Cryptographic and cryptanalytic applications of pairings on elliptic curves

Firas Kraïem, Master 2 CSI, University of Bordeaux
Supervised by Jean Gillibert

March 14, 2014

Introduction

The idea of using the group of points of an elliptic curve instead of the multiplicative group of a finite field in order to obtain hard instances of the discrete logarithm problem to be used in a discrete-logarithm-based cryptosystem was independently proposed by Koblitz [11] and Miller [13] in 1985. The main reason for this proposal was that the discrete logarithm problem in multiplicative groups of finite fields had been found to be solvable in subexponential time using the "index calculus" family of algorithms, but no such algorithms had been found for discrete logarithm problems in groups of points of elliptic curves, where the best known algorithms were the general algorithms which can be used in any group and take exponential time.

This remained true until Menezes, Okamoto, and Vanstone [12] showed in 1993 that the discrete logarithm problem in the groups of points of certain elliptic curves can be reduced to that in the multiplicative group of finite fields sufficiently small as to make it solvable in subexponential time. Their reduction, now referred to as the "MOV attack", makes use of a function introduced by Weil in 1940 [23] and now called the Weil pairing.

It was then shown by Joux [10] that the added structure provided by the Weil pairing could also be used constructively, to provide solutions to problems for which none previously existed. Namely, Joux used the Weil pairing to construct a protocol analogous to the Diffie-Hellman protocol, but which can be used to create a shared secret between three participants instead of two. More constructions using the Weil pairing were subsequently introduced, one of the most significant being a practical identity-based encryption scheme due to Boneh and Franklin [3].

The rest of this work is organised as follows: in Chapter 1, we discuss basic notions about elliptic curves, without going into too much detail for the sake of brevity. In Chapter 2, we discuss the definition and properties of the Weil pairing and a similar pairing due to Tate [21], and in Chapter 3, we discuss some applications of those pairings in cryptography. Namely, we discuss the MOV attack and an analogous attack due to Frey and Rück [6] which uses the Tate pairing in lieu of the Weil pairing, as well as Boneh and Franklin's identity-based encryption scheme.

Contents

Chapter 1

Elliptic curves

1.1 Affine and projective plane curves

For the sake of brevity, we will not be proving the associativity of addition on an elliptic curve, and so we will have little use for the projective plane, treating the point at infinity as just a formal symbol. We nevertheless briefly discuss curves in the projective plane, both for mathematical interest and to have some idea of where the point at infinity comes from. This is essentially a condensed version of the first few sections of [20, Appendix A].

Given a field K, the set of all ordered pairs (x, y) such that $x, y \in K$ is called the *affine plane over* K and denoted $\mathbf{A}^2_K$. Given a polynomial $C \in K[X, Y]$, we can consider its zeros in $\mathbf{A}^2_L$ for any field $L \supseteq K$, which are the pairs $(x, y) \in \mathbf{A}^2_L$ such that $C(x, y) = 0$. The set of all zeros of C in $\mathbf{A}^2_L$ is an *affine plane curve*. It is convenient to denote a curve and its defining polynomial by the same letter, so we will speak of both "the curve C" and "the polynomial C", and if C (as a polynomial) has coefficients in K, we say that C is *defined over* K. The points of C which have coordinates in L are then said to be *rational over* L.

The affine plane alone is not entirely satisfactory from a geometric point of view. For example, we know that two distinct lines intersect in exactly one point in the affine plane, unless they are parallel, in which case they do not intersect at all. In order to solve this discrepancy, we need to work in the *projective plane*. A point in the projective plane over some field K is a triple (x, y, z), with x, y, z in K and not all zero, with the added property that we regard (x, y, z) and (x', y', z') as representing the same point if and only if there is some $\lambda \in K^*$ such that $(x', y', z') = (\lambda x, \lambda y, \lambda z)$. More formally, we define an equivalence relation $\sim$ on $K^3 \setminus \{(0, 0, 0)\}$ as follows: $(x, y, z) \sim (x', y', z')$ if and only if there is some $\lambda \in K^*$ such that $(x', y', z') = \lambda(x, y, z)$. Then, the *projective plane over* K, noted $\mathbf{P}^2_K$, is the quotient set $(K^3 \setminus \{(0, 0, 0)\})/\sim$ (*i.e.*, the set of all equivalence classes). A triple $(x, y, z) \in K^3 \setminus \{(0, 0, 0)\}$ is called a *representative* for the point represented by its equivalence class, which we will denote by $[x, y, z]$.

We divide the points $[x, y, z]$ of $\mathbf{P}^2_K$ into two groups. If $z \neq 0$, we have $[x, y, z] = [x/z, y/z, 1]$, and so any point with $z \neq 0$ can be written as $[x, y, 1]$. We identify the point $[x, y, 1]$ of the projective plane with the point (x, y) of the affine plane, and so in general we identify the point $[x, y, z]$ of the projective plane with the point $(x/z, y/z)$ of the affine plane, if $z \neq 0$. This gives an embedding of the affine plane into the projective plane (the point (x, y) of the affine plane being mapped to the point $[x, y, 1]$ of the projective plane). If $z = 0$, then the point $[x, y, 0]$ does not lie in the affine plane, and is called a *point at infinity*. We will see that two lines which are parallel in the affine plane intersect at one of these points in the projective plane (this formalises the common perception that two parallel lines intersect "at infinity").

We need to be careful in defining curves in the projective plane. For example if we try to define a curve on the projective plane as the set of zeros of the polynomial $C = X^2 - Y$, we will have a problem. It seems that the point $[1, 1, 1]$ is on the curve, since its coordinates give a zero of the polynomial. However, the point $[2, 2, 2]$ is the same point, but its coordinates do not give a zero of the polynomial. In order to solve this problem and make the curve independent of the choice of representatives of a point, we work only with *homogeneous polynomials*. A polynomial is said to be *homogeneous* if all its terms have the same degree. Thus our polynomial above was not homogeneous, since its first term had degree 2 but its second term had degree 1. The polynomial $X^2 - YZ$ is homogeneous, and we see that this determines whether a point of the projective plane is on the curve independently of a choice of representative. Indeed, if $x^2 - yz = 0$, then for any $\lambda \neq 0$ we have

$$(\lambda x)^2 - (\lambda y \lambda z)^2 = \lambda^2 x^2 - \lambda^2 yz = \lambda^2(x^2 - yz) = 0,$$

so if a point $[x, y, z]$ is on the curve, we can take any representative $(\lambda x, \lambda y, \lambda z)$ to verify it.

Thus a curve in the projective plane is defined by a homogeneous polynomial in $K[X, Y, Z]$, and as above its points can be divided into its *affine points*, which are the points with $z = 1$ (those who lie in the copy of the affine plane embedded in the projective plane), and its points at infinity, which are those with $z = 0$. We call the set of all affine points of a projective curve its *affine part*. For example, consider the curve defined by $X^2 - YZ$ in the projective plane over $\mathbf{R}$. To obtain its affine part, we set $z = 1$ and we obtain $X^2 - Y$, so the affine part is just the curve $y = x^2$ in the usual affine plane. To obtain its points at infinity we set $z = 0$ and we obtain the equation $x^2 = 0$, so $x = 0$, and since x, y, z cannot be all zero, we see that any triple $(0, y, 0)$ with $y \neq 0$ gives a solution. But all those triples are representatives of the same point, so the curve has only one point at infinity: the point $[0, 1, 0]$.

The preceeding paragraph should have made clear how to "projectivise" an affine plane curve: just insert appropriate powers of Z in the polynomial so as to make it homogeneous. Since we set $z = 1$ to obtain the affine part of a projective curve, the affine part of the new curve will be the same as the original affine curve. As a last example, we will show that two parallel lines in the affine plane indeed intersect at a point at infinity in the projective plane. Let $y = ax + m$ and $y = ax + m'$, with $m \neq m'$, be two lines in the usual affine plane over $\mathbf{R}$, they are defined by the polynomials $aX - Y + m$ and $aX - Y + m'$ respectively. Adding appropriate powers of Z we obtain $aX - Y + mZ$ and $aX - Y + m'Z$. The affine parts of those lines are just the corresponding lines in the affine plane, which are parallel and thus do not intersect. If we set $z = 0$ however we obtain $aX - Y$, so the triples $(x, ax, 0)$, for $x \neq 0$, give points which lie on both lines. It is easily seen that they all give the same point, so we conclude that the two lines intersect at the point at infinity $[1, a, 0]$.

1.2 Elliptic curves

In most situations we will encounter, an *elliptic curve* E will be a projective plane curve defined over some field K by an equation of the form

$$Y^2 = X^3 + a_4 X + a_6, \tag{1.1}$$

where a_4 and a_6 are in K. As usual, we can consider its points in any field $L \supseteq K$, which form a projective plane curve in $\mathbf{P}^2_L$. To find its points at infinity, we homogeneise the polynomial, which yields

$$Y^2 Z = X^3 + a_4 X Z^2 + a_6 Z^3,$$

and letting $z = 0$ we obtain just $x^3 = 0$. Just like for the curve $X^2 - YZ$ of the preceeding section, we see that the curve has only one point at infinity, namely the point $[0, 1, 0]$. Since there is only one point at infinity, we will in general just work in the affine plane and use the point at infinity on an *ad hoc* basis. We thus let $E(L)$ be the set

$$\{(x, y) \in \mathbf{A}^2_L, y^2 = x^3 + a_4x + a_6\} \cup \{\infty\},$$

and call it the set of points of E in L. An equation as in (1.1) is called a *Weierstrass equation* for the curve E. We also require that the cubic polynomial in X on the right-hand side of (1.1) have no repeated roots, *i.e.*, that its discriminant, which is equal to $-(4a_4^3 + 27a_6^2)$, be non-zero (equivalently, we require that $4a_4^3 + 27a_6^2 \neq 0$).

We will sometimes need to consider equations of a more general form, especially when working over fields of characteristic 2 (which are especially important in cryptography) or 3. A *general Weierstrass equation* for an elliptic curve is an equation of the form

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6, \tag{1.2}$$

where again the coefficients a_i lie in some field K over which the curve is defined. If the characteristic of K is not 2, then we can let $Y' = Y + a_1X/2 + a_3/2$ to obtain

$$Y'^2 = X^3 + a'_2X^2 + a'_4X + a'_6$$

for some constants a'_2, a'_4, a'_6. If also the characteristic of K is not 3, we can let $X' = X + a'_2/3$ and obtain

$$Y'^2 = X'^3 + aX' + b,$$

which is a Weierstrass equation as in (1.1). When we need to be specific, we will call it a *short Weierstrass equation*. Unless mentioned otherwise, all results will be proved for elliptic curves defined by a short Weierstrass equation only, the proofs for general Weierstrass equations are identical in spirit, but require longer calculations.

1.3 The group law

Given an elliptic curve E defined over some field K, we have seen that we can consider its set of points with coordinates in a field $L \supseteq K$, which we denote $E(L)$. $E(L)$ contains all the pairs (x, y) corresponding to the affine points of E in $\mathbf{A}^2_L$, as well as the formal symbol ∞, representing the point at infinity of E in $\mathbf{P}^2_L$. We will see that we can turn $E(L)$ into an *abelian group*.

We first assume that the characteristic of K is not 2 or 3, meaning that an elliptic curve E over K is of the form $Y^2 = X^3 + a_4X + a_6$. For two distinct affine points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ in $E(L)$, for some field $L \supseteq K$, we define the sum $P_1 + P_2$ as follows. First we draw the line through P_1 and P_2. Assuming $x_1 \neq x_2$, the line will be given by the equation

$$Y = m(X - x_1) + y_1,$$

where m is the slope of the line, which is given by

$$m = \frac{y_2 - y_1}{x_2 - x_1}.$$

Substituting $m(X - x_1) + y_1$ for Y in the equation of the curve, we obtain

$$(m(X - x_1) + y_1)^2 = X^3 + a_4X + a_6,$$

which can be rearranged as

$$X^3 - m^2X^2 + \cdots = 0.$$

The roots of this polynomial are the x-coordinates of the points where the curve and the line intersect. Since we already know two of them (x_1 and x_2), and since the coefficient of X^2 is the negative of the sum of the roots, we obtain finally that the line and the curve intersect at a third point, whose x-coordinate is $m^2 - x_1 - x_2$. Its y-coordinate can then be obtained by substituting $m^2 - x_1 - x_2$ for X in the equation of the line: $Y = m(X - x_1) + y_1$. The point $P_1 + P_2$ is the reflection of the point we have just obtained across the x axis, and so, if we call this point P_3 and its coordinates (x_3, y_3), we find

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \qquad y_3 = \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3) - y_1.$$

If $x_1 = x_2$, then the line through P_1 and P_2 is vertical, and it does not intersect the curve at a third point on the affine plane (indeed, for any given value of x, there are at most two values of y such that (x, y) is on the curve). However, it intersects the curve on the projective plane at its point at infinity. Indeed, a vertical line is defined by a polynomial of the form $X - c$. Homogeneising it gives $X - cZ$, and setting $z = 0$ to obtain its points at infinity yields the equation $x = 0$, so the line has exactly one point at infinity, which is $[0, 1, 0]$. The curve also has the point at infinity $[0, 1, 0]$, so it is the third point of intersection. Reflecting it across the x axis yields $[0, -1, 0]$, which is the same point, so we obtain finally that if $x_1 = x_2$, we have $P_1 + P_2 = \infty$.

To add a point $P = (x_1, y_1)$ to itself, we proceed as above, except that we take the tangent line to the curve at P (which we think of as being the line which goes "through P and P"). Its slope can be obtained by implicitly differentiating the equation of the curve, we get

$$2Y\frac{dY}{dX} = 3X^2 + a_4,$$

and so the slope of the tangent at P is

$$m = \frac{dY}{dX}(x_1, y_1) = \frac{3x_1^2 + a_4}{2y_1}.$$

If $y_1 = 0$, the tangent is vertical, so we set as above $P + P = \infty$. Otherwise, the equation of the tangent is

$$Y = m(X - x_1) + y_1.$$

As before, it intersects the curve in three points, but P counts as two of them, so letting (x_2, y_2) be the coordinates of $P + P$, we obtain

$$x_2 = \left(\frac{3x_1^2 + a_4}{2y_1}\right)^2 - 2x_1, \qquad y_2 = \frac{3x_1^2 + a_4}{2y_1}(x_1 - x_2) - y_1.$$

Finally, let $P_1 = (x_1, y_1)$ be an affine point of the curve and $P_2 = \infty$. We have seen that the point at infinity lies on every vertical line, so the line through P_1 and ∞ is the affine vertical line which goes through P_1. This line intersects the curve at a third point, which is the reflection of P_1 across the x axis (if $y_1 = 0$, that's P_1 itself), and reflecting that point across the x axis, we obtain P_1 again. Thus $P_1 + \infty = P_1$ for all P_1. This means that the point at infinity acts as the identity for our group law, so we set finaly $\infty + \infty = \infty$.

It is clear that our law has an identity element (the point at infinity), and that each point has an inverse (its reflection across the x axis). Moreover, the law is clearly commutative, since

the line through P_1 and P_2 is the same as the line through P_2 and P_1. It is more surprising (and difficult to prove) that it is associative. As indicated above, for the sake of brevity, we will not prove the associativity here, and refer to any book on the subject, such as [22, Section 2.4].

1.4 The group law in characteristic 2

The formulas of the preceeding section do not apply in fields of characteristic 2. Since those fields are important in cryptography, we now look at the formulas for the group law of curves defined over fiels of characteristic 2. To be complete, we would also need to give the formulas in characteristic 3 (the formulas of the previous section do apply to elliptic curves given by a short Weierstrass equation in characteristic 3, but not all curves can be described by a short Weierstrass equation), but we will omit those formulas since we will not need them in our setting.

We need to work with the general Weierstrass equation, because the short equation always yields a singular curve in characteristic 2. A point P on a plane curve C is called *singular* if both partial derivatives of C vanish at P, and a curve is called *singular* if it has at least one singular point. The short Weierstrass equation viewed as a polynomial is $Y^2 - X^3 - a_4X - a_6$, and we see that its partial derivative with respect to Y is $2Y = 0$. Its partial derivative with respect to X is $-(3X^2 + a_4)$, and letting first x_0 be any root of $3X^2 + a_4$ in $\overline{K}$ and then y_0 be any square root of $x_0^3 + a_4x_0 + a_6$ gives a singular point (x_0, y_0) on C.

Singular curves are very bad both for geometry and for cryptography, which is why we need to work with the general Weierstrass equation, in order to obtain non-singular curves; we recall that the general Weierstrass equation is

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6.$$

First, we determine the points at infinity of such curves. Homogeneising the polynomial gives

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

and setting $z = 0$ we obtain as before the equation $x^3 = 0$, so $x = 0$ and the curve has only one point at infinity, namely $[0, 1, 0]$. Given an affine point $P_1 = (x_1, y_1)$ on the curve, however, the vertical line through P_1 no longer intersects the curve at $(x_1, -y_1)$, but at $(x_1, -a_1x_1 - a_3 - y_1)$.

We can now describe how to add two affine points P_1 and P_2 on E (of course, we still have $P + \infty = P$ for all P): first draw the line through P_1 and P_2 (the tangent line at P_1 if $P_1 = P_2$), it will intersect the curve at a third point P_3, and $P_1 + P_2$ is then the intersection of the curve and the vertical line through P_3. We now give the formulas for this, which are obtained in exactly the same way as in the previous section, making the appropriate adjustments due to characteristic 2. To keep the formulas from getting lengthy, we will separate the curves given by general Weierstrass equations into two groups, according to whether $a_1 \neq 0$ or $a_1 = 0$.

If $a_1 \neq 0$, then letting

$$X = a_1^2X' + \frac{a_3}{a_1}, \qquad Y = a_1^3Y' + a'^{-3}(a_1^2a_4 + a_3^2)$$

gives an equation of the form

$$Y'^2 + X'Y' = X'^3 + a_2'X'^2 + a_6',$$

and this curve is singular if and only if $a_6' = 0$, so we assume $a_6' \neq 0$. Thus any curve of this form can be written (as a polynomial) as

$$Y^2 + XY + X^3 + a_2X^2 + a_6,$$

and letting $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_3 = P_1 + P_2 = (x_3, y_3)$, the formulas are as follows. If $P_1 \neq P_2$, then

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \frac{y_1 + y_2}{x_1 + x_2} + a_2 + x_1 + x_2$$

and

$$y_3 = \frac{y_1 + y_2}{x_1 + x_2}(x_1 + x_3) + y_1 + x_3$$

(of course if $x_1 = x_2$, the line through P_1 and P_2 is vertical and $P_1 + P_2 = \infty$). If $P_1 = P_2$, we again take the tangent, and we obtain

$$x_3 = x_1^2 + \frac{a_6}{x_1^2}$$

and

$$y_3 = \frac{x_1^2 + y_1}{x_1} x_3 + x_1^2 + x_3$$

(if $x_1 = 0$, then the tangent is vertical and $P_1 + P_1 = \infty$).

If $a_1 = 0$, letting $x = x' + a_2$ and $y' = y$ gives an equation of the form

$$y'^2 + a_3' y' = x'^3 + a_4' x' + a_6',$$

and this curve is singular if and only if $a_3' = 0$, so we assume $a_3' \neq 0$. Any curve of this form can be written as a polynomial as

$$Y^2 + a_3 Y + X^3 + a_4 X + a_6,$$

and again letting $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_3 = P_1 + P_2 = (x_3, y_3)$, the formulas are as follows. If $P_1 \neq P_2$, then

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

and

$$y_3 = \frac{y_1 + y_2}{x_1 + x_2}(x_1 + x_3) + y_1 + a_3$$

(again if $x_1 = x_2$, then $P_1 + P_2 = \infty$). Finally if $P_1 = P_2$, then

$$x_3 = \left(\frac{x_1^2 + a_4}{a_3}\right)^2$$

and

$$y_3 = \frac{x_1^2 + a_4}{a_3}(x_1 + x_3) + y_1 + a_3$$

(recall that we assume $a_3 \neq 0$).

Chapter 2

Pairings on elliptic curves

2.1 Preliminaries

2.1.1 Pairings

Definition 2.1. Let R be a commutative ring with unity, and M, N, L be three R-modules. A *pairing* is a R-bilinear map $e : M \times N \to L$. That is, a map such that

- $e(rm, n) = e(m, rn) = re(m, n)$ for all $r \in R$, $m \in M$ and $n \in N$;
- $e(m_1, n) + e(m_2, n) = e(m_1 + m_2, n)$ for all $m_1, m_2 \in M$ and $n \in N$; and
- $e(m, n_1) + e(m, n_2) = e(m, n_1 + n_2)$ for all $m \in M$ and $n_1, n_2 \in N$.

A familiar example of a pairing is the usual dot-product in Euclidean space. Letting $R = \mathbf{R}$, $M = N = \mathbf{R}^n$ and $L = \mathbf{R}$, M, N, L are R-modules (actually R-vector spaces, since R is a field), and is it well-known that the map $e(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$ satisfies the three conditions above, and is thus a pairing. In the pairings we will use in our elliptic-curve setting, M, N, L will be abelian groups, so we recall that abelian groups are $\mathbf{Z}$-modules with the external law defined by $nx = x + x + \cdots + x$ for n summands.

2.1.2 Torsion subgroups

The *torsion subgroup* of an abelian group G consists of those elements of G which have finite order. That is, an element $g \in G$ lies in the torsion subgroup of G is there is a positive integer n such that $ng = 0$. This is easily seen to be a subgroup of G: it is not empty since it contains 0, and if $mg = nh = 0$, then

$$mn(g - h) = mng - mnh = n0 - m0 = 0 - 0 = 0.$$

(We note that commutativity is essential here; for example in the non-abelian group $\mathrm{GL}_2(\mathbf{C})$, the matrices

$$A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}$$

have finite order, but their product does not.)

We will work with sugroups of the torsion subgroup of a group G. Namely, we will work with the *m-torsion subgroup*, for a positive integer m, which consists of those elements $g \in G$ such that $mg = 0$. This is again a subgroup of G, since it contains 0 and since if $mg = mh = 0$, then

$$m(g - h) = mg - mh = 0 - 0 = 0.$$

Of an elliptic curve

We have seen that given an elliptic curve E defined over a field K, we can consider the set of points of the curve with coordinates in any field $L \supseteq K$. We have also seen that this set can be made into an abelian group, which we denote by $E(L)$. We will consider the m-torsion subgroup of $E(L)$, which we denote by $E(L)[m]$. The next result, which we will not prove, shows that $E(L)[m]$ has a very simple group structure, if we allow L to be sufficiently large.

Proposition 2.2. *Let E be an elliptic curve defined over a field K, and m be an integer which is not a multiple of the characteristic of K (in particular, if the characteristic of K is zero, then m can be any positive integer). Then $E(\overline{K})[m]$ is isomorphic to $\mathbf{Z}/m\mathbf{Z} \times \mathbf{Z}/m\mathbf{Z}$.*

Proof. For a proof, see for example [22, Theorem 3.2] or [19, Corollary III.6.4]. □

In the following, we will denote $E(\overline{K})[m]$ simply by $E[m]$, where K is understood to be a field over which the curve E is defined.

Of roots of unity

If we let G be the multiplicative group of a field K, then the m-torsion subgroup of G is the group of all mth roots of unity in K, which we note $\mu_m(K)$. In other words, $\mu_m(K)$ is the group whose elements are precisely the roots of the polynomial $X^m - 1$ in K. If we make as above the assumption that m is not a multiple of the characteristic of K, then this polynomial does not have any repeated root. Indeed, its derivative is mX^{m-1}, and those two polynomials do not have any common root (the only root of mX^{m-1} is 0, but $0^m - 1 = -1 \neq 0$).

Thus, if $L \supseteq K$ is a field over which $X^m - 1$ splits completely, $\mu_m(L)$ has m elements. Moreover, it is known that a finite subgroup of the multiplicative group of a field is cyclic, so $\mu_m(L)$ is isomorphic to $\mathbf{Z}/m\mathbf{Z}$. This is true in particular if $L = \overline{K}$.

2.2 Rational functions on elliptic curves

In this section, K is an *algebraically closed* field, unless stated otherwise.

2.2.1 On affine elliptic curves

In this section we mostly follow [5, Sections 2.1-2.2]. We will be interested in polynomials and rational functions on elliptic curves. If E is an elliptic curve defined over K and $f \in K[X, Y]$ is a polynomial, then we can consider $f(x, y)$ for any affine point $P = (x, y) \in E(K)$ in the obvious manner, so f induces a map from the set of affine points of $E(K)$ into K, which for now we will call the function induced by f on E. It is clear that if two polynomials f and g differ by a multiple of E (viewed as a polynomial), then they induce the same function on E. Indeed, for any $(x, y) \in E(K)$ we have

$$g(x, y) = f(x, y) + \lambda E(x, y) = f(x, y) + 0 = f(x, y).$$

In fact, the converse is also true: two elements of $K[X,Y]$ which induce the same function on E differ by a multiple of E. To prove this, it is clearly sufficient to prove that if $f \in K[X,Y]$ induces the zero function on E, then it must be a multiple of E. Now, since K is algebraically closed, $E(K)$ contains infinitely many affine points: for every $\alpha \in K$, the polynomial $E(\alpha, Y) \in K[Y]$ has at least one root, so there is at least one point on E whose X-coordinate is α. If $f \in K[X,Y]$ induces the zero function on E, it means that we have $f(x,y) = 0$ for all affine points (x,y) of E. Since also by definition $E(x,y) = 0$ for all such points, this means that there are infinitely many points (x,y) such that $f(x,y) = E(x,y) = 0$. The next result shows that, since E is irreducible, this forces f to be a multiple of E [17, p. 4]. We first need a lemma:

Lemma 2.3. *Let K be any field and E be any elliptic curve defined over K. Then the polynomial defining E is irreducible over K.*

Proof. We recall that E as a polynomial is of the form

$$Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6 \in K[X,Y].$$

Viewing E as a polynomial in $K(X)[Y]$, it is a monic polynomial of degree 2, so assuming for contradiction that it is reducible, it can be written as $E = (Y + f)(Y + g)$, where $f, g \in K(X)$. We then have $E = Y^2 + (f + g)Y + fg$, and so $f + g$ has degree at most 1 while fg has degree 3. This leads to a contradiction, because since fg has odd degree, we must have $\deg f \neq \deg g$. Assuming without loss of generality that $\deg f > \deg g$, we then have $\deg(f + g) = \deg f$, but $\deg f$ must be at least 2. Thus E is irreducible in $K(X)[Y]$, and so it must also be irreducible in $K[X][Y] = K[X,Y]$. □

Proposition 2.4. *Let K be any field, $f \in K[X,Y]$ be any irreducible polynomial, and $g \in K[X,Y]$ be any polynomial not divisible by f. Then there are only finitely many $(x,y) \in \mathbf{A}^2_K$ such that $f(x,y) = g(x,y) = 0$.*

Proof. We view f as a polynomial in $K(Y)[X]$ (that is, a polynomial in one indeterminate X with coefficients in the field $K(Y)$). We first show that there are only finitely many values of β for which there exists some α such that $f(\alpha, \beta) = g(\alpha, \beta) = 0$. If f is a constant polynomial (in $K(Y)[X]$), it means that it is a polynomial in $K[Y]$, and we have that $f(\alpha, \beta) = 0$ if and only if β is a root of $f \in K[Y]$. Since f is irreducible, it cannot be the zero polynomial, and so it can have only finitely many roots.

Say now that $f \in K(Y)[X]$ is not constant. Since f is irreducible in $K[X,Y] = K[Y][X]$, it remains irreducible in $K(Y)[X]$ [9, Lemma 3, Section 2.16]. A similar argument shows that since g is not divisible by f in $K[X,Y]$, it remains not divisible by f in $K(Y)[X]$. Thus f and g are relatively prime in $K(Y)[X]$, and we can write a Bézout relation $f\tilde{u} + g\tilde{v} = 1$ with $\tilde{u}, \tilde{v} \in K(Y)[X]$. Writing the coefficients of $\tilde{u}$ and $\tilde{v}$ with a common denominator $a \in K[Y]$ we obtain $fu + gv = a$, with $u, v \in K[X,Y]$ and $a \neq 0$. Thus if $f(\alpha, \beta) = g(\alpha, \beta) = 0$, then $a(\beta) = 0$, meaning that there are only finitely many suitable values for β.

Finally, for each value of β, the polynomial $f(X, \beta) \in K[X]$ cannot be the zero polynomial (otherwise β would be a root of $f \in K(X)[Y]$, and so f would be divisible by $Y - \beta$). Since $f(\alpha, \beta) = 0$ precisely if α is a root of $f(X, \beta)$, there are only finitely many suitable values of α, which completes the proof. □

Remark 2.5. It is easily seen from the argument above that the number of points (x,y) such that $f(x,y) = g(x,y) = 0$ is at most equal to the product of the degrees of f and g. Bézout's theorem, a classical result in algebraic geometry, tells us that with some additional hypotheses, they cna be rendered exactly equal.

So, in order to study polynomials as function on E, it is sufficient to consider the ring $K[X,Y]/(E)$, which is called the *coordinate ring of* E and noted $K[E]$. Since E is irreducible, $K[E]$ is an integral domain, and we can also consider its field of quotients, which is called the *field of rational functions on* E and noted $K(E)$. Of course, a function $f \in K(E)$ is not necessarily defined at every point $(x,y) \in E(K)$ since its denominator may be zero. Moreover, it is a priori unclear whether the value of a rational function at a point is well-defined. For example if E is defined over $\mathbf{C}$ by $Y^2 = X^3 + X$, the point $P = (0,0)$ is on E. The rational function X/Y appears to not be defined at P, but $XY/Y^2 = XY/(X^3+X) = Y/(X^2+1)$ is the same element of $K(E)$ and is defined at P (with value 0).

Definition 2.6. Let P be a point on E. A rational function $f \in K(E)$ is said to be *defined* (or *regular*) *at* P if there exist $g, h \in K[E]$ such that $f = g/h$ and $h(P) \neq 0$.

Proposition 2.7. *If $f = g/h$ is regular at P and $h(P) \neq 0$ then the value of f at P is $f(P) = g(P)/h(P)$, and is independent of the choice of g and h.*

Proof. Let $f = g_1/h_1 = g_2/h_2$, with $h_1(P), h_2(P)$ both non-zero. Then $g_1h_2 = g_2h_1$ in $K[E]$, meaning that there is some $k \in K[E]$ such that $g_1h_2 - g_2h_1 = kE$. This means that $g_1(P)h_2(P) - g_2(P)h_1(P) = 0$, and so $g_1(P)/h_1(P) = g_2(P)/h_2(P)$, as desired. □

It is easy to see that the rational functions on E which are regular at P form a ring, which is called the *local ring of* E *at* P and noted $\mathcal{O}_P(E)$. Its units are the rational functions f with are regular and non-zero at P, and it is then easy to see that its non-units form an ideal, which means that $\mathcal{O}_P(E)$ is indeed a local ring [4, Théorème 2.77]. The non-units of $\mathcal{O}_P(E)$ form its (unique) maximal ideal, which we denote by $\mathfrak{m}_P(E)$. In fact, $\mathcal{O}_P(E)$ is even a *discrete valuation ring.*

Proposition 2.8. *For an elliptic curve E defined over K and an affine point $P \in E(K)$, the ring $\mathcal{O}_P(E)$ is a discrete valuation ring,* i.e.*, there exists an element $u \in \mathfrak{m}_P(E)$ such that every non-zero element s of $\mathcal{O}_P(E)$ can be written as $s = u^r t$, with r a positive integer and t a unit. The element u is called a* uniformising parameter*, and the value of r is independent of the choice of u.*

Proof. See [5, Theorem 2.11]. □

Since a polynomial is always defined at every point, every polynomial $f \in K[E]$ is in $\mathcal{O}_P(E)$, and if $f \neq 0$ we can write it as $f = u^r t$ as above. The *order of* f *at* P, noted $\mathrm{ord}_P(f)$, is the integer r, and we define also $\mathrm{ord}_P(0) = \infty$. It is easy to see that $\mathrm{ord}_P(fg) = \mathrm{ord}_P(f) + \mathrm{ord}_P(g)$, so we extend the order to any rational function $f = g/h \in K(E)$ by letting $\mathrm{ord}_P(f) = \mathrm{ord}_P(g) - \mathrm{ord}_P(h)$. If $\mathrm{ord}_P(f) > 0$ we say that f has a *zero* at P, and if $\mathrm{ord}_P(f) < 0$ we say that f has a *pole* at P. $|\mathrm{ord}_P(f)|$ is the *multiplicity* of the zero or pole.

Example 2.9. To make the preceeding discussion easier to grasp, it is worthwhile to study an analogous situation in dimension 1, where the involved polynomials are in one indeterminate. We thus let $K = \mathbf{R}$ and consider the subring R of $\mathbf{R}(X)$ (rational functions of the real line to itself) which consists of all f which are regular at 0, in the sense defined above. Then this ring is a discrete valuation ring, a uniformising parameter being for example the polynomial X. Indeed, any polynomial f can be written in a unique way as $f = X^d g$ where g is a unit in R. For example the polynomial $f = X^4 + X^2$ is not a unit in R since its inverse is not regular at 0. However we can write, $f = X^2(X^2+1)$, and X^2+1 is a unit in R, so the order of f at 0 is 2. Similarly, the polynomial $X^3 + X$ has order 1, so we find that the rational function $\frac{X^4+X^2}{X^3+X}$ has a zero of multiplicity 1 at P, which coincides with our usual definition of the multiplicity of a zero.

2.2.2 On projective elliptic curves

We want to consider the order of a rational function on a curve at every point of the curve, including the point at infinity, so we need to generalise the previous discussion to the projective setting. This is a straightforward application of the ideas discussed in Section 1.1. We again follow [5, Sections 2.6-2.7].

We have seen that the values of the zeros of a polynomial in $K[X, Y, Z]$ in the projective plane are well-defined (independent of a choice of representatives) if and only if the polynomial is homogeneous. Similarly, the value of a rational function $f \in K(X, Y, Z)$ at a point $P \in \mathbf{P}^2_K$ is well-defined if and only if $f = g/h$ is a quotient of two homogeneous polynomials of the same degree. Indeed, we then have

$$f(\lambda P) = \frac{g(\lambda P)}{h(\lambda P)} = \frac{\lambda^k g(P)}{\lambda^k h(P)} = \frac{g(P)}{h(P)} = f(P).$$

We define the field of rational functions on the projective elliptic curve E to be the field of all such functions, modulo E. We also introduce some notation for the homogeneisation and dehomogeneisation maps, following [5, Definitions 2.17-2.18].

Definition 2.10. For a point $P = (x, y) \in \mathbf{A}^2_K$, we denote by $P^* = (x, y)^*$ the corresponding point in $\mathbf{P}^2_K$, namely, $(x, y)^* = [x, y, 1]$. For an affine point $P = [x, y, z] \in \mathbf{P}^2_K$ (*i.e.*, a point such that $z \neq 0$), we denote by $P_* = [x, y, z]_*$ the corresponding point in $\mathbf{A}^2_K$, namely, $[x, y, z]_* = (x/z, y/z)$. Those maps are inverse bijections between $\mathbf{A}^2_K$ and the set of affine points of $\mathbf{P}^2_K$.

Definition 2.11. For a polynomial $f \in K[X, Y]$, we denote by f^* the homogeneous polynomial of $K[X, Y, Z]$ obtained by adding powers of Z to all but the highest degree monomials so as to obtain a homogeneous polynomial. More precisely, $f^*(X, Y, Z) = Z^{\deg f} f(X/Z, Y/Z)$. For a homogeneous polynomial $f \in K[X, Y, Z]$, we denote by f_* the polynomial $f(X, Y, 1) \in K[X, Y]$.

As we have seen in Section 1, for any $P \in \mathbf{A}^2_K$ and $f \in K[X, Y]$ we have $f(P) = 0$ if and only if $f^*(P^*) = 0$. If $C \in K[X, Y]$ is an affine curve, C^* is called its *projective closure*, and as before we say that C is the *affine part* of C^*.

Definition 2.12. Let E^* be a projective elliptic curve defined over K. The *field of rational functions on* E^*, noted $K(E^*)$, is the subfield of the field of quotients of the ring $K[E^*] = K[X, Y, Z]/(E^*)$ which consists of the zero function and the functions which are quotients of homogeneous polynomials of the same degree. For a point $P^* \in E^*(K)$, a function $f^* \in K(E^*)$ is *regular* (or *defined*) *at* P^* if there exist $g^*, h^* \in K[E^*]$ homogeneous and of the same degree such that $f^* = g^*/h^*$ and $h^*(P^*) \neq 0$. The functions of $K(E^*)$ which are defined at P^* form a ring, called the *local ring of* E^* *at* P^* and noted $\mathcal{O}_{P^*}(E^*)$. This ring is a local ring.

As before, we will want to work mainly in the affine plane, and use the projective plane only on an *ad hoc* basis when needed. So, given a rational function f on an affine curve E, we want to find a rational function f^* on the projective curve E^* which corresponds to f in the affine plane. We have no problem in going from a rational function on the projective curve E^* to one on the affine curve E: starting from a rational function $f^* = g^*/h^* \in K(E^*)$ we can set $z = 1$ in the numerator and denominator, and obtain the rational function $f = g/h \in K(E)$, whose value at any affine point of E where it is defined is equal to the value of f^* at the corresponding point of E^*. However, we want to go the other way. We cannot just homogeneise g and h as we did before, because g and h need not have the same degree, and homogeneising a polynomial does not change its degree. We solve this problem in a straightforward way: we just multiply the numerator or denominator by the appropriate power of Z to make their degrees equal.

Note that if f is actually a polynomial and is non-constant, the homogeneisation function defined above will differ from the one we use to homogeneise polynomials. For example, the polynomial $X + 1$ will be homogeneised as a polynomial to $X + Z$ and as a rational function to $(X + Z)/Z$. We will use either of those two maps depending on whether we view f as a polynomial or as a rational function, which should be clear from the context.

It is easily verified that the homogeneisation and dehomogeneisation maps described above are inverse field isomorphisms between $K(E)$ and $K(E^*)$, and also that for any affine point $P \in E(K)$ they are inverse ring isomorphisms between $\mathcal{O}_P(E)$ and $\mathcal{O}_{P^*}(E^*)$. Moreover, for all $f \in \mathcal{O}_P(E)$ we have $f(P) = f^*(P^*)$ [5, Proposition 2.23]. Finally, the analogue of Proposition 2.8 holds:

Proposition 2.13. *For a projective elliptic curve E and a point $P \in E(K)$, the ring $\mathcal{O}_{P^*}(E^*)$ is a discrete valuation ring.*

Proof. See [5, Theorem 2.26 and Lemma 2.31]. If P is an affine point, then this is exactly Proposition 2.8, so let $P = \infty$. Then a uniformising parameter is $u = X/Y$. If $f = g/h \in \mathcal{O}_P(E)$ with $h(P) \neq 0$, write $g_* = v + wY \in K[E_*]$, with $v, w \in K[X]$. Then f has a pole of degree $\max\{2 \deg v, 2 \deg w + 3\}$ at ∞. □

2.3 Divisors

In this section we follow [5, Section 2.8] and [22, Section 11.1].

The *divisor* of a rational function $f \in K(E)$ is simply a convenient notation to summarise its zeros and poles and their multiplicities. It is an element of the free abelian group generated by formal symbols corresponding to the points of $E(K)$. We first define divisors in general.

Definition 2.14. Let E be an elliptic curve defined over K. A *divisor on E* is an element of the free abelian group generated by formal symbols corresponding to the points of $E(K)$, *i.e.*, a formal sum

$$\sum_{P \in E(K)} n_P[P],$$

where the n_P are integer coefficients, only finitely many of which are non-zero, and the $[P]$ are formal symbols. The group of divisors on E is noted $\mathrm{Div}(E)$.

We define two functions on divisors:

Definition 2.15. Let $D = \sum_{P \in E(K)} n_P[P] \in \mathrm{Div}(E)$. The *degree of D* is the sum of its coefficients:

$$\deg(D) = \sum_{P \in E(K)} n_P \in \mathbf{Z},$$

and its *sum* is obtained by dropping the square brackets:

$$\mathrm{sum}(D) = \sum_{P \in E(K)} n_P P \in E(K).$$

It is clear that the degree and sum functions give surjective group homomorphisms of $\mathrm{Div}(E)$ into $\mathbf{Z}$ and $E(K)$ respectively. Moreover, the kernel of the degree homomorphism (*i.e.*, the set of divisors of degree 0) is an important subgroup of $\mathrm{Div}(E)$, which we note $\mathrm{Div}^0(E)$. The restriction of the sum homomorphism to $\mathrm{Div}^0(E)$ is still surjective, because for any $P \in E(K)$ we have $\deg([P] - [\infty]) = 1 - 1 = 0$ and $\mathrm{sum}([P] - [\infty]) = P - \infty = P$.

As indicated above, we define the divisor of a rational function $f \in E(K)$:

Definition 2.16. Let $f = g/h \in E(K)$ be a non-zero rational function on E. The *divisor of* f, noted $\mathrm{div}(f)$, is the formal sum

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

We note that this is indeed a divisor (*i.e.*, a *finite* formal sum), since by Proposition 2.4, g and h can only have finitely many common zeros with E (and if f has order non-zero at a point P, at least one of g and h must be zero at P). We note also that since $\mathrm{ord}_P(f_1 f_2) = \mathrm{ord}_P(f_1) + \mathrm{ord}_P(f_2)$, div as a map from $E(K)^\times$ to $\mathrm{Div}(E)$ is a group homomorphism. A divisor $D \in \mathrm{Div}(E)$ is called a *principal divisor* if there is some $f \in E(K)^\times$ such that $\mathrm{div}(f) = D$. The set of all principal divisors is noted $\mathrm{Prin}(E)$ and is clearly a subgroup of $\mathrm{Div}(E)$ (the zero divisor is the divisor of a non-zero constant function, and $\mathrm{div}(f) - \mathrm{div}(g) = \mathrm{div}(f/g)$). The quotient group $\mathrm{Div}(E)/\mathrm{Prin}(E)$ is called the *Picard group* or *divisor class group* of E and noted $\mathrm{Pic}(E)$.

We now wish to completely characterise the principal divisors. The next result, which we will not prove, gives a necessary condition for a divisor to be principal.

Proposition 2.17. *Any non-zero rational function on an elliptic curve has an equal number of zeros and poles, counting multiplicities. In other words,* $\mathrm{Prin}(E)$ *is a subgroup of* $\mathrm{Div}^0(E)$.

Proof. See for example [5, Theorem 2.28]. □

We now wish to relate, for two points $P_1, P_2 \in E(K)$, the divisors $[P_1] + [P_2]$ and $[P_1 + P_2]$. It is clear that they have the same sum, but not the same degree. Suppose that $P_1 \neq -P_2$ and let $aX + bY + c$ be the line through P_1 and P_2, which is thus not vertical. It intersects E at a third point P_3, and so the function $f = aX + bY + c \in K(E)$ has a zero of multiplicity 1 at each of P_1, P_2, P_3. To compute the multiplicity of its pole at ∞ as in Proposition 2.13 we write $f = (aX + c) + bY$ and we see that f has a pole of multiplicity 3 at ∞, so

$$\mathrm{div}(f) = [P_1] + [P_2] + [P_3] - 3[\infty].$$

The line through $P_3 = (x_3, y_3)$ and $-P_3 = (x_3, -y_3)$ is vertical with equation $X - x_3$. Letting $g = X - x_3 \in K(E)$, we find

$$\mathrm{div}(g) = [P_3] + [-P_3] - 2[\infty],$$

and so

$$\mathrm{div}(f/g) = [P_1] + [P_2] - [-P_3] - [\infty].$$

Since $-P_3 = P_1 + P_2$, we obtain finally

$$[P_1] + [P_2] = [P_1 + P_2] + [\infty] + \mathrm{div}(f/g).$$

We need one more lemma, which we will not prove:

Lemma 2.18. *Let* $P, Q \in E(K)$. *If there exists a rational function* $f \in K(E)$ *such that* $\mathrm{div}(f) = [P] - [Q]$, *then* $P = Q$.

Proof. See for example [22, Lemma 11.3]. □

We can now state:

Proposition 2.19. *Let* E *be an elliptic curve, and* $D \in \mathrm{Div}^0(E)$. *Then* D *is a principal divisor if and only if* $\mathrm{sum}(D) = \infty$.

We note that by Proposition 2.17 this is equivalent to saying that a divisor is principal if and only if it has degree zero and sum ∞.

Proof. We have seen that for any $P_1, P_2 \in E(K)$, $[P_1] + [P_2]$ equals $[P_1 + P_2] + [\infty] + \operatorname{div}(g)$ for some $g \in K(E)$. We note that

$$\operatorname{sum}(\operatorname{div}(g)) = P_1 + P_2 - P_1 - P_2 - \infty = \infty.$$

Thus any sum $[P_1]+[P_2]$ can be written as $[P]+k[\infty]$ plus a principal divisor, for some $P \in E(K)$ (possibly ∞) and $k > 0$, and this generalises by induction to any sum $[P_1]+[P_2]+\cdots+[P_n]$. We have similarly $-[P_1]-[P_2]-\cdots-[P_n] = -[P]-k[\infty]$ minus a principal divisor. Thus, summing separately the symbols of D with positive and negative coefficients, we get

$$\begin{aligned} D &= [P] + m[\infty] + \operatorname{div}(f) - [Q] - n[\infty] - \operatorname{div}(g) \\ &= [P] - [Q] + (m-n)[\infty] + \operatorname{div}(f/g), \end{aligned}$$

and since $\operatorname{sum}(\operatorname{div}(f)) = \operatorname{sum}(\operatorname{div}(g)) = \infty$, we have $\operatorname{sum}(\operatorname{div}(f/g)) = \infty$. By Proposition 2.17, we have also $\deg(\operatorname{div}(f/g)) = 0$, and since $\deg(D) = 0$ by assumption, we get

$$\deg(D) = 1 - 1 + (m-n) + 0 = m - n = 0,$$

so $D = [P] - [Q] + \operatorname{div}(f/g)$, and $\operatorname{sum}(D) = P - Q$.

Suppose that $\operatorname{sum}(D) = \infty$, then $P - Q = \infty$, so $P = Q$, and $D = \operatorname{div}(f/g)$, so D is a principal divisor. Conversely, if $D = \operatorname{div}(h)$, then $[P] - [Q] = \operatorname{div}(gh/f)$. Then $P = Q$ by Lemma 2.18, so $\operatorname{sum}(D) = \infty$. □

Corollary 2.20. *$E(K)$ is a group, isomorphic to* $\operatorname{Pic}^0(E) = \operatorname{Div}^0(E)/\operatorname{Prin}(E)$.

Proof. This follows from the isomorphism theorem for quotient groups: as noted after Definition 2.15, the sum homomorphism from $\operatorname{Div}^0(E)$ to $E(K)$ is surjective, and by Proposition 2.19 its kernel is precisely $\operatorname{Prin}(E)$. Although here we have freely used associativity in $E(K)$ to make arguments easier, it is possible to arrive at this result without using it; this is how associativity is proved for example in [5]. ([22] uses a different argument.) □

Finally, we will need the following:

Proposition 2.21. *Two rational functions with the same divisor differ by a multiplicative constant,* i.e.*, if $f, g \in K(E)^\times$ and $\operatorname{div} f = \operatorname{div} g$, then $f = cg$ for some $c \in K^\times$.*

Proof. See [5, Corollary 2.35]. □

2.4 The Weil pairing

2.4.1 Definition and properties

We define the Weil pairing following [8, Section 5.8.3]. This definition differs from (but is of course equivalent to) the usual one (*e.g.*, the one given in [19, Section III.8]), but makes the Weil pairing practical to compute using an algorithm due to Miller [14] (see also [19, Remark XI.8.2]), which we describe in the next section.

Let E be an elliptic curve defined over a field K. We recall that $E[m]$ is the m-torsion subgroup of $E(\overline{K})$, and that it is isomorphic to $\mathbf{Z}_m \otimes \mathbf{Z}_m$ whenever m is not a multiple of the characteristic of K. We wish to define a pairing $e_m : E[m] \times E[m] \to \mu_m(\overline{K})$. Let $P, Q \in E[m]$.

Since $mP = \infty$, the divisor $m[P] - m[\infty]$ has degree zero and sum ∞, so it is principal by Proposition 2.19. Let $m[P] - m[\infty] = \operatorname{div} f_P$, and likewise let $m[Q] - m[\infty] = \operatorname{div} f_Q$. The *Weil pairing* e_m is defined by

$$e_m(P, Q) = \frac{f_P(Q+S)}{f_P(S)} \Big/ \frac{f_Q(P-S)}{f_Q(-S)} ,$$

where $S \in E(K) \setminus \{\infty, P, -Q, P-Q\}$ (this ensures that all the quantities appearing in the right-hand side are defined and non-zero).

Proposition 2.22. *The value of the Weil pairing $e_m(P, Q)$ is independent of the choice of f_P, f_Q and S.*

Proof. That $e_m(P, Q)$, for a given S, is independent from the choice of f_P and f_Q is clear by Proposition 2.21: f_P and f_Q are defined by their divisors up to multiplicative constants, and the constants cancel out in the quotients. To show that it is independent of the choice of S, we consider the function

$$f : S \mapsto \frac{f_P(Q+S)}{f_P(S)} \Big/ \frac{f_Q(Q-S)}{f_Q(-S)} .$$

This function is zero at S if and only if $f_P(Q+S)$ is zero. Since the only zero of f_P is at P with multiplicity m, we see that f has a zero of multipicity m at $P - Q$. Likewise, it has a pole of multiplicity m at $P - Q$, so $\operatorname{div} f$ is the zero divisor, meaning that f is constant. □

Proposition 2.23. *The Weil pairing has the following properties:*

- *It is* bilinear*: for any $P, P_1, P_2, Q, Q_1, Q_2 \in E[m]$, we have*

  $$e_m(P_1, Q)e_m(P_2, Q) = e_m(P_1 + P_2, Q),$$

  and

  $$e_m(Q, P_1)e_m(Q, P_2) = e_m(Q, P_1 + P_2).$$

  This implies that $e_m(P, Q)$ is indeed a mth root of unity for all $P, Q \in E[m]$ since

  $$e_m(P, Q)^m = e_m(P, mQ) = e_m(P, \infty) = 1.$$

- *It is* alternating*: for any $P \in E[m]$, we have*

  $$e_m(P, P) = 1,$$

  which together with bilinearity implies that

  $$e_m(P, Q) = e_m(Q, P)^{-1}$$

  for all $P, Q \in E[m]$.

- *It is* nondegenerate*: for any $P \in E[m]$, if $e_m(P, Q) = 1$ for all $Q \in E[m]$, then $P = \infty$.*

- *It is* Galois-invariant*: for any $\sigma \in \operatorname{Gal}(\overline{K}/K)$, we have $e_m(\sigma(P), \sigma(Q)) = \sigma(e_m(P, Q))$.*

Proof. See for example [22, Theorem 11.7]. □

We now prove some additional properties of the Weil pairing, which are consequences of the above properties, and will be used later. Since $E[m]$ is isomorphic to $\mathbf{Z}_m \otimes \mathbf{Z}_m$ (if the characteristic of K does not divide m), it is a free $\mathbf{Z}_m$-module of rank 2. In other words, there exist two elements $P, Q \in E[m]$ such that any $R \in E[m]$ can be written as $R = aP + bQ$ for some $a, b \in \mathbf{Z}_m$, and it follows immediately that such a decomposition is unique. The set $\{P, Q\}$ is then called a *basis* for $E[m]$.

Proposition 2.24. *If $\{P, Q\}$ is a basis for $E[m]$ as a $\mathbf{Z}_m$-module, then $e_m(P, Q)$ is a primitive mth root of unity.*

Proof. Let $e_m(P, Q) = \zeta$, and $d \in \mathbf{Z}$ be such that $\zeta^d = 1$, we wish to show that d is a multiple of m. We have $e_m(P, dQ) = \zeta^d = 1$, and also $e_m(Q, dQ) = e_m(Q, Q)^d = 1^d = 1$. Let $S = aP + bQ \in E[m]$, then

$$e_m(S, dQ) = e_m(P, dQ)^a e_m(Q, dQ)^b = 1^a 1^b = 1,$$

and since this is true for all S, we have $dQ = \infty$ by the non-degeneracy of the Weil pairing. This implies that d is a multiple of m (and so is zero in $\mathbf{Z}_m$), since $\infty = 0P + 0Q$ is the unique decomposition of ∞ in the basis $\{P, Q\}$ with coefficients in $\mathbf{Z}_m$. □

Proposition 2.25. *If $P, Q \in E[m]$ lie in $E(L)$, then $e_m(P, Q) \in \mu_m(L)$.*

Proof. We know that $e_m(P, Q) \in \mu_m(\overline{K})$. Let $\sigma \in \mathrm{Gal}(\overline{K}/L)$, we have

$$\sigma(e_m(P, Q)) = e_m(\sigma(P), \sigma(Q)) = e_m(P, Q),$$

which shows that $e_m(P, Q) \in L$. □

Corollary 2.26. *If $L \supseteq K$ is such that $E[m] \subseteq E(L)$ (in other words, such that all the m-torsion points have coordinates in L), then $X^m - 1 \in K[X]$ splits in L (in other words, the group $\mu_m(\overline{K})$ of mth roots of unity in $\overline{K}$ is contained in L).*

Proof. Let $P, Q \in E(L)$ such that $\{P, Q\}$ is a basis for $E[m]$, we have $e_m(P, Q) = \zeta$, where $\zeta \in \overline{K}$ is a primitive mth root of unity by Proposition 2.24. Since also $\zeta \in L$ by Proposition 2.25, this proves the result. □

We will also need the following.

Proposition 2.27. *Let $P \in E[m]$ be of order m. Then there is some $Q \in E[m]$ such that $\{P, Q\}$ is a basis for $E[m]$.*

Proposition 2.28. *Let $P \in E[m]$ be of order m. The map $\phi : Q \mapsto e_m(P, Q)$ from $E[m]$ to $\mu_m(\overline{K})$ is a group homomorphism with kernel $\langle P \rangle$. This implies that the group $E[m]/\langle P \rangle$ is isomorphic to $\mu_m(\overline{K})$.*

Proof. That ϕ is a homomorphism follows from the bilinearity of the Weil pairing, and it is clear that its kernel contains $\langle P \rangle$. Suppose now that $Q \notin \langle P \rangle$, and let $\{P, R\}$ be a basis for $E[m]$. We can then write $Q = aP + bR$ with $b \neq 0$ in $\mathbf{Z}_m$. But then $e_m(P, Q) = e_m(P, R)^b$, and since $e_m(P, R)$ is a primitive mth root of unity, it follows that $e_m(P, Q) \neq 1$. □

2.4.2 Computation

Given a point $P \in E(K)$ and a positive integer m, Miller's algorithm efficiently computes a rational function $f_P \in K(E)$ such that

$$\mathrm{div}(f_P) = m[P] - [mP] - (m - 1)[\infty],$$

and so if $P \in E[m]$ the function f_P will have the divisor prescribed by the definition of the Weil pairing in the previous section. It is a "double-and-add" algorithm, similar to the well-known "fast exponentiation" algorithm, and is based on the following result:

Proposition 2.29. *Let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be two points on an elliptic curve E, and let λ be the slope of the line through P and Q (if $P = Q$, that's the tangent to the curve at P, and also if the line is vertical we let $\lambda = \infty$). Define a function $g_{P,Q} \in K(E)$ by*

$$g_{P,Q} = \frac{Y - y_P - \lambda(X - x_P)}{X + x_P + x_Q - \lambda^2}$$

if $\lambda \neq \infty$, and $g_{P,Q} = X - x_P$ if $\lambda = \infty$. Then

$$\operatorname{div}(g_{P,Q}) = [P] + [Q] - [P + Q] - [\infty].$$

Proof. Suppose first that the line through P and Q is not vertical, and let $Y = \lambda X + \nu$ be its equation. This line intersects E at the points P, Q and $-P - Q$, so if we see it as a function, its divisor is

$$\operatorname{div}(Y - \lambda X - \nu) = [P] + [Q] + [-P - Q] - 3[\infty].$$

Also the vertical line $X - x_{P+Q}$ which goes through $P + Q$ (which is not ∞) has divisor

$$\operatorname{div}(X - x_{P+Q}) = [P + Q] + [-P - Q] - 2[\infty],$$

so finally the function

$$g_{P,Q} = \frac{Y - \lambda X - \nu}{X - x_{P+Q}}$$

has the prescribed divisor. By the addition formula, $x_{P+Q} = \lambda^2 - x_P - x_Q$, and replacing ν in the numerator by $y_P - \lambda x_P$ gives the desired function.

If the line through P and Q is vertical, then $P + Q = \infty$, and the function $X - x_P$ has the prescribed divisor (namely, $[P] + [Q] - 2[\infty] = [P] + [Q] - [P + Q] - [\infty]$). □

Miller's algorithm is given as Algorithm 1. The proof of its corectness is a straightforward but lengthy induction (see [19, Theorem XI.8.1]).

Algorithm 1 Miller's algorithm

Input: A point $P \in E(K)$ and an integer $m > 0$ not divisible by the characteristic of K and given as a binary representation $m_0, m_1, \ldots, m_{n-1}$, where m_0 is the least significant bit and $m_{n-1} = 1$

Output: A function $f_P \in K(E)$ such that $\operatorname{div} f_P = m[P] - [mP] - (m-1)[\infty]$

```
T ← P, f ← 1
for i = n − 2 down to 0 do
    f ← f^2 g_{T,T}
    T ← 2T
    if m_i = 1 then
        f ← f g_{T,P}
        T ← T + P
    end if
end for
return f
```

2.5 The Tate pairing

The Tate pairing is another pairing on the group of points of an elliptic curve, which is often used in cryptography in lieu of the Weil pairing since it is somewhat easier to compute. While the Tate pairing can in principle be used for all the applications we discuss, we will only use it to reduce the elliptic-curve discrete logarithm problem to a finite-field discrete logarithm problem. This reduction using the Tate pairing was introduced by Frey and Rück [6], and so we will mostly follow [6] in our treatment of the Tate pairing. However, its treatment is fairly abstract and general, so we will always specialise it to the particular case of elliptic curves.

We have seen before that the group of points of an elliptic curve E is isomorphic to the quotient group $\mathrm{Pic}^0(E) = \mathrm{Div}^0(E)/\mathrm{Prin}(E)$, where $\mathrm{Div}^0(E)$ is the group of divisors on E of degree 0 and $\mathrm{Prin}(E)$ is the group of principal divisors on E. Moreover, a divisor $D \in \mathrm{Div}^0(E)$ is principal if and only if its sum is ∞, which means that two divisors $D, D' \in \mathrm{Div}^0(E)$ are in the same class modulo $\mathrm{Prin}(E)$ if and only if they have the same sum. This means that we can identify a point $P \in E(K)$ with the set of divisors of $\mathrm{Div}^0(E)$ whose sum is P, and a natural choice of a representative from this set is the divisor $[P] - [\infty]$, so in the end we can identify a point $P \in E(K)$ with the divisor $[P] - [\infty] \in \mathrm{Div}^0(E)$. In the following, we will denote the divisor $[P] - [\infty]$ by $\overline{P}$. Also, we assume that E is defined over some finite field $\mathbf{F}_q$, and when working with m-torsion points we let $K = \mathbf{F}_{q^k}$, where k is the smallest integer such that m divides $q^k - 1$ (so $\mu_m(\overline{K}) \subseteq K$).

As before, if P is a m-torsion point, the divisor $m[P] - m[\infty]$ has degree zero and sum ∞, and so is a principal divisor (another way to look at it is to say that (the class of) the divisor $\overline{P}$, which we identify to the point P, is m-torsion in the group $\mathrm{Pic}^0(E) = \mathrm{Div}^0(E)/\mathrm{Prin}(E)$, meaning that $m\overline{P} = m[P] - m[\infty]$ is in $\mathrm{Prin}(E)$). Let again $f_P \in K(E)$ be a function whose divisor is $m[P] - m[\infty]$, and let Q be any point of $E(K)$. The divisor $[Q+S]-[S]$, for any point S, has degree zero and sum Q, and is thus a representative of Q as an element of $\mathrm{Pic}^0(E)$. Moreover, it is always possible to choose S so that the divisors $[P]-[\infty]$ and $[Q+S]-[S]$ have no points in common (unless the curve is very small, and thus useless), by choosing $S \notin \{\infty, P, -Q, P-Q\}$.

If $f \in K(E)$ is a rational function on E and $D = \sum n_i P_i \in \mathrm{Div}(E)$ is a divisor on E such that $\mathrm{div}\, f$ and D have no point in common, we define

$$f(D) = \prod f(P_i)^{n_i} \in K^*.$$

This allows us to define a non-degenerate pairing as follows. Let $D \in \mathrm{Div}^0(E)$ be such that $\overline{D}$ is m-torsion in $\mathrm{Pic}_0(E)$. This means that mD is principal, so let $f \in K(E)$ be a rational function with divisor mD. Further, let $D' \in \mathrm{Div}^0(E)$ be a divisor which has no common point with D, and so with mD. Then our pairing is the map

$$\{\cdot,\cdot\}_m : \mathrm{Pic}^0(E)[m] \times \mathrm{Pic}^0(E)/m\,\mathrm{Pic}^0(E) \to K^*/(K^*)^m$$

defined by $\{\overline{D}, \overline{D'}\}_m = f(D')$. Identifying as above a point $P \in E(K)$ with the set of divisors in $\mathrm{Div}^0(E)$ whose sum is P, we obtain the pairing

$$\{\cdot,\cdot\}_m : E(K)[m] \times E(K)/mE(K) \to K^*/(K^*)^m$$

defined by $\{P, Q\}_m = f_P(D_Q)$, where D_Q is a representative of Q as an element of $\mathrm{Pic}^0(E)$ which has no common point with $[P]-[\infty]$. We have seen that we can let $D_Q = [Q+S]-[S]$ for some $S \notin \{\infty, P, -Q, P-Q\}$, and in that case we have $\{P, Q\}_m = f_P(Q+S)/f_P(S)$ (we note that this is the definition given for example in [8, Section 5.8.3]).

As it is, the values of our pairing are elements of $K^*/(K^*)^m$, that is, they are only defined modulo mth powers, so it would be desirable to modify it so that the values are in K^*. To do

this, we use the map $\phi : x \mapsto x^{(q^k-1)/m}$ from K^* to itself. It is obviously a group homomorphism, and is kernel is exactly $(K^*)^m$ (if we let α be a generator of K^*, then $(\alpha^\ell)^{(q^k-1)/m} = 1$ if and only if ℓ is a multiple of m, in other words α^ℓ is an mth power). Since also $\phi(x)$ is a mth root of unity for all x, we see finally that ϕ gives an isomorphism between $K^*/(K^*)^m$ and $\mu_m(K)$, so we can define a new pairing

$$\tau_m : E(K)[m] \times E(K)/mE(K) \to \mu_m(K)$$

by $\tau_m(P, Q) = \phi(\{P, Q\}_m)$. The first pairing is (a variant of) the *Tate pairing*, and the second is (a variant of) the *modified Tate pairing*. We note that the Tate pairing can be defined in general, while the modified Tate pairing requires that we work in a field which contains the mth roots of unity.

Chapter 3

Applications

3.1 The MOV attack

The original motivation for the use of elliptic curves in cryptography was the existence of sub-exponential algorithms to solve the discrete logarithm problem in the multiplicative group of a finite field (the "index calculus" algorithms). Since no such algorithm is known to solve the discrete logarithm problem in the group of points of a general elliptic curve, cryptographic algorithms based on the discrete logarithm problem, such as the Diffie-Hellman key exchange algorithm or the ElGamal and DSA cryptosystems, can be implemented to use the group of points of an elliptic curve instead of the multiplicative group of a finite field, resulting in shorter keys for the same level of security. Since using shorter keys makes arithmetic operations computationally easier, this is especially attractive on low-resource systems such as embedded systems or smart cards.

However, every cryptosystem has some weak instances, which must be avoided in order to obtain a secure system. For example, in any discrete-logarithm-based cryptosystem it is crucial that the order of the chosen generator in a group should not be easily factored, in order to prevent attacks by the Pohlig-Hellman algorithm. Similarly, Pollard's $p-1$ algorithm easily defeats any RSA implementation in which $p-1$, where p is one of the two prime factors of the modulus, has only small prime factors. One of the first applications of elliptic-curve pairings in cryptography was to use the Weil pairing to reduce the discrete logarithm problem on elliptic curves defined over a finite field $\mathbf{F}_q$ to the discrete logarithm problem in a finite field $\mathbf{F}_{q^k}$. If k is small enough, the discrete logarithm problem in the field $\mathbf{F}_{q^k}$ is significantly easier than the original elliptic-curve discrete logarithm problem, and the benefits of using elliptic curves vanish: the curves for which this happens are among the weak instances of cryptosystems based on the elliptic-curve discrete logarithm problem. This reduction is due to Menezes, Okamoto and Vanstone [12], and is now referred to as the MOV attack.

3.1.1 Generalities

We first give some preliminary results. In all the following, E is an elliptic curve defined over a finite field $\mathbf{F}_q$, where $q = p^r$ for a prime p. We recall that $E(\mathbf{F}_q)$ is the group of points of E with coordinates in $\mathbf{F}_q$.

Theorem 3.1 (Hasse)**.** *The order of the group $E(\mathbf{F}_q)$ satisfies*

$$|E(\mathbf{F}_q)| = q + 1 - t,$$

where $|t| \leq 2\sqrt{q}$. *t is called the* trace of Frobenius.

Proof. See for example [19, Theorem V.1.1]. □

Hasse's theorem gives relatively tight bounds on the order of $E(\mathbf{F}_q)$, but it is often necessary to compute it exactly. Thanks to an algorithm of Schoof [16] (later improved by Elkies and Atkin, and now referred to as the SEA algorithm), this is doable in deterministic polynomial time. We can also be more specific about the group structure of $E(\mathbf{F}_q)$:

Proposition 3.2. *As a group, $E(\mathbf{F}_q)$ is isomorphic to $\mathbf{Z}_{n_1} \otimes \mathbf{Z}_{n_2}$, where $n_1, n_2 \geq 1$ and $n_2 \mid n_1$.*

Proof. See for example [22, Theorem 4.1]. □

We will also need to generate random points in $E(\mathbf{F}_q)$. We do this in a straightforward manner: we pick an element $x_1 \in \mathbf{F}_q$ at random, and see whether there is a point in $E(\mathbf{F}_q)$ with x-coordinate x_1. Determining whether such a point exists amounts to determining whether a quadratic polynomial with coefficients in $\mathbf{F}_q$ has a root in $\mathbf{F}_q$, and this can be done in probabilistic polynomial time [2]. Since there are at least $q+1-2\sqrt{q}$ points in $E(\mathbf{F}_q)$ and since for each value of x_1 there are at most two points with x-coordinate x_1, there are at least $(q+1-2\sqrt{q})/2$ values of x_1 for which such a point exists. Thus if we pick x_1 uniformly at random, the probability that a point with x-coordinate x_1 exists is at least

$$\frac{q+1-2\sqrt{q}}{2q} \geq \frac{1}{2} + \frac{1}{2q} - \frac{1}{\sqrt{q}} \geq \frac{1}{2} - \frac{1}{\sqrt{q}}.$$

We recall that the discrete logarithm problem is, given two elements g, h of a group G, to find an integer k such that $h = g^k$. In our elliptic-curve setting, we are given two points $P, Q \in E(\mathbf{F}_q)$ and wish to find an integer k such that $Q = kP$. A natural question is whether such a k exists at all, in other words, whether Q is in the subgroup of $E(\mathbf{F}_q)$ generated by P. In a cryptographic setting, a user typically computes Q by multiplying P by his private key k (which an attacker wishes to find), so it is assumed that k exists. Nevertheless, the Weil pairing provides a simple way to check this, which also gives the basic idea of the MOV attack. In all the following, we let n be the order of $P \in E(\mathbf{F}_q)$ and assume that $\gcd(n, q) = 1$.

Proposition 3.3. *There exists an integer k such that $Q = kP$ if and only if $nQ = \infty$ and $e_n(P, Q) = 1$.*

Proof. If $Q = kP$, then

$$nQ = nkP = k\infty = \infty,$$

and also

$$e_n(P, Q) = e_n(P, kP) = e_n(P, P)^k = 1^k = 1.$$

Conversely, suppose $nQ = \infty$, so $Q \in E[n]$. Let $\{P, R\}$ be a basis for $E[n]$ (in a sufficiently large field), so $e_n(P, R) = \zeta$ is a primitive nth root of unity, and write $Q = aP + bR$. Then

$$e_n(P, Q) = e_n(P, P)^a e_n(P, R)^b = \zeta^b,$$

but since also $e_n(P, Q) = 1$ by hypothesis and ζ is a primitive nth root of unity, we find that b must be a multiple of n. Thus $bR = \infty$, and $Q = aP$. □

Let n be the order of P in $E(\mathbf{F}_q)$ (for the curves occurring in practice, the order of $E(\mathbf{F}_q)$ is sufficiently small to be factored, and by iterating through its divisors we can quickly determine n), $\mathbf{F}_{q^k}$ be an extension of $\mathbf{F}_q$ such that $E[n]$ is contained in $E(\mathbf{F}_{q^k})$, and $Q \in \mathbf{F}_{q^k}$ be such that $\{P, Q\}$ is a basis for $E[n]$.

Proposition 3.4. *The Weil pairing induces an isomorphism between* $\langle P \rangle$ *and* $\mu_n(\mathbf{F}_{q^k})$ *by* $f : R \mapsto e_n(R, Q)$.

Proof. f is a homomorphism by the bilinearity of the Weil pairing. Moreover, if $f(kP) = e_n(kP, Q) = e_n(P, Q)^k = 1$, then k must be a multiple of n since $e_n(P, Q)$ is a primitive nth root of unity, which means that $kP = \infty$, and f is injective. This shows that f is an isomorphism since the groups are finite. □

The reduction is now straightforward (Algorithm 2). It gives the desired result because

$$\beta = e_n(mP, R) = e_n(P, R)^m = \alpha^m,$$

and so $\ell \equiv m \pmod{n}$, and since the discrete logarithm computation takes place in $\mathbf{F}_{q^k}$, we have reduced the discrete logarithm problem in $E(\mathbf{F}_q)$ to the discrete logarithm in $\mathbf{F}_{q^k}$. However, k will in general be too large for this to provide any advantage, and in any case we have not provided a way to determine k and R. We do this in the next paragraph for a certain class of curves where k is small.

Algorithm 2 MOV reduction

Input: An elliptic curve E defined over $\mathbf{F}_q$, a point $P \in E(\mathbf{F}_q)$ of order n relatively prime with q, and a point $Q \in \langle P \rangle$

Output: An integer m such that $Q = mP$

1. Determine the smallest integer k such that $E[n] \subseteq E(\mathbf{F}_{q^k})$.
2. Find a point $R \in E(\mathbf{F}_{q^k})$ such that $e_n(P, R)$ is a primitive nth root of unity (this is true in particular if $\{P, R\}$ is a basis for $E[n]$) and compute $\alpha = e_n(P, R) \in \mathbf{F}_{q^k}$.
3. Compute $\beta = e_n(Q, R) \in \mathbf{F}_{q^k}$.
4. Solve the discrete logarithm $\beta = \alpha^\ell$ in $\mathbf{F}_{q^k}$.
5. Output ℓ.

3.1.2 Supersingular curves

We have not described in Algorithm 2 how to determine k and R. We will now do this for a special class of curves called *supersingular* curves, for which k is small and in which the MOV attack therefore becomes practical. Those curves are among the weak cases alluded to above. This is unfortunate, because supersingular curves on the other hand allow for faster multiplication of a point by an integer (see [22, p. 132]).

Definition 3.5. An elliptic curve E defined over a finite field $\mathbf{F}_q$ of characteristic p is said to be *supersingular* if its trace of Frobenius $t = q + 1 - |E(\mathbf{F}_q)|$ is a multiple of p.

Proposition 3.6. *Let* E *be a supersingular elliptic curve defined over a finite field* $\mathbf{F}_q$*, and* P *be a point of order* n *in* $E(\mathbf{F}_q)$*. Then* n *is relatively prime with* q*, and there exists an integer* $k \leq 6$ *such that* $E[n] \subseteq E(\mathbf{F}_{q^k})$*. Further, the group* $E(\mathbf{F}_{q^k})$ *is isomorphic to* $\mathbf{Z}_{cn_1} \otimes \mathbf{Z}_{cn_1}$ *for some integer* c *(where* n_1 *is as in Proposition 3.2).*

Proof. Let $q = p^r$. Since p divides t and q, we have $|E(\mathbf{F}_q)| \equiv q + 1 - t \equiv 1 \pmod{p}$. If p divided n, since also n divides $|E(\mathbf{F}_q)|$ it would mean that $|E(\mathbf{F}_q)| \equiv 0 \pmod{p}$, a contradiction. For the other assertions, see [12] and its references. □

Algorithm 3 MOV reduction for supersingular elliptic curves

Input: A supersingular elliptic curve E defined over $\mathbf{F}_q$, a point $P \in E(\mathbf{F}_q)$ of order n, and a point $Q \in \langle P \rangle$

Output: An integer m such that $Q = mP$

1. Determine k and c (see Table 1 in [12]).
2. Let R be a random point in $E(\mathbf{F}_{q^k})$, and $S = (cn_1/n)R$
3. Compute $\alpha = e_n(P, S)$ and $\beta = e_n(Q, S)$.
4. Compute the disctete logarithm $\beta = \alpha^\ell$ in $\mathbf{F}_{q^k}$.
5. If $Q = \ell P$, output ℓ. Otherwise, α is not a primitive nth root of unity, so go back to step 2.

The complete reduction algorithm for supersingular curves is Algorithm 3. The algorithm succeeds in computing m if the point S chosen in step 2 is such that $\alpha = e_n(P, S)$ is a primitive nth root of unity, so we need to estimate the probability that this happens. First we show that since R is chosen uniformly in $E(\mathbf{F}_{q^k})$, the distribution of S in $E[n]$ is also uniform.

Proposition 3.7. *Let G be a group isomorphic to $\mathbf{Z}_{cn} \otimes \mathbf{Z}_{cn}$, and H be the subgroup of G isomorphic to $\mathbf{Z}_n \otimes \mathbf{Z}_n$. Suppose we pick an element $x \in G$ uniformly at random. Then the distribution of the element $cx \in H$ will be uniform in H.*

Proof. It is clear that cx is in H. Let now $(ca, cb) \in H$, we wish to find all the $(a', b') \in G$ such that $c(a', b') = (ca', cb') = (ca, cb)$. We obtain the equation $ca' \equiv ca \pmod{cn}$, which simplifies to $a' \equiv a \pmod{n}$, and so we find that $a' \equiv a + kn \pmod{cn}$ for some $k \in \{0, \ldots, c-1\}$, meaning that there are c possible values for a'. Likewise, there are c possible values for b', so we find c^2 possible values for (a', b'). This accounts for all the $(cn)^2$ elements of G, and the result follows. □

In our setting, $E(\mathbf{F}_{q^k})$ is isomorphic to $\mathbf{Z}_{cn_1} \otimes \mathbf{Z}_{cn_1}$ and $E[n]$ is its subgroup isomorphic to $\mathbf{Z}_n \otimes \mathbf{Z}_n$. Since the chosen element of $E(\mathbf{F}_{q^k})$ is multiplied by cn_1/n and since $cn_1 = n(cn_1/n)$, the situation corresponds to the above proposition, and we can say that S is distributed uniformly in $E[n]$. Now, we have seen that the Weil pairing induces (by $Q \mapsto e_n(P, Q)$) an isomorphism from $E[n]/\langle P \rangle$ to $\mu_n(\mathbf{F}_{q^k})$, which means that for any $x \in \mu_n(\mathbf{F}_{q^k})$ there are exactly n points $Q \in E[n]$ such that $e_n(P, Q) = x$. We want x to be a primitive nth root of unity, in other words, a generator of $\mu_n(\mathbf{F}_{q^k})$. Since $\mu_n(\mathbf{F}_{q^k})$ has order n, it has $\varphi(n)$ generators, and so the probability that $e_n(P, S)$ is a generator is $n\varphi(n)/n^2 = \varphi(n)/n$, which if $n \geq 5$ is at least $\frac{1}{6 \log \log n}$.

3.1.3 Implementation

We implement the MOV reduction in PARI/GP [15] for supersingular elliptic curves. For ease of implementation, we work only in curves defined over $\mathbf{F}_p$ for a (large) prime p. It follows easily from the Hasse bounds that in that case, a supersingular curve must have $t = 0$, and then we have $k = 2$ and $c = 1$ or 2. Also, the following result tells us how to easily construct supersingular curves defined over $\mathbf{F}_p$ for arbitrarily large p.

Proposition 3.8. *Let $p \equiv 2 \pmod 3$ and $b \not\equiv 0 \pmod p$. Then the elliptic curve E defined over $\mathbf{F}_p$ by $Y^2 = X^3 - b$ is supersingular.*

Proof. We want to show that $E(\mathbf{F}_p)$ contains exactly $p+1$ elements. Since $p-1$ is not a multiple of 3, the group $\mathbf{F}_p^\times$ contains no subgroup of order 3, which means that the group endomorphism $\phi : x \mapsto x^3$ is injective, and is thus an automorphism (if $x^3 = 1$ and $x \neq 1$, then x generates a subgroup of order 3, which is impossible, so the kernel of ϕ is trivial). Since also $0^3 = 0$, ϕ induces a permutation of $\mathbf{F}_p$, and this means that for any $a \in \mathbf{F}_p$, the polynomial $X^3 - a$ has exactly one root in $\mathbf{F}_p$. In turn, this means that for any $y_1 \in \mathbf{F}_p$, there is exactly one point in $E(\mathbf{F}_p)$ with y-coordinate y_1: the point (x_1, y_1) where x_1 is the only root of $X^3 - (b + y_1^2)$. Adding the point at infinity, we see that $E(\mathbf{F}_p)$ has $p + 1$ points, as desired. □

The code for the MOV reduction for supersingular elliptic curves is given as Program 1. In order to compare our `elllogmov()` function to PARI's `elllog()` (which, in our setting uses Pollard's ρ algorithm), we proceed as follows:

1. Pick at random a prime p of some prescribed bit length, which is congruent to 2 modulo 3 (this ensures that our elliptic curve will be supersingular) and such that $(p+1)/6$ is prime (this ensures that we can pick a point of large prime order).
2. Pick at random an element $b \in \mathbf{F}_p^*$, and construct the curve $E : Y^2 = X^3 - b$. By Proposition 3.8, this curve is supersingular.
3. Pick at random a point $P \in E(\mathbf{F}_p)$ of (prime) order $(p + 1)/6$.
4. Pick at random an integer $\ell \pmod{(p + 1)/6}$ and compute $Q = \ell P$.

We then recover ℓ from E, P, Q with the two functions, and repeat this procedure 10 times. The results are summarised in Table 3.1 (along with those of the program for the Frey-Rück reduction, introduced in the next section). For a given bit-length n, the given value is the average running time (in seconds) for one run of the above procedure.

3.2 The Frey-Rück attack

The idea of the Frey-Rück attack is completely identical to that of the MOV attack, except that it uses the (modified) Tate pairing instead of the Weil pairing: since the Tate pairing is also nondegenerate, all the results of the previous section apply to it as well. In implementing the attack, we must note that PARI/GP's `elltatepairing()` function implements the original Tate pairing, so we must exponentiate its result in order to obtain the modified Tate pairing. The resulting program is Program 2, and as before we report its running time in Table 3.1 (along with those of the MOV program).

Bit-length of p	`elllogmov()`	`elllogfr()`	`elllog()`
40	0.8	0.7	2.1
45	4.0	4.8	11.7
50	27.0	17.4	102.3
55	65.9	71.6	301.8
60	452.4	373.5	1069.0

Table 3.1: Comparison of `elllogmov()` (Program 1), `elllogfr()` (Program 2), and PARI's `elllog()` on supersingular elliptic curves.

Program 1 PARI/GP code for the MOV reduction for supersingular elliptic curves

```
elllogmov(E, Q, P) = {
    my(p,n1,n2,n,factorn,k,c,u='u,E2,R,S,a,b,l);
    p = E.p;
    \\ The group of points of E is isomorphic to Z_n1 x Z_n2, where n2|n1
    n1 = ellgroup(E)[1];
    n2 = if(length(ellgroup(E)) > 1, ellgroup(E)[2], 1);
    n = ellorder(E, P);
    factorn = factor(n);
    \\ Determine the embedding degree k and integer c such that E(F_{q^k}) is
    \\ isomorphic to Z_{c*n1} x Z_{c*n1}
    k = 2;
    c = if(n2 == 1, 1, 2);

    u = ffgen(p^k, u);
    E2 = ellinit([E.a4, E.a6], u);
    until (fforder(a, n) == n,
        R = random(E2);
        S = ellmul(E2, R, c*n1/n);
        a = ellweilpairing(E2, P, S, n);
    );
    b = ellweilpairing(E2, Q, S, n);
    l = fflog(b, a, [n, factorn]);
    return(l%n);
};
```

3.2.1 Comparison with the MOV attack

Harasawa et al. [7] compare the MOV and Frey-Rück reductions, and conclude that the Frey-Rück reduction is always preferable. Its main advantage is that the Tate pairing requires only one execution of Miller's algorithm (versus two for the Weil pairing used in the MOV reduction), but in some special cases it may also reduce the elliptic-curve discrete logarithm to the discrete logarithm in a smaller finite field.

If E is an elliptic curve defined over a finite field $\mathbf{F}_q$ and $P \in E(\mathbf{F}_q)$ has order m, the MOV reduction reduces the discrete logarithm problem in $E(\mathbf{F}_q)$ with base P to that in $\mathbf{F}_{q^k}$, where k is the smallest integer such that $E[m] \subseteq E(\mathbf{F}_{q^k})$. On the other hand, the Frey-Rück reduction requires only that m divide $q^k - 1$. It is easily seen from Corollary 2.26 that the first condition implies the second. Moreover, Balasubramanian and Koblitz [1] have shown that the converse is true under the additional assumption that m does not divide $q - 1$. Thus, if m divides $q - 1$ but $E[m] \not\subseteq E(\mathbf{F}_q)$, the Frey-Rück reduction will reduce the elliptic curve discrete logarithm to the discrete logarithm in $\mathbf{F}_q$, whereas the MOV reduction will reduce it to the discrete logarithm in $\mathbf{F}_{q^k}$ for some $k > 1$.

3.3 Identity-based encryption

In an *identity-based cryptosystem*, a user can use any bit string as his public key; this is in contrast to systems such as RSA where a user's public key must be of a prescribed form (in RSA, it consists

Program 2 PARI/GP code for the Frey-Rück reduction for supersingular elliptic curves

```
elllogfr(E, Q, P) = {
    my(p,n1,n2,n,factorn,k,c,u='u,E2,R,S,a,b,l);
    p = E.p;
    \\ The group of points of E is isomorphic to Z_n1 x Z_n2, where n2|n1
    n1 = ellgroup(E)[1];
    n2 = if(length(ellgroup(E)) > 1, ellgroup(E)[2], 1);
    n = ellorder(E, P);
    factorn = factor(n);
    \\ Determine the embedding degree k and integer c such that E(F_{q^k}) is
    \\ isomorphic to Z_{c*n1} x Z_{c*n1}
    k = 2;
    c = if(n2 == 1, 1, 2);

    u = ffgen(p^k, u);
    E2 = ellinit([E.a4, E.a6], u);
    until (fforder(a, n) == n,
        R = random(E2);
        S = ellmul(E2, R, c*n1/n);
        a = elltatepairing(E2, P, S, n)^((p^k-1)/n);
    );
    b = elltatepairing(E2, Q, S, n)^((p^k-1)/n);
    l = fflog(b, a, [n, factorn]);
    return(l%n);
};
```

of (binary representations of) the product $n = pq$ of two primes and an encryption exponent relatively prime to $\varphi(n)$). Typically, a user will use some identifying information, such as his e-mail address, as his public key. The idea of identity-based cryptography was first formulated by Shamir in 1984 [18], and a practical identity-based encryption scheme, using the Weil pairing, was devised by Boneh and Franklin in 2001 [3].

3.3.1 Introduction

As introducied by Shamir (and later implemented by Boneh and Franklin), an identity-based encryption scheme requires a trusted third-party, whom we will name Trent and who is responsible for ensuring that only the intended recipient of a message can decrypt it (this will be made precise below), and consists of the four following functions:

- **Setup:** run by Trent once, to set up the system. Takes as input a security parameter k, and returns the (public) parameters params of the system (which include the spaces $\mathcal{M}$ of plaintexts and $\mathcal{C}$ of ciphertexts) and the (private) master key master.
- **Extract:** run by Trent whenever a user wants to use a certain identity ID as a public key. Takes as input params, master and ID, and returns a private key d, with which the user can decrypt messages encrypted with the public key ID. Of course, Trent will only give d to the user if the user is authorised to decrypt messages encrypted with ID.

- **Encrypt:** takes as input params, ID, and a message $M \in \mathcal{M}$, and returns a ciphertext $C \in \mathcal{C}$.
- **Decrypt:** takes as input params, a ciphertext $C \in \mathcal{C}$, and a private key d associated to the identity ID, and returns the plaintext M.

The main feature of identity-based cryptosystems is that when Alice wants to send a message to Bob, she does not need to obtain Bob's public key (and to verify that it is indeed Bob's public key and not an impostor's). Also, Bob need not have obtained the private key associated to the identity used by Alice to encrypt a message at the time she sent it, but may obtain one, and decrypt the message, later.

Boneh and Franklin also give further motivation: key revocation and privilege management. If Alice appends the current date to Bob's identity and uses the resulting string to encrypt a message, Bob will need to obtain the private key associated to that string in order to decrypt the message. Of course, Trent will only give Bob the associated public key if Bob is authorised to access messages sent to the given address at the given date. This makes it easy to revoke Bob's access to the address: just don't give him the private key after a certain date. Alice can further append a string of the form `level=secret` to the string she uses as a public key, and Trent will only give Bob the associated private key if Bob is authorised to access messages at the `secret` level at the given date.

3.3.2 The bilinear Diffie-Hellman problem

We first discuss the problem on which the Boneh-Franklin system is based, which is a variant of the Diffie-Hellman problem. In all the following, G_1 and G_2 are groups of prime order q. This forces G_1 and G_2 to be cyclic, and thus abelian. We are interested in certain kinds of pairings from $G_1 \times G_1$ to G_2 (viewed as $\mathbf{Z}$-modules). To ease notation, for a group G we let $G^* = G \setminus \{1\}$ (where 1 is the identity element of G).

Definition 3.9. A pairing $\hat{e} : G_1 \times G_1 \to G_2$ is called *admissible* if it is not the zero map (*i.e.*, there is some $g, h \in G_1$ such that $\hat{e}(g, h)$ is not the identity element of G_2) and there is an efficient algorithm to compute $\hat{e}(g, h)$ for any $g, h \in G_1$.

Remark 3.10. We note that since G_1 is cyclic, the fact that $\hat{e}$ is not the zero map implies that it is *not* alternating (*i.e.*, we have $\hat{e}(g, g) \neq 1$). So we will not use the Weil pairing as is, and will use a variant which we will introduce later.

The existence of such a pairing has the consequence that the Decisional Diffie-Hellman (DDH) problem in G_1 (given $g, g^a, g^b, g^c \in G_1$, tell whether $g^c = g^{ab}$) is easy: if $g^c = g^{ab}$, then

$$\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab} = \hat{e}(g^{ab}, g) = \hat{e}(g^c, g),$$

and conversely (here we use the fact that $\hat{e}$ is not alternating). However, the Computational Diffie-Hellman (CDH) problem, which is to compute g^{ab} given g, g^a, g^b, is still believed to be difficult (if the groups are chosen carefully). The Boneh-Franklin system uses a variant of the CDH problem called the Bilinear Diffie-Hellman (BDH) problem.

Definition 3.11. Let G_1, G_2 be two groups of prime order q, g be a generator of G_1, and $\hat{e} : G_1 \times G_1 \to G_2$ be an admissible pairing. The *Bilinear Diffie-Hellman Problem* for $(G_1, G_2, \hat{e})$ is to compute $\hat{e}(g, g)^{abc}$ given g, g^a, g^b, g^c.

We make the assumption that the BDH problem is hard in certain groups. Informally, this means that any probabilistic polynomial time algorithm solves the BDH problem with probability $1/2+\epsilon$, where ϵ is negligible (as a function of a security parameter k). More precisely, we assume that we have at our disposal a BDH parameter generator whose output constitutes a hard instance of the BDH problem.

Definition 3.12. A *BDH parameter generator* is a probabilistic algorithm which

1. takes as input a security parameter $k \in \mathbf{Z}_{>0}$;
2. runs in time polynomial in k; and
3. outputs a prime q, two groups G_1, G_2 of order q, and an admissible pairing $\hat{e} : G_1 \times G_1 \to G_2$.

Typically, k will be the size (in bits) of q.

It is easily seen that the BDH problem in $(G_1, G_2, \hat{e})$ can be no harder than the CDH problem in G_2: if we have an algorithm to solve the CDH problem in G_2, then we can compute g^{ab} (from g^a and g^b), and we have $\hat{e}(g^{ab}, g^c) = \hat{e}(g, g)^{abc}$. (The converse is still an open question.) We note also that if the DDH problem is hard in G_2, then the function $f_h : G_1 \to G_2$ which maps g to $\hat{e}(g, h)$ is a one-way function (if $h \in G_1$ is not the identity). Indeed, f_h is a group homomorphism by the bilinearity of $\hat{e}$, and it is non-trivial since

$$\hat{e}(h^a, h^b) = \hat{e}(h^{ab}, h) \neq 0$$

for some a, b. Thus f_h is injective, and so it is an isomorphism since the groups have the same order. If we were able to invert f_h, this would give us an isomorphism from G_2 to G_1, which would allow us to "translate" an instance of the DDH problem in G_2 to one in G_1, and we have seen before that the DDH problem in G_1 is easy.

3.3.3 The general Boneh-Franklin encryption scheme

We first describe the Boneh-Franklin encryption scheme in terms of abstract groups, before discussing its practical implementation using the Weil pairing. In addition to the parameters described above, we will also need two hash functions, which we view as random oracles (this simply means that they are "perfect" hash function: their output is uniformly distributed):

- H_1 takes as input a binary string (of any length) and outputs a generator (*i.e.*, a non-identity element) of G_1; and
- H_2 takes as input an element of G_2, and outputs a binary string of some fixed length n. n determines the length of messages: a plaintext will be a binary string of length n, and a ciphertext will be a pair (g, s), where $g \in G_1^*$ and s is a binary string of length n.

We also let k be a security parameter, and $\mathcal{G}$ be a BDH parameter generator. We can now describe the four functions:

- **Setup:** given a security parameter $k \in \mathbf{Z}_{>0}$:
  1. Run $\mathcal{G}$ on input k to obtain a prime q, two groups G_1, G_2 of order q, and an admissible pairing $\hat{e} : G_1 \times G_1 \to G_2$. Choose a random generator $g \in G_1$.
  2. Choose a random $s \mod q$, $s \neq 0$, and let $h = g^s$.
  3. Choose two hash functions H_1, H_2 as described above.

The space of messages and the space of ciphertexts are as stated above. The (public) system parameters params are $(q, G_1, G_2, \hat{e}, n, g, h, H_1, H_2)$, and the (secret) master key master is s.

- **Extract:** given an identity ID, which is a binary string of unrestricted length:
  1. Compute $h_{\mathsf{ID}} = H_1(\mathsf{ID}) \in G_1^*$.
  2. Output the private key $d_{\mathsf{ID}} = h_{\mathsf{ID}}^s$ (where s is the master key).
- **Encrypt:** given a message $M \in \mathcal{M}$ and an identity ID:
  1. Compute $h_{\mathsf{ID}} = H_1(\mathsf{ID}) \in G_1^*$.
  2. Compute $g_{\mathsf{ID}} = \hat{e}(h_{\mathsf{ID}}, h) \in G_2^*$.
  3. Choose a random $r \mod q$, $r \neq 0$.
  4. Output the ciphertext $C = (g^r, M \oplus H_2(g_{\mathsf{ID}}^r))$ (where $\oplus$ denotes bitwise exclusive or).
- **Decrypt:** given a ciphertext $C = (u, v) \in \mathcal{C}$ and an element $d \in G_1^*$, output

$$v \oplus H_2(\hat{e}(d, u)) \in \mathcal{M}.$$

Showing that the cryptosystem works as intended is easy. If $C = (u, v)$ was encrypted using the identity ID and if $d = d_{\mathsf{ID}}$, then we have

$$\hat{e}(d_{\mathsf{ID}}, u) = \hat{e}(h_{\mathsf{ID}}^s, g^r) = \hat{e}(h_{\mathsf{ID}}, g)^{rs} = \hat{e}(h_{\mathsf{ID}}, h)^r = g_{\mathsf{ID}}^r,$$

which means that the values of H_2 used during encryption and decryption are the same. Moreover, Boheh and Franklin have shown that, if the instance of the BDH problem given by $\mathcal{G}$ is hard, the cryptosystem above, which they call "BasicIdent", is semantically secure against chosen plaintext attacks (for short, they say that it has the $\mathsf{IND\text{-}ID\text{-}CPA}$ property). We will not prove this here, but we will describe this security property in some detail.

3.3.4 Semantic security against chosen plaintext attacks

In a traditional (non identity-based) public key cryptosystem, the notion of semantic security formalises the intuition that an attacker who obtains a given ciphertext should not be able to obtain any information about the plaintext. As is customary to describe cryptographic security properties, sematic security is described as a game between a *challenger* and an *attacker*, which proceeds as follows:

1. The challenger generates a set of parameters for the cryptosystem, and gives the public key to the attacker.
2. The attacker chooses two plaintexts M_0 and M_1 of equal length, and gives them to the challenger.
3. The challenger chooses at random a bit $i \in \{0, 1\}$, encrypts the message M_i, and sends the resulting ciphertext C to the attacker (this is the *challenge*).
4. The attacker chooses a bit $i' \in \{0, 1\}$, and wins if (and only if) $i' = i$.

The cryptosystem is then said to be *semantically secure* if any attacker running in polynomial time wins with probability $1/2 + \epsilon$, where ϵ is negligible as a function of the security parameter k of the cryptosystem.

Remark 3.13. It is well-known that the RSA cryptosystem in its simplest form is not semantically secure. Indeed, if we let $n = pq$ be the public modulus and e be the public encryption exponent, e must be odd since it must be relatively prime with $\varphi(n)$, which is even. Thus, any given message m will have the same Jacobi symbol (mod n) as its ciphertext m^e. This gives an attacker a simple way to win the game with probability 1: send messages M_0 and M_1 with different Jacobi symbols, and then send the bit i' such that C and $M_{i'}$ have the same Jacobi symbol. This shows that RSA must be modified in order to achieve the highest levels of security (the same is true for discrete-logarithm-based cryptosystems such as ElGamal).

Semantic security against chosen plaintext attacks informally means that an attacker who wishes to decrypt a given ciphertext C (or, even better, obtain the private key which is normally needed to decrypt it) is allowed to choose plaintext messages $M_1, M_2, \ldots, M_n$ and obtain the corresponding ciphertexts $C_1, C_2, \ldots, C_n$, but still cannot obtain any information about the plaintext M which was encrypted to produce C. Of course, security against chosen plaintext attacks is implicit in a traditional (non identity-based) public-key cryptosystem: since the parameters of the system are public, anybody can encrypt as many messages as they wish. Thus, semantic security against chosen plaintext attacks for a traditional public-key cryptosystem is just semantic security as described above, and is noted IND-CPA (where IND stands for "indistinguishability").

This definition cannot be used as is for identity-based cryptosystems, because it say nothing about how the attacker is allowed to use the function Extract. Boneh and Franklin defined semantic security against chosen plaintext attacks for identity-based cryptosystems (IND-ID-CPA) by allowing the attacker to

1. choose as many identities as he wishes, and obtain the associated private keys, both before and after the challenger sends the challenge; and

2. choose the identity used to encrypt the challenge.

More formally, the game is as follows:

1. The challenger chooses a security parameter k and runs Setup on k. He then gives the public parameters params of the system to the attacker, and keeps the master key master secret.

2. The attacker chooses identities $\mathsf{ID}_1, \ldots, \mathsf{ID}_m$, and sends them to the challenger. The challenger sends the corresponding private keys to the attacker. (The attacker may choose the identities *adaptively*, which means that he may use the knowledge of the private keys for $\mathsf{ID}_1, \ldots, \mathsf{ID}_k$ in choosing the identity ID_{k+1}.)

3. The attacker chooses an identity $\mathsf{ID} \notin \{\mathsf{ID}_1, \ldots, \mathsf{ID}_m\}$ and two plaintext messages M_0 and M_1 of equal length, and sends them to the challenger. The challenger chooses a random bit $i \in \{0, 1\}$, encrypts the message M_i with the identity ID, and sends the resulting ciphertext C to the attacker.

4. As in step 2, the attacker chooses, possibly adaptively, identities $\mathsf{ID}_{m+1}, \ldots, \mathsf{ID}_n$ such that $\mathsf{ID} \notin \{\mathsf{ID}_{m+1}, \ldots, \mathsf{ID}_n\}$, and sends them to the challenger. The challenger sends the corresponding private keys to the attacker.

5. The attacker sends a bit $i' \in \{0, 1\}$ to the challenger. The attacker wins if (and only if) $i' = i$.

As before, the cryptosystem is said to be semantically secure against chosen plaintext attacks (IND-ID-CPA) if any adversary running in polynomial time wins with probability $1/2+\epsilon$, where ϵ is negligible (as a function of k).

Theorem 3.14. *The identity-based encryption scheme of Section 3.3.3 is IND-ID-CPA secure if the hash functions H_1 and H_2 are random oracles and the BDH problem is hard in groups generated by $\mathcal{G}$.*

Proof. See [3, Theorem 4.1]. □

Remark 3.15. Boneh and Franklin also give a refinement of their system which achieves semantic security against chosen ciphertext attacks (IND-ID-CCA), but we will not discuss it here. Briefly, IND-ID-CCA means that in addition to obtaining the private keys for any identities of his choice (other than the identity used in the challenge), an attacker is also allowed to decrypt any message encrypted with any identity.

3.3.5 Implementation using the Weil pairing

We now discuss how to implement the identity-based encryption scheme of Section 3.3.3, using the Weil pairing. As we noted earlier, the Weil pairing by itself is not an admissible pairing in the sense of Definition 3.9, since it is alternating. In order to modify it and make it admissible, we use a special kind of functions called *distortion maps*. As in [8, Section 5.9.2], we restrict our attention to points of prime order, since it is sufficient for our purposes.

Definition 3.16. Let E be an elliptic curve defined over a field K and $P \in E(K)$ be of prime order $q > 2$. A map ϕ from $E(\overline{K})$ to itself is a *q-distortion map* if it has the following two properties:

1. $\phi(nP) = n\phi(P)$ for all $n > 0$. This ensures that $\phi(P)$ is q-torsion, since we have $(q+1)P = P$, and so $(q+1)\phi(P) = \phi((q+1)P) = \phi(P)$.
2. The value of the Weil pairing $e_q(P, \phi(P))$ is a primitive qth root of unity. This ensures that $\phi(P)$ has order q (we cannot have $\phi(P) = \infty$ since that would imply $e_q(P, \phi(P)) = 1$).
3. There is a polynomial time algorithm to compute ϕ.

Since q is prime the q-torsion group $E[q]$ is a vector space of dimension 2 over the field $\mathbf{Z}/q\mathbf{Z}$, and moreover, any qth root of unity which is not 1 is primitive. This makes the second condition of the above definition very easy to check.

Proposition 3.17. *Let $P, Q \in E[q]$. The following conditions are equivalent:*

1. *$\{P, Q\}$ is a basis for $E[q]$.*
2. *$P \neq \infty$ and Q is not a multiple of P.*
3. *$e_q(P, Q)$ is a primitive qth root of unity.*
4. *$e_q(P, Q) \neq 1$.*

So finally we see that $\phi(P)$ must be an element of $E[q] \setminus \langle P \rangle$. We can now construct an admissible pairing:

Definition 3.18. Let E be an elliptic curve defined over a field K, and $P \in E(K)$ be of prime order $q > 2$. Let ϕ be a q-distortion map, and let $Q, R \in \langle P \rangle$. The *modified Weil pairing on $E[q]$* (relative to ϕ) is the map

$$\hat{e}_q(Q, R) = e_q(Q, \phi(R))$$

from $\langle P \rangle \times \langle P \rangle$ to $\mu_q(\overline{K})$.

Proposition 3.19. *The modified Weil pairing defined above is an admissible pairing.*

Proof. We first note that ϕ as a map from $\langle P \rangle$ to $E[q]$ is a group homomorphism since

$$\phi(aP + bP) = \phi((a+b)P) = (a+b)\phi(P) = a\phi(P) + b\phi(P) = \phi(aP) + \phi(bP),$$

and so

$$\begin{aligned}\hat{e}_q(Q, R_1)\hat{e}_q(Q, R_2) &= e_q(Q, \phi(R_1))e_q(Q, \phi(R_2)) \\ &= e_q(Q, \phi(R_1) + \phi(R_2)) \\ &= e_q(Q, \phi(R_1 + R_2)) \\ &= \hat{e}_q(Q, R_1 + R_2).\end{aligned}$$

Since also

$$\hat{e}_q(Q_1, R)\hat{e}_q(Q_2, R) = e_q(Q_1, \phi(R))e_q(Q_2, \phi(R)) = e_q(Q_1 + Q_2, \phi(R)) = \hat{e}_q(Q_1 + Q_2, \phi(R)),$$

we see that $\hat{e}_q$ is indeed a pairing. We have also $\hat{e}_q(P, P) \neq 1$, and $\hat{e}$ can be computed efficiently since both ϕ and the Weil pairing e_q can. □

It now remains to describe a BDH parameter generator for a concrete implementation of the Boneh-Franklin cryptosystem using the (modified) Weil pairing. For a given value of the security parameter k, let q be a random k-bit prime, and let p be the smallest prime such that $p \equiv 2 \pmod 3$ and $q \mid p + 1$. We have seen before that the elliptic curve E defined over $\mathbf{F}_p$ by $Y^2 = X^3 + 1$ is supersingular, which means that it has exactly $p + 1$ points. We let $P \in E(\mathbf{F}_p)$ be a point of order q, $G_1 = \langle P \rangle$, and $G_2 = \mu_q(\mathbf{F}_{p^2})$ (we note that since q divides $p + 1$, it also divides $p^2 - 1 = (p-1)(p+1)$, so both G_1 and G_2 are cyclic of order q). The bilinear map $\hat{e} : G_1 \times G_1 \to G_2$ is the modified Weil pairing as defined above, so we finally need to describe the distortion map ϕ.

Let ζ be a primitive 3rd root of unity in $\mathbf{F}_{p^2}$, it exists because $p \equiv 2 \pmod 3$, so $p^2 \equiv 1 \pmod 3$, and 3 divides $p^2 - 1$, and moreover we have $\zeta \notin \mathbf{F}_p$ since 3 does not divide $p - 1$. Define $\phi(x, y) = (\zeta x, y)$ (and $\phi(\infty) = \infty$). It is easily seen that ϕ is an automorphism of $E(\mathbf{F}_{p^2})$, and moreover $\phi(P)$ cannot be a multiple of P since it does not have coordinates in $\mathbf{F}_p$. This shows that ϕ is a q-distortion map.

As stated before, the security of this system depends on the hardness of the BDH problem in the given groups, and we have also seen that the BDH problem is at most as difficult as the CDH problem in G_2. Since G_2 has size k^2, this means that we need to take at least $k = 512$, since the CDH problem in groups of size 1024 bits is considered sufficiently difficult. An implementation in PARI/GP [15] is given in Appendix A.

Appendix A

PARI/GP implementation of the Boneh-Franklin scheme

```
\\ Helper functions to convert between various representations of strings.

\\ ASCII code of a hex digit character in [0-9a-f] -> value in 0-15
\\ Examples:
\\ hexchar2int(49) -> 1
\\ hexchar2int(100) -> 13
hexchar2int(c) = {
    if (c >= 48 && c <= 57, return(c-48));
    if (c >= 97 && c <= 102, return(c-87));
    error("Unexpected value in hexchar2int!");
};
\\ Value in 0-15 -> hex digit in [0-9a-f]
\\ Examples:
\\ int2hexchar(2) -> "2"
\\ int2hexchar(12) -> "c"
int2hexchar(n) = {
    if (n >= 0 && n <= 9,
        return(Str(n));
    );
    if (n == 10, return("a"));
    if (n == 11, return("b"));
    if (n == 12, return("c"));
    if (n == 13, return("d"));
    if (n == 14, return("e"));
    if (n == 15, return("f"));
    error("Unexpected value in int2hexchar!");
};
\\ String of hex digits characters -> integer
\\ Reads the string as an integer in base 16 where the leftmost digit is the
\\ most significant.
\\ Examples:
\\ hex2int("10") -> 16
```

```
\\ hex2int("fe") -> 254
hex2int(s) = {
    my(ls,vecs,n,i);
    ls = length(s);
    vecs = Vecsmall(s);
    n = 0;
    for (i = 1, ls,
        n = n + hexchar2int(vecs[ls-i+1])*16^(i-1);
    );
    return(n);
};
\\ Opposite of hex2int()
\\ Examples:
\\ int2hex(123) -> "7b"
int2hex(n) = {
    my(N,s);
    N = n;
    s = "";
    while (N > 0,
        s = Str(int2hexchar(N % 16), s);
        N = N\16;
    );
    return(s);
};
\\ String of arbitrary ASCII characters -> integer
\\ Reads the string as an integer in base 256 where the leftmost digit is the
\\ most significant.
\\ Examples:
\\ ascii2int("ab") -> 24930
\\ ascii2int("hello") -> 448378203247
ascii2int(s) = {
    my(ls,vecs,n,i);
    ls = length(s);
    vecs = Vecsmall(s);
    n = 0;
    for (i = 1, ls,
        n = n + vecs[ls-i+1]*256^(i-1);
    );
    return(n);
};
\\ Opposite of ascii2int()
\\ Examples:
\\ int2ascii(448378203247) -> "hello"
int2ascii(n) = {
    my(N,s);
    N = n;
    s = "";
    while (N > 0,
        s = Str(Strchr(N % 256), s);
```

```
        N = N\256;
    );
    return(s);
};

\\ Hash functions (very rudimentary, definitely not random oracles).

\\ H1
\\ Input: * prime p
\\        * point P of prime order q on the curve Y^2 = X^3 + 1 over F_p
\\        * string ID
\\ Output: non-zero point Q in (P), "hash" of ID
\\ Basically hashes ID with SHA-512 and interprets the hash as a non-zero
\\ integer n mod q, then returns nP.
\\ Probably works best when q is 512 bits long.
H1(p,P,ID) = {
    my(E,q,s);
    E = ellinit([0,1],p);
    q = ellorder(E, P);
    s = externstr(Strprintf("/bin/echo -n '%s' | sha512sum | cut -f 1 -d ' '", ID))[1];
    return(ellmul(E, P, 1 + (hex2int(s) % (q-1))));
};
\\ H2
\\ Input: a finite field element x
\\ Output: a string of n bits (hash of u). n is the maximum possible length
\\         for a plaintext.
\\ We set n = 512 bits (64 ASCII characters, 128 hex digits), and simply return
\\ the SHA-512 hash of the PARI string representation of x. We return it as an
\\ integer, so as to directly pass it to bitxor().
H2(x) = {
    my(s);
    s = externstr(Strprintf("/bin/echo -n '%s' | sha512sum | cut -f 1 -d ' '", Str(x)))[1];
    return(hex2int(s));
};

\\ Modified Weil pairing
\\ Input: * Elliptic curve E defined over F_{p^2}
\\        * Point P on E with coordinates in F_p and of prime order
\\        * Point Q in (P), not infinity
\\        * q order of P
\\        * z primitive 3rd root of unity in F_{p^2}
\\ Output: value of the modified Weil pairing
ellmodweilpairing(E, P, Q, q, z) = {
    if (P == [0] || Q == [0], error("Infinity in ellmodweilpairing!"));
    ellweilpairing(E, P, [z*Q[1],Q[2]], q);
}

\\ Setup the system
\\ Input: positive integer k
```

```
\\ Output: [[p,q,P,Q,u,z,n],s], where
\\         * q is a random k-bit prime
\\         * p is the smallest prime which is congruent to 2 mod 3 and
\\           such that q divides p+1 and q^2 does not divide p+1
\\         * P is a random point of order q on the curve Y^2 = X^3 + 1 defined
\\           over F_p
\\         * s is a random non-zero integer mod q (THIS IS THE SECRET KEY)
\\         * Q = sP
\\         * u is a generator for a finite field F_{p^2}
\\         * z is a primitive 3rd root of unity in the finite field generated
\\           by u
\\         * n is the maximum length of a message (in bits).
\\ The array [p,q,P,Q,u,z,n] is also referred to as params.
Setup(k) = {
    my(p,q,E,P,s,Q,u='u,z,n);
    until(isprime(q), q = 2^(k-1) + random(2^(k-1)));
    p = q-1;
    until(p % 3 == 2 && (p+1) % (q^2) != 0 && isprime(p), p = p+q);
    E = ellinit([0,1],p);
    until(ellorder(E,P) == q, P = random(E));
    s = 1 + random(q-1);
    Q = ellmul(E, P, s);
    u = ffgen(p^2, u);
    until(z != 1, z = random(u)^((p^2-1)/3));
    n = 512;
    return([[p,q,P,Q,u,z,n],s]);
};

\\ Extract the private key for a certain ID
\\ Input: * t = [params,s] as returned by Setup()
\\        * ID a string
\\ Output: * d a point in (P), the private key associated to ID
Extract(t, ID) = {
    my(params,s,E,Q_ID);
    params = t[1];
    s = t[2];
    E = ellinit([0,1], params[1]);
    Q_ID = H1(params[1], params[3], ID);
    return(ellmul(E, Q_ID, s));
};

\\ Encrypt a message
\\ Input: * params as returned by Setup()
\\        * ID the identity of the recipient (string)
\\        * M the message to be encrypted (ASCII string of length at most n/8)
\\ Output: * ciphertext [C1,C2], where C1 is a non-zero point in (P) and C2 is
\\           a string of n/4 hexadecimal digits.
Encrypt(params, ID, M) = {
    my(E,E2,Q_ID,r,g_ID,C1,C2);
```

```
    if (length(M) > params[7]/8,
        error(Strprintf("Message too long! (max length: %d characters)",
                        params[7]/8));
    );
    E = ellinit([0,1], params[1]);
    E2 = ellinit([0,1], params[5]);
    Q_ID = H1(params[1], params[3], ID);
    r = 1 + random(params[2]-1);
    g_ID = ellmodweilpairing(E2, Q_ID, params[4], params[2], params[6]);
    C1 = ellmul(E, params[3], r);
    C2 = bitxor(H2(g_ID^r), ascii2int(M));
    return([C1,int2hex(C2)]);
};

\\ Decrypt a message
\\ Input: * params as returned by Setup()
\\        * C ciphertext as returned by Encrypt()
\\        * d private key as returned by Extract()
\\ Output: * message M as an ASCII string
Decrypt(params, C, d) = {
    my(E2,Mint);
    if (length(C[2]) > params[7]/4,
        error(Strprintf("Ciphertext too long! (max length: %d hex digits)",
                        params[7]/4));
    );
    E2 = ellinit([0,1], params[5]);
    Mint = bitxor(H2(ellmodweilpairing(E2, d, C[1], params[2], params[6])),
                  hex2int(C[2]));
    return(int2ascii(Mint));
};
```

Bibliography

[1] R. Balasubramanian and N. Koblitz, *The Improbability That an Elliptic Curve Has Subexponential Discrete Log Problem under the Menezes-Okamoto-Vanstone Algorithm. Journal of Cryptology*, vol. 11, pp. 141-145, Springer, New York, 1998.

[2] M. Ben-Or, *Probabilistic Algorithms in Finite Fields. 22nd Annual Symposium on Foundations of Computer Science*, 1981.

[3] D. Boneh and M. Franklin, *Identity-Based Encryption from the Weil Pairing. SIAM Journal of Computing*, vol. 32, pp. 586-615, 2003.

[4] J. Calais, *Éléments de théorie des anneaux: Anneaux commutatifs. Mathématiques à l'Université*, Ellipses, Paris, 2006.

[5] A. Enge, *Elliptic Curves and Their Applications to Cryptography: An Introduction.* Kluwer, Dordrecht, 1999.

[6] G. Frey and H.-G. Rück, *A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Mathematics of Computation*, vol. 62, pp. 865-874, 1994.

[7] R. Harasawa et al., *Comparing the MOV and FR Reductions in Elliptic Curve Cryptography.* J. Stern (Ed.), EUROCRYPT'99, *Lecture Notes in Computer Science*, vol. 1592, pp. 190-205, Springer, Berlin Heidelberg, 1999.

[8] J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography. Undergraduate Texts in Mathematics*, Springer, New York, 2008.

[9] N. Jacobson, *Basic Algebra I*, second edition. W. H. Freeman and Company, San Francisco, 1985.

[10] A. Joux, *A One Round Protocol for Tripartite Diffie-Hellman. Journal of Cryptology*, vol. 17, pp. 263-276, Springer, New York, 2004.

[11] N. Koblitz, *Elliptic Curve Cryptosystems. Mathematics of Computation*, vol. 48, pp. 203-209, 1987.

[12] A. J. Menezes, T. Okamoto, and S. A. Vanstone, *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. IEEE Transactions on Information Theory*, vol. 39, pp. 1639-1646, 1993.

[13] V. S. Miller, *Use of Elliptic Curves in Cryptography.* H. C. Williams (Ed.), *Advances in Cryptology – CRYPTO '85*, *Lecture Notes in Computer Science*, vol. 218, pp. 417-426, Springer, Berlin Heidelberg, 1986.

[14] V. S. Miller, *The Weil Pairing, and Its Efficient Calculation. Journal of Cryptology*, vol. 17, pp. 235-261, Springer, New York, 2004.

[15] PARI/GP, version `2.6.2`, Bordeaux, 2014, `http://pari.math.u-bordeaux.fr/`.

[16] R. Schoof, *Elliptic Curves over Finite Fields and the Computation of Square Roots mod p. Mathematics of Computation*, vol. 44, pp. 483-494, 1985.

[17] I. R. Shafarevich, *Basic Algebraic Geometry 1: Varieties in Projective Space*, third edition. Springer, Berlin Heidelberg, 2013.

[18] A. Shamir, *Identity-Based Cryptosystems and Signature Schemes*. G. R. Blakley and D. Cham (Eds.), *Advances in Cryptology – CRYPTO '84*, *Lecture Notes in Computer Science*, vol. 196, pp. 47-53, Springer, Berlin Heidelberg, 1985.

[19] J. H. Silverman, *The Arithmetic of Elliptic Curves*, second edition. *Graduate Texts in Mathematics*, volume 106, Springer, New York, 2009.

[20] J. H. Silverman and J. Tate, *Rational Points on Elliptic Curves. Undergraduate Texts in Mathematics*, Springer, New York, 1992. Appendix A is available online at `http://link.springer.com/content/pdf/bbm%3A978-1-4757-4252-7%2F1.pdf`.

[21] J. Tate, *WC-groups over* `p`*-adic fields. Séminaire N. Bourbaki*, 1956-1958, p. 265-277, 1957. Available online at `http://archive.numdam.org/article/SB_1956-1958__4__265_0.pdf`.

[22] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*, second edition. *Discrete Mathematics and its Applications*, CRC Press, Boca Raton, 2008.

[23] A. Weil, *Sur les fonctions algébriques à corps de constantes fini. Les Comptes Rendus de l'Académie des sciences*, vol. 210, pp. 592-594, 1940. Available online at `http://gallica.bnf.fr/ark:/12148/bpt6k31623/f592.image.langFR`.