts of bits) and examining the resulting decryptions. (In essence, this is what makes FullIdent secure against adaptive chosen ciphertext attacks.)

Theorem 6.2. FullIdent *is consistent.*

Proof. Let $\mathsf{ID} \in \{0,1\}^*$, $M \in \mathcal{M}$ be a message, $d_{\mathsf{ID}} = \mathcal{X}(\mathsf{params}, s, \mathsf{ID}) = sQ_{\mathsf{ID}}$ a private key and $\langle U, V\rangle = \mathcal{E}(\mathsf{params}, \mathsf{ID}, M)$ the encryption of M. Then

$$U = rP, V = \sigma \oplus H_2\big(\hat{e}(Q_{\mathsf{ID}}, P_{pub})^r\big), \text{ and } W = M \oplus H_4(\sigma)$$

Consequently,

$$\begin{aligned}
\mathcal{D}(\mathsf{params}, d_{\mathsf{ID}}, \langle U, V, W\rangle) &= W \oplus H_4\big(V \oplus H_2\big(\hat{e}(d_{\mathsf{ID}}, U)\big)\big) \\
&= M \oplus H_4(\sigma) \oplus H_4\big(\sigma \oplus H_2\big(\hat{e}(Q_{\mathsf{ID}}, P_{pub})^r\big) \oplus H_2\big(\hat{e}(d_{\mathsf{ID}}, U)\big)\big) \\
&= M \oplus H_4(\sigma) \oplus H_4\big(\sigma \oplus H_2\big(\hat{e}(Q_{\mathsf{ID}}, sP)^r\big) \oplus H_2\big(\hat{e}(sQ_{\mathsf{ID}}, rP)\big)\big) \\
&= M \oplus H_4(\sigma) \oplus H_4(\sigma) \\
&= M.
\end{aligned}$$

□

Security

The BasicIdent scheme is proven secure in the random oracle model against IND-ID-CPA attacks (assuming the hardness of BDH) using the following theorem. Due to its length the proof is not included here.

Theorem 6.3 (4.1 in [10])**.** *Suppose the hash functions H_1 and H_2 are random oracles. Then* BasicIdent *is a semantically secure identity based encryption scheme (*IND-ID-CPA*) assuming* BDH *is hard in groups G_1, G_T with bilinear pairing $\hat{e}$. Concretely, suppose there is a polynomially bounded* IND-ID-CPA *adversary $\mathcal{A}$ that has non-negligible advantage $\epsilon(k)$ against the scheme* BasicIdent*. Then there is an algorithm $\mathcal{B}$ that runs in polynomial time and solves* BDH *in groups G_1 and G_T with bilinear pairing $\hat{e}$ with non-negligible advantage.*

Intuitively, efficiently attacking BasicIdent allows one to efficiently solve BDH which contradicts the BDH hardness assumption. In the proof in [10] the efficiency of the security reduction is as follows. Suppose the advantage of $\mathcal{A}$ against FullIdent is ε. If $\mathcal{A}$ makes at most $q_E > 0$ private key extraction queries and $q_{H_2} > 0$ hash queries to H_2, then the advantage of $\mathcal{B}$ is at least $\frac{2\varepsilon(k)}{e(1+q_E)\cdot q_{H_2}}$, where e is the base of the natural logarithm. The running time of $\mathcal{B}$ is $O(\text{time}(\mathcal{A}))$.

The FullIdent scheme is proven secure in the random oracle model against IND-ID-CCA2 attacks (assuming the hardness of BDH) using the following theorem. Again, the proof is omitted here.

Theorem 6.4 (4.4 in [10]). *Let the hash functions H_1, H_2, H_3 and H_4 be random oracles. Then* FullIdent *is* IND-ID-CCA2*-secure assuming* BDH *is hard in groups G_1, G_T with bilinear pairing $\hat{e}$. Concretely, suppose there is a polynomially bounded* IND-ID-CCA2 *adversary $\mathcal{A}$ that has non-negligible advantage against the scheme* FullIdent*. Then there is an algorithm $\mathcal{B}$ that runs in polynomial time and solves* BDH *in groups G_1 and G_T with bilinear pairing $\hat{e}$ with non-negligible advantage.*

Again, intuitively this means efficiently attacking FullIdent allows one to efficiently solve BDH which contradicts the BDH hardness assumption. The efficiency of the security reduction is as follows. Suppose the advantage of $\mathcal{A}$ against FullIdent is ε. If $\mathcal{A}$ makes at most $q_E > 0$ private key extraction queries, at most q_D decryption queries and at most $q_{H_2}, q_{H_3}, q_{H_4}$ hash queries to H_2, H_3, H_4 repsectively, then the advantage of $\mathcal{B}$ is at least

$$2FO_{\text{adv}}\left(\frac{\epsilon(k)}{e(1+q_E+q_D)}, q_{H_4}, q_{H_3}, q_D\right)/q_{H_2}$$

and the running time of $\mathcal{B}$ is at most

$$FO_{\text{time}}(t(k), q_{H_4}, q_{H_3}),$$

where

$$FO_{\text{adv}}(\epsilon(k), q_{H_4}, q_{H_3}, q_D) = \frac{1}{2q_{H_4}+q_{H_3})}[(\epsilon(k)+1)(1-2/q)^{q_D}-1]$$

and

$$FO_{\text{time}}(t(k), q_{H_4}, q_{H_3}) = t(k) + O((q_{H_4}+q_{H_3})\cdot n),$$

where q is the order of the groups G_1 and G_T and n is the length of σ.

Generalization

The FullIdent scheme can further modified to allow the more general setting of asymmetric pairings. This allows for a greater variety of curves and we can use. We follow the construction as given in [50]. First we have to fix four variables $i, j, k, l \in \{1, 2\}$ that decide for certain elements if they are to be chosen from G_1 or G_2. These variables redefine the following sections in the protocol:

(i) The generator $P \in G_1$ is replaced with generators $P_1 \in G_1$ and $P_2 \in G_2$, and now the public parameter is chosen to be $P_{pub} = sP_i \in G_i$.

(j) The cryptographic hash function $H_1 : \{0,1\}^* \to G_1$ becomes a function $H_1 : \{0,1\}^* \to G_j$.

(k) The first element of the ciphertext tuple $rP \in G_1$ becomes $rP_k \in G_k$.

(l) The private key $d_{\mathsf{ID}} \in G_1^*$ is now chosen such that $d_{\mathsf{ID}} = sQ_{\mathsf{ID}} \in G_l$ or $d_{\mathsf{ID}} = \phi(sQ_{\mathsf{ID}}) \in G_l$.

There are certain restrictions on the actual choice of i, j, k and l. For instance, since $Q_{\mathsf{ID}} \in G_j$ and $P_{pub} \in G_i$ we generally need to have $i \neq j$, because the encryption requires a pairing computation of P_{pub} and Q_{ID}. However, if ϕ is efficiently computable, then the pairing of $\phi(P_{pub})$ and Q_{ID} or the pairing of $\phi(Q_{\mathsf{ID}})$ and P_{pub} could be used instead, also allowing the choice of $i = j = 2$. Similarly, the decryption requires a pairing computation of rP_k and d_{ID}, so we generally require $k \neq l$. If ϕ is efficiently computable we can use the pairing of $\phi(rP_k)$ and d_{ID} or the pairing of $\phi(d_{\mathsf{ID}})$ and rP_k instead,

in which case the choice of $k = l = 2$ is also allowed. Also note that if $j = 1$, then $d_{\mathsf{ID}} = sQ_{\mathsf{ID}} \in G_j$ and thus $l = 1$. If $j = 2$, then generally $l = 2$, unless ϕ is efficiently computable, in which case $d_{\mathsf{ID}} = \phi(sQ_{\mathsf{ID}}) \in G_1$ (i.e. $l = 1$) is also allowed. Finally, in the security proof (Lemmas 4.2 and 4.6 in [10]) the random oracle H_1 is simulated by computing $Q_t = b_t P_j$ and $d_t = b_t P_{pub} = b_t s P_j$, while one of the equalities $d_t = sQ_t$ or $d_t = \phi(sQ_t)$ should also hold. This means $i = j$ or if ϕ is efficiently computable $i = 2$ and $j = 1$.

Accounting for these restrictions, we can see that if ϕ is not efficiently computable, then (i, j, k, l) has to be $(2, 1, 2, 1)$. If ϕ is efficiently computable then (i, j, k, l) can be selected from

$$(2, 2, 2, 2), (2, 2, 2, 1), (2, 1, 2, 1), \text{ and } (2, 2, 1, 2).$$

In the previous chapter we concluded that $\mathtt{BDH}^{\phi}_{i,j,k}$ is harder than $\mathtt{BDH}^{\phi}_{i',j',k'}$ if $i + j + k \leq i' + j' + k'$, so it is best to choose $(2, 1, 2, 1)$ or $(2, 2, 1, 2)$. This results in the following full description of the scheme:

$\mathcal{S}$ (setup). Let k be the desired security parameter.

- Generate three cyclic groups G_1, G_2 and G_T of prime order q and a pairing $\hat{e} : G_1 \times G_2 \to G_T$. The size of these groups should depend on the security parameter k. In addition, pick a random generator $P_2 \in G_2$, a random element $s \in (\mathbb{Z}/q\mathbb{Z})^*$ and let $P_{pub} = sP_2$.
- Pick cryptographic hash functions $H_1 : \{0,1\}^* \to G_j^*$, $H_2 : G_T \to \{0,1\}^n$ for some $n \in \mathbb{Z}$, $H_3 : \{0,1\}^n \times \{0,1\}^n \to (\mathbb{Z}/q\mathbb{Z})^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$. For the security proof to work, these hash functions need to be random oracles.
- Let $\mathsf{params} = \langle q, G_1, G_2, G_T, \hat{e}, n, P_2, P_{pub}, H_1, H_2, H_3, H_4 \rangle$ and $\mathsf{master\text{-}key} = s$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = G_k^* \times \{0,1\}^n \times \{0,1\}^n$.
- Return $\langle \mathsf{params}, \mathsf{master\text{-}key} \rangle$.

$\mathcal{X}$ (extract). Let $\mathsf{ID} \in \{0,1\}^*$ be the public key to extract the corresponding private key from and let s be $\mathsf{master\text{-}key}$.

- Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$
- Compute $d_{\mathsf{ID}} = sQ_{\mathsf{ID}} \in G_j^*$
- Return d_{ID}.

$\mathcal{E}$ (encrypt). Let $\mathsf{ID} \in \{0,1\}^*$ be the public key to use for encryption and let $M \in \mathcal{M}$ be the plaintext to encrypt.

- Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$.
- Pick a random $\sigma \in \mathcal{M}$ and let $r = H_3(\sigma, M)$.
- Do one of the following:
  1. Compute $g_{\mathsf{ID}} = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in G_T$, if $j = 1$.
  2. Compute $g_{\mathsf{ID}} = \hat{e}(\phi(P_{pub}), Q_{\mathsf{ID}}) \in G_T$, if $j = 2$.
- Compute $C = \langle rP_k, \sigma \oplus H_2(g_{\mathsf{ID}}^r), M \oplus H_4(\sigma) \rangle$.
- Return C.

$\mathcal{D}$ (decrypt). Let $C = \langle U, V, W \rangle \in \mathcal{C}$ be encrypted with public key ID and let d_{ID} be the corresponding private key.

- Reject the ciphertext if $P_2 \notin G_2^*$.
- Do one of the following:
  1. Compute $V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = \sigma$, if $j = 1$.
  2. Compute $V \oplus H_2(\hat{e}(U, d_{\mathsf{ID}})) = \sigma$, if $j = 2$.
- Compute $W \oplus H_4(\sigma) = M$.
- Let $r = H_3(\sigma, M)$. If $U \neq rP$ then the ciphertext is forged and should be rejected, otherwise M is the decryption of C.
- Return M.

If ϕ is not efficiently computable then the security reduces to the $\mathtt{BDH}^{\phi}_{2,1,2}$ problem or the $\mathtt{coBDH}_{1,2}$ problem, which is a somewhat weaker result. If ϕ is efficiently computable, then the security reduces to the $\mathtt{BDH}^{\phi}_{2,1,2}$ problem or the $\mathtt{BDH}^{\phi}_{2,2,1}$ the problem, but since ϕ is not just given as an oracle to the adversary but is now actually computable, these are equal to the $\mathtt{BDH}_{2,1,2}$ problem and the $\mathtt{BDH}_{2,2,1}$ problem, respectively.

6.2.4 Other identity based encryption schemes

Following the Boneh-Franklin scheme, lots of other identity based encryption have been proposed. Some try to improve on the level of efficiency or security, others try to adapt special types of public key cryptosystems (e.g. hierarchical schemes, fuzzy schemes, etc.) to the setting of identity based encryption. In this section we give a short overview of some important systems that have been developed. For the actual construction and security proof of the systems we refer to their respective articles.

Identity based encryption without random oracles

Because the random oracle model is quite controversial, an important open problem after the construction of the Boneh-Franklin scheme was to develop an identity based encryption scheme which is provably secure in the standard model. As a first step towards this goal, Canetti et al. [13] create an identity based encryption scheme which is provably secure without random oracles, although in a slightly weaker security model. In this weakened model, known as *selective identity security*, an adversary needs to commit to the identity he wishes to attack in advance. In the standard identity based model, the adversary is allowed to adaptively choose his target identity. The security of the scheme depends on the hardness of the DBDH problem and the construction is quite inefficient.

As an improvement, Boneh and Boyen [6] created two efficient identity based encryption schemes, both provably secure in the selective-identity model and also without resorting to random oracle methodology. The first system can be extended to an efficient hierarchical identity based encryption system (see next section) and its security is based on the DBDH problem. The second system is more efficient, but its security reduces to the non-standard DBDHI problem.

A later construction due to Boneh and Boyen [7] is proven fully secure without random oracles. Its security reduces to the DBDH problem. However, the scheme is impractical and was merely given as a theoretical construct to prove that there indeed exist fully secure identity based encryption schemes without having to resort to random oracles.

Finally, Waters [51] improves on this result and constructs a modification of the scheme which is efficient and fully secure without random oracles. Its security also reduces to the DBDH problem.

Hierarchical identity based encryption

The concept of hierarchical identity based encryption was first introduced by Horwitz and Lynn [30]. In traditional public key infrastructures there is a root certificate authority, and possibly a hierarchy of other certificate authorities. The root authority can issue certificates to authorities on a lower level and the lower level certificate authorities can issue certificates to users. To reduce workload, a similar setup could be useful in the setting of identity based encryption. In identity based encryption the trusted party is the private key generator. A natural way to extend this to a two-level hierarchical based encryption is to have a root private key generator and domain private key generators. Users would then be associated with their own primitive identity plus the identity of their respective domain, both arbitrary strings. Users can obtain their private key from a domain private key generator, which in turn obtains its private key from the root private key generator. More levels can be added to the hierarchy by adding subdomains, subsubdomains, etcetera.

The first hierarchical identity based encryption scheme with an arbitrary number of levels is given by Gentry and Silverberg [26]. It is an extension of the Boneh-Franklin scheme and its security depends on the hardness of the BDH problem. It also uses random oracles.

Boneh and Boyen managed to construct a hierarchical based encryption scheme without random oracles based on the BDH problem, but it is secure in the weaker selective-ID model [6].

In the aforementioned constructions, the time needed for encryption and decryption grows linearly in the hierarchy depth, thus becoming less efficient at complex hierarchies. In [9], Boneh, Boyen and Goh give a hierarchical identity based encryption system in which the decryption time is the same at every hierarchy depth. It is selective-ID secure without random oracles and based on the BDHE problem.

Fuzzy identity based encryption

In [45], Sahai and Waters give a *fuzzy identity based encryption system*. In fuzzy identity based encryption, identities are viewed as a set of descriptive attributes, instead of a string of characters. The idea is that private keys can decrypt messages encrypted with the public key ω, but also messages encrypted with the public key ω' if $d(\omega, \omega') < \varepsilon$ for a certain metric d and a fault tolerance value ε. One valuable application of fuzzy identity based encryption is the use of biometric identities. Since two measurements of the same biometric (e.g. an iris scan) will never be exactly the same, a certain amount of error tolerance is required when using such measurements as keys. The security of the Sahai-Waters scheme reduces to the modified DBDH problem.

Identity based encryption schemes without pairings

Another identity based encryption scheme that was published around the same time as the Boneh-Franklin scheme (but turned out to be invented several years earlier) is due to Cocks. The security of the system is based on the quadratic residuosity problem modulo a composite $N = pq$ where $p, q \in \mathbb{Z}$ are prime [17]. Unfortunately, this system produces very large ciphertexts compared to the pairing based systems and thus is not very efficient.

Recently, Boneh et. al. constructed another identity based encryption system that is not based on pairings [11]. It is related to the Cocks system since the security of it is also based on the quadratic residuosity problem. The system is space efficient but encryptions are slow. It is proven secure in the random oracle model.

6.3 Short Signatures

6.3.1 Introduction

Digital signatures are used to authenticate messages in an electronic environment. The concept was first described in 1976 by Diffie and Hellman [18]. Since then, several practical signature schemes have been proposed, but RSA and DSA are the most frequently used schemes today. Both RSA and DSA generate relatively long signatures. In low-bandwidth communication environments, the signatures should be as short as possible. For example, a signature printed on a postage stamp or a signature communicated over the telephone cannot be as long as 320 bits, which is a typical keysize in today's implementation of DSA (due to the required security level). In this section we describe two pairing based signature schemes that provide much shorter signatures while offering similar levels of security.

Definition 6.7. Formally, a signature scheme is specified by four randomized algorithms $(\mathcal{I}, \mathcal{G}, \mathcal{S}, \mathcal{V})$:

- $\mathcal{I}$ (setup). This algorithm takes a security multiplier k as input and returns system parameters params.

  Notation: $\mathsf{params}\rangle \leftarrow \mathcal{I}(1^k)$.

- $\mathcal{G}$ (key generation). When invoked by user A, this algorithm takes the system parameters as input and returns a public key P_{A} (which will be used to verify signatures) and a secret key S_{A} (which will be used to produce signatures).

  Notation: $\langle P_{\mathsf{A}}, S_{\mathsf{A}}\rangle \leftarrow \mathcal{G}(\mathsf{params})$.

- $\mathcal{S}$ (sign). This algorithm takes a message $M \in \mathcal{M}$ and a secret key S_{A} and returns a signature σ.

  Notation: $\sigma \leftarrow \mathcal{S}(\mathsf{params}, M, S_{\mathsf{A}})$.

- $\mathcal{V}$ (verify). This algorithm takes a message $M \in \mathcal{M}$, a signature σ and a public key P_{A} and verifies if σ is a valid signature produced by the secret key S_{A} that belongs to the public key P_{A} for the message M. It returns valid is the signature is valid, and invalid otherwise

Notation: $\mathcal{V}(\mathsf{params}, M, \sigma, P_{\mathsf{A}})$.

Remark 6.1. Usually signature schemes are described by three algorithms instead of four. The setup algorithm is then omitted and the key generation algorithm takes the security multiplier instead. Also, system parameters are not explicitly passed as arguments to the sign and verify algorithms, but are simply assumed to be available.

6.3.2 Security models

Notions of security in signature schemes

To be able to have signature schemes that are provably secure, we first need a formal definition of security. We consider the security models for signature schemes as described by Goldwasser et. al. in their seminal paper [27].

Similar to the security models for encryption, we distinguish between security goals and attack models. Possible goals an adversary could have against a signature scheme of a user A are:

1. *Total break.* Recovering the secret key of A.
2. *Universal forgery.* Being able to forge a signature for *any* message.
3. *Selective forgery.* Forging a signature for a particular message chosen *a priori* by the adversary.
4. *Existential forgery.* Forging a signature for at least one message that has not been signed before. The adversary has no control over the message.
5. *Strong existential forgery* [8]. Forging a signature for at least one message. The message may have been signed before, but the forged signature should be a new signature. The adversary has no control over the message.

These goals are in decreasing order of hardness (e.g. a total break of the signature scheme implies any type of forgery). Therefore, signature schemes resisting existential forgery are the most desirable systems in terms of security.

Remark 6.2. To "forge" a signature for a message M means to produce a new signature for M. Obtaining a valid signature for M from A does not constitute forgery.

The attack models describe how much information is accessible to the adversary:

1. *Key-only attack.* In this attack model the adversary only knows the public key of A.
2. *Known message attack.* In this attack model the adversary is given a set of valid signatures for a set of messages $M_1, M_2, \ldots, M_n$. The adversary is also given the messages, but he cannot choose them.
3. *Generic or weak chosen message attack.* In this attack model the adversary is given a set of valid signatures for a set of messages $M_1, M_2, \ldots, M_n$. These messages are chosen by the adversary but the entire list of messages is fixed before the adversary learns the corresponding signatures and before the adversary even sees the public key of A. Therefore, this attack is the same against every user.

4. *Direct chosen message attack.* In this attack model the adversary is given a set of valid signatures for a set of messages $M_1, M_2, \ldots, M_n$. These messages are chosen by the adversary and this time the adversary is allowed to construct the list of messages after learning the public key of the user A. The attack is still non-adaptive - the attacker only sees the signatures after fixing the entire list.

5. *Adaptive chosen message attack.* This attack model is the most general. The adversary is given access to an oracle that computes signatures of A for any message. This attack is adaptive - the oracle queries can depend on the results of previous oracle queries.

The adaptive chosen message attack is the most natural attack and being able to successfully perform such an attack implies being able to successfully perform any of the other attacks. The usual notion of security employed is resistance against existential forgery under adaptive chosen message attacks, but in some situations resistance against strong existential forgery under adaptive chosen message attacks is desirable.

Definition 6.8. A formal security definition can be given as follows. Consider the following game, played by an adversary $\mathcal{A}$ and a challenger:

- **Setup.** The challenger runs $\mathcal{G}(1^k)$ for a given security multiplier k, gives the adversary the public key P and keeps the secret key S to itself.

- **Signature Queries.** The adversary is allowed to do sign queries. In such a query the adversary sends the challenger a message $M \in \mathcal{M}$. The challenger computes $\sigma_i \leftarrow \mathcal{S}(M_i, S_{\mathsf{A}})$ and sends σ_i to the adversary. The adversary does not have to commit to the list of queries in advance; each query can depend on the result of former queries. The adversary decides when this phase is over.

- **Output.** The adversary outputs a pair (M, σ) and wins the game if

  1. $\mathcal{V}(M, \sigma, P_{\mathsf{A}}) = \mathsf{valid}$.
  2. $M \notin \{M_1, M_2, \ldots, M_n\}$.

The *advantage* of the adversary $\mathcal{A}$ attacking the signature scheme is defined as follows:

$$\mathrm{Adv}_{\mathcal{A}}(k) = \Pr[\mathcal{V}(M, \sigma, P_{\mathsf{A}}) = \mathsf{valid}]$$

where the probability is taken over the random bits used by both the challenger and the adversary.

We consider the signature scheme to be *secure against existential forgery under adaptive chosen message attacks* if $\mathrm{Adv}_{\mathcal{A}}$ is negligible as a function of k for every polynomially adversary $\mathcal{A}$.

Complexity assumptions

The security of the short signature schemes we will describe rely on the following complexity assumptions.

Definition 6.9. An algorithm $\mathcal{A}$ is said to have *advantage* ε in solving co-CDH in (G_1, G_2) if

$$\Pr\left[\mathcal{A}(P_2, aP_2, Q) = aQ \,\middle|\, a \leftarrow (\mathbb{Z}/p\mathbb{Z})^*, Q \leftarrow G_1^*\right] \geq \varepsilon.$$

Definition 6.10. An algorithm $\mathcal{A}$ is said to have *advantage* ε in solving q-SDH in (G_1, G_2) if

$$\Pr\left[\mathcal{A}(P_1, xP_1, \ldots, x^qP_1, P_2, xP_2) = \left(c, \frac{1}{x+c}P_1\right) \,\middle|\, \begin{array}{l} x \leftarrow (\mathbb{Z}/p\mathbb{Z})^*, \\ P_1 \leftarrow G_1^*, P_2 \leftarrow G_2^* \end{array}\right] \geq \varepsilon.$$

These problems are said to be hard in groups (G_1, G_2) if no polynomially bounded algorithm has non-negligible advantage in solving the problems in (G_1, G_2). The asymptotic formulation of these assumption requires a bilinear group generator again, as defined in the previous section.

Definition 6.11. Assuming the existence of a bilinear group generator $\mathcal{G}$, the advantage of an algorithm $\mathcal{A}$ in solving co-CDH for $\mathcal{G}$ is defined as follows:

$$\mathrm{Adv}^{\text{co-CDH}}_{\mathcal{G},\mathcal{A}}(k) = \Pr\left[\mathcal{A}(q, G_1, G_2, G_T, \hat{e}, P_2, aP_2, Q) = aQ \,\middle|\, \begin{array}{l} \langle q, G_1, G_2, G_T, \hat{e}\rangle \leftarrow \mathcal{G}(1^k), \\ Q \leftarrow G_1^*, a \leftarrow (\mathbb{Z}/q\mathbb{Z})^* \end{array}\right].$$

If for any randomized polynomial time (in k) algorithm $\mathcal{A}$ we have that $\mathrm{Adv}^{\text{co-CDH}}_{\mathcal{G},\mathcal{A}}$ is a negligible function, then $\mathcal{G}$ is said to satisfy the co-CDH-*assumption* and co-CDH is said to be *hard* in groups generated by $\mathcal{G}$.

Definition 6.12. Assuming the existence of a bilinear group generator $\mathcal{G}$, the advantage of an algorithm $\mathcal{A}$ in solving q-SDH for $\mathcal{G}$ is defined as follows:

$$\mathrm{Adv}^{q\text{-SDH}}_{\mathcal{G},\mathcal{A}}(k) = \Pr\left[\begin{array}{l} \mathcal{A}(p, G_1, G_2, G_T, \hat{e}, P_1, \\ xP_1, \ldots, x^qP_1, P_2, xP_2) \end{array} = \left(c, \frac{1}{x+c}P_1\right) \,\middle|\, \begin{array}{l} \langle p, G_1, G_2, G_T, \hat{e}\rangle \leftarrow \mathcal{G}(1^k), \\ P_1 \leftarrow G_1^*, P_2 \leftarrow G_2^* \\ x \leftarrow (\mathbb{Z}/p\mathbb{Z})^* \end{array}\right].$$

If for any randomized polynomial time (in k) algorithm $\mathcal{A}$ we have that $\mathrm{Adv}^{q\text{-SDH}}_{\mathcal{G},\mathcal{A}}$ is a negligible function, then $\mathcal{G}$ is said to satisfy the q-SDH-*assumption* and q-SDH is said to be *hard* in groups generated by $\mathcal{G}$.

6.3.3 The Boneh-Lynn-Shacham short signature scheme

The Boneh-Lynn-Schacham (or BLS) signature scheme [12] is one of the earlier pairing based protocols. It offers signature lengths half the size of signatures generated by DSA, but at a similar level of security. We describe the revised version which uses an asymmetric pairing.

Definition

The BLS-scheme is originally constructed using any Gap co-Diffie-Hellman pair, but since we are focusing on pairing based cryptography we will translate the algorithms to the pairing based setting:

$\mathcal{I}$ (setup). Let k be the desired security parameter.

- Generate three cyclic groups G_1, G_2 and G_T of prime order and a pairing $\hat{e} : G_1 \times G_2 \to G_T$. The size of these groups should depend on the security parameter k. In addition, pick a random generator $P \in G_1$ and a random generator $Q \in G_2$.
- Pick cryptographic hash functions $H : \{0,1\}^* \to G_1^*$. For the security proof to work, this hash function needs to be a random oracle.
- Let $\mathsf{params} = \langle G_1, G_2, G_T, \hat{e}, n, P, Q, H\rangle$. The message space is $\mathcal{M} = \{0,1\}^n$.
- Return params.

$\mathcal{G}$ (key generation).

- Pick a random $x \in \mathbb{Z}/p\mathbb{Z}$ and compute $V = xQ \in G_2$.
- The public key is V, and the secret key is x.
- Return $\langle V, x\rangle$.

$\mathcal{S}$ (sign). Let $M \in \mathcal{M}$ be the message to sign and let x be the secret key.

- Compute $R = H(M) \in G_1$.
- Compute $\sigma = xR \in G_1$.
- Return σ.

$\mathcal{V}$ (verify). Let $M \in \mathcal{M}$ be a message, $\sigma \in G_1$ a signature and $v \in G_2$ a public key.

- Compute $R = H(M) \in G_1$.
- Return valid if $\hat{e}(\sigma, Q) = \hat{e}(R, V)$. Otherwise, return invalid.

Any adversary has access to the public key $xQ \in G_2$ and the generators $P \in G_1$ and $Q \in G_2$. Forging a signature means creating an element $xR \in G_1$ when $R \in G_1$ is given which is equivalent to solving the co-CDH problem in (G_1, G_2), which is assumed to be hard. On the other hand, $xR \in G_1$ is a valid signature generated using the public key $yQ \in G_2$ if $x = y$. Thus, verifying a signature is equivalent to solving the co-DDH problem in (G_1, G_2), which is easy due to the existence of the efficiently computable pairing.

Theorem 6.5. *The BLS-scheme is consistent.*

Proof. Let $M \in \mathcal{M}$ be a message and $\sigma \in G_1$ be a signature on this message generated by the signing procedure using the public key V. We need to show that the

$$\mathcal{V}(\mathsf{params}, M, V) = \mathsf{valid}.$$

Indeed

$$\hat{e}(\sigma, Q) = \hat{e}(xR, Q) = \hat{e}(R, Q)^x \quad \text{and} \quad \hat{e}(R, V) = \hat{e}(R, xQ) = \hat{e}(R, Q)^x.$$

□

The scheme becomes efficient when we instantiate it with groups that allow a short representation of elements. Luckily, elliptic curves (i.e. G_1 and G_2 are subgroups of the group of points on an elliptic curve $\mathcal{E}/\mathbb{F}_q$) have a short representation. Instead of taking σ as the signature, we can store just the x-coordinate of σ. In the verification algorithm we simply pick a point σ having said x-coordinate. Since there could be two points having this x-coordinate it is possible we have picked $-\sigma$ instead. Therefore, to preserve consistency of the scheme we also accept a signature if $\hat{e}(\sigma, Q)^{-1} = \hat{e}(R, V)$.

Security

The security of the scheme in the random oracle model is proven by the following theorem.

Theorem 6.6 (3.2 in [12]). *Suppose the hash function H is a random oracle. Then the signature scheme on (G_1, G_2) is secure against existential forgery under an adaptive chosen-message attack assuming* co-CDH *is hard in (G_1, G_2). Concretely, suppose there is a forger algorithm $\mathcal{A}$ that has non-negligible advantage against the signature scheme and runs in polynomial time. Then there is an algorithm $\mathcal{B}$ that solves* co-CDH *in (G_1, G_2) with non-negligible advantage and runs in polynomial time.*

Intuitively, this theorem states that being able to efficiently forge a signatures allows one to efficiently solve co-CDH in (G_1, G_2), which contradicts the co-CDH hardness assumption. The efficiency of the security reduction in the proof in [12] is as follows. Suppose $\mathcal{A}$ has advantage ε against the signature scheme and runs in time t. Suppose $\mathcal{A}$ makes at most $q_s > 0$ signature queries and at most $q_H > 0$ hash queries to H. Then the constructed algorithm $\mathcal{B}$ solves co-CDH in (G_1, G_2) with advantage ε' and runs in time t' such that

$$\varepsilon \geq e(q_s + 1) \cdot \varepsilon' \quad \text{and} \quad t \leq t' - c_{G_1}(q_H + 2q_s),$$

where c_{G_1} is a constant that depends on G_1. Here e is the base of the natural logarithm.

6.3.4 The Boneh-Boyen short signature scheme

Another short signature scheme based on pairings is due to Boneh en Boyen [8]. The signatures are just as short as the signatures in the BLS-scheme and the scheme is much more efficient. The security is not based on random oracles, but uses the Strong Diffie-Hellman (SDH) problem as a complexity assumption. The scheme is proven to be existentially unforgable against an adaptive chosen message attack.

Definition

The signature scheme is defined by the following set of algorithms:

$\mathcal{I}$ (setup). Let k be the desired security parameter.

- Generate three cyclic groups G_1, G_2 and G_T of prime order q and a pairing $\hat{e} : G_1 \times G_2 \to G_T$. The size of these groups should depend on the security parameter k.
- Let params $= \langle q, G_1, G_2, G_T, \hat{e} \rangle$. The message space is $\mathcal{M} = \mathbb{Z}/q\mathbb{Z}$.
- Return params.

$\mathcal{G}$ (key generation).

- Pick random generators $P_1 \in G_1$ and $P_2 \in G_2$, and random integers $x, y \in (\mathbb{Z}/q\mathbb{Z})^*$.
- Compute $U = xP_2 \in G_2$ and $V = yP_2 \in G_2$ Also compute $z = e(P_1, P_2) \in G_T$.
- The public key is the tuple $\langle P_1, P_2, U, V, z \rangle$ and the secret key is the tuple $\langle P_1, x, y \rangle$.

- Return $\langle\langle P_1, P_2, U, V, z\rangle, \langle P_1, x, y\rangle\rangle$.

$\mathcal{S}$ (sign). Let $m \in \mathbb{Z}/q\mathbb{Z}$ be the message to sign and let $\langle P_1, x, y\rangle$ be the secret key.

- Pick a random $r \in (\mathbb{Z}/q\mathbb{Z}) - \{\frac{x+m}{y}\}$.
- Compute $\sigma = (x + m + yr)^{-1} P_1 \in G_1$ (the inverse $(x + m + yr)^{-1}$ is computed modulo p.
- Return $\langle \sigma, r\rangle$.

$\mathcal{V}$ (verify). Let $m \in \mathbb{Z}/q\mathbb{Z}$ be a message, $\langle \sigma, r\rangle$ a signature and $\langle P_1, P_2, U, V, z\rangle$ a public key.

- Return valid if $\hat{e}(\sigma, U + mP_2 + rV) = z$. Otherwise, return invalid.

Theorem 6.7. *The Boneh-Boyen signature scheme is consistent.*

Proof. Let $m \in \mathbb{Z}/q\mathbb{Z}$ be a message and $\langle \sigma, r\rangle$ be a signature on this message generated the signing procedure using the public key $\langle P_1, P_2, U, V, z\rangle$. We need to show that

$$\mathcal{V}(\mathsf{params}, M, \langle P_1, P_2, U, V, z\rangle) = \mathsf{valid}.$$

Indeed

$$\begin{aligned}\hat{e}(\sigma, U + mP_2 + rV) &= \hat{e}\big((x + m + yr)^{-1} P_1, xP_2 + mP_2 + ryP_2\big)\\ &= \hat{e}\big((x + m + yr)^{-1} P_1, (x + m + ry)P_2\big)\\ &= \hat{e}\big(P_1, P_2\big)^{(x+m+yr)^{-1}(x+m+yr)}\\ &= \hat{e}\big(P_1, P_2\big) = z.\end{aligned}$$

□

Security

The security of this signature scheme depends on a the strong Diffie-Hellman assumption, which is the discrete logarithm analogue of the strong RSA assumption. The SDH assumption is new and not as well studied as the BDH assumption, so in order to build confidence in this assumption, the authors derive a lower bound on the on the computational complexity of an algorithm solving q-SDH in generic groups [8]. The signature scheme is proven to be secure in the standard model (i.e. without the use of random oracles) using the following theorem.

Theorem 6.8 (3.2 in [8])**.** *The signature scheme on (G_1, G_2) is secure against existential forgery under an adaptive-chosen message attack assuming q-SDH is hard in (G_1, G_2). Suppose there is a forger algorithm $\mathcal{A}$ that has non-negligible advantage against the signature scheme and runs in polynomial time. Then there is an algorithm $\mathcal{B}$ that solves q-SDH in (G_1, G_2) with non-negligible advantage and runs in polynomial time.*

Intuitively, this theorem states that being able to efficiently forge a signatures allows one to efficiently solve q-SDH, which contradicts the q-SDH hardness assumption. The efficiency of the security reduction in the proof in [8] is as follows. Suppose $\mathcal{A}$ has advantage ε against the signature time and runs in time t. Suppose $\mathcal{A}$ makes at most $q_s > 0$ signature queries. Then the constructed algorithm $\mathcal{B}$ solves q-SDH in (G_1, G_2) with advantage ε' and runs in time t' such that

$$q > q_s, \qquad \varepsilon \geq 2(\varepsilon' + \frac{q_s}{p}) \approx 2\varepsilon', \qquad \text{and} \qquad t \leq t' - \theta(q^2 T)$$

where T is the maximum time for a multiplication in G_1 and G_2.

Chapter 7

Conclusions

We have defined elliptic and hyperelliptic curves over finite fields and shown how to compute the group operation on them.

The Weil pairing and Tate-Lichtenbaum pairing are efficiently computable bilinear mappings defined on elliptic and hyperelliptic curves. It seems the Tate-Lichtenbaum pairing on elliptic curves is the most efficient and practical pairing to use. We have shown how to compute both pairings using Miller's algorithm.

Using the group law on elliptic and hyperelliptic curves, it is possible to define the discrete logarithm problem on these curves. On curves with a low embedding degree (i.e. supersingular curves), pairings can be used to reduce the problem to the discrete logarithm problem on a finite field, where sub-exponential attacks can be used.

We have shown how pairings give rise to mathematical problems (e.g. the bilinear Diffie-Hellman problem) that can be used as the basis for cryptographic schemes. We have discussed the consequences of being able to invert pairings.

Pairings can be used to create interesting cryptographic protocols. For instance, they can be used to create a novel cryptographic scheme, known as identity based encryption, for which currently no other practical construction is known. In addition, pairings can be used to construct a signature scheme that has very short signature representations.

Bibliography

[1] Nuttapong Attrapadung, Yang Cui, Goichiro Hanaoka Hideki Imai, Kanta Matsuura, Peng Yang, and Rui Zhang, *Relations among notions of security for identity based encryption schemes*, LATIN 2006: Theoretical Informatics, 7th Latin American Symposium, vol. 3887, Springer-Verlag, 2006, pp. 130–141.

[2] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway, *Relations among notions of security for public-key encryption schemes*, CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology (London, UK), Springer-Verlag, 1998, pp. 26–45.

[3] Mihir Bellare and Phillip Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, ACM Conference on Computer and Communications Security, 1993, pp. 62–73.

[4] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels, *Advances in elliptic curve cryptography (london mathematical society lecture note series)*, Cambridge University Press, New York, NY, USA, 2005.

[5] Ian F. Blake, G. Seroussi, and N. P. Smart, *Elliptic curves in cryptography*, Cambridge University Press, New York, NY, USA, 1999.

[6] D. Boneh and X. Boyen, *Efficient selective-ID secure identity-based encryption without random oracles*, Advances in Cryptology (EUROCRYPT 2004), LNCS, vol. 3027, Springer, 2004, pp. 223–238.

[7] ———, *Secure identity based encryption without random oracles*, Proceedings of Crypto 2004, Lecture Notes in Computer Science, Springer-Verlag, 2004.

[8] ———, *Short signatures without random oracles*, Lecture Notes in Computer Science (EUROCRYPT'04), vol. 3027, Springer, 2004, pp. 382–400.

[9] D. Boneh, X. Boyen, and E. Goh, *Hierarchical identity based encryption with constant size ciphertext*, Proceedings of Eurocrypt '05, 2005.

[10] Dan Boneh and Matt Franklin, *Identity-based encryption from the Weil pairing*, Lecture Notes in Computer Science **2139** (2001), 213–??

[11] Dan Boneh, Craig Gentry, and Michael Hamburg, *Space-efficient identity based encryption without pairings*, FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (Washington, DC, USA), IEEE Computer Society, 2007, pp. 647–657.

[12] Dan Boneh, Ben Lynn, and Hovav Shacham, *Short signatures from the Weil pairing*, Lecture Notes in Computer Science **2248** (2001), 514–??

[13] R. Canetti, S. Halevi, and J. Katz, *A forward-secure public-key encryption scheme*, Advances in Cryptology (Eurocrypt 2003). Lecture Notes in Computer Science, vol. 2656, Springer-Verlag, 2003, pp. 255–271.

[14] Ran Canetti, Oded Goldreich, and Shai Halevi, *The random oracle methodology, revisited*, J. ACM **51** (2004), no. 4, 557–594.

[15] J. Cheon and D. Lee, *Diffie-hellman problems and bilinear maps*, Cryptology ePrint Archive: Report 2002., 2002.

[16] YoungJu Choie and Eunjeong Lee, *Implementation of the tate pairing on hyperelliptic curves of genus 2*, vol. 2791, 2004, pp. 97–111.

[17] Clifford Cocks, *An identity based encryption scheme based on quadratic residues*, Proceedings of the 8th IMA International Conference on Cryptography and Coding. Lecture Notes in Computer Science, vol. 2260, 2001.

[18] Whitfield Diffie and Martin E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **IT-22** (1976), no. 6, 644–654.

[19] D. Freeman, M. Scott, and E. Teske, *A taxonomy of pairing-friendly elliptic curves*, Preprint, 2006.

[20] Eiichiro Fujisaki and Tatsuaki Okamoto, *Secure integration of asymmetric and symmetric encryption schemes*, Lecture Notes in Computer Science **1666** (1999), 537–554.

[21] Steven Galbraith, Florian Hess, and Frederik Vercauteren, *Hyperelliptic pairings*, Lecture Notes in Computer Science, vol. 4575, Springer, 2007, pp. 108–131.

[22] ———, *Aspects of pairing inversion*, IEEE Transactions on Information Theory, 2008, pp. 5719–5728.

[23] Steven D. Galbraith, *Supersingular curves in cryptography*, Lecture Notes in Computer Science **2248** (2001), 495–??

[24] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart, *Pairings for cryptographers*, Discrete Appl. Math. **156** (2008), no. 16, 3113–3121.

[25] P. Gaudry, F. Hess, and N. P. Smart, *Constructive and destructive facets of Weil descent on elliptic curves*, Journal of Cryptology: the journal of the International Association for Cryptologic Research **15** (2002), no. 1, 19–46.

[26] Craig Gentry and Alice Silverberg, *Hierarchical id-based cryptography*, ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security (London, UK), Springer-Verlag, 2002, pp. 548–566.

[27] Shafi Goldwasser, Silvio Micali, and Ron L. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal on Computing **17** (1988), no. 2, 281–308.

[28] R. Granger, F.Hess, R. Oyono, N. Theriault, and F. Vercauteren, *Ate pairing on hyperelliptic curves*, Advances in Cryptology (2007), 430–447.

[29] F. Hess, *A note on the tate pairing of curves over finite fields*, Arch. Math, 2004, pp. 28–32.

[30] Jeremy Horwitz and Ben Lynn, *Toward hierarchical identity-based encryption*, Theory and Application of Cryptographic Techniques, 2002, pp. 466–481.

[31] Klaus Hulek, *Elementary algebraic geometry*, American Mathematical Society, 2000.

[32] Antoine Joux, *A one round protocol for tripartite diffie-hellman*, ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory (London, UK), Springer-Verlag, 2000, pp. 385–394.

[33] Kanayama, Kobayashi, Saito, and Uchiyama, *Remarks on elliptic curve discrete logarithm problems*, TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems (2000).

[34] Anthony W. Knapp, *Elliptic curves*, Mathematical Notes, vol. 20, Princeton University Press, 1992.

[35] Neil Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation, vol. 48, 1987, pp. 203–209.

[36] ———, *Hyperelliptic cryptosystems*, J. Cryptol. **1** (1989), no. 3, 139–150.

[37] Neil Koblitz, Alfred J. Menezes, Yi-Hong Wu, and Robert J. Zuccherato, *Algebraic aspects of cryptography*, Springer-Verlag New York, Inc., New York, NY, USA, 1998.

[38] Alfred Menezes, Scott Vanstone, and Tatsuaki Okamoto, *Reducing elliptic curve logarithms to logarithms in a finite field*, STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing (New York, NY, USA), ACM Press, 1991, pp. 80–89.

[39] Alfred J. Menezes, Yi-Hong Wu, and Robert J. Zuccherato, *An elementary introduction to hyperelliptic curves*, 1996.

[40] Victor S. Miller, *Short programs for functions on curves*, IBM Thomas J. Watson Research Center, 1986.

[41] ———, *Use of elliptic curves in cryptography*, Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85 (New York, NY, USA), Springer-Verlag New York, Inc., 1986, pp. 417–426.

[42] Tatsuaki Okamoto and David Pointcheval, *The gap-problems: A new class of problems for the security of cryptographic schemes*, PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography (London, UK), Springer-Verlag, 2001, pp. 104–118.

[43] Grace Orzech and Morris Orzech, *Plane algebraic curves*, Marcel Dekker, Inc., 1981.

[44] Hans-Georg Rück, *On the discrete logarithm in the divisor class group of curves*, Math. Comput. **68** (1999), no. 226, 805–806.

[45] Amit Sahai and Brent Waters, *Fuzzy identity based encryption*, Lectures Notes in Computer Science, vol. 3494, Springer, 2005, pp. 457–473.

[46] Oliver Schirokauer, Damian Weber, and Thomas F. Denny, *Discrete logarithms: The effectiveness of the index calculus method*, ANTS, 1996, pp. 337–361.

[47] Adi Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology, Proceedings of CRYPTO '84, Lecture notes in Computer Science, 1984, pp. 47–53.

[48] Joseph H. Silverman, *The arithmetic of elliptic curves*, Graduate Texts in Mathematics, vol. 106, Springer, 1986.

[49] N. P. Smart, *The discrete logarithm problem on elliptic curves of trace one*, Journal of Cryptology: the journal of the International Association for Cryptologic Research **12** (1999), no. 3, 193–196.

[50] Nigel Smart and Frederik Vercauteren, *On computable isomorphisms in efficient asymmetric pairing based systems*, 2007, pp. 538–547.

[51] B. Waters, *Efficient identity-based encryption without random oracles*, Advances in Cryptology – EUROCRYPT 2005, Lecture Notes in Computer Science, vol. 3404, 2005, pp. 114–127.

[52] André Weil, *Sur les fonctions algebriques à corps de constantes finis.*, C. R. Academie des Sciences, 1946.