Pairings for beginners

by

Craig Costello

Contents

Symbols and abbreviations

(f)	divisor of the function f
$[n]P$	scalar multiplication (exponentiation) of P by $n \in \mathbb{Z}$
$\#E$	number of points on E
$\mathbb{A}^n(K)$	affine n-space over the field K
$\epsilon(D)$	effective part of the divisor D
η_T	eta (T) pairing
$\mathbb{F}_q$	finite field with q elements
$\mathbb{F}_{q^k}$	full extension field
$\mathbb{G}_1$	base field subgroup: $E[r] \cap \ker(\pi - [1])$ (in Type 3 pairing)
$\mathbb{G}_2$	trace-zero subgroup: $E[r] \cap \ker(\pi - [q])$ (in Type 3 pairing)
$\mathbb{G}_T$	order r subgroup of $\mathbb{F}_{q^k}^*$ (commonly the r-th roots of unity μ_r)
$\mathcal{O}$	point at infinity on an elliptic curve E
$\overline{K}$	algebraic closure of the field K
$\mathbb{P}^n(K)$	projective n-space over the field K
ϕ	occurs as the distortion map on supersingular curves and as the GLV endomorphism
Φ_i	i-th cyclotomic polynomial
π	q-power Frobenius endomorphism: $(x, y) \mapsto (x^q, y^q)$

Ψ	the (un)twisting isomorphism
ψ	occurs as both the isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ and as the GLS isomorphism
$\psi_\ell(x)$	ℓ-th division polynomial on E (for odd ℓ)
ρ	ratio between base field size and subgroup size for a pairing-friendly curve
a_T	ate pairing
C	an arbitrary curve
C_g	(imaginary quadratic) hyperelliptic curve of genus g
D	occurs as both a divisor on E and the CM discriminant of E
d	degree of twist
D_P	divisor $(P) - (\mathcal{O})$
D_Q	divisor $(Q) - (\mathcal{O})$
E	an elliptic curve
e	a general pairing
E'	twisted curve (defined over $\mathbb{F}_{q^{k/d}}$)
$E(K)$	set of K-rational points on E
$e(P, Q)$	pairing of P and Q (the paired value)
E/K	elliptic curve defined over K
$E[r]$	the (entire) r-torsion
$f_{m,P}$	function with divisor $(f_{m,P}) = m(P) - ([m]P) - (m-1)(\mathcal{O})$
g	genus of a curve
K	arbitrary field
k	embedding degree of E (with respect to q and r)

n_P	multiplicity of point P in associated divisor
P	generator of $\mathbb{G}_1$
Q	generator of $\mathbb{G}_2$
r	order of the large prime subgroup in $E(\mathbb{F}_q)$
T	ate pairing loop parameter ($T = t - 1$)
t	trace of Frobenius
$T_r(P, Q)$	order r reduced Tate pairing
$t_r(P, Q)$	order r Tate pairing
$w_r(P, Q)$	order r Weil pairing
aTr	anti-trace map
BKLS − GHS	Barreto-Kim-Lynn-Scott/Galbraith-Harrison-Soldera algorithm
BLS	Barreto-Lynn-Scott families
BN	Barreto-Naehrig family with $k = 12$
CM	complex multiplication
Deg(D)	degree of the divisor D
$\mathrm{Div}^0(E)$	group of degree zero divisors on E
$\mathrm{Div}_{\overline{\mathbb{F}}_q}(E)$	group of divisors on $E/\mathbb{F}_q$
DLP	discrete logarithm problem
ECC	elliptic curve cryptography
ECDLP	elliptic curve discrete logarithm problem
End(E)	endomorphism ring of E
Gal(L/K)	Galois group of L over K
GLS	Galbraith-Lin-Scott method

GLV	Gallant-Lambert-Vanstone method
HECC	hyperelliptic curve cryptography
KSS	Kachisa-Schaefer-Scott families
MNT	Miyaji-Nakabayashi-Takano (construction/criteria)
NIST	National Institute of Standards and Technology
NSS	not supersingular curves
$\mathrm{ord}_P(f)$	the multiplicity of f at P on E
PBC	pairing-based cryptography
$\mathrm{Pic}^0(E)$	Picard group of E
$\mathrm{Prin}(E)$	group of principal divisors on E
$\mathrm{QR}(q)$	set of quadratic residues modulo q
$\mathrm{supp}(D)$	support of the divisor D
Tr	trace map

Chapter 1

Introduction

Aficionados of cryptographic pairing computation are often asked by interested newcomers to point towards literature that is a good starting point. My answer usually differs depending on the mathematical background volunteered from the "pairing beginner", but almost always involves accordingly picking a subset of the following excellent references.

- Galbraith's chapter [Gal05] is a stand-out survey of the field (up until 2005). It provides several theorems and proofs fundamental to pairing-based cryptography and gives some useful toy examples that illustrate key concepts.

- Lynn's thesis [Lyn07] is also a great survey of the entire arena of pairing computation (up until 2007), and gives all the details surrounding the pioneering papers he co-authored [BKLS02, BLS02, BLS03, BLS04], which are themselves good starting points.

- The first chapter of Naehrig's thesis [Nae09, Ch. 1] conveniently presents the necessary algebro-geometric results required to be able to read most of the literature concerning pairing computation.

- Scott's webpage [Sco04] gives a short and very friendly introduction to the basics of the groups involved in pairing computations by means of an illustrative toy example.

- In his new chapter entitled Algorithmic Aspects of Elliptic Curves, Silverman's second edition [Sil09, Ch. XI.7] includes a concise introduction to pairing-based cryptography that also points to foundational results found elsewhere in his book.

In addition, digging up talks from some of the big players in the field is usually (but not always!) a good way to avoid getting bogged down by minor technical details that slow one's progress in grasping the main ideas. In particular, we refer to the nice talks by Scott [Sco07a, Sco07b] and Vercauteren [Ver06b, Ver06a].

In any case, correctly prescribing the best reading route for a beginner naturally requires individual diagnosis that depends on their prior knowledge and technical preparation. A student who is interested in learning pairings, but who has never seen or played with an elliptic curve, may quickly become overwhelmed if directed to dive straight into the chapters of Silverman's book or Naehrig's thesis. This is not due to lack of clarity, or to lack of illuminating examples (both chapters are ample in both), but perhaps more because of the vast amount of technical jargon that is necessary for one to write a complete and self-contained description of cryptographic pairings. On the other hand, an informal, example-driven approach to learning the broad field of pairing computation may ease the beginner's digestion in the initial stages. For instance, a novice would be likely to find it more beneficial to first see the simple toy example of the quadratic twisting isomorphism in action on Scott's webpage [Sco04], before heading to Silverman's book [Sil09, Ch. X.5.4] to see all possible twisting isomorphisms formally defined, and then later returning to his earlier chapters (specifically Ch. II.2) to read about maps between curves in full generality.

In this light we discuss the major aim of this text. We intend to let illustrative examples drive the discussion and present the key concepts of pairing computation with as little machinery as possible. For those that are fresh to pairing-based cryptography, it is our hope that this chapter might be particularly useful as a first read and prelude to more complete or advanced expositions (e.g. the related chapters in [Gal12]).

On the other hand, we also hope our beginner-friendly intentions do not leave any sophisticated readers dissatisfied by a lack of formality or generality, so in cases where our discussion does sacrifice completeness, we will at least endeavour to point to where a more thorough exposition can be found.

One advantage of writing a survey on pairing computation in 2012 is that, after more than a decade of intense and fast-paced research by mathematicians and cryptographers around the globe, the field is now racing towards full maturity. Therefore, an understanding of this text will equip the reader with most of what they need to know in order to tackle any of the vast literature in this remarkable field, at least for a while yet. Anyone who understands our examples will also comfortably absorb the basic language of algebraic geometry in the context of curve-based cryptography. Since we are aiming the discussion at active readers, we have matched every example with a corresponding snippet of (hyperlinked) Magma [BCP97] code[1], where we take inspiration from the helpful Magma pairing tutorial by Dominguez Perez *et al.* [DKS09]. In the later chapters we build towards a full working pairing code that encompasses most of the high-level optimisations; this culminates to finish the chapter in Example 7.5.1.

The text is organised as follows. We start in Chapter 2 by giving an overview of elliptic curve cryptography (ECC). Indeed, elliptic curves are the main object on which cryptographic pairings take place, so this first chapter forms a basis for the entire text. In Chapter 3 we introduce the important concept of divisors, as well as other essential theory from algebraic geometry that is needed to properly understand cryptographic pairings. In Chapter 4 we detail the specific elliptic curve groups that are employed in a cryptographic pairing, before presenting Miller's algorithm to compute the Weil and Tate pairings in Chapter 5. In Chapter 6 we introduce the notion of *pairing-friendly curves* and give a brief survey of the most successful methods of constructing them. In Chapter 7 we bring the reader up to speed with the landmark achievements and improvements that have boosted pairing computation to the point it is today.

[1]If one does not have access to Magma, the scripts we provide can be run at the online Magma calculator: `http://magma.maths.usyd.edu.au/calc/`

Chapter 2

Elliptic curves as cryptographic groups

The purpose of this chapter is to introduce elliptic curves as they are used in cryptography. Put simply, an elliptic curve is an abstract type of *group*.

Perhaps a newcomer will find this abstractness apparent immediately when we insist that to understand elliptic curve groups in cryptography, the reader should be familiar with the basics of *finite fields* $\mathbb{F}_q$. This is because, more generally, elliptic curves are groups which are defined on top of (over) fields. Even though elliptic curve groups permit only one binary operation (the so called *group law*), the operation itself is computed within the *underlying field*, which by definition permits two operations (and their inverses). For a general field K, the group elements of an elliptic curve E are *points* whose (x, y) coordinates come from $\overline{K}$ (the algebraic closure of K), and which satisfy the (affine) curve equation for E, given as

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

where $a_1, ..., a_6 \in \overline{K}$. Equation (2.1) is called the *general Weierstrass equation* for elliptic curves. Aside from all the $(x, y) \in \overline{K}$ solutions to the equation above, there is one extra point which can not be defined using the affine equation, but which must be included to complete the group definition. This point is called the *point at infinity*, which we denote by $\mathcal{O}$, and we will define it properly in a

moment.

If $a_1, ..., a_6 \in K$, then we say E is *defined over* K, and write this as E/K (the same goes for any extension field L of K). Before we go any further, we make a convenient simplification of the general Weierstrass equation. If the field characteristic is not 2 or 3, then divisions by 2 and 3 in K permit the substitutions $y \mapsto (y - a_1x - a_3)/2$ to give $E : y^2 = 4x^3 + b_2x^2 + 2b_4x + b_6$, and then $(x, y) \mapsto \left(\frac{x-3b_2}{36}, \frac{y}{108}\right)$, which (upon appropriate rescaling) yields the following simplified equation.

$$E : y^2 = x^3 + ax + b. \tag{2.2}$$

Equation (2.2) is called the *short Weierstrass equation* for elliptic curves, and will be used all the way through this text. Namely, we will always be working over large prime fields, where the short Weierstrass equation covers all possible isomorphism classes of elliptic curves, so the curves we use will always be an instance of (2.2).

Example 2.0.1 (Magma script). $E/\mathbb{Q} : y^2 = x^3 - 2$ is an elliptic curve. Along with the point at infinity $\mathcal{O}$ (which we are still yet to define), the set of points over $\mathbb{Q}$ is written as $E(\mathbb{Q})$, and is defined as $E(\mathbb{Q}) = \{(x, y) \in \mathbb{A}^2(\mathbb{Q}) : y^2 = x^3 - 2\} \cup \{\mathcal{O}\}$. The point $P = (x_P, y_P) = (3, 5)$ lies in $E(\mathbb{Q})$, as do $Q = (x_Q, y_Q) = \left(\frac{129}{100}, \frac{-383}{1000}\right)$ and $R = (x_R, y_R) = \left(\frac{164323}{29241}, \frac{-66234835}{5000211}\right)$, so we can write $P, Q, R \in E(\mathbb{Q})$. We usually write E to represent the group of points over the full algebraic closure, so for example, the point $S = (x_S, y_S) = \left(0, \sqrt{-2}\right) \in E = E(\overline{\mathbb{Q}})$, but $S \notin E(\mathbb{Q})$. Soon we will be defining the binary group operation $\oplus$ on E using rational formulas in the underlying field, so an active reader can return to this example with these formulas to verify that $R = P \oplus Q$, where x_R, y_R are computed from x_P, y_P, x_Q, y_Q using additions and multiplications (also subtractions and inversions) in $\mathbb{Q}$. Furthermore, it can also be verified that $Q = P \oplus P$, so that $R = P \oplus P \oplus P$; we usually write these as $Q = [2]P$ and $R = [3]P$, where $\underbrace{P \oplus P \cdots \oplus P}_{n} = [n]P$ in general. To finish this example, we remark that if $(x', y') \in E$, then $(x', -y') \in E$ (but is not distinct if $y' = 0$), which is true for any elliptic curve in short Weierstrass form.

Example 2.0.2 (Magma script). $E/\mathbb{F}_{11} : y^2 = x^3 + 4x + 3$ is an elliptic curve. $E(\mathbb{F}_{11})$ has 14 points: $(0, 5)$, $(0, 6)$, $(3, 3)$, $(3, 8)$, $(5, 4)$,$(5, 7)$, $(6, 1)$, $(6, 10)$, $(7, 0)$, $(9, 3)$, $(9, 8)$, $(10, 3)$, $(10, 8)$, not forgetting the point at infinity $\mathcal{O}$. Notice that all

but two points come in pairs (x', y') and $(x', -y')$, the exceptions being $(x', y') = (7, 0)$ (since $y' = -y' = 0$) and $\mathcal{O}$. If we form the quadratic extension $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$ with $i^2 + 1 = 0$, then considering E over $\mathbb{F}_{q^2}$ will allow many more solutions, and give many more points: namely, $\#E(\mathbb{F}_{q^2}) = 140$. In addition to the points in $E(\mathbb{F}_q)$, $E(\mathbb{F}_{q^2})$ will also contain those points with x-coordinates in $\mathbb{F}_q$ that did not give x^3+4x+3 as a quadratic residue in $\mathbb{F}_q$ (but necessarily do in $\mathbb{F}_{q^2}$), and many more with both coordinates in $\mathbb{F}_{q^2} \setminus \mathbb{F}_q$. Examples of both such points are $(2, 5i)$ and $(2i + 10, 7i + 2)$ respectively. It is not a coincidence that $\#E(\mathbb{F}_q) \mid \#E(\mathbb{F}_{q^2})$, since $E(\mathbb{F}_q)$ is a subgroup of $E(\mathbb{F}_{q^2})$.

Not every tuple $(a, b) \in K \times K$ gives rise to the curve given by $f(x, y) = y^2 - (x^3 + ax + b) = 0$ being an elliptic curve. If there exists $P = (x_P, y_P)$ on f such that both partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ vanish simultaneously at P, then P is called a *singular* point and f is also deemed singular. Conversely, if no such point exists, f is called *non-singular*, or *smooth*, and is then an elliptic curve. It is easy enough to show that a singularity occurs if and only if $4a^3 + 27b^2 = 0$ (see [Sil09, Ch. III.1, Prop. 1.4]), so as long as $4a^3 + 27b^2 \neq 0$ in K, then $E/K : y^2 = x^3 + ax + b$ is an elliptic curve.

In cryptography we only ever instantiate elliptic curves defined over finite fields, but it is often conceptually helpful to view graphs of elliptic curves over $\mathbb{R}$. We illustrate the difference between singular and non-singular (smooth) elliptic curves in Figures 2.1-2.4.

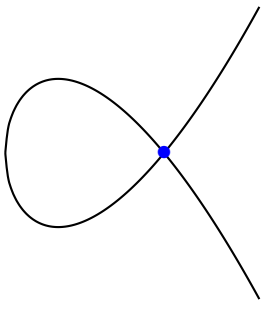


Figure 2.1: Singular curve $y^2 = x^3 - 3x + 2$ over $\mathbb{R}$.

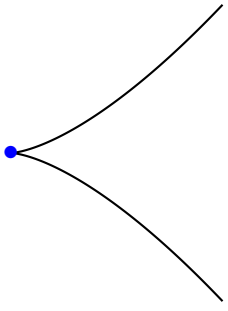


Figure 2.2: Singular curve $y^2 = x^3$ over $\mathbb{R}$.

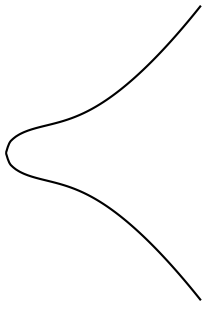


Figure 2.3: Smooth curve $y^2 = x^3 + x + 1$ over $\mathbb{R}$.

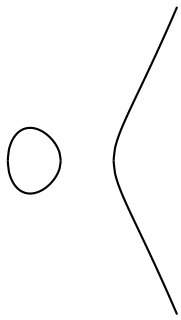


Figure 2.4: Smooth curve $y^2 = x^3 - x$ over $\mathbb{R}$.

2.1 The group law: the chord-and-tangent rule

We now turn to describing the elliptic curve group law, and it is here that viewing pictures of elliptic curves over $\mathbb{R}$ is especially instructive. We start with a less formal description until we define the role of the point at infinity $\mathcal{O}$. The group law exploits the fact that, over any field, a line (a degree one equation in x and y) intersects a cubic curve (a degree three equation in x and y) in three places (this is a special case of a more general theorem due to Bezout [Har77, I.7.8]). Namely, if we run a line $\ell : y = \lambda x + \nu$ between two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ on E, then substituting this line into $E : y^2 = x^3 + ax + b$ will give a cubic polynomial in x, the roots of which are the x-coordinates of the three points of intersection between ℓ and E. Knowing the two roots (x_P and x_Q) allows us to determine a unique third root that corresponds to the third and only other point in the affine intersection $\ell \cap E$, which we denote by $\ominus R$ (the reason will become clear in a moment). The point $\ominus R$ is then "flipped" over the x-axis to the point R. In general, the elliptic curve composition law $\oplus$ is defined by this process, namely $R = P \oplus Q$. When computing $R = P \oplus P$, the line ℓ is computed as the tangent to E at P. That is, the derivatives of ℓ and E are matched at P, so (counting multiplicities) ℓ intersects E "twice" at P. Figures 2.5 and 2.6 illustrate why this process is aptly named the *chord-and-tangent* rule.

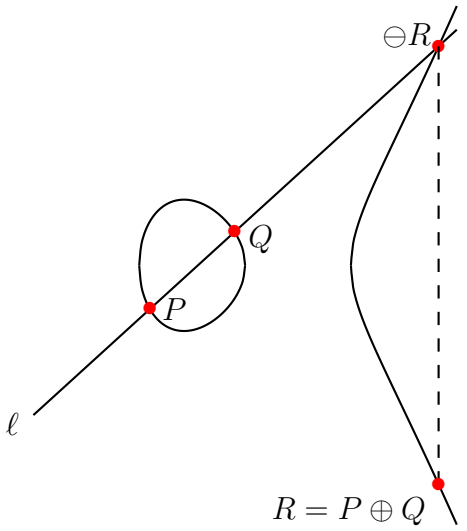


Figure 2.5: Elliptic curve addition.

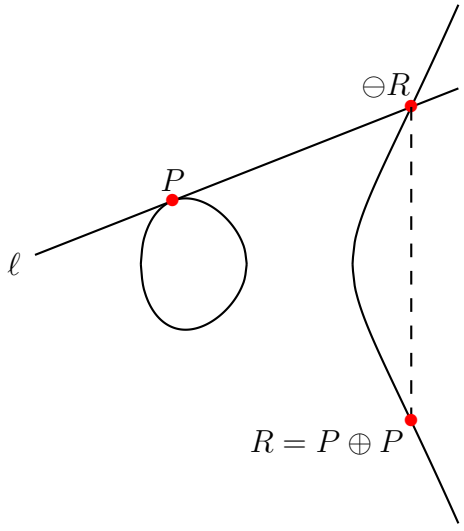


Figure 2.6: Elliptic curve doubling.

Having loosely defined the general group operation, we can now (also loosely)

define the role of the point at infinity $\mathcal{O}$. To try and place it somewhere in the above diagrams, one can think of $\mathcal{O}$ as being a point that simultaneously sits infinitely high and infinitely low in the y direction. This allows us to informally conceptualise two properties of elliptic curve groups: firstly, that the point at infinity $\mathcal{O}$ plays the role of the *identity* of the group; and secondly, that the unique inverse of a point is its reflected image over the x-axis (e.g. the $\ominus R$'s in Figures 2.5 and 2.6 are the respective inverses of the R's, and vice versa). If we apply the process in the previous paragraph to compute $R \oplus (\ominus R)$, we start by finding the vertical line that connects them (the dashed lines in Figures 2.5 and 2.6). This line also intersects E (twice) at the point at infinity $\mathcal{O}$, which is then reflected back onto itself, giving $R \oplus (\ominus R) = \mathcal{O}$. Thus, if we define the identity of the group to be $\mathcal{O}$, then the inverse of any element $R = (x_R, y_R)$ is taken as $\ominus R = (x_R, -y_R)$.

Example 2.1.1 (Magma script). $E/\mathbb{R} : y^2 = x^3 - 2x$ is an elliptic curve. The points $(-1,-1)$, $(0,0)$ and $(2,2)$ are all on E, and are also on the line $\ell : y = x$. Applying the technique described above to compute some example group law operations via the line ℓ, we have $(-1,-1) \oplus (0,0) = (2,-2)$, $(2,2) \oplus (0,0) = (-1,1)$, and $(-1,-1) \oplus (2,2) = (0,0)$. All but four points come in pairs with their inverse (i.e. (x', y') and $(x', -y')$); the exceptions being $(0,0)$, $(\sqrt{2}, 0)$, $(-\sqrt{2}, 0)$ (notice the vertical tangents when $y = 0$ in these cases), and $\mathcal{O}$, which are all their own inverse, e.g. $(0,0) = \ominus(0,0)$, so $(0,0) \oplus (0,0) = \mathcal{O}$ on E. The tangent line ℓ' to E at $(-1,-1)$ is $\ell' : y = -\frac{1}{2}x - \frac{3}{2}$, and it intersects E once more at $(\frac{9}{4}, -\frac{21}{8})$, which gives $(-1,-1) \oplus (-1,-1) = [2](-1,-1) = (\frac{9}{4}, \frac{21}{8})$.

Example 2.1.2 (Magma script). In this example we consider the same curve equation as the last example, but this time over a small finite field, namely $E/\mathbb{F}_{11} : y^2 = x^3 - 2x$. Rational points are injected naturally across to the finite field case (as long as there is no conflict with the characteristic), so we can immediately find the points $(0,0)$, $(2,2)$ and $(-1,-1) = (10,10)$ (and their inverses) in Figure 2.9. In this case, consider performing the group law operation between the (blue) points $(5,7)$ and $(8,10)$. The line ℓ that joins them is $y = x + 2$, which intersects E once more at $(10,1)$. Negating the y-coordinate finds the other point on the dashed line, and gives $(5,7) \oplus (8,10) = (10,10)$.

Example 2.1.2 is also intended to justify why, although (in cryptography) we only ever use elliptic curves over finite fields, we often opt to illustrate the group law by drawing the continuous pictures of curves over $\mathbb{R}$.

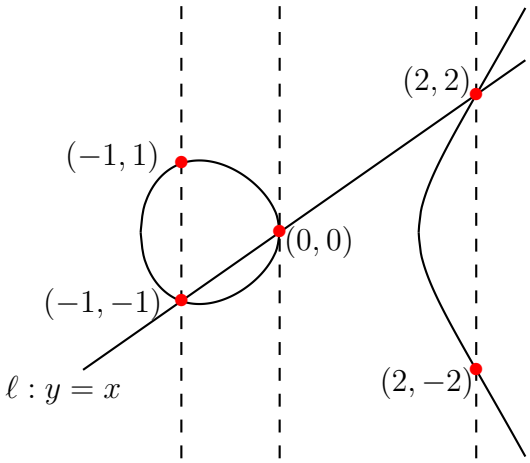


Figure 2.7: Addition in $\mathbb{R}$.

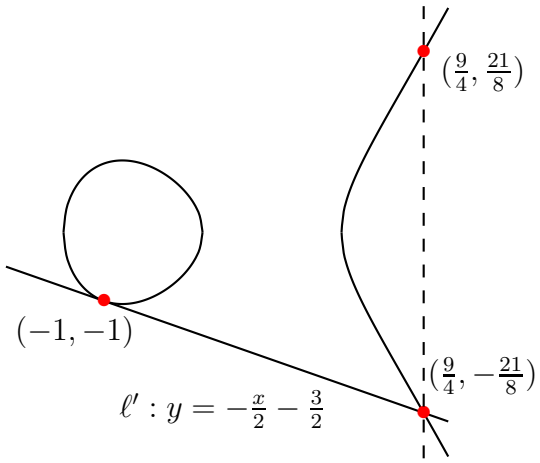


Figure 2.8: Doubling in $\mathbb{R}$.

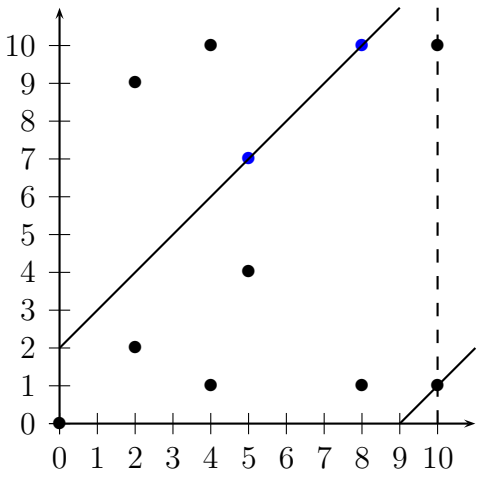
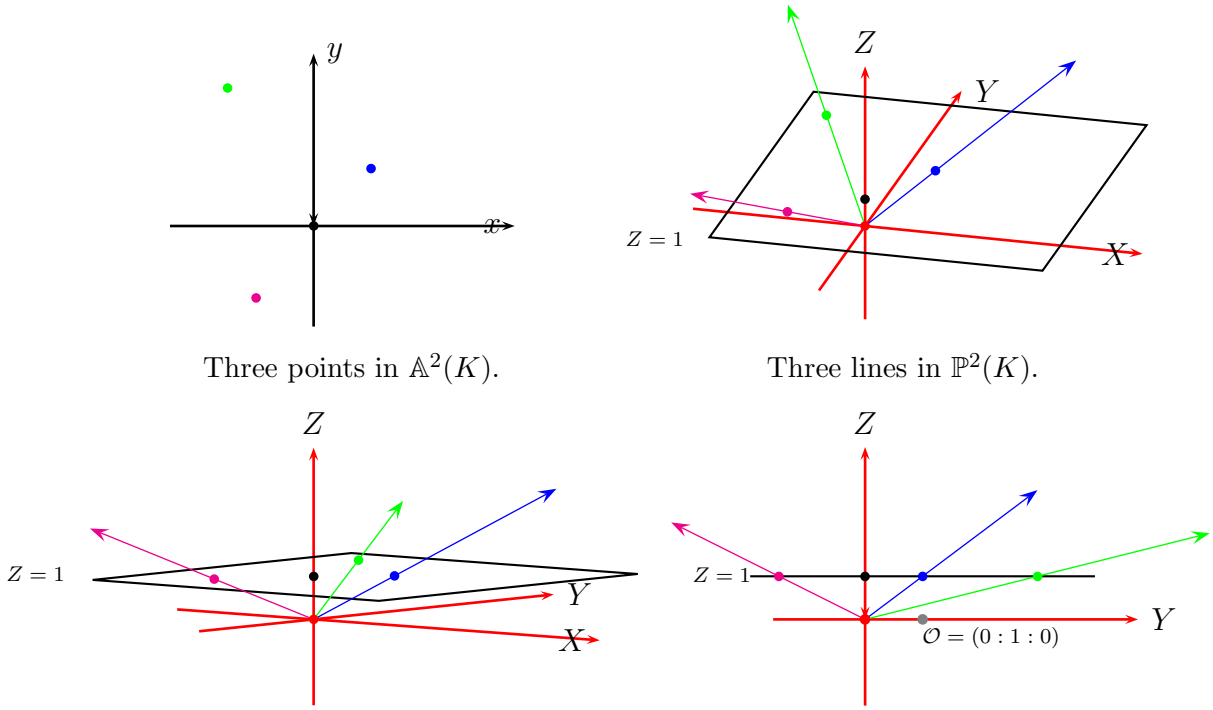


Figure 2.9: The points (excluding $\mathcal{O}$) on $E(\mathbb{F}_{11})$.

2.1.1 The point at infinity in projective space

We now focus our attention on giving a more formal definition for the point at infinity. So far we have been describing elliptic curves in *affine space* as a set of affine points together with the point at infinity: $E = \{(x, y) \in \mathbb{A}^2(\overline{K}) : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$. In general, a more precise way to unify (or include) points at infinity with the affine points is to work in *projective space*: essentially, instead of working with points in n-space, we work with lines that pass through the origin in $(n+1)$-space. For our purposes, this means our affine points in 2-space become lines in 3-space, namely that $(x, y) \in \mathbb{A}^2(\overline{K})$ corresponds to the line defined by all points of the form $(\lambda x, \lambda y, \lambda) \in \mathbb{P}^2(\overline{K})$, where $\lambda \in \overline{K}^*$. That is, $\mathbb{P}^2$ is $\mathbb{A}^3 \setminus$

$\{(0,0,0)\}$ modulo the following congruence condition: $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$ if there exists $\lambda \in \overline{K}^*$ such that $(x_1, y_1, z_1) = (\lambda x_2, \lambda y_2, \lambda z_2)$. Figure 2.10 illustrates the relationship between points in $\mathbb{A}^2$ with their congruence classes (lines) in $\mathbb{P}^2$; the lines in 3-space should also extend "downwards" into the region where $Z < 0$ but we omitted this to give more simple pictures. We reiterate that these lines do not include the point $(0, 0, 0)$.



Three points in $\mathbb{A}^2(K)$. Three lines in $\mathbb{P}^2(K)$.

Three lines in $\mathbb{P}^2(K)$. Three lines in $\mathbb{P}^2(K)$.

Figure 2.10: Identifying points in $\mathbb{A}^2$ with lines in $\mathbb{P}^2$

We usually use capital letters and colons to denote a (representative of a) congruence class in projective coordinates, so that in general $(X : Y : Z)$ represents the set of all points on the "line" in $\mathbb{P}^2$ that correspond to $(x, y) \in \mathbb{A}^2$. There are many copies of $\mathbb{A}^2$ in $\mathbb{P}^2$, but we traditionally map the affine point $(x, y) \in \mathbb{A}^2$ to projective space via the trivial inclusion $(x, y) \mapsto (x : y : 1)$, and for any $(X : Y : Z) \neq \mathcal{O} \in \mathbb{P}^2$, we map back to $\mathbb{A}^2$ via $(X : Y : Z) \mapsto (X/Z, Y/Z)$. The point at infinity $\mathcal{O}$ is represented by $(0 : 1 : 0)$ in projective space (see the last diagram in Figure 2.10), for which we immediately note that the map back to $\mathbb{A}^2$ is ill-defined.

Example 2.1.3 (Magma script). $E/\mathbb{R} : y^2 = x^3 + 3x$ is an elliptic curve. $P =$

$(3,6) \in \mathbb{A}^2(\overline{\mathbb{R}})$ is a point on E. In projective space, P becomes $P = (3 : 6 : 1) \in \mathbb{P}^2(\overline{\mathbb{R}})$, which represents all points in $(3\lambda, 6\lambda, \lambda)$ for $\lambda \in \overline{\mathbb{R}} \setminus \{0\}$. For example, the points $(12, 24, 4)$, $(-3\sqrt{-1}, -6\sqrt{-1}, -1\sqrt{-1})$, $(3\sqrt{2}, 6\sqrt{2}, \sqrt{2})$ in $\mathbb{A}^3(\overline{\mathbb{R}})$ are all equivalent (modulo the congruence condition) in $\mathbb{P}^2(\overline{\mathbb{R}})$, where they are represented by P. As usual, the point at infinity on E is $\mathcal{O} = (0 : 1 : 0)$.

The way we define the collection of points in projective space is to *homogenise* $E : y^2 = x^3 + ax + b$ by making the substitution $x = X/Z$ and $y = Y/Z$, and multiplying by Z^3 to clear the denominators, which gives

$$E_{\mathbb{P}} : Y^2Z = X^3 + aXZ^2 + bZ^3. \tag{2.3}$$

The set of points (X, Y, Z) with coordinates in $\overline{K}$ that satisfies (2.3) is called the *projective closure* of E. Notice that $(0, \lambda, 0)$ is in the projective closure for all $\lambda \in \overline{K}^*$, and that all such points cannot be mapped into $\mathbb{A}^2$, justifying the representative of point at infinity being $\mathcal{O} = (0 : 1 : 0)$.

Example 2.1.4 (Magma script). Consider $E/\mathbb{F}_{13} : y^2 = x^3 + 5$. There are 15 affine points $(x, y) \in \mathbb{A}^2(\mathbb{F}_{13})$ on E, which (with the point at infinity $\mathcal{O}$) gives $\#E(\mathbb{F}_{13}) = 16$. On the other hand, if we homogenise (or projectify) E to give $E_{\mathbb{P}}/\mathbb{F}_{13} : Y^2Z = X^3 + 5Z^3$, then there are 16 classes $(X : Y : Z) \in \mathbb{P}^2(\mathbb{F}_{13})$: $(0 : 1 : 0)$, $(2 : 0 : 1)$, $(4 : 2 : 1)$, $(4 : 11 : 1)$, $(5 : 0 : 1)$, $(6 : 0 : 1)$, $(7 : 6 : 1)$, $(7 : 7 : 1)$, $(8 : 6 : 1)$, $(8 : 7 : 1)$, $(10 : 2 : 1)$, $(10 : 11 : 1)$, $(11 : 6 : 1)$, $(11 : 7 : 1)$, $(12 : 2 : 1)$, $(12 : 11 : 1)$. Each of these classes represents several points $(X, Y, Z) \in \mathbb{A}^3(\mathbb{F}_{13})$ whose coordinates satisfy $Y^2Z = X^3 + 5Z^3$ (there are actually 195 such points, but this is not important). In fact, each class represents infinitely many points on $E_{\mathbb{P}}(\overline{\mathbb{F}}_{13})$. Any reader that is familiar with Magma, or has been working through our examples with the accompanying Magma scripts, will recognise the representation of points as representatives in $\mathbb{P}^2$.

The projective coordinates (X, Y, Z) used to replace the affine coordinates (x, y) above are called *homogenous projective coordinates*, because the projective version of the curve equation in (2.3) is homogeneous. These substitutions ($x = X/Z$, $y = Y/Z$) are the most simple (and standard) way to obtain projective coordinates, but we are not restricted to this choice of substitution. For example, many papers in ECC have explored more general substitutions of the form $x = X/Z^i$ and $y = Y/Z^j$ on various elliptic curves [BL07a].

Example 2.1.5 (Magma script). Consider $E/\mathbb{F}_{41} : y^2 = x^3 + 4x - 1$. Using

homogeneous coordinates gives rise to the projective equation $Y^2Z = X^3 + 4XZ^2 - Z^3$, with the point at infinity being $\mathcal{O} = (0 : 1 : 0)$. An alternative projection we can use is $x = X/Z$ and $y = Y/Z^2$, which in this instance give the projective equation $Y^2 = X^3Z + 4XZ^3 - Z^4$, from which the point at infinity is seen (from putting $Z = 0$) to be $\mathcal{O} = (1 : 0 : 0)$. Another commonly used coordinate system is Jacobian coordinates, which use the substitutions $x = X/Z^2$ and $y = Y/Z^3$ to give the projective equation $Y^2 = X^3 + 4XZ^4 - Z^6$. In this case, we substitute $Z = 0$ to see that the point at infinity is defined by the line $\mathcal{O} = (\lambda^2 : \lambda^3 : 0) \in \mathbb{P}^2(\mathbb{F}_{41})$.

2.1.2 Deriving explicit formulas for group law computations

We are now in a position to give explicit formulas for computing the elliptic curve group law. The chord-and-tangent process that is summarised in Figures 2.5 and 2.6 allows a simple derivation of these formulas. We derive the formulas in affine space, but will soon transfer them into projective space as well. The derivation of the formulas for point additions $R = P \oplus Q$ and for point doublings $R = P \oplus P$ follow the same recipe, the main difference being in the calculation of the gradient λ of the line $\ell : y = \lambda x + \nu$ that is used. We will first derive the formulas for the addition $R = P \oplus Q$ in the general case, and will then make appropriate changes for the general doubling formulas. By "general case", we mean group law operations between points where neither point is $\mathcal{O}$, and the points that are being added are not each inverses of one another; we will handle these special cases immediately after the general cases. Referring back to Figure 2.5, the line $\ell : y = \lambda x + \nu$ that intersects $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ has gradient $\lambda = (y_Q - y_P)/(x_Q - x_P)$. From here, ν can simply be calculated as either $\nu = y_P - \lambda x_P$ or $\nu = y_Q - \lambda x_Q$, but in the literature we will often see an unbiased average of the two as $\nu = (y_Q x_P - y_P x_Q)/(x_P - x_Q)$. From here we substitute $\ell : y = \lambda x + \nu$ into $E : y^2 = x^3 + ax + b$ to find the third affine point of intersection, $\ominus R$, in $\ell \cap E$. Finding the coordinates of $\ominus R$ trivially reveals the coordinates of $R = (x_R, y_R)$, since $\ominus R = (x_R, -y_R)$; the roots of the cubic that

result will be x_P, x_Q and x_R. Namely,

$$\begin{aligned}(x - x_P)(x - x_Q)(x - x_R) &= (x^3 + ax + b) - (\lambda x + \nu)^2 \\ &= x^3 - \lambda^2 x^2 + (a - 2\lambda\nu)x + b - \nu^2.\end{aligned}$$

We only need to look at the coefficient of x^2 to determine x_R, since the coefficient on the left hand side is $-(x_P + x_Q + x_R)$. From here, recovering the y-coordinate is simple, since $-y_R$ lies on ℓ, so

$$x_R = \lambda^2 - x_P - x_Q; \qquad\qquad y_R = -(\lambda x_R + \nu).$$

This finishes the description of addition in the general case. When adding P to itself (i.e. doubling P – refer back to Figure 2.6), the line $\ell : y = \lambda x + \nu$ is the tangent to E at P. Thus, its gradient λ is the derivative function dy/dx of E, evaluated at P. To obtain dy/dx, we differentiate the curve equation implicitly, as

$$\begin{aligned}\frac{d}{dx}(y^2) &= \frac{d}{dx}(x^3 + ax + b) \\ \frac{d}{dy}(y^2)\frac{dy}{dx} &= 3x^2 + a \\ \frac{dy}{dx} &= \frac{3x^2 + a}{2y}.\end{aligned}$$

Thus, $\lambda = \frac{dy}{dx}(P) = (3x_P^2 + a)/(2y_P)$, and $\nu = y_P - \lambda x_P$. Again, we substitute ℓ into E, but this time two of the roots of the resulting cubic are x_P, so we obtain x_R and y_R as

$$x_R = \lambda^2 - 2x_P; \qquad\qquad y_R = -(\lambda x_R + \nu).$$

This finishes the derivation of doubling formulas in the general case. We now complete the group law description by looking at the special cases. The point at infinity $\mathcal{O}$ is the identity, or neutral element, so any operation involving it is trivial. Otherwise, any operation between elements P and Q with different x-coordinates employs the general addition. This leaves the remaining cases of $x_P = x_Q$: (i) if $y_P = -y_Q$, then P and Q are inverses of each other and $P \oplus Q = \mathcal{O}$ (note that this includes $y_P = y_Q = 0$), and (ii) if $y_P = y_Q \neq 0$, then $P = Q$ and we use the point doubling formulas.

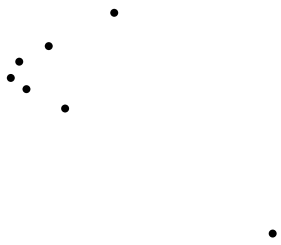
Much of the literature concerning the elliptic curve group law tends to present the complete description in the previous paragraph using an "if-then-else" style algorithm, where the "if" statements distinguish which of the above scenarios we are in. In optimised cryptographic implementations however, this is not the way that the group law operation is coded. This is because the groups we use are so large that the chances of running into a special case (that is not general doubling or general addition) randomly is negligible. Moreover, the parameters are usually chosen so that we are guaranteed not to run into these cases. In this light then, it will soon become clear that the major operations we are concerned with are point additions $R = P \oplus Q$ and point doublings $R = P \oplus P$, the formulas for which are summarised in (2.4) and (2.5) respectively.

$$\text{(Affine addition)} \qquad \lambda = \frac{y_Q - y_P}{x_Q - x_P}; \qquad \nu = y_P - \lambda x_P;$$
$$(x_P, y_P) \oplus (x_Q, y_Q) = (x_R, y_R) = \left(\lambda^2 - x_P - x_Q, -(\lambda x_R + \nu)\right). \quad (2.4)$$

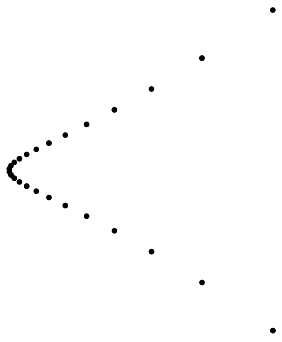
$$\text{(Affine doubling)} \qquad \lambda = \frac{3x_P^2 + a}{2y_P}; \qquad \nu = y_P - \lambda x_P;$$
$$[2](x_P, y_P) = (x_P, y_P) \oplus (x_P, y_P) = (x_R, y_R) = \left(\lambda^2 - 2x_P, -(\lambda x_R + \nu)\right). \quad (2.5)$$

Example 2.1.6 (Magma script). We revisit the curve $E/\mathbb{Q} : y^2 = x^3 - 2$ from Example 2.0.1 to verify the group law calculations that were stated. We start with the point doubling of $P = (x_P, y_P) = (3, 5)$, to compute $Q = [2]P = P \oplus P$ using (2.5). Here, $\lambda = \frac{3x_P^2+a}{2y_P} = \frac{3\cdot 3^2+0}{2\cdot 5} = \frac{27}{10}$, from which ν follows as $\nu = y_P - \lambda x_P = 5 - \frac{27}{10} \cdot 3 = -\frac{31}{10}$. Thus, $x_Q = \lambda^2 - 2x_P = (\frac{27}{10})^2 - 2 \cdot 3 = \frac{129}{100}$, and $y_Q = -(\lambda x_Q + \nu) = -(\frac{27}{10} \cdot \frac{129}{100} - \frac{31}{10}) = -\frac{383}{1000}$, giving $(x_Q, y_Q) = [2](x_P, y_P) = (\frac{129}{100}, -\frac{383}{1000})$. For the addition $R = P \oplus Q$, we use the formulas in (2.4), so $\lambda = \frac{y_Q - y_P}{x_Q - x_P} = (-\frac{383}{1000} - 5)/(\frac{129}{100} - 3) = \frac{5383}{1710}$, and $\nu = y_P - \lambda x_P = 5 - \frac{5383}{1710} \cdot 3 = -\frac{2533}{570}$. Thus, $x_R = \lambda^2 - x_P - x_Q = (\frac{5383}{1710})^2 - 3 - \frac{129}{100} = \frac{164323}{29241}$, and $y_R = \lambda x_R + \nu = \frac{5383}{1710} \cdot \frac{164323}{29241} - \frac{2533}{570} = -\frac{66234835}{5000211}$, so $(x_R, y_R) = (\frac{164323}{29241}, -\frac{66234835}{5000211})$. Since $Q = [2]P = P \oplus P$, then $R = P \oplus Q = [3]P$. We finish this example with a remark that further justifies the use of finite fields as the underlying fields in cryptography. It is not too painful to show that $P = (3, 5)$ and $\ominus P = (3, -5)$ are the only integral points on E [Sil09, Ch. IX, Prop. 7.1(b)], or that $E(\mathbb{Q})$ is actually *infinite cyclic* [Sil09, Ch. IX, Remark 7.1.1], meaning that among

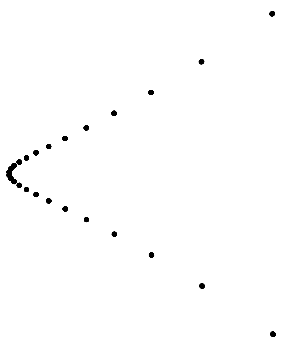
infinitely many rational points, only two have integer coordinates. Besides the infinite nature of $E(\mathbb{Q})$ (the lack of any finite subgroups is not useful in the context of discrete logarithm based cryptographic groups), observing the growing size of the numerators and denominators in $[n]P$, even for very small values of n, shows why using $E(\mathbb{Q})$ would be impractical. Using Magma, we can see that the denominator of the y-coordinate of $[10]P$ is 290 bits, whilst the denominator in $[100]P$ is 29201 bits, which agrees with the group law formulas in (2.4) and (2.5) that suggest that denominators of successive scalar multiples of P would grow quadratically; even Magma takes its time computing $[1000]P$, whose denominator is 2920540 bits, and Magma could not handle the computation of $[10000]P$. In Figure 2.11 we plot multiples of $P = (3, 5)$ that fall within the domain $x < 6$.



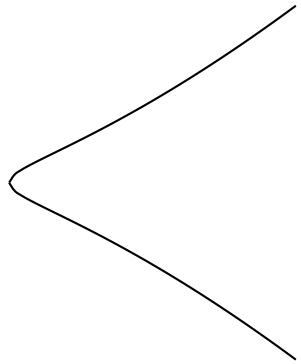
Of the first 10 multiples of $P = (3, 5)$ in $E(\mathbb{Q})$, 7 had $x < 6$.



Of the first 100 multiples of $P = (3, 5)$ in $E(\mathbb{Q})$, 64 had $x < 6$.



Of the first 1000 multiples of $P = (3, 5)$ in $E(\mathbb{Q})$, 635 had $x < 6$.



$E : y^2 = x^3 - 2$ over $\mathbb{R}$.

Figure 2.11: More and more points (with $x < 6$) in the infinite group $E(\mathbb{Q})$

From now on we will only be working with elliptic curves over finite fields. We start with a simple example of basic group law computations on $E(\mathbb{F}_q)$ to

summarise the discussion up until this point.

Example 2.1.7 (Magma script). $E/\mathbb{F}_{23} : y^2 = x^3 + 5x + 7$ is an elliptic curve, and both $P = (x_P, y_P) = (2, 5)$ and $Q = (x_Q, y_Q) = (12, 1)$ are on E. Using the affine point addition formulas in (2.4), we find $R = P \oplus Q$ by first computing $\lambda = \frac{y_Q - y_P}{x_Q - x_P} = \frac{1-5}{12-2} = -4 \cdot 10^{-1} = -28 = 18$, from which ν follows as $\nu = y_P - \lambda x_P = 5 - 18 \cdot 2 = -31 = 15$, so $\ell : y = 18x + 15$ is the line running through P and Q. We then compute $(x_R, y_R) = (\lambda^2 - x_P - x_Q, -(\lambda x_R + \nu))$, so $x_R = 18^2 - 2 - 12 = 11$ and $y_R = -(18 \cdot 11 + 15) = 17$, meaning $R = (11, 17)$. Applying (2.5) to compute $S = [2]P$ gives $\lambda' = \frac{3x_P^2 + a}{2y_P} = \frac{3 \cdot 2^2 + 5}{2 \cdot 5} = 17 \cdot 10^{-1} = 17 \cdot 7 = 4$, and ν' follows as $\nu' = y_P - \lambda' x_P = 5 - 4 \cdot 2 = 20$, so $\ell' : y = 4x + 20$ is the tangent line that intersects E with multiplicity two at P. We then compute $(x_S, y_S) = (\lambda'^2 - 2x_P, -(\lambda' x_S + \nu'))$, so $x_S = 4^2 - 2 \cdot 2 = 12$ and $y_S = -(4 \cdot 12 + 20) = -68 = 1$, meaning $S = (12, 1)$.

We now give an example of the *multiplication-by-m* map on E, defined as

$$[m] : E \to E, \qquad P \mapsto [m]P,$$

and illustrate the straightforward way to compute it in practice. This operation is analogous to exponentiation $g \mapsto g^m$ in $\mathbb{Z}_q^*$, and is the central operation in ECC, as it is the *one-way* operation that buries discrete logarithm problems in $E(\mathbb{F}_q)$. To efficiently compute the exponentiation g^m in $\mathbb{Z}_q^*$, we *square-and-multiply*, whilst to compute the scalar multiplication $[m]P$ in $E(\mathbb{F}_q)$, we (because of the additive notation) *double-and-add*.

Example 2.1.8 (Magma script). Let $E/\mathbb{F}_{1021} : y^2 = x^3 - 3x - 3$ so that $r = \#E(\mathbb{F}_q) = 1039$ is prime. Let $P = (379, 1011) \in E$ and $m = 655$, and suppose we are to compute $[m]P = [655](379, 1011)$. To double-and-add, we write the (10-bit) binary representation of m as $m = (m_9, ..., m_0)_2 = (1, 0, 1, 0, 0, 0, 1, 1, 1, 1)$. Initialising $T \leftarrow P$, and starting from the second most significant bit m_8, we successively compute $T \leftarrow [2]T$ for each bit down to m_0, and whenever $m_i = 1$ we compute $T \leftarrow T + P$. So, in our case it takes 9 doublings $T \leftarrow [2]T$ and 5 additions $T \leftarrow T + P$ to compute $[m]P$, which ends up being $[655](379, 1011) = (388, 60)$. In general then, this straightforward double-and-add algorithm will take $\log_2 m$ doublings and roughly half as many additions to compute $[m]P$ (if m is randomly chosen).

2.1.3 The group axioms

All but one of the group axioms are now concrete. Namely, for *closure*, if we start with two points in $E(K)$, then the chord-and-tangent process gives rise to a cubic polynomial in K for which two roots (the two x-coordinates of the points we started with) are in K, meaning the third root must also be in K; the explicit formulas affirm this. The *identity* and *inverse* axioms are fine, since $P \oplus \mathcal{O} = P$, and the element $\ominus P$ such that $P \oplus (\ominus P) = \mathcal{O}$ is clearly unique and well defined for all P. We also note that the group is *abelian*, since the process of computing $P \oplus Q$ is symmetric. The only non-obvious axiom is *associativity*, i.e. showing $(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$. An elementary approach using the explicit formulas above can be used to show associativity by treating all the separate cases, but this approach is rather messy [Fri05]. Silverman gives a much more instructive proof [Sil09, Ch. III.3.4e] using tools that we will develop in the following chapter, but for now we offer some temporary intuition via the illustration in Figures 2.12 and 2.13.

2.1.4 Speeding up elliptic curve computations

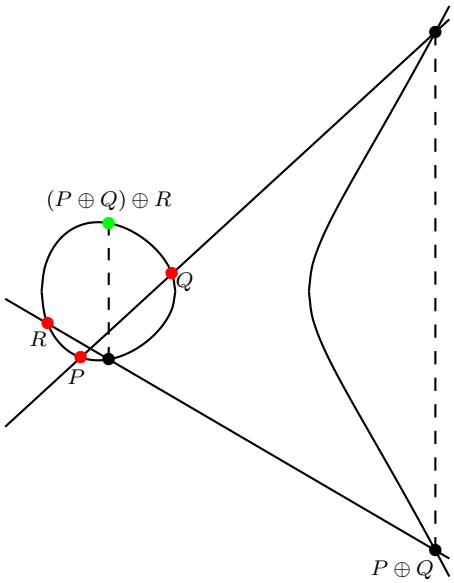


Figure 2.12: $(P \oplus Q) \oplus R$.

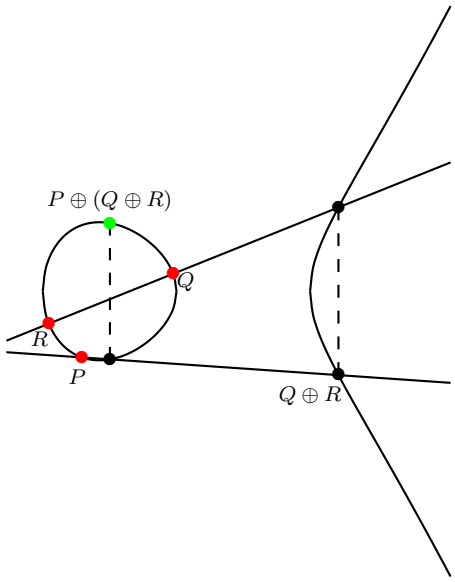


Figure 2.13: $P \oplus (Q \oplus R)$.

Group law computations on elliptic curves are clearly more complicated than computations in traditional groups that facilitate discrete logarithm based protocols like $\mathbb{F}_q^*$; the explicit formulas in (2.4) and (2.5) use many field operations.

However, in the context of cryptography, the more abstract nature of elliptic curve groups actually works in their favour. This is essentially because attackers aiming to solve the discrete logarithm problem on elliptic curves also face this abstractness. The subexponential algorithms that apply to finite field discrete logarithms[1] do not translate to the elliptic curve setting, where the best available attacks remain generic, exponential algorithms like Pollard rho [Pol78]. This means that elliptic curve groups of a relatively small size achieves the same conjectured security as multiplicative groups in much larger finite fields, i.e. $E(\mathbb{F}_{q_1})$ and $\mathbb{F}_{q_2}^*$ achieve similar security when $q_2 \gg q_1$. For example, an elliptic curve defined over a 160-bit field currently offers security comparable to a finite field of 1248 bits [Sma10, Table 7.2]. Thus, although more field operations are required to perform a group law computation, these operations take place in a field whose operational complexity is much less, and this difference is more than enough to tip the balance in the favour of elliptic curves. In addition, the smaller group elements in $E(\mathbb{F}_{q_1})$ implies much smaller key sizes, greatly reducing storage and bandwidth requirements. These are some of the major reasons that elliptic curves have received so much attention in the realm of public-key cryptography; the field of elliptic curve cryptography (ECC) has been thriving since Koblitz [Kob87] and Miller [Mil85] independently suggested their potential as alternatives to traditional groups.

One avenue of research that has given ECC a great boost is that of optimising the group law computations. The explicit formulas in affine coordinates ((2.4) and (2.5)) would not be used to compute the group law in practice, and in fact the Weierstrass model $E : y^2 = x^3 + ax + b$ is often not the optimal curve model either. A huge amount of effort has been put towards investigating other models and coordinate systems in order to minimise the field operations required in group law computations. One of the initial leaps forward in this line of research was the observation that performing computations in projective space avoids field inversions, which are extremely costly in practice. We illustrate these techniques in the following examples.

Example 2.1.9 (Magma script). Consider a general Weierstrass curve $E(\mathbb{F}_q) : y^2 = x^3 + ax + b$ where q is a large prime, and let $\mathbf{M}$, $\mathbf{S}$ and $\mathbf{I}$ represent the cost of computing multiplications, squarings and inversions in $\mathbb{F}_q$ respectively. To compute a general affine point doubling $(x_R, y_R) = [2](x_P, y_P)$ using (2.5) costs

[1] See Diem's notes on *index calculus* for a nice introduction [Die12].

$2\mathbf{M}+2\mathbf{S}+\mathbf{I}$, and to compute a general affine point addition $(x_R, y_R) = (x_P, y_P) \oplus (x_Q, y_Q)$ using (2.4) costs $2\mathbf{M} + \mathbf{S} + \mathbf{I}$. On the other hand, we can transform the formulas into homogeneous projective space according to the substitutions $x = X/Z$ and $y = Y/Z$, and we can consider computing $(X_R : Y_R : Z_R) = [2](X_P : Y_P : Z_P)$ and $(X_R : Y_R : Z_R) = (X_P : Y_P : Z_P) \oplus (X_Q : Y_Q : Z_Q)$ on $E : Y^2Z = X^3 + aXZ^2 + bZ^3$. For the addition case, substituting $x_i = X_i/Z_i$ and $y_i = Y_i/Z_i$ for $i \in \{P, Q, R\}$ into the affine formulas

$$x_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q; \qquad y_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x_R) - y_P$$

taken from (2.4), gives

$$\frac{X_R}{Z_R} = \left(\frac{\frac{Y_Q}{Z_Q} - \frac{Y_P}{Z_P}}{\frac{X_Q}{Z_Q} - \frac{X_P}{Z_P}}\right)^2 - \frac{X_P}{Z_P} - \frac{X_Q}{Z_Q}; \qquad \frac{Y_R}{Z_R} = \left(\frac{\frac{Y_Q}{Z_Q} - \frac{Y_P}{Z_P}}{\frac{X_Q}{Z_Q} - \frac{X_P}{Z_P}}\right)\left(\frac{X_P}{Z_P} - \frac{X_R}{Z_R}\right) - \frac{Y_P}{Z_P}.$$

After a little manipulation, we can then set Z_R to be the smallest value that contains both denominators above, and update the numerators accordingly to give

$$\begin{aligned}
X_R &= (X_PZ_Q - X_QZ_P)\left(Z_PZ_Q(Y_PZ_Q - Y_QZ_P)^2 - (X_PZ_Q - X_QZ_P)^2(X_PZ_Q + X_QZ_P)\right);\\
Y_R &= Z_PZ_Q(X_QY_P - X_PY_Q)(X_PZ_Q - X_QZ_P)^2\\
&\quad - (Y_PZ_Q - Y_QZ_P)\left((Y_PZ_Q - Y_QZ_P)^2Z_PZ_Q - (X_PZ_Q + X_QZ_P)(X_PZ_Q - X_QZ_P)^2\right);\\
Z_R &= Z_PZ_Q(X_PZ_Q - X_QZ_P)^3.
\end{aligned}$$

The explicit formulas database (EFD) [BL07a] reports that the above formulas can be computed in a total of $12\mathbf{M} + 2\mathbf{S}$. The real power of adopting projective coordinates for computations becomes apparent when we remark that most optimised implementations of $\mathbb{F}_q$ arithmetic have $\mathbf{I} \gg 20\mathbf{M}$, and the multiplication to inversion ratio is commonly reported to be $80 : 1$ or higher. Thus, the $12\mathbf{M} + 2\mathbf{S}$ used for additions in projective space will be much faster than the $2\mathbf{M} + \mathbf{S} + \mathbf{I}$ for affine additions. For completeness, we remark that deriving the projective formulas for computing $(X_R : Y_R : Z_R) = [2](X_P : Y_P : Z_P)$ is analogous (but substantially more compact since we only have the projective coordinates of P to deal with), and the EFD reports that this can be done in $5\mathbf{M}+6\mathbf{S}$, which will again be much faster than the $2\mathbf{M} + 2\mathbf{S} + I$ in affine space.

The Weierstrass model for elliptic curves covers all isomorphism classes, meaning that every elliptic curve can be written in Weierstrass form. Other

models of elliptic curves are usually available if some condition holds, and (if this is the case) it can be advantageous to adopt such a model, as the following example shows.

Example 2.1.10 (Magma script). If $x^3 + ax + b$ has a root in $\mathbb{F}_q$, then Billet and Joye [BJ03, Eq. 8-10] show that instead of working with $E : y^2 = x^3 + ax + b$, we can work with the (birationally equivalent) *Jacobi-quartic* curve $J : v^2 = au^4 + du^2 + 1$, for appropriately defined a, d (that depend on the root). Here we write J using (u, v) coordinates so back-and-forth mappings are defined without confusion. Thus, consider $E/\mathbb{F}_{97} : y^2 = x^3 + 5x + 5$, for which $x^3 + 5x + 5$ has 34 as a root, so we will work on the isomorphic curve $J/\mathbb{F}_{97} : v^2 = 73u^4 + 46u^2 + 1$. Instead of homogeneous projective coordinates, [BJ03] projectified under the substitution $u = U/W$ and $v = V/W^2$, which gives the (non-homogeneous) projective closure as $J : V^2 = 73U^4 + 46U^2W^2 + W^4$. Any point $(x, y) \neq \mathcal{O}$ on E can be taken straight to the projective closure of J via

$$(x, y) \mapsto \left(2(x - 34) : (2x + 34)(x - 34)^2 - y^2 : y\right),$$

with the reverse mapping given by

$$(U : V : W) \mapsto \left(2\frac{V + W^2}{U^2} - 17, W\frac{4(V + W^2) - 5U^2}{U^3}\right).$$

For example $(x, y) = (77, 21)$ maps to $(U : V : W) = (86 : 8 : 21)$, and vice versa. We now look at the formulas for the point addition $(U_3 : V_3 : W_3) = (U_1 : V_1 : W_1) \oplus (U_2 : V_2 : W_2)$ on $J : V^2 = aU^4 + dU^2W^2 + W^4$, taken from [BJ03, Eq. 11], as

$$\begin{aligned}
U_3 &= U_1W_1V_2 + U_2W_2V_1, \\
V_3 &= \left((W_1W_2)^2 + a(U_1U_2)^2\right)(V_1V_2 + dU_1U_2W_1W_2) + 2aU_1U_2W_1W_2(U_1^2W_2^2 + U_2^2W_1^2), \\
W_3 &= (W_1W_2)^2 - a(U_1U_2)^2,
\end{aligned}$$

where we immediately highlight the relative simplicity of the above formulas in comparison to the homogeneous projective formulas derived in the previous example. Unsurprisingly then, the fastest formulas for Jacobi-quartic additions and doublings outdo those for general Weierstrass curves in homogeneous projective space. Namely, the current fastest formulas for doublings on Jacobi-quartics cost $2\mathbf{M} + 5\mathbf{S}$ and additions cost $6\mathbf{M} + 4\mathbf{S}$ [HWCD09], whilst in the previous

example we had $5\mathbf{M} + 6\mathbf{S}$ for doublings and $12\mathbf{M} + 2\mathbf{S}$ for additions.

The Jacobi-quartic curves discussed above are just one example of dozens of models that have been successful in achieving fast group law computations, and therefore fast cryptographic implementations. Other well known models include Edwards curves [Edw07, BL07b], Hessian curves [JQ01, Sma01] and Montgomery curves [Mon87]. We refer to the EFD [BL07a] for a catalogue of all the fastest formulas for the popular curve models, and to Hisil's thesis [His10] for a general method of (automatically) deriving fast group law algorithms on arbitrary curve models. For any reader wishing to delve even further into group law arithmetic on elliptic curves, we also recommend the recent, advanced works by Castryck and Vercauteren [CV11], and by Kohel [Koh11].

2.2 Torsion, endomorphisms and point counting

We now turn our focus to the behaviour of elliptic curve groups, as they are used in cryptography. We start by importantly discussing the possible structures exhibited by the finite group $E(\mathbb{F}_q)$. It turns out that $E(\mathbb{F}_q)$ is either itself cyclic, or isomorphic to a product of two cyclic groups $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ with $n_1 \mid n_2$ [ACD+05, Prop. 5.78]. In cryptography, we would like the group $E(\mathbb{F}_q)$ to be *as cyclic as possible*, so we usually prefer the former case, or at the very least for n_1 to be very small. In most cases of practical interest, we can generate curves that are cyclic with relative ease, so throughout this thesis it is to safe assume that $E(\mathbb{F}_q)$ is cyclic (but to see the real depth of this question in general, we refer to [MS07]). The following example illustrates that $E(\mathbb{F}_q) = \langle P \rangle$ obeys all the usual rules that apply to cyclic groups, and introduces the important notion of *r-torsion*.

Example 2.2.1 (Magma script). Consider $E/\mathbb{F}_{101} : y^2 = x^3 + x + 1$. The group order is $\#E(\mathbb{F}_q) = 105 = 3 \cdot 5 \cdot 7$, and $P = (47, 12) \in E$ is a generator. Lagrange's theorem says that points (and subgroups) over the base field will have order in $\{1, 3, 5, 7, 15, 21, 35, 105\}$. Indeed, to get a point of order $r \mid 105$, we simply multiply P by the appropriate *cofactor*, which is $h = \#E/r$. For example, a point of order 3 is $[35](47, 12) = (28, 8)$, a point of order 21 is $[5](47, 12) = (55, 65)$, and a point of order 1 is $[105](47, 12) = \mathcal{O}$ (which is the only such point). By

definition, a point is "killed" (sent to $\mathcal{O}$) when multiplied by its order. Any point over the full closure $E(\overline{\mathbb{F}}_q)$ that is killed by r is said to be in the r-torsion. So, the point $(55, 65)$ above is in the 21-torsion, as is the point $(28, 8)$. There are exactly 21 points in $E(\mathbb{F}_q)$ in the 21-torsion, but there are many more in $E(\overline{\mathbb{F}}_q)$.

The whereabouts and structure of r-torsion points in $E(\overline{\mathbb{F}}_q)$ (alluded to at the end of Example 2.2.1) plays a crucial role in pairing-based cryptography; we will be looking at this in close detail in Chapter 4.

In ECC we would like the group order $\#E(\mathbb{F}_q)$ to be as close to prime as possible. This is because the (asymptotic) complexity of the ECDLP that attackers face is dependent on the size of the largest prime subgroup of $E(\mathbb{F}_q)$. Even if the particular instance of the discrete logarithm problem uses a generator of the whole group, the attacker can use the known group order to solve smaller instances in subgroups whose orders are pairwise prime, and then reconstruct the answer using the Chinese Remainder Theorem (CRT). We make this clear in the following two examples: the first is a toy example, whilst the second shows the difference between two curves of the same cryptographic size; one that is currently considered secure and one that is completely breakable using modern attacks.

Example 2.2.2 (Magma script). Consider $E/\mathbb{F}_{1021} : y^2 = x^3 + 905x + 100$, with group order $\#E(\mathbb{F}_q) = 966 = 2 \cdot 3 \cdot 7 \cdot 23$, and generator $P = (1006, 416)$. Suppose we are presented with an instance of the ECDLP: namely, we are given $Q = (612, 827)$, and we seek to find k such that $[k]P = Q$. For the sake of the example, suppose our best "attack" is trivial: trying every multiple $[i]P$ of P until we hit the correct one ($i = k$). Rather than seeking i in the full group ($2 \leq i \leq 965$), we can map the instance into each prime order subgroup by multiplying by the appropriate cofactor, and then solve for $k_j \equiv k \bmod j$, $j \in \{2, 3, 7, 23\}$. For $j = 2$, we have $P_j = P_2 = [966/2]P = [483](1006, 416) = (174, 0)$, and $Q_j = Q_2 = [483](612, 827) = (174, 0)$, so $Q_2 = [k_2]P_2$ gives $k_2 = 1$. For $j = 3$, we have $P_3 = [322]P = (147, 933)$ and $Q_3 = [322]P = \mathcal{O}$, so $Q_3 = [k_3]P_3$ gives $k_3 = 3$. For $j = 7$, we have $P_7 = [138]P = (906, 201)$ and $Q_7 = [138]Q = (906, 201)$, so $Q_7 = [k_7]P_7$ gives $k_7 = 1$. For $j = 23$, we have $P_{23} = [42]P = (890, 665)$ and $Q_{23} = [42]Q = (68, 281)$. For $Q_{23} = [k_{23}]P_{23}$, we exhaust $k_{23} \in \{1, .., 22\}$ to see that $k_{23} = 20$. Now, we can use the Chinese Remainder Theorem to solve

$$k \equiv k_2 = 1 \bmod 2; \quad k \equiv k_3 = 0 \bmod 3; \quad k \equiv k_7 = 1 \bmod 7; \quad k \equiv k_{23} = 20 \bmod 23,$$

which gives $k \equiv 687 \bmod \#E$, solving the ECDLP instance. Notice that the

hardest part was exhausting the set $\{1, .., 22\}$ to find $k_{23} = 20$, so the largest prime order subgroup becomes the bottleneck of the algorithm, giving intuition as to why the largest prime order subgroup defines the attack complexity when groups of a cryptographic size are used.

Example 2.2.3 (Magma script). For our real world example, we take the curve P-256 from the NIST recommendations [NIS99], which currently achieves a similar security level (resistance against best known attacks) to the 128-bit Advanced Encryption Standard (AES) for symmetric encryption. The curve is defined as $E/\mathbb{F}_q : y^2 = x^3 - 3x + b$, with prime order $r = \#E$, and generator $G = (x_G, y_G)$, where

$q = 115792089210356248762697446949407573530086143415290314195533631308867097853951,$
$r = 115792089210356248762697446949407573529996955224135760342422259061068512044369,$
$b = 41058363725152142129326129780047268409114441015993725554835256314039467401291,$
$x_G = 48439561293906451759052585252797914202762949526041747995844080717082404635286,$
$y_G = 36134250956749795798585127919587881956611106672985015071877198253568414405109,$
$x_H = 53987601597021778433910548064987973235945515666715026302948657055639179420355,$
$y_H = 53690949263410447908824456005055253553237881490194075871737490561466076234637.$

We give another point $H = (x_H, y_H)$ to pose $H = [k]G$ as an intractable instance of the ECDLP; this 256-bit prime field (and group order) is far beyond the reach of current attacks. For example, there is currently a campaign underway to solve a discrete logarithm problem over a 130-bit field using a cluster of servers that have already been running for two years (see `http://ecc-challenge.info/`), so (assuming the best known attacks stay exponential) it seems the above ECDLP should be safe for a while yet. We remark that the prime characteristic q is given by $q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$; such primes are preferred in ECC as they allow for faster finite field multiplication and reduction routines, greatly enhancing the speed of $\mathbb{F}_q$ arithmetic. We now give a curve over the same field $\mathbb{F}_q$, for which the ECDLP is well within reach of the best known attacks. Namely, consider the alternative curve with $b = 0$, namely $\tilde{E}/\mathbb{F}_q : y^2 = x^3 - 3x$, whose group order $n = \#\tilde{E}$ is given as

$$n = 115792089210356248762697446949407573530086143415290314195533631308867097853952,$$
$$= 2^{96} \cdot 7 \cdot 274177 \cdot 67280421310721 \cdot 1131830892797394193140491410 3.$$

This time, the largest prime divisor of the group order is only 94 bits long, and the complexity of solving the ECDLP in $\tilde{E}(\mathbb{F}_q)$ is governed by the difficulty of solving the ECDLP instance in this largest prime subgroup, which could be done in a small amount of time on a desktop computer.

The above example provides clear motivation as to the importance of counting points on elliptic curves. The largest prime factor of the group order determines the difficulty that attackers face when trying to solve the ECDLP, so we would like to be able to count points on curves quickly enough to find those whose order is prime or almost prime (i.e. has a small cofactor), or have methods of prescribing such a group order before searching for the curve. Fortunately, on elliptic curves we have efficient algorithms to do both.

We start our brief discussion on elliptic curve point counting by referring back to the two group orders in Example 2.2.3, and observing that both group orders share the first half of their digits with those of the field characteristic q. This suggests that the number of points on an elliptic curve is close to q, which is indeed the case in general; the *Hasse bound* [Sil09, Ch. 5, Th. 1.1] says the most that $\#E(\mathbb{F}_q)$ can differ from $q+1$ is $2\sqrt{q}$, i.e. $|\#E(\mathbb{F}_q) - (q+1)| \leq 2\sqrt{q}$. This offset between $\#E(\mathbb{F}_q)$ and $(q+1)$ is called the *trace of Frobenius*, and is denoted by t, so

$$\#E(\mathbb{F}_q) = q + 1 - t, \qquad |t| \leq 2\sqrt{q} \tag{2.6}$$

We will discuss where t comes from and provide some more intuition behind the above formula in a moment, but what the Hasse bound tells us is that the group order lies somewhere in the interval $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$. In fact, Deuring [Deu41] showed that when q is prime[2], then every value $N \in [q+1-2\sqrt{q}, q+1+2\sqrt{q}]$ can be found as a group order $\#E(\mathbb{F}_q)$ for some E.

Example 2.2.4 (Magma script). Let $q = 23$, so that the Hasse interval becomes $[q+1-2\sqrt{q}, q+1+2\sqrt{q}] = [15, 33]$, meaning that there are exactly 19 different

[2]When q is a prime power, there are a very small number of explicitly described exceptions.

group orders taken by elliptic curves over $\mathbb{F}_{23}$. For example, $E/\mathbb{F}_{23} : y^2 = x^3 + 18x + 3$ has $\#E = 15$, whilst $\tilde{E}/\mathbb{F}_{23} : y^2 = x^3 + 13x + 7$ has $\#\tilde{E} = 33$. We give 19 (a, b) pairs such that the corresponding curves $E : y^2 = x^3 + ax + b$ have group orders in ascending order spanning the whole interval, as follows: $(18, 3)$, $(7, 22)$, $(19, 14)$, $(17, 17)$, $(12, 5)$, $(7, 12)$, $(8, 10)$, $(17, 18)$, $(20, 20)$, $(2, 3)$, $(20, 3)$, $(6, 8)$, $(16, 8)$, $(16, 22)$, $(9, 16)$, $(19, 6)$, $(20, 8)$, $(22, 9)$, $(13, 7)$.

A rough (but elementary and instinctive) argument as to why $\#E \approx q$ is that approximately half of the values $x \in [0, .., q-1]$ will give a quadratic residue $x^3 + ax + b \in \mathrm{QR}(q)$, which gives rise to two points $(x, \pm\sqrt{x^3 + ax + b}) \in E(\mathbb{F}_q)$, the only exception(s) being when $x^3 + ax + b = 0$ which obtains one point. The sophisticated explanation requires a deeper knowledge than our introduction offers, but for the purposes of this introductory text we get almost all that we need from Equation (2.6); the derivation of which makes use of the following definition. If E is defined over $\mathbb{F}_q$, then the *Frobenius endomorphism* π is defined as

$$\pi : E \to E, \qquad (x, y) \mapsto (x^q, y^q). \tag{2.7}$$

We note that the Frobenius endomorphism maps any point in $E(\overline{\mathbb{F}}_q)$ to a point in $E(\overline{\mathbb{F}}_q)$, but the set of points fixed by π is exactly the group $E(\mathbb{F}_q)$. Thus, π only acts non-trivially on points in $E(\overline{\mathbb{F}}_q) \setminus E(\mathbb{F}_q)$, and more generally, $\pi^i : (x, y) \mapsto (x^{q^i}, y^{q^i})$ only acts non-trivially on points in $E(\overline{\mathbb{F}}_q) \setminus E(\mathbb{F}_{q^i})$.

Example 2.2.5 (Magma script). Let $q = 67$, and consider $E/\mathbb{F}_q : y^2 = x^3 + 4x + 3$, and let $\mathbb{F}_{q^2} = \mathbb{F}_q(u)$ where $u^2 + 1 = 0$, and further let $\mathbb{F}_{q^3} = \mathbb{F}_q(v)$ where $v^3 + 2 = 0$. For $P_1 = (15, 50) \in E(\mathbb{F}_q)$, we have $\pi_q(P_1) = (15^q, 50^q) = (15, 50)$. For $P_2 = (2u + 16, 30u + 39)$, we have $\pi_q(P_2) = ((2u + 16)^q, (30u + 39)^q) = (65u + 16, 39 + 37u)$; it is easy to see in this example that computing $\pi_q(Q)$ for any $Q \in E(\mathbb{F}_{q^2})$ involves a simple "complex conjugation" on each coordinate, which also agrees with $\pi_q^2(Q) = Q$. Let $P_3 = (15v^2 + 4v + 8, 44v^2 + 30v + 21)$, $\pi_q(P_3) = (33v^2 + 14v + 8, 3v^2 + 38v + 21)$, $\pi_q^2(P_3) = (19v^2 + 49v + 8, 20v^2 + 66v + 21)$, and $\pi_q^3(P_3) = P_3$.

We can now return to sketch the derivation of Equation (2.6) by skimming over results that are presented in full in Silverman's book [Sil09, Ch. V, Th. 1.1]. We now know that $P \in E(\mathbb{F}_q)$ if and only if $\pi(P) = P$ (i.e. $([1] - \pi)P = \mathcal{O}$), and thus $\#E(\mathbb{F}_q) = \#\ker([1] - \pi)$. It is not too hard to show that the map

$[1] - \pi$ is separable, which means that $\#E(\mathbb{F}_q) = \#\ker([1] - \pi) = \deg([1] - \pi)$. We can then make use of (a special case of) a version of the Cauchy-Schwarz inequality [Sil09][Ch. V, Lemma 1.2], to give $|\deg([1] - \pi) - \deg([1]) - \deg(\pi)| \leq 2\sqrt{\deg([1])\deg(\pi)}$, from which Equation (2.6) follows from $\deg(\pi) = q$.

The theory of elliptic curves makes constant use of the *endomorphism ring* of E, denoted $\mathrm{End}(E)$, which (as the name suggests) is the ring of all maps from E to itself; addition in the ring is natural, i.e. $(\psi_1 + \psi_2)(P) = \psi_1(P) + \psi_2(P)$, and multiplication in $\mathrm{End}(E)$ is composition $(\psi_1\psi_2)(P) = \psi_1(\psi_2(P))$. The *multiplication-by-m* map $[m]$ is trivially in $\mathrm{End}(E)$ for all $m \in \mathbb{Z}$, and when E is defined over a finite field, then clearly π is too, so we are usually interested in any extra endomorphisms that shed more light on the behaviour of E.

Example 2.2.6 (Magma script). Consider $E/\mathbb{F}_q : y^2 = x^3 + b$. The map ξ, defined by $\xi : (x, y) \mapsto (\xi_3 x, y)$ with $\xi_3^3 = 1$ and $\xi_3 \neq 1$, is a non-trivial endomorphism on E, so $\xi \in \mathrm{End}(E)$. If $\xi_3 \in \mathbb{F}_q$, then ξ will be defined over $\mathbb{F}_q$, otherwise $\xi_3 \in \mathbb{F}_{q^2}$ in which case ξ is not *defined over* $\mathbb{F}_q$, but over $\mathbb{F}_{q^2}$. We will observe both cases. Firstly, cubic roots of unity will be defined in $\mathbb{F}_q$ if and only if $q \equiv 1 \bmod 3$, so let us take $q \equiv 19$, $b = 5$, which gives $E/\mathbb{F}_{19} : y^2 = x^3 + 5$. Let $\xi_3 = 7$ so that $\xi_3^3 = 1$ (we could have also taken $\xi_3^2 = 11$), so that $\xi : (x, y) \mapsto (7x, y)$ is an endomorphism on E. Applying this to, say $P = (-1, 2)$, gives $\xi(P) = (-7, 2) \in E$. Taking the same curve over $\mathbb{F}_{23}$, i.e. $E/\mathbb{F}_{23} : y^2 = x^3 + 5$, for which $P = (-1, 2)$ is a again a point, we no longer have a non-trivial $\xi_3 \in \mathbb{F}_{23}$, so we must form a quadratic extension $\mathbb{F}_{q^2}(u)$, $u^2 + 1 = 0$. Now, we can take $\xi_3 = 8u + 11$ (the other option is $\xi_3^2 = 15u + 11$), so that $\xi(P) = (-(8u + 11), 2) = (15u + 12, 2) \in E(\mathbb{F}_{q^2})$. Notice that P started in $E(\mathbb{F}_q)$, but landed in $E(\mathbb{F}_{q^2})$ under ξ. The endomorphism ξ has an inverse ξ^{-1} (which is defined the same way but with ξ_3^2 instead), so ξ is actually an automorphism of E, written as $\xi \in \mathrm{Aut}(E)$.

The definition of $\xi : (x, y) \mapsto (\xi_3 x, y)$ in the above example gives an endomorphism on $E : y^2 = x^3 + b$ regardless of the field that E is defined over. If there exists a non-trivial map (like ξ) for an elliptic curve E, we say E has *complex multiplication*. To be more precise, all elliptic curve endomorphism rings trivially contain $\mathbb{Z}$, since every $m \in \mathbb{Z}$ corresponds to the multiplication-by-m map $[m] \in \mathrm{End}(E)$. However, if non-trivial endomorphisms exist that make $\mathrm{End}(E)$ strictly larger than $\mathbb{Z}$, then we say E has complex multiplication (CM). Thus, by this definition, every elliptic curve defined over $\mathbb{F}_q$ has CM, because the existence of the Frobenius endomorphism $\pi \in \mathrm{End}(E)$ makes $\mathrm{End}(E)$ larger than $\mathbb{Z}$.

However, if we discuss whether E has CM without yet stipulating the underlying finite field, then the question becomes non-trivial in general, because the answer depends on the existence of non-trivial maps. We use Silverman's example to illustrate [Sil09, Ch. 3, Eg. 4.4].

Example 2.2.7 (Magma script). Consider $E/K : y^2 = x^3 + ax$. The map $\zeta : (x, y) \mapsto (-x, iy)$, where $i^2 = -1$ in K is an endomorphism, so E has CM. Clearly, ζ will be defined over K if and only if $i \in K$. Observe that $\zeta \circ \zeta(x, y) = \zeta(-x, iy) = (x, -y) = -(x, y)$, so $\zeta \circ \zeta = [-1]$ (i.e. ζ^2 is equivalent to negation). Thus, there is a ring homomorphism $\mathbb{Z}[i] \to \mathrm{End}(E)$ defined by $m + ni \mapsto [m] + [n] \circ \zeta$. If $\mathrm{Char}(K) \neq 0$, then this map is an isomorphism, thus $\mathrm{End}(E) \cong \mathbb{Z}[i]$, and $\mathrm{Aut}(E) \cong \mathbb{Z}[i]^*$.

The trace of Frobenius t in Equation (2.6) is named so because of the role it plays in the characteristic polynomial satisfied by π, which is given as

$$\pi^2 - [t] \circ \pi + [q] = 0 \qquad \text{in } \mathrm{End}(E), \tag{2.8}$$

meaning that for all $(x, y) \in E(\overline{\mathbb{F}}_q)$, we have

$$(x^{q^2}, y^{q^2}) - [t](x^q, y^q) + [q](x, y) = \mathcal{O}. \tag{2.9}$$

Example 2.2.8 (Magma script). We use our results from Example 2.2.5 to illustrate, so as before $E/\mathbb{F}_{67} : y^2 = x^3 + 4x + 3$, $\mathbb{F}_{q^2} = \mathbb{F}_q(u)$ where $u^2 + 1 = 0$, and $\mathbb{F}_{q^3} = \mathbb{F}_q(v)$ where $v^3 + 2 = 0$. The trace of Frobenius is $t = -11$, so $\#E(\mathbb{F}_q) = q + 1 - t = 79$. For $P_1 = (15, 50) \in E(\mathbb{F}_q)$, we trivially had $\pi^2(P_1) = \pi(P_1) = P_1$, so $P_1 - [t]P_1 + [q]P_1 = ([1] - [t] + [q])P_1 = [\#E(\mathbb{F}_q)]P_1 = \mathcal{O}$. For $P_2 = (2u+16, 30u+39)$, we had $\pi^2(P_2) = P_2$ and $\pi(P_2) = (65u+16, 37u+39)$, so we are computing $P_2 - [-11]\pi(P_2) + [67]P_2 = [68](2u+16, 30u+39) + [11](65u+16, 37u+39)$, which is indeed $\mathcal{O}$. $P_3 \in E(\mathbb{F}_{q^3})$ is the only case where both π and π^2 act non-trivially, so we compute $(19v^2+49v+8, 20v^2+66v+21) - [-11](33v^2+14v+8, 3v^2+38v+21) + [67](15v^2+4v+8, 44v^2+30v+21)$, which is $\mathcal{O}$.

We now give a brief sketch of Schoof's algorithm for counting points on elliptic curves [Sch85]. Understanding the algorithm is not a prerequisite for understanding pairings, but it certainly warrants mention in any overview text on elliptic curves in cryptography, since it is essentially the algorithm that made ECC practical. Before Schoof's polynomial-time algorithm, all algorithms for point counting on elliptic curves were exponential and therefore cryptographi-

cally impractical. Besides, to sketch his idea, we need to introduce the notion of *division polynomials*, which are a useful tool in general. Put simply, division polynomials are polynomials whose roots reveal torsion points: namely, for odd[3] ℓ, the ℓ-th division polynomial $\psi_\ell(x)$ on E solves to give the x-coordinates of the points of order ℓ. They are defined recursively and depend on the curve constants a and b, but rather than giving the recursions here, we point the reader to [Sil09, Ch. III, Exer. 3.7], and opt instead for an example that illustrates their usefulness.

Example 2.2.9 (Magma script). Recall the curve $E/\mathbb{F}_{101} : y^2 = x^3 + x + 1$ from Example 2.2.1 with group order $\#E(\mathbb{F}_q) = 105 = 3 \cdot 5 \cdot 7$. The x-coordinates of the points of order 2 are found as the roots of $\psi_2(x) = 4x^3 + 4x + 4$, which is irreducible in $\mathbb{F}_q[x]$, so there are no 2-torsion points in $E(\mathbb{F}_q)$. For $r = 3$, $\psi_3(x) = 3x^4+6x^2+12x+100 \in \mathbb{F}_q[x]$ factors into $\psi_3(x) = (x+73)(x+84)(x^2+45x+36)$, so we get two solutions over $\mathbb{F}_q$, namely $x = 17$ and $x = 28$. This does not mean that the points implied by both solutions are in $\mathbb{F}_q$: namely, $x = 28$ gives $x^3 + x + 1 \in \mathrm{QR}(q)$, so two points in the 3-torsion follow as $(28, 8)$ and $(28, 93)$. Conversely, $x = 17$ gives $x^3+x+1 \notin \mathrm{QR}(q)$, so the two points implied by $x = 17$ will be defined over $\mathbb{F}_{q^2}$. For $\psi_5(x) = 5x^{12} + ... + 16$, the factorisation in $\mathbb{F}_q[x]$ is $\psi_5(x) = (x + 15)(x + 55)(x^5 + ... + 1)(x^5 + ... + 100)$, which gives $x = 46$ and $x = 86$ as solutions. This time, both x values give rise to two points, giving four non-trivial 5-torsion points in total: $(46, 25)$, $(46, 76)$, $(86, 34)$, $(86, 67)$. $\psi_7(x)$ is degree 24, and gives three linear factors in $\mathbb{F}_q[x]$, all of which result in two 7-torsion points, giving 6 non-trivial torsion points in total: $(72, 5)$, $(72, 96)$, $(57, 57)$, $(57, 44)$, $(3, 43)$, $(3, 58)$. Other division polynomials have roots in $\mathbb{F}_q$, but these roots will not give rise to points defined over $\mathbb{F}_q$. For example, $\psi_{11}(x)$ has 5 roots over $\mathbb{F}_q$ (13, 18, 19, 22, 63), but none of them give points in $E(\mathbb{F}_q)$, meaning we will have to extend to $E(\mathbb{F}_{q^2})$ to collect any 11-torsion points. The only division polynomials whose roots produce points defined over $\mathbb{F}_q$ are the $\psi_d(x)$ with $d \mid 105$. This generalises to imply that the only division polynomials whose roots produce points defined over $\mathbb{F}_{q^n}$ are $\psi_d(x)$, where $d \mid \#E(\mathbb{F}_{q^n})$.

We are now in a position to shed light on Schoof's algorithm. Equation (2.6) means that computing $E(\mathbb{F}_q)$ immediately reduces to computing the (much smaller) trace of Frobenius, t. At the highest level, Schoof's idea is to compute

[3]When ℓ is even, the division polynomial is of the form $\psi_\ell(x, y) = y \cdot \tilde{\psi}_\ell(x)$ since $y = 0$ gives points of order two, which are in the ℓ-torsion.

$t_\ell \equiv t \bmod \ell$ for enough co-prime ℓ's to be able to uniquely determine t within the interval $-2\sqrt{q} \leq t \leq 2\sqrt{q}$ via the Chinese Remainder Theorem. Namely, when $\prod_\ell t_\ell \geq 4\sqrt{q}$, then we have enough relations to determine the correct t. To compute t_ℓ for various primes ℓ, Schoof looked to consider Equation (2.9) "modulo ℓ", restricting the points (x, y) to come from the ℓ-torsion, and trying to solve

$$(x^{q^2}, y^{q^2}) - [t_\ell](x^q, y^q) + [q_\ell](x, y) = \mathcal{O}, \tag{2.10}$$

for t_ℓ, where $q_\ell \equiv q \bmod \ell$. The problem for general ℓ is, that since we do not know the group order, we cannot explicitly use ℓ-torsion points in (2.10), nor do we know if they are even defined over $\mathbb{F}_q$, or where they *are* defined, so we have to work with (2.10) implicitly. Namely, we restrict (2.10) to the ℓ-torsion by working modulo $\psi_\ell(x)$: we do not work with Equation (2.10) on $E(\mathbb{F}_q)$, but rather in the polynomial ring $R_\ell = \mathbb{F}_q[x, y]/\langle \psi_\ell(x), y^2 - (x^3 + ax + b)\rangle$, where the size of the polynomials $f(x, y)$ we deal with in R_ℓ are bounded by the degrees of the division polynomials $\psi_\ell(x)$. Even for very large prime fields $\mathbb{F}_q$ of cryptographic size, the number of different primes used is small enough to keep this algorithm very practical. For example, finding the group order of the curve defined over a 256-bit prime q in Example 2.2.3 would require solving (2.10) for the 27 primes up to $\ell = 107$, at which point the product of all the primes used exceeds $4\sqrt{q}$. It is not too difficult to deduce that the asymptotic complexity of Schoof's algorithm is $O\left((\log q)^8\right)$ (see [Sil09, Ch. XI.3] for details, and further improvements).

Example 2.2.10 (Magma script). Consider $E/\mathbb{F}_{13} : y^2 = x^3 + 2x + 1$; we seek $\#E(\mathbb{F}_{13})$. Schoof's algorithm actually begins with $\ell = 3$ [Sil09, Ch. XI.3]; so since $14 < 4\sqrt{13} < 15$, we only need to solve (2.10) with $\ell = 3$ and $\ell = 5$. For $\ell = 3$, $\psi_3(x) = 3x^4 + 12x^2 + 12x + 9$, so we work in the ring $R_3 = \mathbb{F}_q[x, y]/\langle 3x^4 + 12x^2 + 12x + 9, y^2 - (x^3 + 2x + 1)\rangle$ with $q_\ell = 1$, to find that $t_3 = 0$. For $\ell = 5$, $\psi_5(x) = 5x^{12} + ... + 6x + 7$, so we work in the ring $R_5 = \mathbb{F}_q[x, y]/\langle 5x^{12} + ... + 6x + 7, y^2 - (x^3 + 2x + 1)\rangle$ with $q_\ell = 3$ to find that $t_5 = 1$. For both cases we had to compute $[q_\ell](x, y)$ in R_ℓ using the affine formulas (2.4) and (2.5), compute (x^q, y^q) and (x^{q^2}, y^{q^2}) in R_ℓ, and then test incremental values of t_ℓ until $[t_\ell](x^q, y^q)$ (also computed with the affine formulas) satisfies (2.10). The CRT with $t \equiv 0 \bmod 3$ and $t \equiv 1 \bmod 5$ gives $t \equiv 6 \bmod 15$, which combined with $-7 \leq t \leq 7$ means $t = 6$, giving $\#E = q + 1 - t = 8$.

We finish this chapter by briefly discussing one more improvement to ECC

that will essentially bring the reader up to speed with major milestones that contribute to the current state-of-the-art implementations. The technique was introduced by Gallant, Lambert and Vanstone (GLV) [GLV01], and recently generalised by Galbraith, Lin and Scott (GLS) [GLS11]. It exploits the existence of an efficiently computable endomorphism ψ that allows us to instantly move P to a large multiple $\psi(P) = [\lambda]P$ of itself, so that (in the simplest case) the scalar multiplication $[m]P$ can be split into $[m]P = [m_0]P + [m_1]\psi(P)$, where if $|m| \approx r$ (the large subgroup order), then $|m_0|$, $|m_1| \approx \sqrt{r}$. The values m_0 and m_1 are found by solving a closest vector problem in a lattice [GLV01, §4]. We apply an example from the GLV paper (which was itself taken from Cohen's book [Coh96, §7.2.3]) that is actually exploiting a special case of the endomorphism we described in Example 2.2.7.

Example 2.2.11 (Magma script). Let $q \equiv 1 \bmod 4$ be prime, $E/\mathbb{F}_q : y^2 = x^3 + ax$, and let $i^2 = -1$. The map defined by $\psi : (x, y) \mapsto (-x, iy)$ and $\psi : \mathcal{O} \mapsto \mathcal{O}$ is an endomorphism defined over $\mathbb{F}_q$ ($\psi = \zeta$ from 2.2.7). Let $P \in E(\mathbb{F}_q)$ have prime order r, then $\psi(Q) = [\lambda]Q$ for all $Q \in \langle P \rangle$, and λ is the integer satisfying $\lambda^2 = -1 \bmod r$. We give a specific example: $q = 1048589$, $E/\mathbb{F}_q : y^2 = x^3 + 2x$ with $\#E = 2r$, where $r = 524053$; we further have $i = 38993$, and $\lambda = 304425$. $P = (609782, 274272) \in E$ has $|\langle P \rangle| = r$, so we can take any element in $\langle P \rangle$, say $Q = (447259, 319154)$, and compute $\psi(Q) = (-447259, i \cdot 319154) = (601330, 117670) = [304425](447259, 319154) = [\lambda]Q$. Computing a random multiple of Q, say $[m]Q$ with $m = 103803$, can be done by decomposing m into (in this case) $(m_0, m_1) = (509, 262)$, and instead computing $[m]Q = [m_0]Q + [m_1]\psi(Q)$. Here m is 17 bits, whilst m_0 and m_1 are both 9 bits. Doing the scalar multiples $[m_0]Q$ and $[m_1]\psi(Q)$ separately would therefore give no savings, but where the GLV/GLS methods gain a substantial speed-up is in merging the doublings required in both of the multiplications by the "mini-scalars", which halves the number of doublings required overall; again, see [GLV01, GLS11] for futher details.

2.3 Chapter summary

We defined the elliptic curve group law $\oplus$ via the chord-and-tangent method, and discussed that elliptic curve groups are an attractive setting for discrete-log based cryptosystems because of the relative security obtained for the sizes of the

fields they are defined over. We also exemplified many improvements in the context of cryptographic implementations, where the fundamental operation (that creates ECDLP instances) is computing large scalar multiples $[m]P$ of $P \in E$. Namely, we showed that group law computations in finite fields can be much faster in projective coordinates, i.e. computing $(X_1 : Y_1 : Z_1) \oplus (X_2 : Y_2 : Z_2)$ rather than $(x_1, y_1) \oplus (x_2, y_2)$, and that other (non-Weierstrass) curve models also offer advantages. We gave an explicit equation for the number of points in $E(\mathbb{F}_q)$, and briefly discussed Schoof's polynomial-time algorithm that facilitates point counting on curves of cryptographic size. We also introduced the notion of the endomorphism ring $\mathrm{End}(E)$ of E, and finished by showing that non-trivial elements of $\mathrm{End}(E)$ can be used to further accelerate ECC. A reader that is comfortable with the exposition in this chapter is equipped with many of the tools required to tackle the vast literature in this field, and is somewhat up-to-date with the state-of-the-art ECC implementations. For example, in the context of chasing ECC speed records, some authors have applied alternative projective coordinate systems to the Edwards model to give very fast scalar multiplications [HWCD08], whilst others have investigated higher dimension GLV/GLS techniques (Example 2.2.11 above was 2-dimensional) to gain big speed-ups [HLX12]; visit `http://bench.cr.yp.to/supercop.html` for comprehensive and up-to-date benchmarkings of a wide number of implementations that are pushing ECC primitives to the limit.

Relaxed notation. Our last order of business before proceeding into the next chapter is to relax some notation in order to agree with the rest of the literature. Rather than writing "$\oplus$" for the elliptic curve group law, from hereon we simply use "$+$". Similarly, for the inverse of the point P, we use $-P$ instead of $\ominus P$.

Chapter 3

Divisors

In this chapter we introduce some basic language and definitions from algebraic geometry that are fundamental to the understanding of cryptographic pairing computations. We continue with our example-driven approach and illustrate each concept and definition as it arises. We will essentially just be expanding on the more concise section found in Galbraith's chapter [Gal05, §IX.2]. However, we only focus on what we need to describe elliptic curve pairings, so we refer any reader seeking a more general and thorough treatment to Galbraith's new book [Gal12, Ch.7-9]. Since our exposition targets the newcomer, we begin by assuring such a reader that their persistence through the definitions and examples will be amply rewarded. On becoming comfortable with the language of divisors, one can immediately start to appreciate how pieces of the "pairings puzzle" fit together very naturally, and might even enjoy feeling intuition behind important theorems that would otherwise appear foreign.

The following statements apply to all curves C over any perfect field K and its closure $\overline{K}$ (see [Sil09, p. 17, p. 1] for the respective definitions). However, for now we place the discussion in our context and specialise to the case where C is an elliptic curve E over a finite field $K = \mathbb{F}_q$. Later in this chapter we will expand to more general examples and statements in time to present the important theorems in their full generality. A *divisor* D on E is a convenient

way to denote a multi-set of points on E, written as the formal sum

$$D = \sum_{P \in E(\overline{\mathbb{F}}_q)} n_P(P),$$

where all but finitely many $n_P \in \mathbb{Z}$ are zero. The standard parentheses $(\cdot)$ around the P's and the absence of square parentheses $[\cdot]$ around the n_P's is what differentiates the formal sum in a divisor from an actual sum of points (i.e. using the group law) on E. The set of all divisors on E is denoted by $\mathrm{Div}_{\overline{\mathbb{F}}_q}(E)$ and forms a group, where addition of divisors is natural, and the identity is the divisor with all $n_P = 0$, the zero divisor $0 \in \mathrm{Div}_{\overline{\mathbb{F}}_q}(E)$. The *degree* of a divisor D is $\mathrm{Deg}(D) = \sum_{P \in E(\overline{\mathbb{F}}_q)} n_P$, and the *support* of D, denoted $\mathrm{supp}(D)$, is the set $\mathrm{supp}(D) = \{P \in E(\overline{\mathbb{F}}_q) : n_P \neq 0\}$.

Example 3.0.1 (Magma script). Let $P, Q, R, S \in E(\overline{\mathbb{F}}_q)$. Let $D_1 = 2(P) - 3(Q)$, and $D_2 = 3(Q) + (R) - (S)$, so that $\mathrm{Deg}(D_1) = 2 - 3 = -1$, and $\mathrm{Deg}(D_2) = 3+1-1 = 3$. The sum $D_1 + D_2 = 2(P)+(R)-(S)$, and naturally $\mathrm{Deg}(D_1+D_2) = \mathrm{Deg}(D_1) + \mathrm{Deg}(D_2) = 2$. The supports are $\mathrm{supp}(D_1) = \{P, Q\}$, $\mathrm{supp}(D_2) = \{Q, R, S\}$, and $\mathrm{supp}(D_1 + D_2) = \{P, R, S\}$.

Associating divisors with a function f on E is a convenient way to write down the intersection points (and their multiplicities) of f and E. Let $\mathrm{ord}_P(f)$ count the multiplicity of f at P, which is positive if f has a zero at P, and negative if f has a pole at P. We write the *divisor of a function* f as (f), and it is defined as the divisor

$$(f) = \sum_{P \in E(\overline{\mathbb{F}}_q)} \mathrm{ord}_P(f)(P).$$

Example 3.0.2 (Magma script). We have already seen examples of functions on E in the previous section, namely the lines $\ell : y = \lambda x + \nu$ used in the chord-and-tangent rule, and it is natural that we are really only interested in the points of intersection of ℓ and E, which is exactly what the divisor (ℓ) tells us. The chord ℓ in Figure 3.1 intersects E in P, Q and $-(P + Q)$, all with multiplicity 1, and (as we will discuss further in a moment) ℓ also intersects E with multiplicity -3 at $\mathcal{O}$, i.e. ℓ has a pole of order 3 at $\mathcal{O}$. Thus, ℓ has divisor $(\ell) = (P) + (Q) + (-(P + Q)) - 3(\mathcal{O})$. The tangent ℓ in Figure 3.2 intersects E with multiplicity 2 at P, with multiplicity 1 at $-[2]P$, and again with multiplicity -3 at $\mathcal{O}$, so in this case $(\ell) = 2(P) + (-[2]P) - 3(\mathcal{O})$. Notice that in both cases

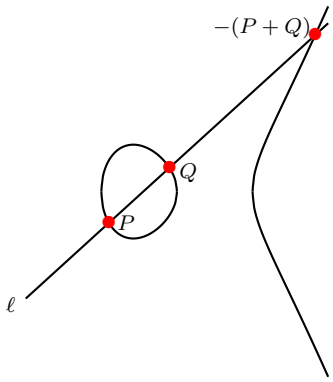


Figure 3.1: $(\ell) = (P) + (Q) + (-(P + Q)) - 3(\mathcal{O})$.

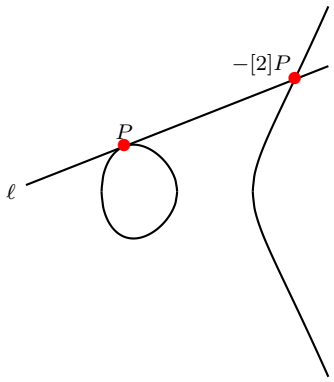


Figure 3.2: $(\ell) = 2(P) + (-[2]P) - 3(\mathcal{O})$.

we have $\mathrm{Deg}\,((\ell)) = 0$.

The balance that occurred between the zeros and poles in Example 3.0.2 that led to $\mathrm{Deg}((\ell)) = 0$ is not a coincidence. In fact, a fundamental result that lies at the heart of the discussion is that this always happens: namely, for any function f on E, we always have $\mathrm{Deg}((f)) = 0$. An instructive proof of this result is in Galbraith's book [Gal12, Th. 7.7.1], but roughly speaking this property follows from observing that the degree of the affine equation that solves for the zeros of f on E matches the degree of the projective equation that determines the multiplicity of the pole of f at $\mathcal{O}$, i.e. the projective version of f is g/h where g and h both have the same degree as f. We revisit Example 3.0.2 and illustrate in this special case.

Example 3.0.3 (Magma script). We already know that three zeros (counting multiplicities) will always arise from substituting $\ell : y = \lambda x + \nu$ into $E/\mathbb{F}_q : y^2 = x^3 + ax + b$, but we have only considered ℓ on the affine curve $E \cap \mathbb{A}^2$, where ℓ has no poles. To consider ℓ on E at $\mathcal{O} = (0 : 1 : 0)$ (in $\mathbb{P}^2(\mathbb{F}_q)$), we need to take $x = X/Z$ and $y = Y/Z$ which gives $(\frac{\lambda X+\nu Z}{Z})^2 = (\frac{X}{Z})^3 + a(\frac{X}{Z}) + b$, for which we clearly have a pole of order 3 when $Z = 0$.

The algebra between functions naturally translates across to the algebra between their divisors, so $(fg) = (f) + (g)$ and $(f/g) = (f) - (g)$, $(f) = 0$ if and only if f is constant, and thus if $(f) = (g)$, then $(f/g) = 0$ so f is a constant multiple of g, which means that the divisor (f) determines f up to non-zero scalar multiples.

Example 3.0.4 (Magma script). Let $\ell : y = \lambda_1 x + \nu_1$ be the chord (through P and

Q) with divisor $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of

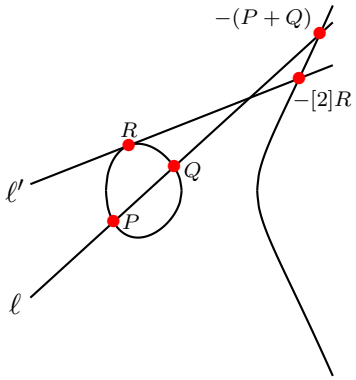


Figure 3.3: Two functions ℓ and ℓ' on E.

the function $\ell_{\text{prod}} = \ell\ell'$ is $(\ell_{\text{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P + Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\text{quot}} = \ell/\ell'$ is $(\ell_{\text{quot}}) = (\ell) - (\ell') = (P) + (Q) + (-(P+Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{quot} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{y-\lambda_1 x+\nu_1}{y-\lambda_2 x+\nu_2}$ gives $\frac{Y-\lambda_1 X+\nu_1 Z}{Y-\lambda_2 X+\nu_2 Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda_1 x - \nu_1)(y - \lambda_2 x - \nu_2)$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{x^3+ax+b+(\lambda_1 x+\nu_1)(\lambda_2 x+\nu_2)}{(\lambda_1+\lambda_2)x+\nu_1+\nu_2}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

3.1 The divisor class group

We can now start introducing important subgroups of the group of divisors $\text{Div}_{\overline{\mathbb{F}}_q}(E)$ on E. We temporarily drop the subscript, and write $\text{Div}(E)$ as the group of all divisors on E. The set of degree zero divisors $\{D \in \text{Div}(E) : \text{Deg}(D) = 0\}$ forms a proper subgroup, which we write as $\text{Div}^0(E) \subset \text{Div}(E)$. If a divisor D on E is equal to the divisor of a function, i.e. $D = (f)$, then D is called a *principal* divisor, and the set of principal divisors naturally form a group, written as $\text{Prin}(E)$. We already know (from Example 3.0.3 and the preceding discussion) that principal divisors have degree zero, but there are also degree zero divisors that are not the divisors of a function, so the degree zero subgroup is strictly larger than the principal divisors, i.e. $\text{Prin}(E) \subset \text{Div}^0(E)$.

There is, however, an extra condition on elements of $\mathrm{Div}^0(E)$ that *does* allow us to write an "if-and-only-if": $D = \sum_P n_P(P) \in \mathrm{Div}^0(E)$ is principal if and only if $\sum_P [n_P]P = \mathcal{O}$ on E [Gal05, Th. IX.2]. We illustrate this statement, and the relationship between the three groups

$$\mathrm{Prin}(E) \subset \mathrm{Div}^0(E) \subset \mathrm{Div}(E) \tag{3.1}$$

in Example 3.1.1.

Example 3.1.1 (Magma script). Consider $E/\mathbb{F}_{103} : y^2 = x^3 + 20x + 20$, with points $P = (26, 20)$, $Q = (63, 78)$, $R = (59, 95)$, $S = (24, 25)$, $T = (77, 84)$, $U = (30, 99)$ all on E. The divisor $(S) + (T) - (P) \in \mathrm{Div}(E)$ is clearly not in the subgroup $\mathrm{Div}^0(E)$, since it has degree 1; there are also infinitely many other trivial examples. The divisor $(P) + (Q) - (R) - (S)$ is in $\mathrm{Div}^0(E)$, but is not principal since $P + Q - R - S = (18, 49) \neq \mathcal{O}$ on E. Thus, a function f with $(f) = (P) + (Q) - (R) - (S)$ does not exist. On the other hand, the divisor $(P)+(Q)-(R)-(T)$ is principal, since it is degree 0 and $P+Q-R-T = \mathcal{O}$ on E. Thus, there is some function f on E such that $(f) = (P) + (Q) - (R) - (T)$; it is $f = \frac{6y+71x^2+91x+91}{x^2+70x+11}$. The sum $R+T$ on E is actually U, thus $P+Q-U = \mathcal{O}$ on E, but this time there is no function with divisor $(P) + (Q) - (U)$ because the degree of this divisor is not zero; however, we can keep the sum on E as $\mathcal{O}$ but manipulate the degree by instead taking the divisor $(P) + (Q) - (U) - (\mathcal{O})$, which must be in $\mathrm{Prin}(C)$, guaranteeing the existence of a function g with $(g) = (P)+(Q)-(U)-(\mathcal{O})$, namely $g = \frac{y+4x+82}{x+73}$. Observe the difference between f and g in projective space, where $f = \frac{6YZ+71X^2+91XZ+91Z^2}{X^2+70XZ+11Z^2}$ and $g = \frac{Y+4X+82Z}{X+73Z}$. For f, the point at infinity $\mathcal{O} = (0 : 1 : 0)$ zeros both the numerator and denominator, giving a zero and a pole which cancels out its contribution to (f), whilst for g, the point at infinity only zeros the denominator, which is why $\mathcal{O} \in \mathrm{supp}((g))$, whereas $\mathcal{O} \notin \mathrm{supp}((f))$.

Returning to the subscript notation for a moment, the three subgroups (and other related groups) in Equation (3.1) are often accompanied by the field they apply to, e.g. for a general field K, they are written as $\mathrm{Prin}_K(E)$, $\mathrm{Div}^0_K(E)$, and $\mathrm{Div}_K(E)$. Here $\mathrm{Div}_K(E) \subset \mathrm{Div}(E)$ is formally defined as the set of divisors invariant under the action of $\mathrm{Gal}(\overline{K}/K)$, where $\sigma \in \mathrm{Gal}(\overline{K}/K)$ acts on $D = \sum_P n_P(P)$ to give $D^\sigma = \sum_P n_P(\sigma(P))$, so that $D \in \mathrm{Div}_K(E)$ if $D = D^\sigma$. This is very natural in the contexts we consider, so we will continue on without subscripts.

Before we define the *divisor class group* of E, we look at the important notion of divisor equivalence in $\mathrm{Div}(E)$. We call the divisors D_1 and D_2 *equivalent*, written as $D_1 \sim D_2$, if $D_1 = D_2 + (f)$ for some function f.

Example 3.1.2 (Magma script). Consider $P = (57, 24)$, $Q = (25, 37)$, $R = (17, 32)$ and $S = (42, 35)$ on $E/\mathbb{F}_{61} : y^2 = x^3 + 8x + 1$. The divisors $D_1 = (P) + (Q) + (R)$ and $D_2 = 4(\mathcal{O}) - (S)$ are equivalent as follows. The function $f : y = 33x^2 + 10x + 24$, which intersects E at P, Q, R and S with multiplicity 1, and therefore has a pole of order 4 at infinity, has divisor $(f) = (P) + (Q) + (R) + (S) - 4(\mathcal{O})$, meaning $D_1 = D_2 + (f)$, so $D_1 \sim D_2$. Alternatively, if we did not want to find f, we could have used $D_1 - D_2 = (P) + (Q) + (R) + (S) - 4(\mathcal{O})$, which has degree zero, and computed that $P + Q + R + S - [4]\mathcal{O} = \mathcal{O}$ on E, which means $D_1 - D_2 \in \mathrm{Prin}(E)$, so that $D_1 - D_2 = (f)$ for some function f.

The *divisor class group*, or *Picard group*, of E is defined as the quotient group

$$\mathrm{Pic}^0(E) = \mathrm{Div}^0(E)/\mathrm{Prin}(E), \tag{3.2}$$

i.e. the divisor class group is the group of all degree zero divisors modulo the principal divisors on E. At first read, this notion of equivalence (modulo divisors of functions) may seem a little abstract, but once we see it in action (particularly in more general scenarios than elliptic curves), it becomes very natural. We will first use this notion to describe the elliptic curve group law in terms of divisors, following along the lines of Galbraith [Gal05, §IX.2].

Example 3.1.3 (Magma script). Referring back to Figure 2.5 (or Figure 2.6 in the case that $Q = P$), the line ℓ joining P and Q has divisor $(\ell) = (P)+(Q)+(-R)-3(\mathcal{O})$, whilst the vertical line $v = x - x_R$ has divisor $(v) = (-R) + (R) - 2(\mathcal{O})$. The quotient $\frac{\ell}{v}$ has divisor $(\frac{\ell}{v}) = (P) + (Q) - (R) - (\mathcal{O})$. Thus, the equation $R = P + Q$ on E is the same as the divisor equality $(R) - (\mathcal{O}) = (P) - (\mathcal{O}) + (Q) - (\mathcal{O}) - (\frac{\ell}{v})$, and the map of points to divisor classes $P \mapsto (P) - (\mathcal{O})$ is a group homomorphism. To concretely connect this back to Equation (3.2), both $(R) - (\mathcal{O})$ and $(P) + (Q) - 2(\mathcal{O})$ are clearly in $\mathrm{Div}^0(E)$, but they represent the same class in $\mathrm{Pic}^0(E)$, because the divisor $(\frac{l}{v}) = (P) + (Q) - (R) - (\mathcal{O})$ (which is their difference) is principal, and therefore zero in $\mathrm{Pic}^0(E)$.

3.2 A consequence of the Riemann-Roch Theorem

The notion of equivalence allows us to *reduce* divisors of any size $D \in \mathrm{Pic}^0(E)$ into much smaller divisors. We will make this statement precise after an example, but we must first define what we mean by "size". A divisor $D = \sum_P n_P(P)$ is called *effective* if $n_P \geq 0$ for all $P \in E$. The only divisor in $\mathrm{Div}^0(E)$ that is effective is the zero divisor. Thus, we define the *effective part* of a divisor D as $\epsilon(D) = \sum_P n_P(P)$, where $n_P \geq 0$. For example, the divisor $D = (P)+(Q)-2(\mathcal{O})$ is not effective, but the effective part is $\epsilon(D) = (P) + (Q)$. By the *size* of D, we mean the degree of the effective part, so in our example, although $\mathrm{Deg}(D) = 0$, it is size 2, since $\mathrm{Deg}(\epsilon(D)) = 2$.

Example 3.2.1 (Magma script). Consider the divisor $D = (P_1)+...+(P_{11})-11(\mathcal{O})$ (with $\mathrm{Deg}(\epsilon(D)) = 11$) as an element of $\mathrm{Pic}^0(E)$ on $E/\mathbb{F}_q : y^2 = x^3 + ax + b$, where the P_i are not necessarily distinct. To find a divisor that is equivalent to D, we can construct function $\ell_{10} : y = a_{10}x^{10} + ... + a_1x + a_0$ to interpolate the distinct points in $\mathrm{supp}(D)$ with appropriate multiplicities. Substituting ℓ_{10} into E gives a degree 20 polynomial in x, the roots of which reveals the 20 affine points of intersection (counting multiplicities) between ℓ_{10} and E. We already know 11 of these points (the P_i's), so let $P'_1, ...P'_9$ be the other 9. An important point to note is that these points are not necessarily defined over $\mathbb{F}_q$. Since $(\ell_{10}) = \sum_{i=1}^{11}(P_i) + \sum_{i=1}^{9}(P'_i) - 20(\mathcal{O}) \in \mathrm{Prin}(E)$, $D' = -(\sum_{i=1}^{9}(P'_i) - 9(\mathcal{O}))$ is a divisor equivalent to D in $\mathrm{Pic}^0(E)$, i.e. $D' \sim D$. We can repeat this process, interpolating the points in $\mathrm{supp}(D')$ with a degree 8 polynomial $\ell_8 : y = a'_8x^8 + ... + a'_1x + a'_0$, which will intersect E (in the affine sense) 16 times, giving 7 new intersection points, thereby finding a divisor $D'' = \sum_{i=1}^{7}(P''_i)-7(\mathcal{O})$ equivalent to D', meaning $D'' \sim D$. It is easy to infer that the number of new roots (maximum number of divisors in the consecutive supports) decreases each time by two, so that in two more steps we will arrive at $\tilde{D} = (\tilde{P}_1)+(\tilde{P}_2)+(\tilde{P}_3)-3(\mathcal{O})$. We can interpolate the three points in $\mathrm{supp}(\tilde{D})$ with a quadratic function $\tilde{\ell} : y = \tilde{a}_2x^2 + \tilde{a}_1x + \tilde{a}_0$ that clearly intersects E at one more affine point, say Q. That is, $(\tilde{\ell}) = (\tilde{P}_1) + (\tilde{P}_2) + (\tilde{P}_3) + (Q) - 4(\mathcal{O})$, and since $(\tilde{\ell}) \in \mathrm{Prin}(E)$, then $(\tilde{D}) \sim (\mathcal{O}) - (Q)$. Lastly, the vertical line $\tilde{v}$ has divisor $(\tilde{v}) = (Q) + (R) - 2(\mathcal{O})$, meaning $(\mathcal{O}) - (Q) \sim (R) - (\mathcal{O})$, which gives $(\tilde{D}) \sim (R) - (\mathcal{O})$. To summarise, we started with a divisor $D = (P_1) + ...(P_{11}) - 11(\mathcal{O})$ which had size 11, and

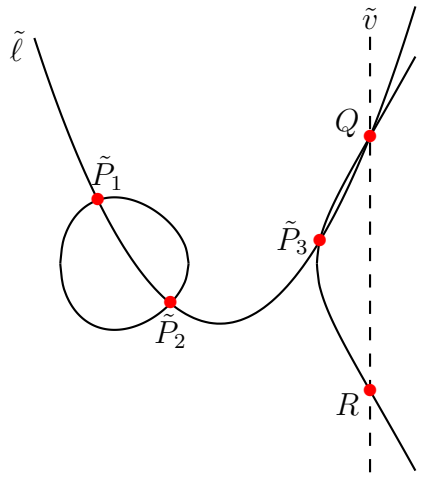


Figure 3.4: Reducing $\tilde{D}$ to $(R) - (\mathcal{O})$ in $\mathrm{Pic}^0(E)$.

reduced to the equivalent divisor $(R) - (\mathcal{O}) \sim D$ in $\mathrm{Pic}^0(E)$ which has size 1.

The above example illustrates a key consequence of one of the most central theorems in our study: the *Riemann-Roch* theorem. To present the theorem in its generality requires a few more definitions than we need for our exposition, so for the full story we refer the reader to any of [Ful08, §8], [Sil09, §II.5], [Gal12, §8.7]. The important corollary we use is the following: for any curve C, there is a unique minimal integer g, called the *genus* of C, such that any divisor $D \in \mathrm{Pic}^0(C)$ is equivalent to a divisor D' with $\mathrm{Deg}(\epsilon(D')) \leq g$. Elliptic curves E are curves of genus $g = 1$, meaning that every $D \in \mathrm{Pic}^0(E)$ can be written as $(P_1) - (Q_1)$; this is why we were able to *reduce* the divisor in Example 3.2.1 to $(R) - (\mathcal{O})$.

We will only be dealing with elliptic curves in this text, since they have proved most successful in the context of pairings, but for now it aids one's understanding to see where elliptic curves fit in a slightly broader context. Assuming an odd characteristic field, a general ("imaginary quadratic") hyperelliptic curve of genus g is a generalisation of an elliptic curve, which can be written as

$$C_g : y^2 = x^{2g+1} + f_{2g}x^{2g} + ... + f_1 x + f_0. \tag{3.3}$$

Each divisor $D \in \mathrm{Pic}^0(C_g)$ has a unique *reduced* representative of the form

$$(P_1) + (P_2) + ... + (P_n) - n(\mathcal{O}),$$

where $n \leq g$, $P_i \neq -P_j$ for all $i \neq j$, and no P_i satisfying $P_i = -P_i$ appears more than once [BBC+09, §2.3]. The following examples illustrate this in the case of

genus 2 and genus 3 respectively.

Example 3.2.2 (Magma script). A general (odd characteristic field) hyperelliptic curve of genus $g = 2$ is given (via Equation (3.3)) as $C_2 : y^2 = x^5 + f_4x^4 + ... + f_0$; we give a typical depiction in Figure 3.5. Suppose we have a divisor $D = (P_1) + (P_2) + (P_3) + (P_4) - 4(\mathcal{O}) \in \mathrm{Pic}^0(C_2)$, the affine support of which is depicted in red.

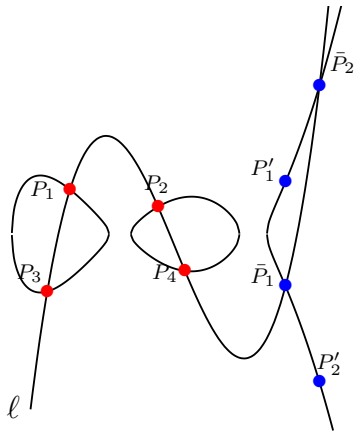


Figure 3.5: Reducing $D = \sum_{i=1}^{4}((P_i) - (\mathcal{O}))$ to $D' = \sum_{i=1}^{2}((P_i') - (\mathcal{O})) \sim D$.

The Riemann-Roch theorem guarantees a (unique) equivalent divisor of the form $(P_1') + (P_2') - 2(\mathcal{O})$. We find it by constructing the cubic function $\ell : y = a_3x^3 + ... + a_0$ that has 4 zeros corresponding to the effective part of D, and therefore 4 poles at $\mathcal{O}$. Substitution of ℓ into E reveals two more points of intersection, $\bar{P}_1$ and $\bar{P}_2$, meaning $(\ell) = (P_1)+(P_2)+(P_3)+(P_4)+(\bar{P}_1)+(\bar{P}_2)-6(\mathcal{O})$. Since $(\ell) \in \mathrm{Prin}(C_2)$, then $D = D - (\ell)$ in $\mathrm{Pic}^0(C_2)$ meaning $D \sim 2(\mathcal{O}) - (\bar{P}_1) - (\bar{P}_2)$. As usual, we reverse the ordering (so the effective part is affine) by making use of the vertical lines v_1 and v_2 with divisors $(v_1) = (\bar{P}_1) + (P_1') - 2(\mathcal{O})$ and $(v_2) = (\bar{P}_2) + (P_2') - 2(\mathcal{O})$, to write $2(\mathcal{O}) - (\bar{P}_1) - (\bar{P}_2) = 2(\mathcal{O}) - (\bar{P}_1) - (\bar{P}_2) + (v_1) + (v_2) = (P_1') + (P_2') - 2(\mathcal{O}) = D'$, meaning $D \sim D'$. We have reduced a divisor D with $\mathrm{Deg}(\epsilon(D)) = 4$ to a divisor D' with $\mathrm{Deg}(\epsilon(D')) = 2 \leq g$. Note that the points in the support of D' are not necessarily defined over $\mathbb{F}_q$. Also note that trying to reduce D' any further, say by running a line $\ell' : y = \lambda x + \nu$ through P_1' and P_2', will not work in general, since this line will intersect E in 3 more places, creating an unreduced divisor D'' with $\mathrm{Deg}(\epsilon(D'')) = 3 > g$.

Example 3.2.3 (Magma script). Consider a general genus 3 hyperelliptic curve $C_3 : y^2 = x^7 + f_6x^6 + ... + f_0$; a typical depiction is given in Figure 3.6, with a

vertically magnified Figure version in 3.7. Consider the divisor $D = \sum_{i=1}^{6}((P_i) - (\mathcal{O})) \in \mathrm{Pic}^0(C_3)$, the affine support of which is the red points in Figure 3.6.

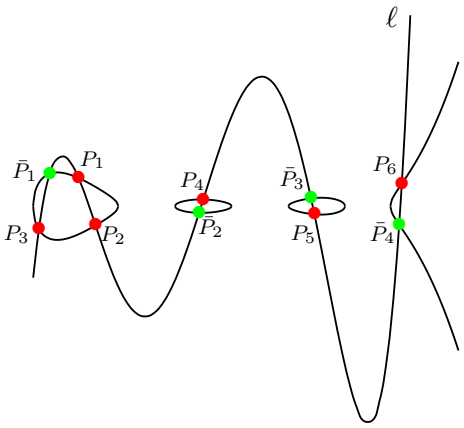


Figure 3.6: The first stage of reducing $D = \sum_{i=1}^{6}((P_i) - (\mathcal{O}))$.

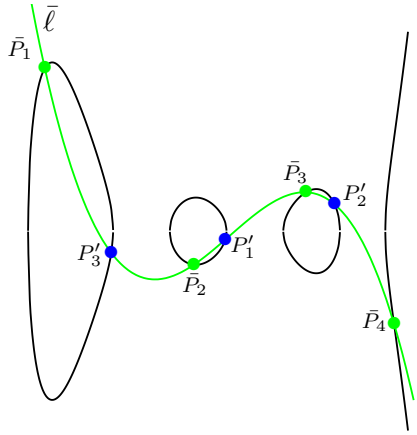


Figure 3.7: The second (and final) stage of divisor reduction.

We reduce D by determining the other points of intersection between the quintic interpolator $\ell : y = a_5x^5 + \ + a_0$ and C_3, of which there are 4: $\bar{P}_1, ..., \bar{P}_4$ depicted in green on C_3. $(\ell) = 0$ in the divisor class group so $\sum_{i=1}^{6}((P_i) - (\mathcal{O})) + \sum_{i=1}^{4}((\bar{P}_i) - (\mathcal{O})) = 0$, but the degree of the effective part of $\sum_{i=1}^{4}((\bar{P}_i) - (\mathcal{O}))$ is still larger than g, so obtaining the unique reduced divisor requires further reduction. Namely, the cubic function $\bar{\ell} : y = \bar{a}_3x^3 + ... + \bar{a}_0$ (depicted in green) interpolates the four green points and (when substituted into C_3) clearly intersects C_3 in another 3 affine points, depicted in blue. Thus, $\sum_{i=1}^{4}((\bar{P}_i) - (\mathcal{O})) + \sum_{i=1}^{3}((P'_i) - (\mathcal{O})) = 0$, which means that $D \sim D' = \sum_{i=1}^{3}((P'_i) - (\mathcal{O}))$ in the divisor class group, and D' is the unique representative of D since $\mathrm{Deg}(\epsilon(D')) = 3 \leq g$.

As mentioned prior to these higher genus examples, the reason this text will only be discussing (genus 1) elliptic curves is because in the arena of pairing-based cryptography, the raw speed of elliptic curves is currently unrivalled by their higher genus counterparts, and all of the state-of-the-art implementations take place in the genus 1 setting.

The elliptic curve group law enjoys a (relatively speaking) very simple, almost entirely elementary description, the only exception being the introduction of

projective space for the formal definition of $\mathcal{O}$. Namely, we were able to describe the chord-and-tangent rule without the language of divisors or the definition of the divisor class group, which is not the case for other curves or general abelian varieties. This is because of the one-to-one correspondence between the divisor class group $\mathrm{Pic}^0(E)$ and the points on E we briefly mentioned in Example 3.1.3, i.e. the group homomorphism $P \mapsto (P) - (\mathcal{O})$ (see [Sil09, III.3.4] [Gal12, Th. 7.9.8, Th. 7.9.9]). Thus, in the elliptic curve setting, we can simply talk about the group elements being points, rather than divisors. In higher genera this does not happen; group elements are no longer points, but rather divisor classes in $\mathrm{Pic}^0(E)$ with multiple elements in their support.

Nevertheless, as we will see in the coming chapters, the language of divisors is absolutely essential in the description of elliptic curve pairings, where the objective is to compute very large (degree) functions on E with prescribed divisors, and then evaluate these functions at other divisors[1]. Evaluating a function $f \in \mathbb{F}_q(E)$ at a divisor $D = \sum_{P \in E} n_P(P)$ has a natural definition, provided the divisors (f) and D have disjoint supports:

$$f(D) = \prod_{P \in E} f(P)^{n_P}. \tag{3.4}$$

The stipulation of disjoint supports is clearly necessary for $f(D)$ to be non-trivial, since $P \in \mathrm{supp}((f))$ implies P is a zero or pole of f on E, meaning $f(P)^{n_P}$ would be either zero or infinity respectively.

Example 3.2.4 (Magma script). Consider $E/\mathbb{F}_{163} : y^2 = x^3 - x - 2$, with $P = (43, 154)$, $Q = (46, 38)$, $R = (12, 35)$ and $S = (5, 66)$ all on E. Let $\ell_{P,Q}$, $\ell_{P,P}$ and $\ell_{Q,Q}$ be the lines joining P and Q, tangent to P, and tangent to Q on E respectively, computed as $\ell_{P,Q} : y+93x+85$, $\ell_{P,P} : y+127x+90$, $\ell_{Q,Q} : y+13x+16$. Let $D_1 = 2(R) + (S)$, $D_2 = 3(R) - 3(S)$ and $D_3 = (R) + (S) - 2(\mathcal{O})$. We can compute $\ell_{P,Q}(D_1) = (y_R + 93x_R + 85)^2(y_S + 93x_S + 85) = 122$, or $\ell_{P,P}(D_2) = (y_R + 127x_R + 90)^3/(y_S + 127x_S + 90)^3 = 53$, but we can not evaluate any of these functions at D_3, since $\mathcal{O} \in \mathrm{supp}(D_3)$, and $\mathcal{O}$ is also in the supports of $(\ell_{P,Q})$, $(\ell_{P,P})$, $(\ell_{Q,Q})$. Let $\ell'_{P,P} = 17\ell_{P,P}$ so that $\ell'_{P,P} = 17y + 40x + 63$, and that $\ell'_{P,P}(D_2) = (17y_R + 40x_R + 63)^3/(17y_S + 40x_S + 63)^3 = 53 = \ell_{P,P}(D_2)$. This is true in general, i.e. that if $g = cf$ for some constant $c \in \overline{\mathbb{F}}_q$, then $f(D) = g(D)$

[1] We will also see that we do not actually compute these very large functions explicitly before evaluating them.

if D has degree zero; the constant c will cancel out because $\mathrm{Deg}(D) = 0$ implies the numerator and denominator of $f(D)$ (identically $g(D)$) have the same total degree.

3.3 Weil reciprocity

We conclude our chapter on divisors (as Galbraith does [Gal05, §IX.2, Th. IX.3], where he also gives a proof) with a central theorem that lies at the heart of many of the proofs of cryptographic pairing properties.

Theorem 3.1 (Weil reciprocity)**.** *Let f and g be non-zero functions on a curve such that (f) and (g) have disjoint supports. Then $f((g)) = g((f))$.*

Most of the functions on E that we have seen so far contain $\mathcal{O}$ in their support. In the first example (3.3.1) we will choose one of the functions such that this is not the case, meaning that Theorem 3.1 can be applied instantly, whilst in the second example we will show how to alleviate this problem when it arises by modifying either of the functions.

Example 3.3.1 (Magma script). Let $E/\mathbb{F}_{503} : y^2 = x^3 + 1$. Consider the functions $f : \frac{20y+9x+179}{199y+187x+359} = 0$ and $g : y + 251x^2 + 129x + 201 = 0$ on E. The divisor of f is $(f) = 2(433, 98) + (232, 113) - (432, 27) - 2(127, 258)$, and the divisor of g is $(g) = (413, 369) + (339, 199) + (147, 443) + (124, 42) - 4(\mathcal{O})$. The supports are clearly disjoint, so we first compute $f((g))$ as

$$\frac{\left(\frac{20\cdot 369+9\cdot 413+179}{199\cdot 369+187\cdot 413+359}\right)\cdot\left(\frac{20\cdot 199+9\cdot 339+179}{199\cdot 199+187\cdot 339+359}\right)\cdot\left(\frac{20\cdot 443+9\cdot 147+179}{199\cdot 443+187\cdot 147+359}\right)\cdot\left(\frac{20\cdot 42+9\cdot 124+179}{199\cdot 42+187\cdot 124+359}\right)}{\left(\frac{20\cdot 1+9\cdot 0+179\cdot 0}{199\cdot 1+187\cdot 0+359\cdot 0}\right)^4} = 321.$$

Notice that f was cast into projective space as $f : \frac{20Y+9X+179Z}{199Y+187X+359Z}$ for the evaluation at $\mathcal{O} = (0 : 1 : 0)$ on the denominator. Now, for $g((f))$ we have

$$\frac{\left(98 + 251\cdot 433^2 + 129\cdot 433 + 201\right)^2\cdot\left(113 + 251\cdot 232^2 + 129\cdot 232 + 201\right)}{\left(258 + 251\cdot 127^2 + 129\cdot 127 + 201\right)^2\cdot\left(27 + 251\cdot 432^2 + 129\cdot 432 + 201\right)} = 321.$$

Example 3.3.2 (Magma script). Let $P, Q, R, S, T, U \in E$, such that $T = -(R + S)$. Further let $\ell' : y = (\lambda' x + \nu')$ be the tangent to E at P and $\ell : y = (\lambda x + \nu)$ be the line between R, S and T depicted in Figure 3.8, so that $(\ell') = 2(P) + (-[2]P) - 3(\mathcal{O})$ and $(\ell) = (R) + (S) + (T) - 3(\mathcal{O})$. Suppose we wish to compute $\ell(\ell')$.

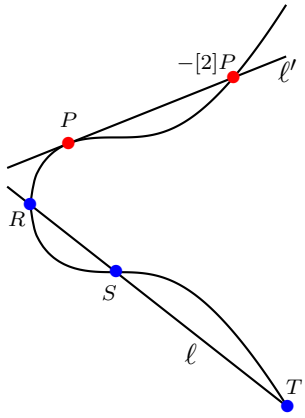


Figure 3.8: $\mathrm{supp}(\epsilon((\ell)))$ and $\mathrm{supp}(\epsilon((\ell')))$.

At this point it does not make sense to compute $\ell(\ell')$ (or $\ell'(\ell)$) since $\mathrm{supp}((\ell)) \cap \mathrm{supp}((\ell')) = \{\mathcal{O}\}$. We can fix this by finding a divisor equivalent to, say (ℓ), whose support is disjoint to $\mathrm{supp}((\ell'))$. This is easily done by picking a random point $U \notin \mathrm{supp}(\ell')$ and defining $D = (R+U)+(S+U)+(T+U)-3(U)$. To see that $D \sim \ell$, observe that $(R+U)-(U) = (R)-(\mathcal{O})$ by writing down the divisor of the quotient of the sloped and vertical lines in the addition of R and U on E. Computing $\ell(\ell')$ is therefore the same as computing $D(\ell')$, but this computation would then require finding a new function on E with divisor D, so we can invoke Theorem 3.1 and instead compute $\ell'(D)$ as

$$\ell'(D) = \frac{(y_{R'} - (\lambda' x_{R'} + \nu'))\,(y_{S'} - (\lambda' x_{S'} + \nu'))\,(y_{T'} - (\lambda' x_{T'} + \nu'))}{(y_U - (\lambda' x_U + \nu'))^3},$$

where $R' = (x_{R'}, y_{R'}) = R+U$, $S' = (x_{S'}, y_{S'}) = S+U$ and $T' = (x_{T'}, y_{T'}) = T+U$ are all such that $R', S', T' \notin \mathrm{Supp}(\ell')$, so that $\ell'(D)$ is the same as $\ell(\ell')$ by Weil reciprocity.

3.4 Chapter summary

We introduced the important concept of divisors on curves. We illustrated their particular usefulness when used to describe functions on curves, since such a function is well defined (up to constant) by its points of intersection with a curve, and these are precisely what the divisor of the function encapsulates. We

defined the *divisor class group* of a (hyperelliptic) curve and discussed that for the case of elliptic curves, there is a bijection between this group and the set of points on the curve, so that we can simply talk about group elements as points on E rather than divisors. We further illustrated several useful properties and theorems that play a big role in the realm of algebraic geometry, most notably the Riemann-Roch theorem and Weil reciprocity. For the most part we specified the context to elliptic curves over finite fields, but all of the results and properties discussed above apply to arbitrary curves over arbitrary fields.

Chapter 4

Elliptic curves as pairing groups

The purpose of this chapter is to define the elliptic groups that are used in cryptographic pairings. We start with the most abstract definition [Sil10]: a pairing is a *bilinear* map on an abelian group M taking values in some other abelian group R

$$\langle \cdot\, , \cdot \rangle : M \times M \to R.$$

Suppose that the binary group operations in M and R are respectively denoted by $+$ and $*$. The bilinearity property of the above map (that classifies it a pairing) means that, for $x, y, z \in M$, we have

$$\begin{aligned} \langle x + y, z \rangle &= \langle x, z \rangle * \langle y, z \rangle, \\ \langle x, y + z \rangle &= \langle x, y \rangle * \langle x, z \rangle. \end{aligned}$$

That is, the map $\langle \cdot, \cdot \rangle$ is linear in both inputs.

It is this bilinearity property that makes pairings such a powerful primitive in cryptography. For our purposes we often find it advantageous to slightly relax the condition that the two arguments in the map come from the same group, and allow them to come from cyclic groups of the same order (which are therefore isomorphic). Thus, in the abundance of literature related to cryptography, the

notation commonly used for the bilinear map is

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T.$$

Our primary objective in this chapter is to define the two groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The definition of $\mathbb{G}_T$ will come with the definition of the pairings in the next chapter.

Currently, the only known instantiations of pairings suitable for cryptography are the Weil and Tate pairings on divisor class groups of algebraic curves, and in the simplest and most efficient cases, on elliptic curves. Let $\mathbb{F}_{q^k}$ be some finite extension of $\mathbb{F}_q$ with $k \geq 1$. The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are defined in $E(\mathbb{F}_{q^k})$, and the *target group* $\mathbb{G}_T$ is defined in the multiplicative group $\mathbb{F}_{q^k}^*$, so we usually write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively, whilst we write $\mathbb{G}_T$ multiplicatively. Thus, for $P, P' \in \mathbb{G}_1$ and $Q, Q' \in G_2$, the bilinearity of e means that

$$\begin{aligned} e(P + P', Q) &= e(P, Q) \cdot e(P', Q), \\ e(P, Q + Q') &= e(P, Q) \cdot e(P, Q'), \end{aligned}$$

from which it follows that, for scalars $a, b \in \mathbb{Z}$, we have

$$e([a]P, [b]Q) = e(P, [b]Q)^a = e([a]P, Q)^b = e(P, Q)^{ab} = e([b]P, [a]Q). \tag{4.1}$$

Even though we are yet to define $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$, and we are still a while away from beginning the discussion of how the pairing $e(P, Q)$ is computed, it helps to immediately see the bilinearity property of pairings in context.

Example 4.0.1 (Magma script). Let $q = 7691$ and let $E/\mathbb{F}_q : y^2 = x^3 + 1$. Suppose $\mathbb{F}_{q^2}$ is constructed $\mathbb{F}_{q^2} = \mathbb{F}_q(u)$ where $u^2 + 1 = 0$. Let $P = (2693, 4312) \in E(\mathbb{F}_q)$ and $Q = (633u + 6145, 7372u + 109) \in E(\mathbb{F}_{q^2})$. $\#E(\mathbb{F}_q) = 2^2 \cdot 3 \cdot 641$ and $\#E(\mathbb{F}_{q^2}) = 2^4 \cdot 3^2 \cdot 641^2 = \#E(\mathbb{F}_q)^2$. P and Q were especially chosen (we will see why later) to be in different subgroups of the same prime order $r = |\langle P \rangle| = |\langle Q \rangle| = 641$. The Weil pairing $e(\cdot, \cdot)$ of P and Q is $e(P, Q) = 6744u + 5677 \in \mathbb{F}_{q^2}^*$. In fact, $r \mid \#\mathbb{F}_{q^2}$, and $e(P, Q)$ actually lies in a subgroup of $\mathbb{F}_{q^2}$, namely the r-th roots of unity $\mu_r \in \mathbb{F}_{q^2}$, meaning that $e(P, Q)^r = 1$. We are now in a position to illustrate some examples of bilinearity. Thus, take any $a \in \mathbb{Z}_r$ and $b \in \mathbb{Z}_r$, say $a = 403$ and $b = 135$, and see that $[a]P = (4903, 2231)$ and $[b]Q = (5806u + 1403, 6091u + 2370)$. We can compute $e([a]P, Q) = 3821u + 7025$

and verify that $e([a]P, Q) = 3821u + 7025 = (6744u + 5677)^{403} = e(P, Q)^a$; or $e(P, [b]Q) = 248u + 5$ to see that $e(P, [b]Q) = 248u + 5 = (6744u + 5677)^{135} = e(P, Q)^b$; or $e([a]P, [b]Q) = 2719u + 2731 = (6744u + 5677)^{561} = e(P, Q)^{a \cdot b \bmod r}$. Note that since $e(P, Q) \neq 1 \in \mu_r$, $e([a]P, [b]Q)$ will only be trivial if $r \mid ab$, which implies $r \mid a$ or $r \mid b$, meaning either (or both) of $[a]P$ or $[b]Q$ must be $\mathcal{O}$. Thus, $e(P, Q) \neq 1$ guarantees non-trivial pairings for non-trivial arguments; this is a cryptographically necessary property that is called *non-degeneracy*.

Following Example 4.0.1 above, if a pairing e is bilinear, non-degenerate and efficiently computable, e is called an *admissible pairing*.

Remark 4.0.1 (ECC vs. PBC). This informal remark is intended as a point of clarification for PBC newcomers. Our confusion in the early days of digesting the vast amount of literature was in part alleviated by one paragraph in Lynn's thesis that helped put the relationship between ECC and PBC in a wider context. The only known admissible pairings that are suitable for cryptography are the Weil and Tate pairings on algebraic curves. The fact that these pairings can be defined on elliptic curves, which were already a highly attractive cryptographic setting before pairings arrived on the scene, is, as Lynn puts it, a "happy coincidence". Cryptographers would have welcomed secure, admissible pairings in any suitable form, but the fact that they were handed down from the realm of algebraic geometry and are computed on elliptic curves makes them "even more attractive" [Lyn07, §2.9].

In cryptography we need more properties than the three which constitute an admissible pairing. The magic of the bilinearity property in (4.1) that gives pairing-based primitives increased functionality over traditional primitives is useless unless discrete logarithm related problems within all three groups remain intractable. Example 4.0.1 gives an admissible pairing, but because the toy sizes of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ clearly offer no resistance in regards to their respective discrete logarithm problems, such a pairing instance would clearly never be used. However, if the size r of all three groups was inflated to be much larger (say 512 bits), then the corresponding pairing could meet current security requirements and resist all known attacks. We present an alternative bilinear pairing that meets the admissible requirements, but (regardless of how large the group sizes are) is still not suitable for PBC. This example too, is taken from Lynn's thesis [Lyn07, §1.9].

Example 4.0.2 (Magma script). Let $r > 1$ be an integer. Suppose $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ has $\mathbb{G}_1 = \mathbb{G}_T = \mathbb{Z}_r^*$ and $\mathbb{G}_2 = \mathbb{Z}_{r-1}^+$, and is defined by $e : (g, a) = g^a$. Notice that for $g, g' \in \mathbb{G}_1$, we have $e(g \cdot g', a) = e(g, a) \cdot e(g', a)$, and for $a, a' \in \mathbb{G}_2$ we have $e(g, a + a') = e(g, a) \cdot e(g, a')$. Although e is then clearly bilinear, the discrete logarithm problem in $\mathbb{G}_2$ is easy, so the power of the bilinear map becomes somewhat redundant. It is interesting to see, however, that we can still state some of the classical problems in terms of the above pairing. For example, if we set r to be a large prime, then the standard discrete logarithm problem becomes: given $g \in \mathbb{G}_1$, $h \in \mathbb{G}_T$, find $a \in \mathbb{G}_2$ such that $e(g, a) = h$.

4.1 The r-torsion

We now turn our focus towards concretely defining the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Having not yet seen how pairings are computed, we will need to make some statements regarding what we need out of $\mathbb{G}_1$ and $\mathbb{G}_2$ that will really only tie together when the definitions of the Weil and Tate pairings come in the following chapter. The main such statement is that computing the pairing $e(P, Q)$, in either the Weil or Tate sense, requires that P and Q come from disjoint cyclic subgroups of the same prime[1] order r. At this point we can only hint towards why by referring back to the stipulation of disjoint supports that was made in the statement of Weil reciprocity (Theorem 3.1), and claiming that if P and Q are in the same cyclic subgroup, then the pairing computation essentially fails because supports of the associated divisors are forced to undesirably coincide.

We have already seen an example (4.0.1) of how we can find more than one cyclic subgroup of order r, when $E(\mathbb{F}_q)$ itself only contains one subgroup. Namely, we extended $\mathbb{F}_q$ to $\mathbb{F}_{q^2}$ and saw that $E(\mathbb{F}_{q^2}) \setminus E(\mathbb{F}_q)$ had at least one other subgroup of order r, where we were able to define Q and subsequently compute $e(P, Q)$. This is precisely the way we obtain two distinct order-r subgroups in general: we find the smallest extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$ such that $E(\mathbb{F}_{q^k})$ captures more points of order r. The integer $k \geq 1$ that achieves this is called the *embedding degree*, and it plays a crucial role in pairing computation. Also at the heart of our discussion then, is the entire group of points of order r on $E(\overline{\mathbb{F}}_q)$, called the r-torsion, which is denoted by $E[r]$ and defined as $E[r] = \{P \in E : [r]P = \mathcal{O}\}$.

[1] There has been some work that exploits additional functionality if r is composite, e.g. an RSA modulus $n = pq$, but we do not consider this much less common and much less efficient setting – see [BGN05, Fre10, BRS11, Lew12] for more details.

The following result (see [ACD+05, Th. 13.13] or [Sil09, Ch. III, Cor. 6.4(b)]) is quite remarkable; it tells us not only the cardinality of $E[r]$, but its structure too. If K is any field with characteristic zero or prime to r, we have

$$E[r] \cong \mathbb{Z}_r \times \mathbb{Z}_r. \tag{4.2}$$

This means that in general, $\#E[r] = r^2$. Furthermore, since the point at infinity $\mathcal{O}$ overlaps into all order r subgroups, Equation (4.2) implies that (for prime r) the r-torsion consists of $r+1$ cyclic subgroups of order r. The following equivalent conditions for the embedding degree k also tell us precisely where $E[r]$ lies in its entirety. We note that the embedding degree is actually a function $k(q, r)$ of q and r, but we just write k since the context is usually clear.

- k is the smallest positive integer such that $r \mid (q^k - 1)$;
- k is the smallest positive integer such that $\mathbb{F}_{q^k}$ contains all of the r-th roots of unity in $\overline{\mathbb{F}}_q$ (i.e. $\mu_r \subset \mathbb{F}_{q^k}$);
- k is the smallest positive integer such that $E[r] \subset E(\mathbb{F}_{q^k})$.

If $r \| \#E(\mathbb{F}_q)$ (i.e. $r \mid \#E(\mathbb{F}_q)$ but $r^2 \nmid \#E(\mathbb{F}_q)$), then the r-torsion subgroup in $E(\mathbb{F}_q)$ is unique. In this case, $k > 1$ and (4.2) implies that $\mathbb{F}_{q^k}$ is the smallest field extension of $\mathbb{F}_q$ which produces any more r-torsion points belonging to $E(\mathbb{F}_{q^k}) \setminus E(\mathbb{F}_q)$. In other words, once the extension field is big enough to find one more point of order r (that is not defined over the base field), then we actually find all of the points in $E[r] \cong \mathbb{Z}_r \times \mathbb{Z}_r$. Scott [Sco04] describes this phenomenon more poetically:

> "... something rather magical happens when a curve with the same equation is considered over the field $\mathbb{F}_{q^k}$ for a certain value of k. The group structure undergoes a strange blossoming, and takes on a new, more exotic character."

We also find Scott's depiction of the torsion subgroup $E[r]$ especially instructive [Sco04, Sco07a], so we use it in the following examples and throughout the rest of this chapter.

Example 4.1.1 (Magma script). Let $q = 11$, and consider $E/\mathbb{F}_q : y^2 = x^3 + 4$. $E(\mathbb{F}_q)$ has 12 points, so take $r = 3$ and note (from Equation (4.2)) that there are 9 points in the 3-torsion. Only 3 of them are found in $E(\mathbb{F}_q)$, namely $(0, 2)$,

$(0, 9)$ and $\mathcal{O}$, which agrees with the fact that the embedding degree $k \neq 1$, since $(q^1 - 1) \not\equiv 0 \bmod r$. However, $(q^2 - 1) \equiv 0 \bmod r$ which means that the embedding degree is $k = 2$, so we form $\mathbb{F}_{q^2} = \mathbb{F}_q(u)$, with $u^2 + 1$. Thus, we are guaranteed to find the whole 3-torsion in $E(\mathbb{F}_{q^2})$, and it is structured as 4 cyclic subgroups of order 3; $\mathcal{O}$ overlaps into all of them – see Figure 4.1. We point out that although $\mathcal{O}$ is in the 3-torsion, it does not have order 3, but rather order 1 – points of order $d \mid r$ are automatically included in the r-torsion. Take any two

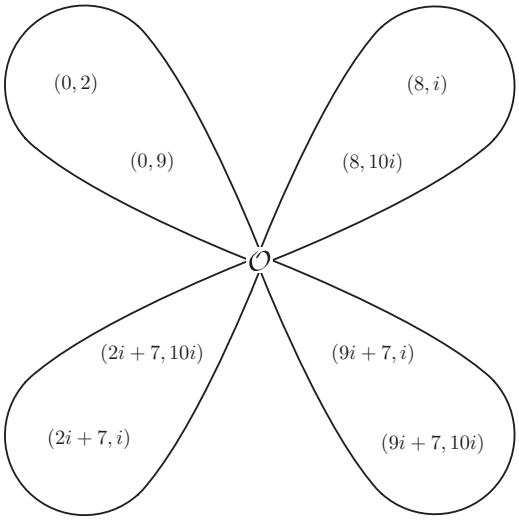


Figure 4.1: The 3-torsion: $E[3]$.

points $P, Q \in E[3] \setminus \{\mathcal{O}\}$ that are not in the same subgroup, neither of which are $\mathcal{O}$. The translation of Equation (4.2) is that any other point in $E[3]$ can be obtained as $[i]P + [j]Q$, $i, j \in \{0, 1, 2\}$. Fixing $P \neq \mathcal{O}$ and letting j run through $0, 1, 2$ lands $P + [j]Q$ in the other three subgroups of $E[3]$ (that are not $\langle Q \rangle$ – this corresponds to $P = \mathcal{O}$).

Example 4.1.2 (Magma script). In the rare case that $r^2 \mid \#E$, it is possible that the entire r-torsion can be found over $E(\mathbb{F}_q)$, i.e. that the embedding degree is $k = 1$. Consider $E/\mathbb{F}_{31} : y^2 = x^3 + 13$, which has 25 points, so take $r = 5$. Since $r \mid q - 1$, $k = 1$ and therefore $E[r] \subseteq E(\mathbb{F}_q)$; Figure 4.2 show the 6 cyclic subgroups of order 5 constituting $E[5] \cong \mathbb{Z}_5 \times \mathbb{Z}_5$. Of course, $r^2 \mid \#E(\mathbb{F}_q)$ does not necessarily imply that $E[r] \subseteq E(\mathbb{F}_q)$, as points of order r^2 are possible.

Before the next example, we introduce an important map that plays an intricate role within the r-torsion subgroups. Since we are working over finite extension fields of $\mathbb{F}_q$, it is natural that we find a useful contribution from Galois

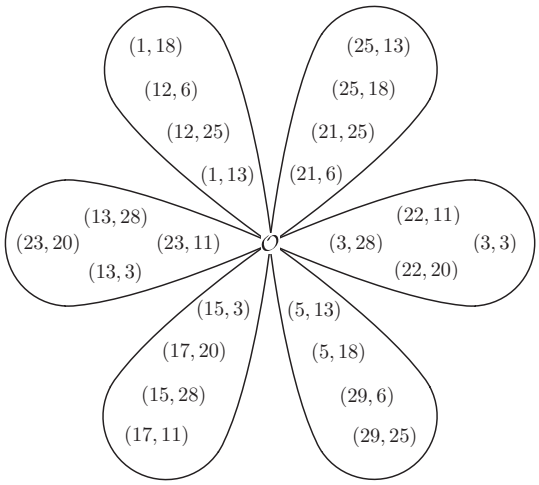


Figure 4.2: The 5-torsion: $E[5]$.

theory. Namely, the *trace map* of the point $P = (x, y) \in E(\mathbb{F}_{q^k})$ is defined as

$$\mathrm{Tr}(P) = \sum_{\sigma \in \mathrm{Gal}(\mathbb{F}_{q^k}/\mathbb{F}_q)} \sigma(P) = \sum_{i=0}^{k-1} \pi^i(P) = \sum_{i=0}^{k-1} (x^{q^i}, y^{q^i}),$$

where π is the q-power Frobenius endomorphism defined in Equation (2.7). Galois theory tells us that $\mathrm{Tr} : E(\mathbb{F}_{q^k}) \to E(\mathbb{F}_q)$, so when $r \| \#E(\mathbb{F}_q)$ (which will always be the case from now on), then this map, which is actually a group homomorphism, sends all torsion points into one subgroup of the r-torsion. We illustrate in Example 4.1.3 before painting the general picture.

Example 4.1.3 (Magma script). We take $q = 11$ again, but this time with $E/\mathbb{F}_q : y^2 = x^3+7x+2$. $E(\mathbb{F}_q)$ has 7 points, so take $r = 7$. We already have $E(\mathbb{F}_q)[r]$, but to collect $E[r]$ in its entirety we need to extend $\mathbb{F}_q$ to $\mathbb{F}_{q^k}$. This time, the smallest integer k such that $(q^k - 1) \bmod 7 \equiv 0$ is $k = 3$, so we form $\mathbb{F}_{q^3} = \mathbb{F}_q(u)$ with $u^3 + u + 4 = 0$, and we are guaranteed that $E[7] \subset E(\mathbb{F}_{q^3})$. The entire 7-torsion has cardinality 49 and splits into 8 cyclic subgroups, as shown in Figure 4.2. To fit the points in, we use the power representation of elements in $\mathbb{F}_{q^3} = \mathbb{F}_q(u)$. In this case, for $P \in E(\mathbb{F}_{q^3})$, the trace map on E is $\mathrm{Tr}(P) = (x, y) + (x^q, y^q) + (x^{q^2}, y^{q^2})$. For the unique torsion subgroup $E(\mathbb{F}_q)[r]$, the Frobenius endomorphism is trivial ($\pi(P) = P$) so the trace map clearly acts as multiplication by k, i.e. $\mathrm{Tr}(P) = [k]P$. However, Tr will send every other element in the torsion into $E(\mathbb{F}_q)[r]$. For example, for $Q = (u^{481}, u^{1049})$ (in the subgroup pointing upwards), we have

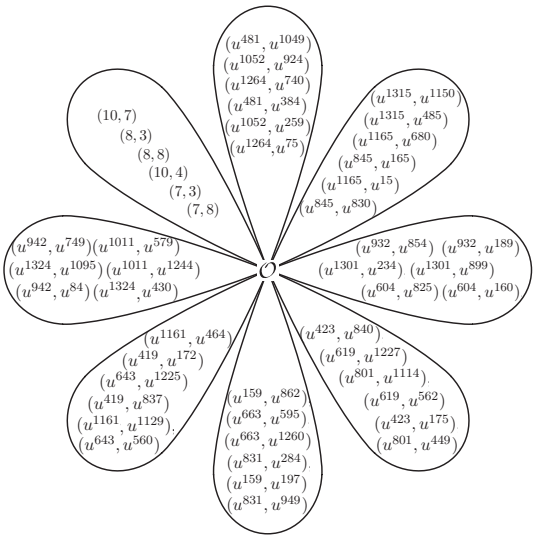


Figure 4.3: The 7-torsion: $E[7]$.

$\mathrm{Tr}(Q) = (8, 8)$; for $R = (u^{423}, u^{840})$ (the lower right subgroup), we have $\mathrm{Tr}(R) = (10, 7)$; for $S = (u^{1011}, u^{1244})$, we have $\mathrm{Tr}(S) = (8, 3)$. There is one other peculiar subgroup in $E[7]$ however, for which the trace map sends each element to $\mathcal{O}$. This occurs in general, and we are about to see that this has important consequences in PBC, but in our case this subgroup is the upper right group containing $T = (u^{1315}, u^{1150})$, i.e. $\mathrm{Tr}(T) = \mathcal{O}$, so $\mathrm{Tr} : \langle T \rangle \to \{\mathcal{O}\}$. One final point to note is that the embedding degree $k = 3$ also implies that the (six) non-trivial 7-th roots of unity are all found in $\mathbb{F}_{q^3}$ (but not before), i.e. $\mu_7 \setminus \{1\} \in \mathbb{F}_{q^3} \setminus \mathbb{F}_{q^2}$.

We now give a general depiction of the r-torsion $E[r]$. To do so, we need to discuss a few assumptions that apply most commonly to the scenarios we will be encountering. Firstly, we assume that $r \| \#E(\mathbb{F}_q)$ is prime and the embedding degree k (with respect to r) is $k > 1$. Thus, there is a unique subgroup of order r in $E[r]$ which is defined over $\mathbb{F}_q$, called the *base-field* subgroup; it is denoted by $\mathcal{G}_1$. Since the Frobenius endomorphism π acts trivially on $\mathcal{G}_1$, but nowhere else in $E[r]$, then it can be defined as $\mathcal{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$. That is, $\mathcal{G}_1$ is the $[1]$-*eigenspace* of π restricted to $E[r]$. There is another subgroup of $E[r]$ that can be expressed using an eigenspace of π. Referring back to Equation (2.8), we can easily deduce that the other eigenvalue of π is q, and we define another subgroup $\mathcal{G}_2$ of $E[r]$ as $\mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$. It turns out that this subgroup is precisely the peculiar subgroup we alluded to in Example 4.1.3. We call $\mathcal{G}_2$ the *trace zero* subgroup, since all $P \in \mathcal{G}_2$ have $\mathrm{Tr}(P) = \mathcal{O}$; this result is attributed

to Dan Boneh [Gal05, Lemma IX.16]. We illustrate in Figure 4.4.

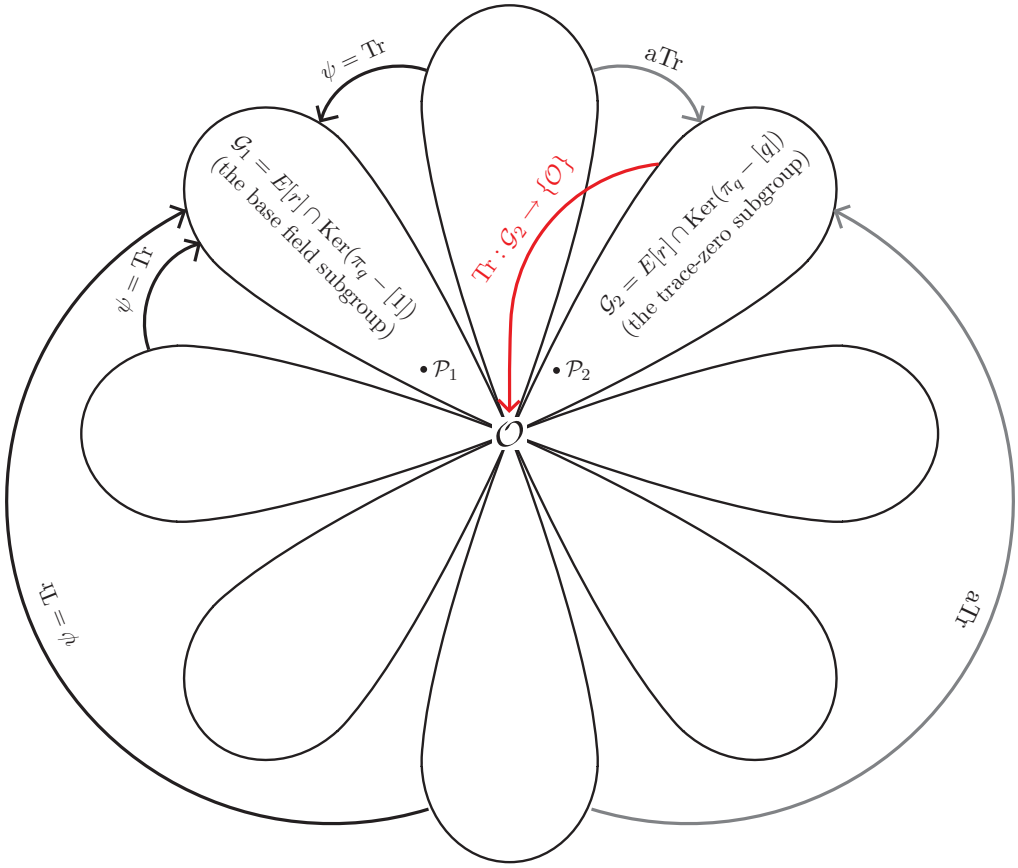


Figure 4.4: The behaviour of the trace and anti-trace maps on $E[r]$.

We can also map any $P \in E[r]$ to the trace zero subgroup $\mathcal{G}_2$ via the *anti-trace map* $\mathrm{aTr} : P \mapsto P' = [k]P - \mathrm{Tr}(P)$; showing that $\mathrm{Tr}(P') = \mathcal{O}$ is a worthwhile exercise for the reader.

To define our pairing, we need to specify the two groups $\mathbb{G}_1$ and $\mathbb{G}_2$: these $\mathbb{G}$'s are not to be confused with the $\mathcal{G}$'s that stand for two specific r-torsion subgroups, as $\mathbb{G}_1$ and $\mathbb{G}_2$ can be defined as any of the $r + 1$ groups in $E[r]$. As we will see however, there are many reasons we would like to specifically set $\mathbb{G}_1 = \mathcal{G}_1$ and $\mathbb{G}_2 = \mathcal{G}_2$, but as we will also see there are reasons that we may not want this to be the case. The existence of maps to and from the different torsion subgroups affects certain functionalities that cryptographers desire in a pairing-based protocol. These functionalities and the choices that are available to us will be discussed in a moment, but we must first look at one last map that

is available for a special class of curves.

Over prime fields, we call an elliptic curve E *supersingular*[2] if $\#E(\mathbb{F}_q) = q+1$. There are several other equivalent conditions [Sil09, Ch. V, Th. 3.1(a)], but the most meaningful property for our purposes is that a supersingular curve comes equipped with a *distortion map* ϕ; this is a non-($\mathbb{F}_q$-)rational map that takes a point in $E(\mathbb{F}_q)$ to a point in $E(\mathbb{F}_{q^k})$ [Gal05, §IX.7.2]. A curve which is not supersingular is called an *ordinary* curve, and it does not have such a map [Ver01, Th. 11]. We give two examples of elliptic curves that are supersingular, and show the behaviour of the distortion map ϕ within the torsion.

Example 4.1.4 (Magma script). Let $q = 59$, for which $E/\mathbb{F}_q : y^2 = x^3 + 1$ is supersingular, meaning $\#E(\mathbb{F}_q) = q + 1 = 60$, so take $r = 5$. The embedding degree is $k = 2$, so we construct the extension as $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$, $i^2 + 1 = 0$. $\xi_3 = 24i + 29$ is a cube root of unity, for which the associated distortion map is $\phi : (x, y) \mapsto (\xi_3 x, y)$. The fact that ϕ^3 is equivalent to the identity map on E is illustrated in Figure 4.5.

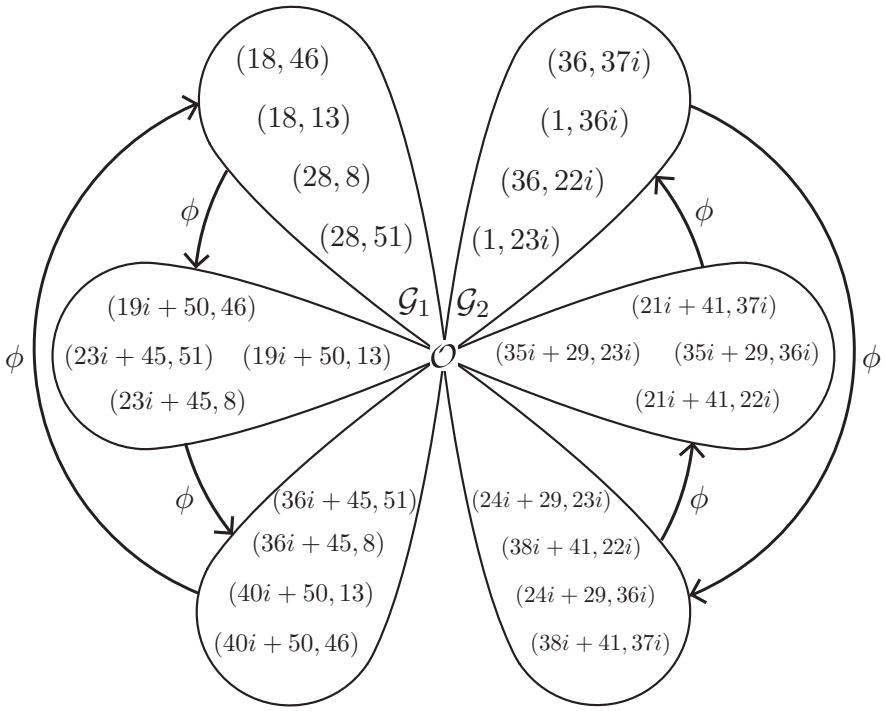


Figure 4.5: The distortion map $\phi : (x, y) \mapsto (\xi_3, y)$ on $E[5]$.

[2]This terminology should not be confused with the *singular* vs. *non-singular* definitions illustrated in, and discussed above, Figures 2.1-2.4.

Example 4.1.5 (Magma script). We take the same fields as the last example ($q = 59$, $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$, $i^2 + 1 = 0$), but instead use the supersingular curve $E/\mathbb{F}_q : y^2 = x^3 + x$, which therefore also has $\#E(\mathbb{F}_q) = 60$. This time, the distortion map is $\phi : (x, y) \mapsto (-x, iy)$, from which it is easy to see that ϕ^4 is equivalent to the identity map on E. In Figure 4.6, we see that (in this case) the

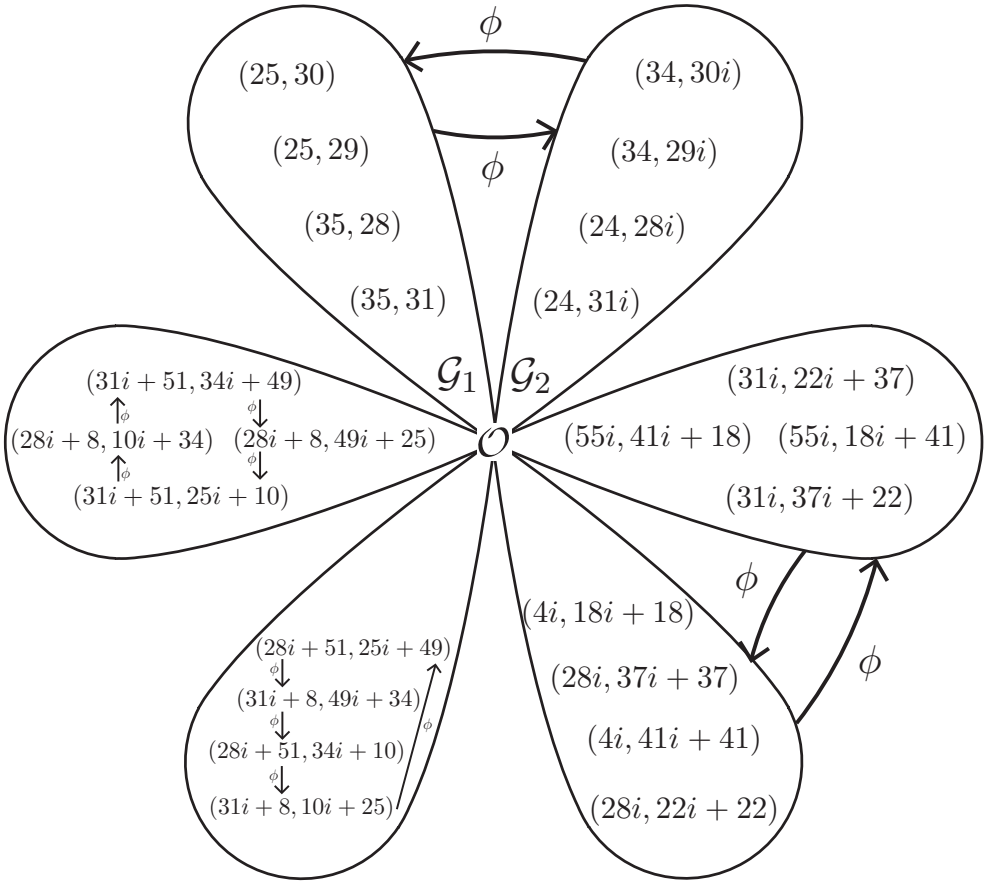


Figure 4.6: The distortion map $\phi : (x, y) \mapsto (-x, iy)$ on $E[5]$.

distortion map does not always move elements out of their subgroup, but rather restricting ϕ to, say the torsion subgroup generated by $(28i + 51, 25i + 49)$, gives an endomorphism on $\langle(28i + 51, 25i + 49)\rangle$. This hints towards one of the major optimisations in pairing computations. Namely, in Chapter 2 we saw the power of endomorphisms applied to ECC (specifically in Example 2.2.11), and in Chapter 7 we are going to see that endomorphisms on torsion subgroups (like the one above) can be used to great effect in PBC.

We summarise the available maps within the r-torsion. From any subgroup

in $E[r]$ that is not $\mathcal{G}_1$ or $\mathcal{G}_2$, we can always map into either $\mathcal{G}_1$ or $\mathcal{G}_2$ via the trace and anti-trace maps respectively. If E is ordinary, we do not have computable maps out of $\mathcal{G}_1$ or $\mathcal{G}_2$, otherwise if E is supersingular then the distortion map ϕ is a homomorphic map out of these two subgroups.

4.2 Pairing types

As we mentioned before the previous two examples, the interplay between the maps that are available in any given scenario gives rise to different functionalities within a pairing-based protocol. Galbraith *et al.* [GPS08] were the first to identify that all of the potentially desirable properties in a protocol cannot be achieved simultaneously, and therefore classified pairings into certain *types*. There are now four pairing types in the literature; Galbraith *et al.* originally presented three, but a fourth type was added soon after by Shacham [Sha05]. The pairing types essentially arise from observing the (practical) implications of placing $\mathbb{G}_1$ and $\mathbb{G}_2$ in different subgroups of $E[r]$; in fact, it will soon become obvious that it is always best to set $\mathbb{G}_1 = \mathcal{G}_1$, so the four types really are tied to the definition of $\mathbb{G}_2$. The main factors affecting the classification are the ability to hash and/or randomly sample elements of $\mathbb{G}_2$, the existence of an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ which is often required to make security proofs work (see [GPS08]), and (as always) issues concerning storage and efficiency.

We follow the notation and descriptions of Chen *et al.* [CCS07], and describe each pairing type in turn. The illustrations of each type are in Figures 4.7-4.10, where the base-field group $\mathcal{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ with generator $\mathcal{P}_1$ is always in the top left, whilst the trace-zero subgroup $\mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$ with generator $\mathcal{P}_2$ is always in the top right. Let P_1 be the generator of $\mathbb{G}_1$ and P_2 be the generator of $\mathbb{G}_2$. It should be born in mind that the pairing $e(P, Q)$ will only compute non-trivially if P and Q are in different subgroups.

- *Type 1 pairings.* This is the scenario where E is supersingular, meaning we can map out of $\mathcal{G}_1$ with ϕ. Thus, we set $\mathbb{G}_1 = \mathbb{G}_2 = \mathcal{G}_1$ (with $P_1 = P_2 = \mathcal{P}_1$). When it comes time to compute a pairing e between say P and Q, we can use ϕ to map Q to $\phi(Q)$ and define $e(P, Q) = \hat{e}(P, \phi(Q))$, where $\hat{e}$ is the Weil or Tate pairing. There are no hashing problems (getting into $E(\mathbb{F}_q)[r]$ requires a simple cofactor multiplication once we have hashed into $E(\mathbb{F}_q)$) and we trivially have an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$. The drawback

of Type 1 pairings comes when considering bandwidth and efficiency: as we will see in Chapter 6, the condition that E be supersingular is highly restrictive when it comes to optimising the speed of computing the pairing. See Figure 4.7.

The remaining three cases are defined over ordinary elliptic curves, so (as we will again see in Chapter 6) there are no restrictions imposed on the choice of elliptic curve that lead to a loss of efficiency. For all these situations we have $\mathbb{G}_1 = \mathcal{G}_1$ and $P_1 = \mathcal{P}_1$ (where hashing is relatively easy), so we only need to discuss the choices for $\mathbb{G}_2$ and P_2.

- *Type 2 pairings.* In this situation we take $\mathbb{G}_2$ to be any of the $r-1$ subgroups in $E[r]$ that is not $\mathcal{G}_1$ or $\mathcal{G}_2$. We have the map $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ as the trace map Tr. We can also use the anti-trace map to move elements from $\mathbb{G}_2$ into $\mathcal{G}_2$ for efficiency purposes. The drawback is that there is no known way of hashing into $\mathbb{G}_2$ specifically, or to generate random elements of $\mathbb{G}_2$. The best we can do here is to specify a generator $P_2 \in \mathbb{G}_2$ and generate elements via scalar multiplications of P_2, but this is often undesirable in protocols since we cannot generate random elements without knowing the discrete logarithm with respect to P_2. See Figure 4.8.

- *Type 3 pairings.* In this scenario we take $\mathbb{G}_2 = \mathcal{G}_2$, the trace zero subgroup. We can now hash into $\mathbb{G}_2$, at the very least by following a cofactor multiplication in $E(\mathbb{F}_{q^k})$ by the anti-trace map aTr : $E[r] \to \mathcal{G}_2$ (we will soon see that there is a much more efficient way than this). The ironic drawback here is that the only subgroup (besides $\mathcal{G}_1$) that we can hash into is also the only subgroup we can not find a map out of. An isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ trivially exists, we just do not have an efficient way to compute it. Thus, security proofs that rely on the existence of such a ψ are no longer applicable, unless the underlying problem(s) remains hard when the adversary is allowed oracle access to ψ [SV07]. See Figure 4.9.

- *Type 4 pairings.* In this situation we take $\mathbb{G}_2$ to be the whole r-torsion $E[r]$, which is a group of order r^2. Hashing into $\mathbb{G}_2$ is possible, but not very efficient, however we cannot hash into the particular subgroup generated by any specific P_2 (i.e. $\mathbb{G}_2$ is not cyclic). Note that hashing into $E[r]$ will only give an element in $\mathcal{G}_1$ or $\mathcal{G}_2$ (which is undesirable in this case) with negligibly low probability for large r. See Figure 4.10.

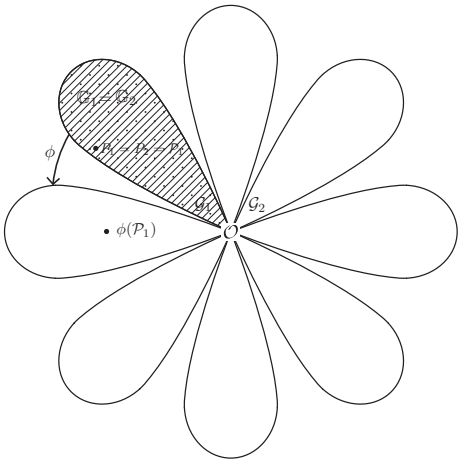


Figure 4.7: Type 1 pairings.

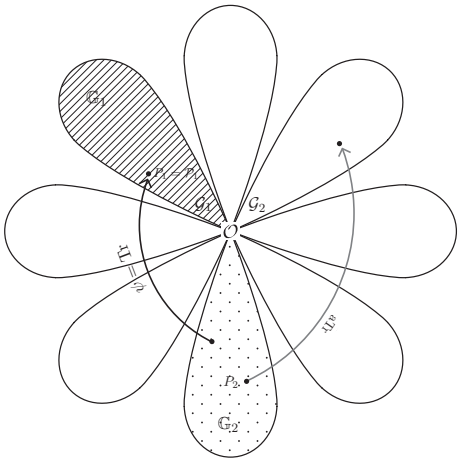


Figure 4.8: Type 2 pairings.

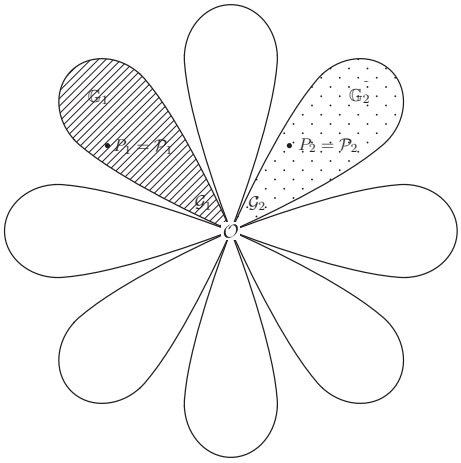


Figure 4.9: Type 3 pairings.

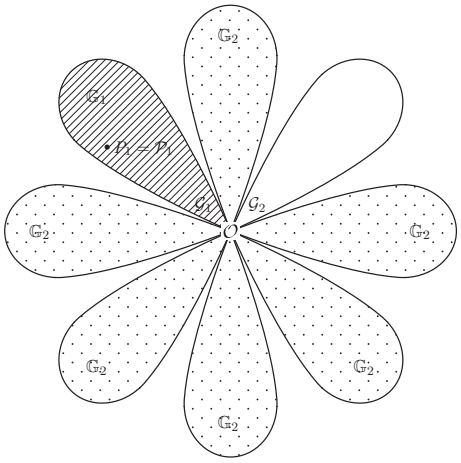


Figure 4.10: Type 4 pairings.

Prior to these different situations being brought to the attention of the PBC community [GPS08], authors publishing pairing-based protocols were often incorrectly assuming combinations of the associated properties that could not be achieved in practice. The message to designers of pairing-based protocols was that individual attention is required to prescribe the pairing type which best suits any particular pairing instantiation. Whilst some authors have since followed this advice closely, a good example being [CCS07, Tables 1-6], it still seems most common that designers of pairing protocols take the easy way out and assume a Type 1 pairing. This approach is somewhat justified, as it allows cryptographers to avoid getting bogged down in the complex details of pairings whilst still enjoying all their functional properties, but overall it is less than sat-

isfactory. The reason is that, at current levels of security, a Type 1 pairing is orders of magnitude more costly than say, a Type 3 pairing. Nowadays all of the state-of-the-art implementations of pairings take place on ordinary curves that assume the Type 3 scenario, where the only potential[3] sacrifice is the map $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Moreover, Chatterjee and Menezes [CM09] paid closer attention to the role of ψ in protocol (proof) designs and essentially argue that there is no known protocol/proof of security that cannot be translated into the Type 3 setting, claiming that Type 2 pairings (which are less efficient but have ψ) are merely inefficient implementations of Type 3 pairings. We note that their claim is only based on empirical evidence; they posed a counter-example as an open problem. Nevertheless, the final message of Menezes' related ECC2009 talk is that "protocol designers who are interested in the performance of their protocols should describe and analyse their protocols using Type 3 pairings" [Men09].

For the remainder of this text then, and unless otherwise stated, the reader should assume we are in the Type 3 scenario where $\mathbb{G}_1 = \mathcal{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ and $\mathbb{G}_2 = \mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$.

4.3 Twisted curves

Before moving our focus to the algorithm for computing pairings, we have one final point to discuss; namely, the most efficient way to hash to, and represent elements in $\mathbb{G}_2$. This discussion brings up the crucial notion of *twists* of elliptic curves, which was first applied to pairings by Barreto *et al.* [BLS03]. We start with an example.

Example 4.3.1 (Magma script). Recall the curve used in Example 4.1.1: $q = 11$, $E/\mathbb{F}_q : y^2 = x^3 + 4$, $\#E(\mathbb{F}_q) = 12$ and $r = 3$. Excluding $\mathcal{O}$, the trace zero subgroup $\mathcal{G}_2$ consists of points defined in $E(\mathbb{F}_{q^2})$, namely $(8, i)$ and $(8, 10i)$. Define the curve $E'/\mathbb{F}_q : y^2 = x^3 - 4$ and observe that the map Ψ^{-1} defined by $\Psi^{-1} : (x, y) \mapsto (-x, iy)$ takes points from E to E', i.e. $\Psi^{-1} : E \to E'$. Restricting Ψ^{-1} to $\mathcal{G}_2$ actually gives a map that takes elements defined over $\mathbb{F}_{q^2}$ to elements defined over $\mathbb{F}_q$: $\Psi^{-1}((8, i)) = (3, 10)$ and $\Psi^{-1}((8, 10i)) = (3, 1)$. The convention is to write Ψ for the reverse map $\Psi : E' \to E$ which in this case is defined by $\Psi : (x', y') \mapsto (-x', y'/i) = (-x', -y'i)$. We call E' a *twist* of E. Every twist

[3]The are some protocols whose security actually relies on the inability to compute ψ efficiently.

has a degree d, which tells us the extension field of $\mathbb{F}_q$ where E and E' become isomorphic. For our purposes, d is also the degree of its field of definition of E' as a subfield of $\mathbb{F}_{q^k}$, i.e. a degree d twist E' of E will be defined over $\mathbb{F}_{q^{k/d}}$. In this example, $k = 2$ and E' is defined over $\mathbb{F}_q$, so we are using a $d = 2$ twist, called a *quadratic twist*. Ordinarily, computations in the group $\mathbb{G}_2 = \mathcal{G}_2$ would

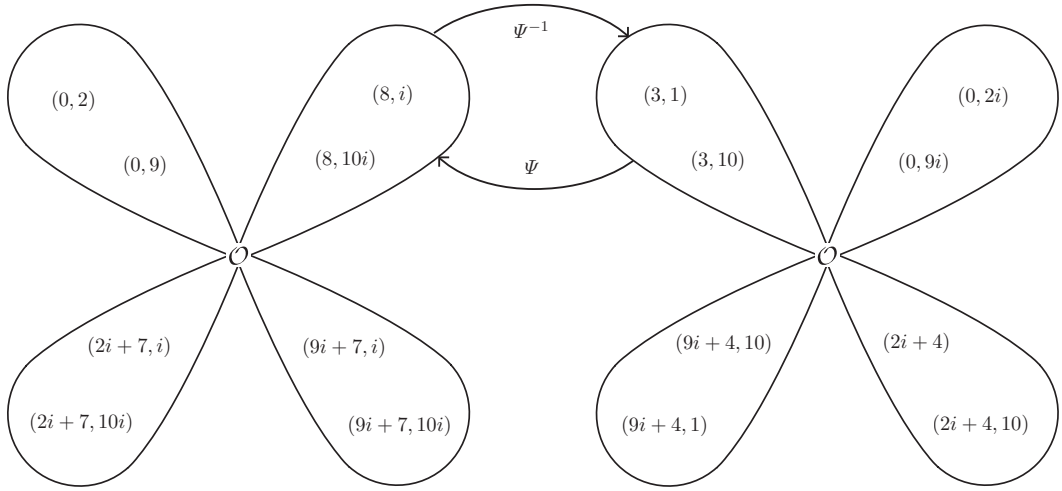


Figure 4.11: E (left) and the quadratic twist E' (right).

require (point doubling/addition) operations in the extension field $\mathbb{F}_{q^2}$, but we can use Ψ^{-1} to instead perform these operations in $E'(\mathbb{F}_q)$, before mapping the result back with Ψ. Moreover, if we restrict the maps to $E[r]$, then Ψ^{-1} takes elements of the trace zero subgroup $\mathcal{G}_2$ of E and moves them to the base field subgroup $\mathcal{G}_1'$ of E'. Note that computing Ψ and Ψ^{-1} is essentially cost free.

We give a larger example that better illustrates the power of employing twisted curves.

Example 4.3.2 (Magma script). Let $q = 103$ and consider $E/\mathbb{F}_q : y^2 = x^3 + 72$, which has $\#E(\mathbb{F}_q) = 84$, so let $r = 7$. The embedding degree (with respect to r) is $k = 6$, so form $\mathbb{F}_{q^6} = \mathbb{F}_q(u)$ with $u^6 + 2 = 0$. The trace zero subgroup $\mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$ is defined over $\mathbb{F}_{q^6}$, and is generated by $(35u^4, 42u^3)$ (see Figure 4.12). We define the degree $d = 6$ *sextic twist* E' of E as $E' : y^2 = x^3 + 72u^6$, where the back-and-forth isomorphisms are defined as $\Psi : E' \to E$, $(x', y') \mapsto (x'/u^2, y'/u^3)$ and $\Psi : E \to E'$, $(x, y) \mapsto (u^2 x, u^3 y)$. Observe that Ψ^{-1} maps elements in $\mathcal{G}_2 \in E(\mathbb{F}_{q^k})[r] = E(\mathbb{F}_{q^6})[r]$ to elements in $E'(\mathbb{F}_{q^{k/d}})[r] = E'(\mathbb{F}_q)[r]$. Thus, when performing group operations in $\mathbb{G}_2 = \mathcal{G}_2$, we gain the advantage of working over

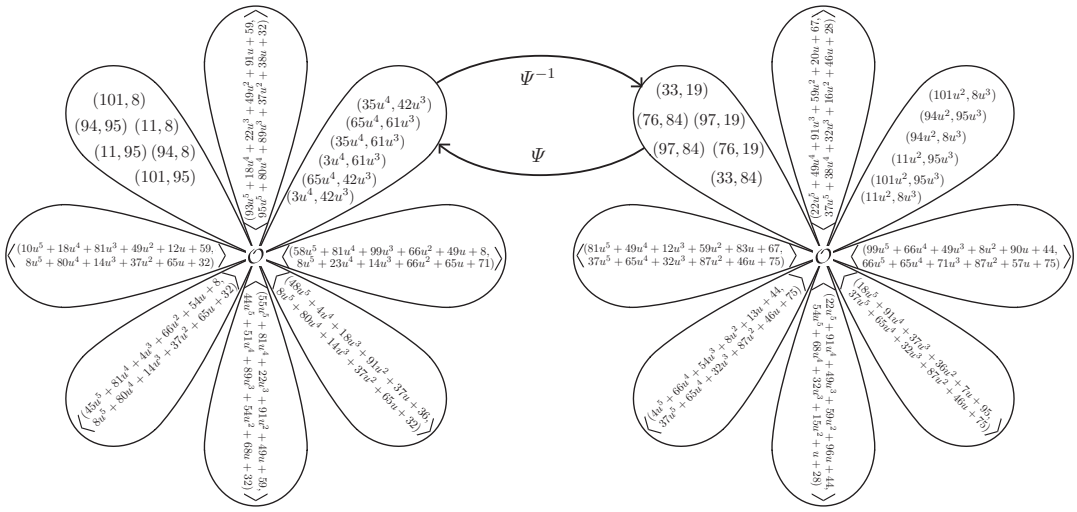


Figure 4.12: E (left) and the (correct) sextic twist E' (right)

$\mathbb{F}_q$ instead of $\mathbb{F}_{q^6}$, a dramatic improvement in computational complexity.

In both Example 4.3.1 and Example 4.3.2 above, we had $k = d$, so the twist allowed us to work in the base field $\mathbb{F}_q$, rather than $\mathbb{F}_{q^k}$. In the general case though, the twist will pull computations back into the subfield $\mathbb{F}_{q^{k/d}}$ of $\mathbb{F}_{q^k}$. For example, if the embedding degree was $k = 12$, a quadratic twist ($d = 2$) would allow computations in $\mathbb{G}_2$ to be performed in $\mathbb{F}_{q^6}$ rather than $\mathbb{F}_{q^{12}}$, whilst a sextic twist ($d = 6$) would allow us to instead work in $\mathbb{F}_{q^2}$. Thus, we would clearly prefer the degree d of the twist to be as high as possible. As it turns out, $d = 6$ is the highest degree available on elliptic curves, where the only possibilities are $d \in \{2, 3, 4, 6\}$ [Sil09, Prop. X.5.4]. For $d > 2$, we also require special subclasses of curves that depend on d, so following [Sil09, Prop. X.5.4] (see also [HSV06, Prop. 6, Prop. 8]) we describe all four cases individually. In the general case according to our context, a twist of $E : y^2 = x^3 + ax + b$ is given by $E' : y^2 = x^3 + a\omega^4 x + b\omega^6$, with $\Psi : E' \to E : (x', y') \mapsto (x'/\omega^2, y'/\omega^3)$, $\omega \in \mathbb{F}_{q^k}$. We can only achieve specific degrees d through combinations of zero and non-zero values for a and b.

- $d = 2$ *quadratic twists.* Quadratic twists are available on any elliptic curve, so if $E/\mathbb{F}_q : y^2 = x^3 + ax + b$, then a quadratic twist is given by $E'/\mathbb{F}_{q^{k/2}} : y^2 = x^3 + a\omega^4 x + b\omega^6$, with $\omega \in \mathbb{F}_{q^k}$ but $\omega^2 \in \mathbb{F}_{q^{k/2}}$. Since $\omega^3 \in \mathbb{F}_{q^k}$, the isomorphism $\Psi : E' \to E$ defined by $\Psi : (x', y') \mapsto (x'/\omega^2, y'/\omega^3)$ will take elements in $E'(\mathbb{F}_{q^{k/2}})$ to elements in $E(\mathbb{F}_{q^k})$, whilst Ψ^{-1} will do the

opposite.

- $d = 3$ *cubic twists.* Degree $d = 3$ twists can only occur when $a = 0$, so if $E/\mathbb{F}_q : y^2 = x^3 + b$, then $E'/\mathbb{F}_{q^{k/3}} : y^2 = x^3 + b\omega^6$, with $\omega^3, \omega^6 \in \mathbb{F}_{q^{k/3}}$, but $\omega^2 \in \mathbb{F}_{q^k} \setminus \mathbb{F}_{q^{k/3}}$. Thus, the isomorphism $\Psi : E' \to E$ (defined as usual) will take elements in $E'(\mathbb{F}_{q^{k/3}})$ to elements in $E(\mathbb{F}_{q^k})$, whilst Ψ^{-1} does the opposite.

- $d = 4$ *quartic twists.* Degree $d = 4$ twists are available when $b = 0$, so if $E/\mathbb{F}_q : y^2 = x^3 + ax$, then $E'/\mathbb{F}_{q^{k/4}} : y^2 = x^3 + a\omega^4 x$, with $\omega^4 \in \mathbb{F}_{q^{k/4}}$, $\omega^2 \in \mathbb{F}_{q^{k/2}}$ and $\omega^3 \in \mathbb{F}_{q^k} \setminus \mathbb{F}_{q^{k/2}}$. Thus, Ψ will move elements in $E'(\mathbb{F}_{q^{k/4}})$ up to elements in $E(\mathbb{F}_{q^k})$, whilst Ψ^{-1} will move elements from $E(\mathbb{F}_{q^k})$ down to $E'(\mathbb{F}_{q^{k/4}})$.

- $d = 6$ *sextic twists.* Sextic twists are only available when $a = 0$, so if $E/\mathbb{F}_q : y^2 = x^3 + b$, then $E'/\mathbb{F}_{q^{k/6}} : y^2 = x^3 + b\omega^6$, with $\omega^6 \in \mathbb{F}_{q^{k/6}}$, $\omega^3 \in \mathbb{F}_{q^{k/3}}$ and $\omega^2 \in \mathbb{F}_{q^{k/2}}$. Thus, Ψ pushes elements in $E'(\mathbb{F}_{q^{k/6}})$ up to $E(\mathbb{F}_{q^k})$, whilst Ψ^{-1} pulls elements from $E(\mathbb{F}_{q^k})$ all the way down to $E'(\mathbb{F}_{q^{k/6}})$.

We make the remark that, for our purposes, a specific twist can only be applied if the curve is of the corresponding form above *and* the embedding degree k has d as a factor. Thus, attractive embedding degrees are those which have any of $d = \{2, 3, 4, 6\}$ as factors, but preferably $d = 4$ or $d = 6$ for increased performance. This will be discussed in detail in Chapter 6. Very fortunately, we will also see in that chapter that almost all of the popular techniques for constructing curves suitable for pairing computation give rise to curves of the form $y^2 = x^3 + b$ or $y^2 = x^3 + ax$, which facilitate the high-degree twists above.

4.4 Chapter summary

We started by discussing that cryptographic pairings are bilinear maps from two elliptic curve groups to a third (finite field) group $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We then claimed that, in general, to define a useful pairing on $\mathbb{G}_1$ and $\mathbb{G}_2$, we must be able to define more than one subgroup in the r-torsion of E, where the most cryptographically useful case is that r is a large prime. We then defined the embedding degree k of E (with respect to r), and showed that we must extend

the field $\mathbb{F}_q$ to $\mathbb{F}_{q^k}$ in order to find more than one such subgroup. In fact, we showed that $E(\mathbb{F}_{q^k})$ actually contains the entire r-torsion, which has cardinality r^2 and consists of $r+1$ cyclic subgroups of order r. These $r+1$ subgroups (and the existence of maps between them) facilitate several choices for the definitions of $\mathbb{G}_1$ and $\mathbb{G}_2$, which gives rise to four pairing types. We argued that the most popular pairing type is a Type 3 pairing, which sets $\mathbb{G}_1$ and $\mathbb{G}_2$ as the two eigenspaces of the Frobenius endomorphism, namely $\mathbb{G}_1 = \mathcal{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ is the base field subgroup, and $\mathbb{G}_2 = \mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$ is the trace zero subgroup.

The definitions of the Weil and Tate pairings in the next chapter inherently justify the claim we made in this chapter that, in general, the arguments P and Q in the pairing $e(P, Q)$ must come from distinct torsion subgroups.

Chapter 5

Miller's algorithm for the Weil and Tate pairings

This chapter defines the Weil and Tate pairings and presents Miller's algorithm for computing them. As usual, we state the definitions in our context (on elliptic curves over finite fields), but the more general definitions are analogous (see [Sil09, Gal05]).

Notation. In this chapter we will use the notation $w_r(P, Q)$ for the (order r) Weil pairing of P and Q and $t_r(P, Q)$ for their (order r) Tate pairing, as this will help when discussing differences and relationships between them. After this chapter though, it will always be clear which pairing we mean and what the value of r is (the largest prime factor of $\#E(\mathbb{F}_q)$), so we will return to the notation most commonly seen in the literature and simply write $e(P, Q)$.

Both pairings make use of a special case of the following fact we recall from Chapter 3: a divisor $D = \sum_P n_P(P)$ is principal (i.e. the divisor of a function) if and only if $\sum_P n_P = 0$ and $\sum_P [n_P]P = \mathcal{O}$ on E. For any $m \in \mathbb{Z}$ and $P \in E$, it follows that there exists a function $f_{m,P}$ with divisor

$$(f_{m,P}) = m(P) - ([m]P) - (m-1)(\mathcal{O}), \tag{5.1}$$

where we note that for $m = 0$, one can take $f_{0,P} = 1$ with $(f_{0,P})$ the zero divisor.

Thus, if $P \in E[r]$, then $f_{r,P}$ has divisor

$$(f_{r,P}) = r(P) - r(\mathcal{O}). \tag{5.2}$$

Observe that $(f_{m+1,P}) - (f_{m,P}) = (P) + ([m]P) - ([m+1]P) - (\mathcal{O})$, which is exactly the divisor of the function $\ell_{[m]P,P}/v_{[m+1]P}$, where $\ell_{[m]P,P}$ and $v_{[m+1]P}$ are the sloped and vertical lines used in the chord-and-tangent addition of the point $[m]P$ and P (see Figure 5.1). This means we can build $f_{m+1,P}$ from $f_{m,P}$ via $f_{m+1,P} = f_{m,P} \cdot \frac{\ell_{[m]P,P}}{v_{[m+1]P}}$.

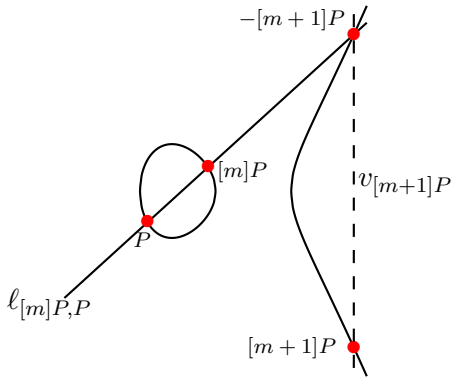


Figure 5.1: $\left(\frac{\ell_{[m]P,P}}{v_{[m+1]P}}\right) = (\ell_{[m]P,P}) - (v_{[m+1]P}) = (P) + ([m]P) - ([m+1]P) - (\mathcal{O})$.

Example 5.0.1 (Magma script). Let $q = 23$, and consider $E/\mathbb{F}_q : y^2 = x^3 + 17x + 6$ which has $\#E(\mathbb{F}_q) = 30$, and which has $P = (10, 7)$ as a point of order 5. Thus, we are guaranteed the existence of a function $f_{5,P}$ on E with divisor $(f_{5,P}) = 5(P) - 5(\mathcal{O})$. Starting with $m = 2$, we will build $f_{5,P}$ by using $f_{m+1,P} = f_{m,P} \cdot \frac{\ell_{[m]P,P}}{v_{[m+1]P}}$ (note that $(f_{1,P})$ is the zero divisor). The function $f_{2,P}$ with divisor $(f_{2,P}) = 2(P) - ([2]P) - (\mathcal{O})$ is the tangent line $l_{P,P}$ at P divided by the vertical line $v_{[2]P}$ through $[2]P$, which is $f_{2,P} = \frac{y+2x+19}{x+16}$. We compute the function $f_{3,P}$ as $f_{3,P} = f_{2,P} \cdot \frac{l_{P,[2]P}}{v_{[3]P}}$, where $l_{P,[2]P}$ is the chord through P and $[2]P$ and $v_{[3]P}$ is the vertical line at $[3]P$. Thus, $f_{3,P} = \frac{y+2x+19}{x+16} \cdot \frac{y+x+6}{x+16} = \frac{3y+x^2+9x+19}{x+16}$. Similarly, multiplication by the chord $l_{P,[3]P}$ through P and $[3]P$ and division by the vertical line $v_{[4]P}$ through $[4]P$ will advance us from $f_{3,P}$ to $f_{4,P}$, as $f_{4,P} = f_{3,P} \cdot \frac{l_{P,[3]P}}{v_{[4]P}} = \frac{3y+x^2+9x+19}{x+16} \cdot \frac{y+2x+19}{x+13} = \frac{(x+22)y+5x^2+3x+5}{x+13}$; this function has divisor $(f_{4,P}) = 4(P) - (4P) - 3(\mathcal{O})$. The last update we require is the function with divisor $(P) + (4P) - (5P) - (\mathcal{O})$, which would ordinarily be the quotient of lines in the addition of P and $4P$, but since P has order 5, we know that $P = -4P$, so this function actually has divisor $(P) + (-P) - 2(\mathcal{O})$. Thus, our last update

to the function is simply the vertical line at P, i.e. $(x-10)$, which gives the final function as $f_{5,P} = (x-10) \cdot \frac{(x+22)y+5x^2+3x+5}{x+13} = (x+22)y + 5x^2 + 3x + 5$; this function has a zero of order 5 on E at P, and a pole of order 5 on E at $\mathcal{O}$.

5.1 The Weil pairing

For a point $P \in E[r]$, the function $f_{r,P}$ with divisor $r(P) - r(\mathcal{O})$ is at the heart of both the Weil and Tate pairing definitions.

Definition 5.1 (The Weil pairing (over finite fields))**.** *Let $P, Q \in E(\mathbb{F}_{q^k})[r]$ and let D_P and D_Q be degree zero divisors with disjoint supports such that $D_P \sim (P) - (\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$. There exist functions f and g such that $(f) = rD_P$ and $(g) = rD_Q$. The Weil pairing w_r is a map*

$$w_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})[r] \to \mu_r,$$

defined as

$$w_r(P, Q) = \frac{f(D_Q)}{g(D_P)}.$$

Among other properties, the Weil pairing is bilinear and non-degenerate. We refer the reader to [Sil09, Ch. III, Prop. 8.1-8.2] for the proofs and full list of properties.

An important point to note is that we can not simply define the Weil pairing as $w_r(P, Q) = f_{r,P}(D_Q)/f_{r,Q}(D_P)$, because $(f_{r,P}) = r(P) - r(\mathcal{O})$ and $(f_{r,Q}) = r(Q) - r(\mathcal{O})$; this corresponds to the divisors $D_P = (P) - (\mathcal{O})$ and $D_Q = (Q) - (\mathcal{O})$, which does not adhere to the requirement that D_P and D_Q have disjoint supports.

Example 5.1.1 (Magma script). Let $q = 23$, and consider $E/\mathbb{F}_q : y^2 = x^3 - x$, which (is supersingular and therefore) has $\#E(\mathbb{F}_q) = q + 1 = 24$. The point $P = (2, 11)$ is a point of order $r = 3$ and the embedding degree with respect to r is $k = 2$. Take $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$ with $i^2+1 = 0$, from which we obtain a point Q of order 3 (that is not in $\langle P \rangle$) as $Q = (21, 12i)$, which is actually in the trace zero subgroup, i.e. $\pi(Q) = [q]Q$. Suppose we wish to compute the Weil pairing $w_r(P, Q)$ of P and Q. For illustrative purposes, we will start by computing $f_{r,P}$ and $f_{r,Q}$ and then updating according to the above paragraph. Following the same technique as the last example, we get $f_{r,P}$ and $f_{r,Q}$ as $f_{r,P} = y+11x+13$ and $f_{r,Q} = y+11ix+10i$,

which have divisors $(f_{r,P}) = 3(P) - 3(\mathcal{O})$ and $(g_{r,P}) = 3(Q) - 3(\mathcal{O})$ respectively. We need to find divisors D_P and D_Q that have distinct supports but which are respectively equivalent to $(P) - (\mathcal{O})$ and $(Q) - (\mathcal{O})$. Note that only one of these divisors needs to be updated (so that its support does not contain $\mathcal{O}$), but we will update both in the name of symmetry. Thus, take two more (random) points in $E(\mathbb{F}_{q^2})$ as $R = (17i, 2i + 21)$ and $S = (10i + 18, 13i + 13)$, and set $D_P = (P + R) - (R)$ and $D_Q = (Q + S) - (S)$. We find f as a function with divisor D_P and g as a function with divisor D_Q as $f = f_{r,P}/(\ell_{P,R}/v_{P+R})^3$ and $g = g_{r,Q}/(\ell_{Q,S}/v_{Q+S})^3$ respectively, where $\ell_{P,R}/v_{P+R}$ is the quotient of the chord between P and R and the vertical line through $P + R$ (and similarly for $\ell_{Q,S}/v_{Q+S}$). We can now compute the Weil pairing according to Definition 5.1 as

$$\begin{aligned} w_r(P, Q) &= f(D_Q)/g(D_P) \\ &= \frac{f(Q + S) \cdot g(R)}{f(S) \cdot g(P + R)}. \\ &= 15i + 11. \end{aligned}$$

Observe that $(15i + 11)^3 = 1$ so $w_r(P, Q) \in \mu_r$. Repeating the whole process with $[2]P$ instead gives $w_r([2]P, Q) = 8i + 11 = w_r(P, Q)^2$, or with $[2]Q$ gives $w_r(P, [2]Q) = 8i + 11 = w_r(P, Q)^2$, or with both $[2]P$ and $[2]Q$ gives $w_r([2]P, [2]Q) = 15i + 11 = w_r(P, Q)^4 = w_r(P, Q)$, which is about as much of the bilinearity of w_r that we can illustrate in this toy example.

5.2 The Tate pairing

The formal definition of the Tate pairing requires that only one argument comes from the r-torsion. For our purposes, the other argument can be any point of $E(\mathbb{F}_{q^k})$, but we will soon see that in general it is still advantageous to choose both points from (distinct subgroups in) the r-torsion. In order to define the Tate pairing correctly though, we need to properly define the groups involved. We assume the standard setting that is of most interest to us: $k > 1$, $r \| \#E(\mathbb{F}_q)$ and, since there are r^2 points in the subgroup $E(\mathbb{F}_{q^k})[r]$, we usually have $r^2 \| \#E(\mathbb{F}_{q^k})$. Thus, let $h = \#E(\mathbb{F}_{q^k})/r^2$ be the cofactor that sends points in $E(\mathbb{F}_{q^k})$ to points

in $E(\mathbb{F}_{q^k})[r]$. Let $rE(\mathbb{F}_{q^k})$ be the *coset* of points in $E(\mathbb{F}_{q^k})$ defined by

$$rE(\mathbb{F}_{q^k}) = \{[r]P : P \in E(\mathbb{F}_{q^k})\}.$$

The number of elements in $rE(\mathbb{F}_{q^k})$ is h and it contains $\mathcal{O}$; from here we will simply denote this coset as rE. Following [Sco04], we can obtain another distinct coset of $E(\mathbb{F}_{q^k})$ by adding a random element R (not in $E[r]$) to each element of rE. In this way we can obtain precisely r^2 distinct, order h cosets. The quotient group E/rE is the group whose elements are these cosets. We note that elements belonging to each coset do not have the same order, nor do they form a (sub)group. In the quotient group E/rE, points belonging to the same coset (group element) can be used to *represent* the coset. Any two points in the same coset differ from one another by an element in rE, so one can think of E/rE as the set of equivalence classes of points in $E(\mathbb{F}_{q^k})$ under the equivalence relation $P_1 \equiv P_2$ if and only if $P_1 - P_2 \in rE$ [Gal05, IX.3].

Example 5.2.1 (Magma script). Let $q = 5$, and consider $E/\mathbb{F}_q : y^2 = x^3 - 3$, which has $\#E(\mathbb{F}_q) = 6$. Thus, taking $r = 3$ gives $k = 2$, so take $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$, where $i^2 + 2 = 0$. Further, note that $\#E(\mathbb{F}_{q^2}) = 36 = hr^2$, so $h = 4$, and thus taking $rE = \{[3]P : P \in E(\mathbb{F}_{q^2})\}$ gives $rE = \{\mathcal{O}, (3i + 4, 0), (2i + 4, 0), (2, 0)\}$, with $\#rE = h$. Each of the other 8 cosets in E/rE are shown in Figure 5.2, where we importantly note that each coset has a unique *representative element* that lies in the r-torsion (see Figure 5.3). Consider the coset containing $P_1 = (2i, 4i + 3)$ $P_2 = (4, 1)$, $P_3 = (3, 2)$ and $P_4 = (3i, i + 3)\}$. All of the non-trivial pairwise differences are (defined by) $P_1 - P_2 = P_3 - P_4 = (3i + 4, 0)$, $P_1 - P_3 = P_2 - P_4 = (2i + 4, 0)$ and $P_1 - P_4 = P_2 - P_3 = (2, 0)$, which are all in rE.

For our purposes, $E[r]$ and the quotient group E/rE both have r^2 elements[1], but although it was the case in Example 5.2.1, it is not necessarily the case that the elements of $E[r]$ each represent a unique coset of E/rE (see [Gal05, IX.3] for a counterexample). However, if $r^2 \| \#E(\mathbb{F}_{q^k})$, then $E[r] \cap rE = \mathcal{O}$, which means that adding a unique torsion element to all of the elements in rE will generate a unique coset in E/rE. That is, $r^2 \| \#E(\mathbb{F}_{q^k})$ implies that $E[r]$ does represent E/rE (see [Gal05, Th. IX.22] for the proof in the supersingular scenario), and this will always be the case for us. This is particularly convenient when it comes to defining the Tate pairing, since the "second" group in the (order r) Tate

[1] In fact, they always have the same number of elements, but there are cases when the cardinality is not r^2 – see [Gal05, IX.3, IX.7.3]

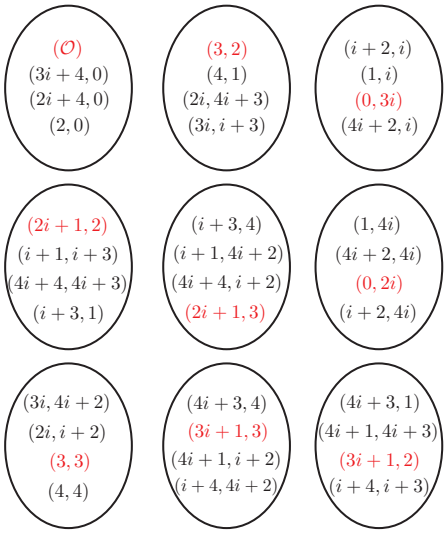


Figure 5.2: The r^2 cosets in the quotient group $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$.

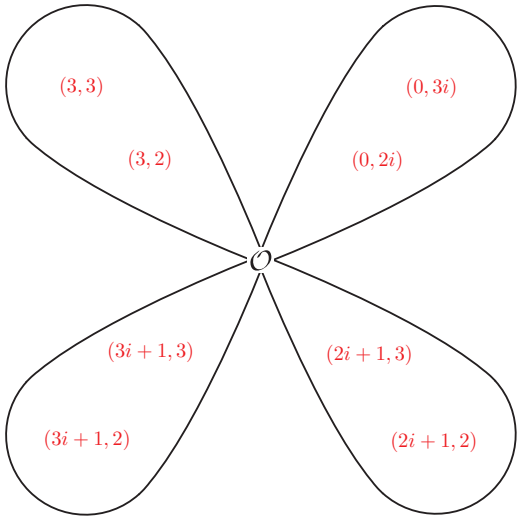


Figure 5.3: The r-torsion, where each $P \in E[r]$ is in a distinct coset of E/rE.

pairing is E/rE. As we will see after the definition, $E[r]$ representing E/rE allows us to take both groups from the r-torsion, which matches the somewhat simpler Weil pairing group definitions.

We note that although we refer to the following pairing as the Tate pairing throughout, it is often aptly called the Tate-Lichtenbaum pairing [Sil09, XI.9]. This is because Lichtenbaum [Lic69] specialised Tate's more general pairing to the case of Jacobians of curves (over local fields) which facilitates explicit computation [Gal05, IX.3].

Definition 5.2 (The Tate pairing (over finite fields))**.** *Let $P \in E(\mathbb{F}_{q^k})[r]$, from which it follows that there is a function f whose divisor is $(f) = r(P) - r(\mathcal{O})$. Let $Q \in E(\mathbb{F}_{q^k})$ be any representative in any equivalence class in $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$, and let D_Q be a degree zero divisor defined over $\mathbb{F}_{q^k}$ that is equivalent to $(Q) - (\mathcal{O})$, but whose support is disjoint to that of (f). The Tate pairing t_r is a map*

$$t_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r,$$

defined as

$$t_r(P, Q) = f(D_Q).$$

Again, we remark that among other properties, the Tate pairing is bilinear

and non-degenerate. We refer the reader to [Sil09, XI.9] and [Gal05, IX.4] for the proofs and full list of properties.

The quotient group $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$ is defined as we would expect. Namely, $(\mathbb{F}_{q^k}^*)^r$ is a subgroup of $\mathbb{F}_{q^k}^*$ defined as $(\mathbb{F}_{q^k}^*)^r = \{u^r : u \in \mathbb{F}_{q^k}^*\}$, so $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$ is the set of equivalence classes of $\mathbb{F}_{q^k}^*$ under the equivalence relation $a_1 \equiv a_2$ if and only if $a_1/a_2 \in (\mathbb{F}_{q^k}^*)^r$.

Example 5.2.2 (Magma script). We continue with the parameters from Example 5.2.1. Let $P = (3, 2) \in E[r]$ (see Figure 5.2) and let $Q = (i+1, 4i+2) \in E(\mathbb{F}_{q^k})$. The function $f : y + 2x + 2 = 0$ on E has divisor $3(P) - 3(\mathcal{O})$, so to compute the Tate pairing we need to find an appropriate $D_Q \sim (Q) - (\mathcal{O})$ but with $P, \mathcal{O} \notin \text{supp}(D_Q)$. Take R (randomly) as $R = (2i, i + 2)$, and let $D_Q = (Q + R) - (R)$, where $Q + R = (3i + 1, 2)$. The Tate pairing is computed as

$$t_r(P, Q) = f(D_Q) = \frac{f(Q + R)}{f(R)} = \frac{2 + 2 \cdot (3i + 1) + 2}{(i + 2) + 2 \cdot 2i + 2} = 4i + 4.$$

To illustrate bilinearity, computing $t_r(P, [2]Q)$ with $D_{[2]Q} = ([2]Q + R) - (R)$ where $[2]Q + R = (i + 2, i)$ gives

$$t_r(P, [2]Q) = f(D_{[2]Q}) = \frac{f([2]Q + R)}{f(R)} = \frac{i + 2 \cdot (i + 2) + 2}{(i + 2) + 2 \cdot 2i + 2} = 2i + 4,$$

or computing $t_r([2]P, Q)$, where $\tilde{f} = y + 3x + 3$ has divisor $\tilde{f} = r([2]P) - r(\mathcal{O})$, gives

$$t_r([2]P, Q) = \tilde{f}(D_Q) = \frac{\tilde{f}(Q + R)}{\tilde{f}(R)} = \frac{2 + 3 \cdot (3i + 1) + 3}{(i + 2) + 3 \cdot 2i + 3} = 3i + 2.$$

Note that $t_r(P, Q) = 4i + 4$, $t_r(P, [2]Q) = 2i + 4 = t_r(P, Q)^2$, but $t_r([2]P, Q) = 3i + 2$, i.e. $t_r(P, [2]Q), t_r([2]P, Q) \notin (\mathbb{F}_{q^k}^*)^r$, but $t_r(P, [2]Q)/t_r([2]P, Q) \in (\mathbb{F}_{q^k}^*)^r$, so $t_r(P, [2]Q) \equiv t_r([2]P, Q) \equiv t_r(P, Q)^2$ in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$.

The above example illustrates an important point: in the context of cryptography, the standard Tate pairing has an undesirable property that its output lies in an equivalence class, rather than being a unique value. A necessary attribute for the Tate pairing to be useful in cryptography is that different parties must compute the exact same value under the bilinearity property, rather than values which are the same under the above notion of equivalence. Thus, to be suitable in practice, we must update the definition of the Tate pairing to make sure the

mapping produces unique values.

Definition 5.3 (The reduced Tate pairing)**.** *Let P, Q, f and D_Q be as in Definition 5.2. Over finite fields, the reduced Tate pairing T_r is a map*

$$T_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) \to \mu_r,$$

defined as

$$\begin{aligned} T_r(P,Q) &= t_r(P,Q)^{\#\mathbb{F}_{q^k}/r} \\ &= f_{r,P}(D_Q)^{(q^k-1)/r}. \end{aligned}$$

Exponentiating elements in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$ to the power of $(q^k-1)/r$ kills r-th powers and sends the paired value to an exact r-th root of unity in μ_r.

From now on we will also take the second argument of the (reduced) Tate pairing from the r-torsion. In fact, we will further assume a Type 3 pairing. Therefore, in the pairing of P and Q, we will assume $P \in \mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ and $Q \in \mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$. One should note that these choices are not restrictions, as far as what values the pairing can take: fixing P and letting Q run through $\langle Q \rangle$ (which has order r) will give each value in μ_r, and vice versa. Thus, for any $\tilde{P}, \tilde{Q}$ pair chosen from anywhere in the torsion, there exists a scalar $0 \leq a \leq r-1$ such that $T_r([a]P, Q) = T_r(P, [a]Q) = T_r(\tilde{P}, \tilde{Q})$.

Example 5.2.3 (Magma script). Let $q = 19$, $E/\mathbb{F}_q : y^2 = x^3 + 14x + 3$, giving $\#E(\mathbb{F}_q) = 20$, so take $r = 5$. The embedding degree is $k = 2$, so let $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$ with $i^2 + 1 = 0$. The points $P = (17, 9)$ and $Q = (16, 16i)$ are in the r-torsion subgroups $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ and $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$ respectively. The Tate pairing of P and Q is $t_r(P, Q) = 7i + 3$, whilst the reduced Tate pairing is $T_r(P, Q) = 15i + 2$. Let $\exp : \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r \to \mu_r$ be the map defined by the exponentiation $\exp : a \mapsto a^{(q^k-1)/r}$, i.e. $\exp : t_r(P, Q) \mapsto T_r(P, Q)$. Observe the difference between the Tate pairing t_r and reduced Tate pairing T_r for the following computations.

$$\begin{array}{cccc} t_r(P,Q)^4 & t_r([4]P,Q) & t_r(P,[4]Q) & t_r([2]P,[2]Q) \\ = 3i+7 & = 7i+16 & = 12i+3 & = 2i+14 \\ \Big\downarrow \exp & \Big\downarrow \exp & \Big\downarrow \exp & \Big\downarrow \exp \\ T_r(P,Q)^4 & T_r([4]P,Q) & T_r(P,[4]Q) & T_r([2]P,[2]Q) \\ = 4i+2 & = 4i+2 & = 4i+2 & = 4i+2 \end{array}$$

We note that none of the t_r lie in $(\mathbb{F}_{q^k})^5$, but the quotient of any two of them does lie there, so all the t_r pairings on the top level are equivalent in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$. On the other hand, T_r ensures that each of the above pairings (that should be equivalent) take exactly the same value in $\mu_r \subset \mathbb{F}_{q^k}$.

From now on, when we say Tate pairing, we mean the reduced Tate pairing T_r in Definition 5.3.

5.3 Miller's algorithm

We briefly recap the pairing definitions from the previous two sections. For the r-torsion points P and Q, the Weil and Tate pairings are respectively computed as $\frac{f_{r,P}(D_Q)}{f_{r,Q}(D_P)}$ and $f_{r,P}(D_Q)^{(q^k-1)/r}$, where the divisors D_P and D_Q are chosen such that their supports are disjoint from the supports of $(f_{r,Q})$ and $(f_{r,P})$ respectively. For any points P and Q belonging to distinct subgroups in $E[r]$, we have already seen how to compute $f_{r,P}(D_Q)$ in the previous sections, but this was only for very small values of r. In practice r will be huge (i.e. at the very least 2^{160}), and since $f_{r,P}$ is a function of degree approximately r, it is not hard to see that computing this function as we did in the previous examples is impossible. In this section we describe Miller's algorithm [Mil04], which makes this computation very feasible. More precisely, the naive method of computing $f_{r,P}(D_Q)$ that we have been using has exponential complexity $O(r)$, whilst the algorithm we are about to describe for this computation has polynomial complexity $O(\log r)$. To put it simply, Miller's algorithm makes pairings practical; without this algorithm, secure cryptographic pairings would only be of theoretical value[2].

We start by referring back to the discussion at the beginning of this chapter. Following Equation (5.1), we saw that the divisor $(f_{m,P}) = m(P) - ([m]P) - (m-1)(\mathcal{O})$ could be updated to the divisor $(f_{m+1}) = (m+1)(P) - ([m+1]P) - m(\mathcal{O})$ by adding the divisor $(\ell_{[m]P,P}/v_{[m+1]P}) = (P) + ([m]P) - ([m+1]P) - (\mathcal{O})$; this corresponds to the multiplication of functions $f_{m+1} = f_m \cdot \ell_{[m]P,P}/v_{[m+1]P}$. Starting with $f_{2,P} = 2(P) - ([2]P) - (\mathcal{O})$ then, we can repeat this process roughly $r-1$ times to obtain the desired function $f_{r,P} = r(P) - ([r]P) - (r-1)(\mathcal{O}) = r(P) - r(\mathcal{O})$. We note that for the last step (i.e. when $m = r-1$) we have $f_{r-1,P} = (r-1)(P) - ([r-1]P) - (r-2)(\mathcal{O})$, so the required divisor is $(P) +$

[2]This is no longer entirely true. In 2007 Stange derived an alternative method to Miller's algorithm for efficiently computing the Tate pairing [Sta07], but it is currently less efficient than Miller's algorithm.

$([r-1]P) - 2(\mathcal{O})$ which corresponds to (a multiplication by!) the vertical line $v_{[r-1]P} = v_{-P} = v_P$; note that this is the same vertical line that appears on the denominator of $\ell_{[r-2]P,P}/v_{[r-1]P}$. Thus, the *pairing evaluation function* $f_{r,P}$ is the product

$$f_{r,P} = \ell_{[r-2]P,P} \cdot \prod_{i=1}^{r-3} \frac{\ell_{[i]P,P}}{v_{[i+1]P}}. \tag{5.3}$$

The first four sloped lines $\ell_{[i]P,P}$ and corresponding vertical lines $v_{[i+1]P}$ from the numerator and denominator of the product in (5.3) are shown in Figure 5.4 and Figure 5.5 respectively. We have seen that the product in (5.3) is (in the most

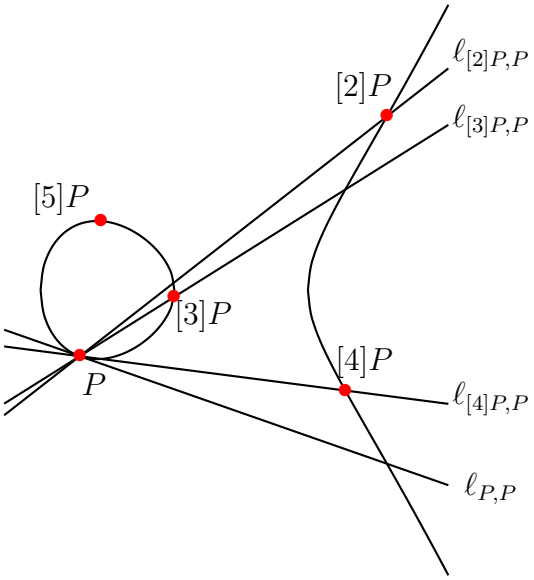


Figure 5.4: The first four sloped lines in the product (5.3).

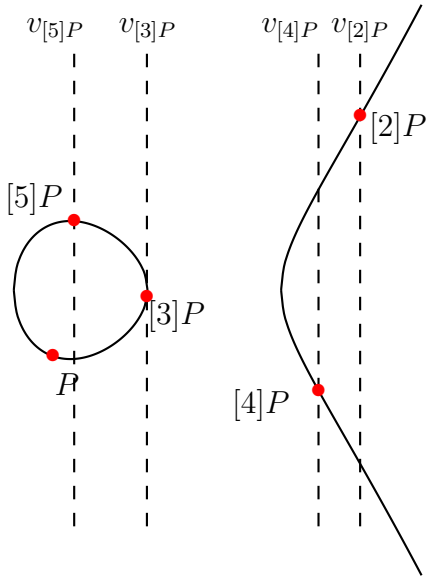


Figure 5.5: The first four vertical lines in the product (5.3).

naive way) built up incrementally by absorbing each of the $\frac{\ell_{[m]P,P}}{v_{[m+1]P}}$ terms into $f_{m,P}$ to increment to $f_{m+1,P}$, eventually arriving at $f_{r,P}$. Alternatively, it can help to see the divisor sum written out in full, to see the contributions of each

of the functions $\frac{\ell_{[i]P,P}}{v_{[i+1]P}}$ in the product all at once.

$$\begin{aligned}
\ell_{P,P}/v_{[2]P}: &\quad (P)+(P)-([2]P)-(\mathcal{O})\\
\ell_{[2]P,P}/v_{[3]P}: &\quad (P)+([2]P)-([3]P)-(\mathcal{O})\\
\ell_{[3]P,P}/v_{[4]P}: &\quad (P)+([3]P)-([4]P)-(\mathcal{O})\\
\vdots &\quad \vdots \quad \vdots \quad \vdots \quad \vdots\\
\ell_{[r-4]P,P}/v_{[r-3]P}: &\quad (P)+([r-4]P)-([r-3]P)-(\mathcal{O})\\
\ell_{[r-3]P,P}/v_{[r-2]P}: &\quad (P)+([r-3]P)-([r-2]P)-(\mathcal{O})\\
\ell_{[r-2]P,P} &\quad (P)+([r-2]P)+(-[r-1]P)-3(\mathcal{O})
\end{aligned}$$

When summing all of the above divisors, most of the inner terms cancel out with one another to leave $(r-1)(P)+(-[r-1]P)-r(\mathcal{O})$, and since $[r-1]P=-P$, we get the divisor of the product being $r(P)-r(\mathcal{O})$.

Roughly speaking, $f_{r,P}=g(x,y)/h(x,y)$, where g and h are degree r functions on E. The above method computes $f_{r,P}$ by successively increasing the degrees of g and h by one each time $f_{m,P}$ is incremented. This is why, when r is (exponentially) large, this naive method has exponential complexity. Miller's algorithm naturally overcomes this through the following observation. The function $f_{m,P}$ has m zeros at P and $(m-1)$ poles at $\mathcal{O}$. Rather than adding one zero and one pole via multiplying $f_{m,P}$ by linear functions, we can double the number of zeros at P and the number of poles at $\mathcal{O}$ if we instead square $f_{m,P}$. Observe that since $(f_{m,P})=m(P)-([m]P)-(m-1)(\mathcal{O})$, then

$$(f_{m,P}^2)=2m(P)-2([m]P)-2(m-1)(\mathcal{O}),$$

which is almost the same as $f_{2m,P}$, whose divisor is

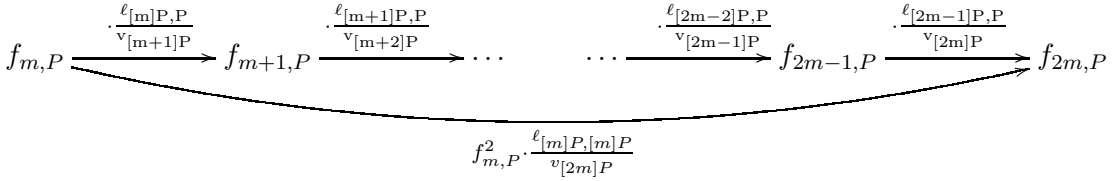
$$(f_{2m,P})=2m(P)-([2m]P)-(2m-1)(\mathcal{O});$$

the difference between the two divisors being $(f_{2m,P})-(f_{m,P}^2)=2([m]P)-([2m]P)-(\mathcal{O})$, which corresponds to a function with two zeros at $[m]P$, a pole at $[2m]P$ and another pole at $\mathcal{O}$. We have seen such a function many times already; this is simply the quotient of the tangent line at $[m]P$ and the vertical line at $[2m]P$ – the lines used to double the point $[m]P$. Thus, we can advance

from $f_{m,P}$ to $f_{2m,P}$ via

$$f_{2m,P} = f_{m,P}^2 \cdot \frac{\ell_{[m]P,[m]P}}{v_{[2m]P}}.$$

We depict the jump from $f_{m,P}$ to $f_{2m,P}$ (as opposed to the naive method of progressing one-by-one) below.



Since, for any m, we can now advance to either $f_{m+1,P}$ or $f_{2m,P}$ quickly, Miller observed that this gives rise to a double-and-add style algorithm to reach $f_{2,r}$ in $O(\log(r))$ steps. However, the degree of $f_{m,P}$ grows linearly in the size of m, so (en route to $m = r$) the function $f_{m,P}$ becomes too large to store explicitly. Thus, the last piece of the puzzle in Miller's derivation of the pairing algorithm was to, at every stage, evaluate $f_{m,P}$ at the given divisor, i.e. $f_{m,P}(D_Q)$. This means that at any intermediate stage of the algorithm we will not be storing an element of the function field $f_{m,P} \in \mathbb{F}_{q^k}(E)$, but rather its evaluation at D_Q which is the value $f_{m,P}(D_Q) \in \mathbb{F}_{q^k}$. At each stage then, the updates that build the function are evaluated at D_Q before being absorbed into intermediate pairing value that is carried through the routine. This is summarised in Algorithm 5.1 below, where the binary representation of r governs the double-and-add route taken to compute $f_{r,P}(D_Q)$, in an identical fashion to the standard double-and-add routine for scalar multiplications on E (see Example 2.1.8).

Miller's algorithm is essentially the straightforward double-and-add algorithm for elliptic curve point multiplication (see Example 2.1.8) combined with evaluations of the functions (the chord and tangent lines) used in the addition process.

Example 5.3.1 (Magma script). We will compute both the Weil and Tate pairings using Miller's algorithm. Let $q = 47$, $E/\mathbb{F}_q : y^2 = x^3 + 21x + 15$, which has $\#E(\mathbb{F}_q) = 51$, so we take $r = 17$. The embedding degree k with respect to r is $k = 4$, thus take $\mathbb{F}_{q^4} = \mathbb{F}_q(u)$ where $u^4 - 4u^2 + 5 = 0$. The point $P = (45, 23)$ has order 17 in $E(\mathbb{F}_q)$ which (because $k > 1$) means $P \in \mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$. The group order over the full extension field is $\#E(\mathbb{F}_{q^4}) = 3^3 \cdot 5^4 \cdot 17^2$, so take $h = 3^3 \cdot 5^4$ as the cofactor. Taking a random point from $E(\mathbb{F}_{q^4})$ and multiplying by h will (almost always) give a point $Q \in E[r]$, but it is likely to land outside of $\mathbb{G}_1 \cup \mathbb{G}_2$,

Algorithm 5.1 Miller's algorithm.

Input: $P \in E(\mathbb{F}_{q^k})[r]$, $D_Q \sim (Q) - (\mathcal{O})$ with support disjoint from $(f_{r,P})$, and $r = (r_{n-1} \dots r_1 r_0)_2$ with $r_{n-1} = 1$.
Output: $f_{r,P}(D_Q) \leftarrow f$.

1: $R \leftarrow P$, $f \leftarrow 1$.
2: **for** $i = n-2$ down to 0 **do**
3: Compute the line functions $\ell_{R,R}$ and $v_{[2]R}$ for doubling R.
4: $R \leftarrow [2]R$.
5: $f \leftarrow f^2 \cdot \frac{\ell_{R,R}}{v_{[2]R}}(D_Q)$.
6: **if** $r_i = 1$ **then**
7: Compute the line functions $\ell_{R,P}$ and v_{R+P} for adding R and P.
8: $R \leftarrow R + P$.
9: $f \leftarrow f \cdot \frac{\ell_{R,P}}{v_{R+P}}(D_Q)$.
10: **end if**
11: **end for**
12: **return** f.

so to move into $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$, we can use the anti-trace map (see Figure 4.4) and take $Q \leftarrow [k]Q - \mathrm{Tr}(Q)$. For example, $Q = (31u^2+29, 35u^3+11u)$ is one of 17 points in $\mathbb{G}_2$. The Tate pairing is $T_r(P,Q) = f_{r,P}(D_Q)^{(q^k-1)/r}$, whilst the Weil pairing is $w_r(P,Q) = \frac{f_{r,P}(D_Q)}{f_{r,Q}(D_P)}$. We will illustrate Miller's algorithm to compute $f_{r,P}(D_Q)$, since it appears in both. The binary representation of r is $r = (1,0,0,0,1)_2$. We will take D_Q as $D_Q = ([2]Q) - (Q)$, which clearly has support disjoint to $(f_{r,P})$ and is equivalent to $(Q) - (\mathcal{O})$. The table below shows the stages of Miller's algorithm for computing $f_{r,P}(D_Q)$: it shows the intermediate values of R, and of the function ℓ/v which corresponds to $\frac{\ell_{R,R}}{v_{[2]R}}$ or $\frac{\ell_{R,P}}{v_{R+P}}$ depending on whether we are in the doubling stage (steps 3-5 of Algorithm 5.1) or the addition stage (steps 6-10 of Algorithm 5.1); the table also shows the progression of the paired value f. To complete the Tate pairing, we compute

i/ r_i	steps of Alg. 5.1	point R	update ℓ/v	update at $[2]Q$ = $\frac{\text{update at } [2]Q}{\text{update at } Q}$ = $\frac{\ell(D_Q)}{v(D_Q)}$ = $\frac{\ell/v([2]Q)}{\ell/v(Q)}$	paired value f
	1	$(45,23)$			1
3/0	3-5	$(12,16)$	$\frac{y+33x+43}{x+35}$	$\frac{20u^3+21u^2+9u+4}{6u^3+19u^2+36u+33} = 41u^3+32u^2+2u+21$	$41u^3+32u^2+2u+21$
2/0	3-5	$(27,14)$	$\frac{y+2x+7}{x+20}$	$\frac{40u^3+18u^2+38u+9}{39u^3+8u^2+20u+18} = 4u^3+5u^2+28u+17$	$22u^3+27u^2+30u+33$
1/0	3-5	$(18,31)$	$\frac{y+42x+27}{x+29}$	$\frac{29u^3+15u^2+8u+14}{18u^3+32u^2+41u+30} = 6u^3+13u^2+33u+28$	$36u^3+2u^2+21u+37$
0/1	3-5	$(45,24)$	$\frac{y+9x+42}{x+2}$	$\frac{10u^3+3u^2+14u+19}{21u^3+26u^2+25u+20} = 46u^3+45u^2+u+20$	$10u^3+21u^2+40u+25$
	6-10	$\mathcal{O}$	$x+2$	$\frac{7u^2+27}{31u^2+31} = 6u^2+43$	$17u^3+6u^2+10u+22$
	12	$f_{r,P}(D_Q) \leftarrow 17u^3+6u^2+10u+22$			

$$t_r(P,Q) = f_{r,P}(D_Q)^{(q^k-1)/r} = (17u^3+6u^2+10u+22)^{287040} = 33u^3+43u^2+45u+39.$$

For the Weil pairing, we require another run of Miller's algorithm, this time reversing the roles of P and Q to compute $f_{r,Q}(D_P) = 2u^2 + 6u + 40$, which gives the Weil pairing as $w_r(P,Q) = \frac{f_{r,P}(D_Q)}{f_{r,Q}(D_P)} = \frac{17u^3+6u^2+10u+22}{2u^2+6u+40} = 22u^3+12u^2+32u+13$. Notice that, in line with Equation 5.3 (and the preceding discussion), the vertical line $x + 2 = 0$ that corresponds to the final addition in this example appears in the denominator of the previous ℓ/v function used for the doubling, and could therefore be cancelled out. We will see that this occurs in general, and is perhaps the least significant of many improvements to Miller's initial algorithm that have accelerated pairings over the last decade. Indeed, in Chapter 7 we will be looking at several more major optimisations to Miller's algorithm. 5.1.

5.4 Chapter summary

We started with the more simple description of the Weil pairing, before moving to the definition of the Tate pairing. This is because both the elliptic curve groups in the raw definition of the Weil pairing are torsion subgroups, which were discussed at length in the previous chapter. On the other hand, one of the groups in the general Tate pairing definition required us to introduce the quotient group $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$. However, we soon showed that (for cases of cryptographic interest) it is no problem to represent this quotient group by a torsion subgroup, thereby unifying the group definitions needed for the Weil and Tate pairing and solidifying the choices of $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ and $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$, which will be standard for the remainder of this text. We saw that at the heart of both the Weil and Tate pairings is the computation of the pairing evaluation function $f_{r,P}(D)$, where $P \in E$ and D is an appropriately defined divisor on E. We finished the chapter by presenting Miller's algorithm, which is the first practical algorithm to compute $f_{r,P}(D)$ for cases of cryptographic interest, and which remains the fastest algorithm for computing pairings to date.

Chapter 6

Pairing-friendly curves

To realise pairing-based cryptography in practice, we need two things [Sco07a]:

- efficient algorithms for computing pairings; and
- suitable elliptic curves.

The former was briefly outlined in the last chapter (and will be taken much further in the next), whilst this chapter is dedicated to the latter.

6.1 A balancing act

Pairings are fundamentally different to traditional number-theoretic primitives, in that they require multiple groups that are defined in different settings. Namely, $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curve groups, whilst $\mathbb{G}_T$ is a multiplicative subgroup of a finite field. All three groups must be secure against the respective instances of the discrete logarithm problem, which means attackers can break the system by solving either the DLP in $\mathbb{G}_T$ or the EDCLP in $\mathbb{G}_1$ or $\mathbb{G}_2$. As we discussed in Section 2.1, elliptic curve groups currently obtain much greater security per bit than finite fields; this is because the best attacks on the ECDLP remain generic attacks like Pollard rho [Pol78] which have exponential complexity, whilst the best attacks on the DLP have sub-exponential complexity. In other words, to achieve the same security, a finite field group needs to have a much greater cardinality than an elliptic curve group. It is standard to state the complexity of asymmetric primitives in terms of the equivalent symmetric key size. For example, the most

recent ECRYPT recommendations (see `http://www.keylength.com/en/3/`) say that to achieve security comparable to AES-128 (i.e. 128-bit security), we need an elliptic curve group of approximately 256 bits[1] and a finite field of approximately 3248 bits. We give an example of a curve in the context of pairings for which $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ meet these particular requirements.

Example 6.1.1 (Magma script). Let $E/\mathbb{F}_q : y^2 = x^3 + 14$ be the curve with order $\#E(\mathbb{F}_q)$ having large prime factor r, where q and r are given as

$$\begin{aligned} q = {} & 421943326900167228539204394914103813941511244153259151125138128777531 \\ & 7505016692408034796798044263154903329667 \quad (369 \text{ bits}), \\ r = {} & 223697061107578678950388273657862788561030003896406213385139113737656 \\ & 9980702677867 \quad (271 \text{ bits}). \end{aligned}$$

The embedding degree is $k = 9$, i.e. $q^9 - 1 \equiv 0 \bmod r$. Thus, the two elliptic curve groups $\mathbb{G}_1 \in E[r]$ and $\mathbb{G}_2 \in E[r]$ have an order of 271 bits, which meets the current requirements for 128-bit security. Although $\mathbb{G}_T$ is a subgroup of order r (in $\mathbb{F}_{q^k}^*$), the attack complexity is determined by the full size of the field $\mathbb{F}_{q^9}$, which is 3248 bits, also meeting the requirements for 128-bit security.

We discuss an important point with reference to the above example. Namely, if we were to use primes q and r of the same bit-sizes as Example 6.1.1, but which corresponded to a curve with a larger embedding degree k, then this would not increase the security level offered by the pairing. For example, even though $k = 18$ gives a finite field of 6496 bits, which on its own corresponds to a much harder DLP ($\approx$ 175-bit security), the overall complexity of attacking the protocol remains the same, because the attack complexity of the ECDLP has not changed. Such an increase in k unnecessarily hinders the efficiency of the pairing, since the most costly operations in Miller's algorithm take place in $\mathbb{F}_{q^k}$. Thus, the ideal approach is to optimise the balance between r and $\mathbb{F}_{q^k}$ so that both can be as small as possible whilst simultaneously meeting the particular security level required. This was achieved successfully in our example, where $\mathbb{F}_{q^k}$ was exactly the recommended size, and r was only a few bits larger than what is needed to claim 128-bit security.

[1] The "half-the-size" principle between elliptic curve groups and the equivalent asymmetric key size is standard [Sma10, §6.1], since attacks against elliptic curves with order r subgroup have running time $O(\sqrt{r})$. Obtaining the equivalent finite field group size is not as trivial – see [Sma10, §6.2].

Nevertheless, we can still obtain a significant improvement on the parameters used in Example 6.1.1; we can keep all three group sizes the same, whilst decreasing the size of the base field $\mathbb{F}_q$. The Hasse bound (see Eq. (2.6)) tells us that the bit-length of $\#E$ and the bit-length of q will be the same. Thus, it is possible that we can find curves defined over smaller fields whose largest prime order subgroup has the same bit-size as that in Example 6.1.1, and whose embedding degree is large enough to offset the decrease in q and therefore that the corresponding full extension field also meets the security requirements. We give a "prime" example.

Example 6.1.2 (Magma script). Let $E/\mathbb{F}_q : y^2 = x^3 + 2$ be the curve with prime order $r = \#E(\mathbb{F}_q)$, where q and r are given as

$$q = 28757880164823737284021204980065523467377219983513098565427519263513769\\ 64733335173 \quad (271 \text{ bits}).$$

$$r = 28757880164823737284021204980065523467376683719770479098963148984065605\\ 60716472109 \quad (271 \text{ bits}).$$

The embedding degree is $k = 12$, i.e. $q^{12} - 1 \equiv 0 \bmod r$, giving $\mathbb{F}_{q^{12}}$ as a 3248-bit field, which is exactly the same size as the $k = 9$ curve in Example 6.1.1. Thus, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ have orders of the same bit-lengths as before, but using this curve instead means that arithmetic in $\mathbb{F}_q$ will be substantially faster; a 271-bit field in this case, compared to 369-bit field in the last.

In light of the difference between Example 6.1.1 and Example 6.1.2, an important parameter associated with a curve that is suitable for pairings is the ratio between the field size q and the large prime group order r, which we call the ρ-value, computed as

$$\rho = \frac{\log q}{\log r}.$$

Referring back to the two curves above, we have $\rho = \frac{\log q}{\log r} = \frac{369}{271} = 1.36$ in Example 6.1.1, whilst $\rho = \frac{\log q}{\log r} = \frac{271}{271} = 1$ in Example 6.1.2. The ρ-value essentially indicates how much (ECDLP) security a curve offers for its field size, and since we generally prefer the largest prime divisor r of $\#E$ to be as large as possible, $\rho = 1$ is as good as we can get. Indeed, the curve in Example 6.1.2 with $\rho = 1$ belongs to the famous Barreto-Naehrig (BN) family of curves [BN05], which all have $k = 12$ and for which the ratio between the sizes of r and $\mathbb{F}_{q^k}$ make them perfectly suited

to the 128-bit security level. This ratio between these group sizes is $\rho \cdot k$ (i.e. $\frac{\log q^k}{\log r} = k \cdot \frac{\log q}{\log r}$), so for commonly used security levels, Figure 6.1 gives the value of $\rho \cdot k$ that balances the current attack complexities of the DLP and ECDLP. Different information security and/or intelligence organisations from around the

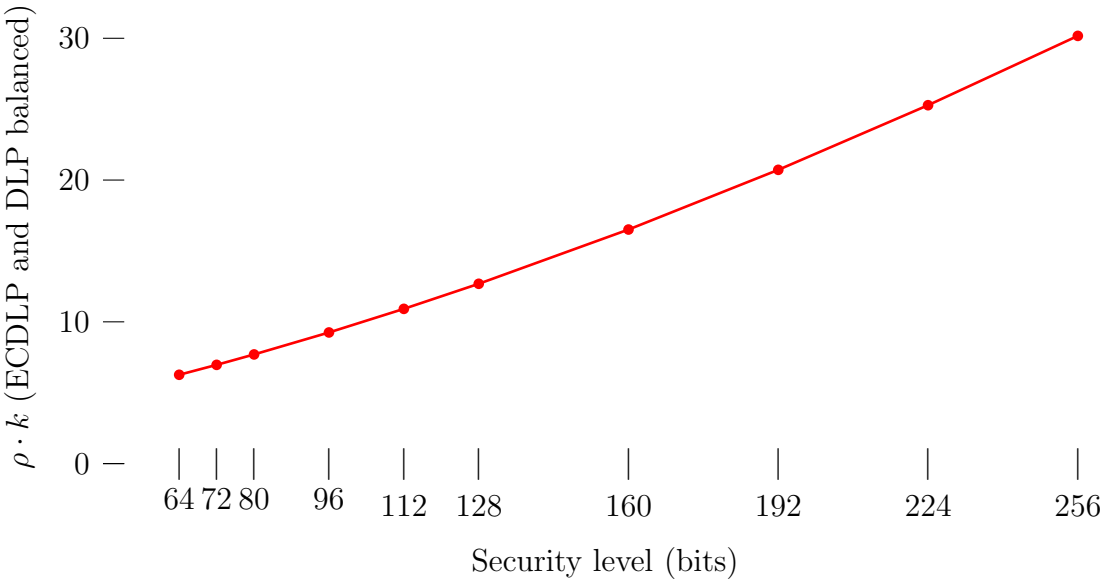


Figure 6.1: The value of $\rho \cdot k$ that balances the complexity of the DLP and ECDLP for commonly used security levels.

globe, such as NIST (the USA) and FNISA (France), give slightly different key size recommendations and complexity evaluations of the algorithms involved; all of this information is conveniently collected at `http://www.keylength.com/`. We have chosen to generate Figure 6.1 according to the numbers in the (most) recent ECRYPT II report [Sma10], which is also summarised there.

Having seen two examples above, we are now in a position to define a *pairing-friendly* curve. Following [FST10], we say that a curve is pairing-friendly if the following two conditions hold:

- there is a prime $r \geq \sqrt{q}$ dividing $\#E(\mathbb{F}_q)$ (i.e. $\rho \leq 2$), and
- the embedding degree k with respect to r is less than $\log_2(r)/8$.

Thus, in their widely cited taxonomy paper, Freeman *et al.* [FST10] consider pairing-friendly curves up to $k = 50$, which is large enough to cover recommended levels of security for some decades yet.

Balasubramanian and Koblitz [BK98] show that, for q of any suitable cryptographic size, the chances of a randomly chosen curve over $\mathbb{F}_q$ being pairing-

friendly is extremely small. Specifically, they essentially show that the embedding degree (with respect to r) of a such a curve is proportional to (and therefore is of the same order as) r, i.e. $k \approx r$. Very roughly speaking, such an argument is somewhat intuitive since (for a random curve) $\#E$ can fall anywhere in the range $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$, so r can be thought of as independent of q, meaning that the order of q in $\mathbb{Z}_r^*$ is random (but see [BK98] for the correct statements). Therefore, imposing that k is small enough to work with elements in $\mathbb{F}_{q^k}$ is an extremely restrictive criterion, so one can not hope to succeed if randomly searching for pairing-friendly curves amongst arbitrary elliptic curves. Thus, in general, pairing-friendly curves require very special constructions.

In Section 6.2 we briefly discuss supersingular elliptic curves, which always possess embedding degrees $k \leq 6$ [MOV93, §4], and (so long as $r \geq \sqrt{q}$) are therefore always pairing-friendly. Referring back to Figure 6.1 though, we can see that having $k > 6$ is highly desirable for efficient pairings at the widely accepted security levels, and thus in Section 6.3 we focus on the ordinary (non-supersingular) case and outline the constructions that achieve pairing-friendly curves with $k > 6$.

6.2 Supersingular curves

Recall from Section 4.1 that an elliptic curve E is characterised as supersingular if and only if a distortion map exists on E. There are essentially five types of supersingular curves that are of interest in PBC [Gal05, Table IX.1], but here we will only mention two. This is because we are only concerned with prime fields in this text, and the other three are either defined over $\mathbb{F}_{p^2}$, $\mathbb{F}_{2^m}$ or $\mathbb{F}_{3^m}$. As Galbraith mentions, a problem in characteristic 2 and 3 is that there is only a small number of curves and fields to choose from, so there is an element of luck in the search for a curve whose order contains a large prime factor. Another problem in small characteristic is that there exist enhanced algorithms for discrete logarithms (see [Gal05, Ch. IX.13]).

All supersingular curves over large prime fields have $\#E(\mathbb{F}_q) = q+1$, from which it follows that $k = 2$, i.e. regardless of the prime factor $r \neq 2$, $r \mid q+1$ implies $r \nmid q-1$ but $r \mid q^2-1$. We have already seen examples of the two popular supersingular choices in Section 4.1, whose general forms are given in Table 6.1.

We give another example of both cases below, but we choose the parameter

q	E	distortion map ϕ	e.g.
2 mod 3	$y^2 = x^3 + b$	$(x, y) \mapsto (\zeta_3 x, y)$, $\zeta_3^3 = 1$	Eg. 4.1.4 (Fig. 4.5)
3 mod 4	$y^2 = x^3 + ax$	$(x, y) \mapsto (-x, iy)$, $i^2 = -1$	Eg. 4.1.5 (Fig. 4.6)

Table 6.1: The two types of popular supersingular curves over prime fields.

sizes to serve another purpose: to show how important it is to employ ordinary curves with higher embedding degrees.

Example 6.2.1 (Magma script). We will choose $q \equiv 11 \bmod 12$ so we can define both examples in Table 6.1 over the same field, but also so that the security of these curves in the context of PBC matches the security of the curve with $k = 12$ in Example 6.1.2. For the ECDLP security to be 128 bits, r still only needs to be 256 bits in size. However, since $k = 2$, for $\mathbb{F}_{q^k}$ to be around 3248 bits, q needs to be around 1624 bits:

$q =$425708693169757088196017853607835113595127103859429924930531263283244403251872949802982860038531930965867890444658222153407204383584492024637762799391807569669124814253270947366226515064812665901907204494611177526596015257984009814596057160388672298355821309046798841446111721495601835913381835880170934319890420895521320439930666405003725309562669243847766834546592867695533445054256132471093279787853214492986394176521193456205570309658462204234557728373615304193316916440130004424612327.

Consider $E_1/\mathbb{F}_q : y^2 = x^3 + 314159$ and $E_2/\mathbb{F}_q : y^2 = x^3 + 265358x$. Both curves have order $\#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q) = q + 1 = hr$, where h is a 1369-bit cofactor and r is the 256-bit prime given as

$r =$5789604461865809771178549250434395392663499233282028201972879200395656482006 3.

The distortion maps are defined over $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$, where $i^2 + 1 = 0$ – see Table 6.1 or Examples 4.1.4 and 4.1.5. The huge size of q stresses the importance of adhering to the optimal ratio of $\rho \cdot k$ suggested by Figure 6.1. A rough but conservative approximation of the complexity of field multiplications in the 1624-bit field, compared to the 271-bit field in Example 6.1.2 gives a ratio of at least 25 : 1. Referring back to the discussion of pairing types in Section 4.2, this gives some idea of the computational price one pays when insisting on the

computability of ψ (as well as the other desired properties offered by a Type 1 pairing), rather than adopting a Type 3 pairing and trusting in the heuristics of Chatterjee and Menezes in the absence of such a ψ [CM09, Men09].

We round out this section by remarking that although supersingular elliptic curves are limited to $k \leq 6$, Rubin and Silverberg give a practical way to obtain larger values of k using Weil descent [RS02]. Alternatively, one can employ a higher genus supersingular curve to obtain a higher embedding degree [Gal01, RS02]. As Galbraith remarks however, there are severe efficiency limitations in both scenarios, and we achieve faster pairings in practice by using ordinary pairing-friendly elliptic curves [Gal05, Ch. IX.15].

6.3 Constructing ordinary pairing-friendly curves

There are three main methods of constructing ordinary pairing-friendly elliptic curves. The two most general methods, the Cocks-Pinch [CP01] and Dupont-Enge-Morain [DEM05] algorithms, produce curves with $\rho = 2$, which is more often than not undesirable when compared to the ρ-values obtained by the third method. Moreover, the third method encompasses all constructions that produce *families* of pairing-friendly elliptic curves, which have been the most successful methods of producing curves that are suitable for current and foreseeable levels of security.

All of the constructions in the literature essentially follow the same idea: fix k and then compute integers t, r, q such that there is an elliptic curve $E/\mathbb{F}_q$ with trace of Frobenius t, a subgroup of prime order r, and an embedding degree k. The *complex multiplication method* (CM method) of Atkin and Morain [AM93] can then be used to find the equation of E, provided the CM discriminant D of E is not too large: D is the square-free part of $4q - t^2$, i.e.

$$Df^2 = 4q - t^2, \tag{6.1}$$

for some integer f. Equation (6.1) is often called the CM equation of E, and by "D not too large" we mean D is less than, say 10^{12} [Sut12].

In 2001, Miyaji, Nakabayashi and Takano [MNT01] gave the first constructions of ordinary pairing-friendly elliptic curves. Their method has since been greatly extended and generalised, but all of the constructions of families essentially followed from their idea, which is aptly named the MNT strategy or MNT

criteria [FST10, Gal05]. For some special cases, Miyaji *et al.* used the fact that if k is the (desired) embedding degree, then $r \mid q^k - 1$ implies $r \mid \Phi_k(q)$, since the k-th cyclotomic polynomial $\Phi_k(x)$ is the factor of $x^k - 1$ that does not appear as a factor of any polynomial $(x^i - 1)$ with $i < k$ [Gal05, IX.15.2]. For these cases they were also the first to parameterise *families* of pairing-friendly curves, by writing t, r and q as polynomials $t(x)$, $r(x)$ and $q(x)$ in terms of a parameter x. Miyaji *et al.* focussed on embedding degrees $k = 3, 4, 6$ and assumed that the group order was to be prime, i.e. $r(x) = q(x) + 1 - t(x)$ (from (2.6)). They proved that the only possibilities for $t(x)$ and $q(x)$ (and hence $r(x)$) are

$$\begin{aligned} k = 3: &\quad t(x) = -1 \pm 6x &&\text{and} \quad q(x) = 12x^2 - 1; \\ k = 4: &\quad t(x) = -x, x + 1 &&\text{and} \quad q(x) = x^2 + x + 1; \\ k = 6: &\quad t(x) = 1 \pm 2x &&\text{and} \quad q(x) = 4x^2 + 1. \end{aligned}$$

Example 6.3.1 (Magma script). We kick-start a search for a $k = 4$ (toy) MNT curve with $x = 10$, incrementing by 1 until $q(x) = x^2 + x + 1$ and $r(x) = q(x)+1-t(x)$ (with either of $t(x) = -x$ or $t(x) = x+1$) are simultaneously prime. At $x = 14$, both $q = q(x) = 211$ and $r = r(x) = 197$ (with $t(x) = x+1$) are prime, so we are guaranteed an elliptic curve $E/\mathbb{F}_q$ with r points and embedding degree $k = 4$ (notice $q^4 - 1 \equiv 0 \bmod r$). The CM equation yields $Df^2 = 4q - t^2 = 619$, which itself is prime, so $f = 1$ and thus we seek a curve over $\mathbb{F}_q$ with CM discriminant $D = 619$. The CM method produces one such curve as $E/\mathbb{F}_q : y^2 = x^3 + 112x + 19$. Notice that $\phi_4(q(x)) = q(x)^2 + 1 = (x^2 + 1) \cdot (x^2 + 2x + 2)$, both factors being the possibilities for $r(x)$.

Notice that the toy example above has $\rho = \frac{\log q}{\log r} = \frac{\log 211}{\log 197} = 1.01$. For x of cryptographically large size though, we will get $\rho = 1$ since $q(x) = x^2 + x + 1$ and $r(x) = x^2 + 2x + 2$ or $r(x) = x^2 + 1$ have the same degree. In general parameterised families then, we use the degrees of $q(x)$ and $r(x)$ to state ρ as

$$\rho = \frac{\deg(q(x))}{\deg(r(x))}.$$

A number of works followed the MNT paper and gave useful generalisations of their results. In particular, we mention the work by Barreto *et al.* [BLS02], Scott and Barreto [SB06], and Galbraith *et al.* [GMV07], all three of which obtain more parameterised families by relaxing the condition that the group order is prime

and allowing for small cofactors so that $\#E = hr$. Another observation made by Barreto *et al.* that somewhat simplifies the process is the following: $r \mid \Phi_k(q)$ and $q + 1 - t \equiv 0 \bmod r$ combine to give that $\Phi_k(t-1) \equiv 0 \bmod r$ [BLS02, Lemma 1]. Substituting $hr = q + 1 - t$ into the CM equation in (6.1) gives

$$Df^2 = 4hr - (t-2)^2. \tag{6.2}$$

In Section 3.1 of [BLS02], Barreto *et al.* obtain many nice parameterised families for various k by considering a special case of the above equation with $t(x) = x+1$, $D = 3$ and (since $r \mid \Phi_k(x)$) finding $f(x)$ and $m(x)$ to fit

$$3f(x)^2 = 4m(x)\Phi_k(x) - (x-1)^2. \tag{6.3}$$

We note that curves with CM discriminant $D = 3$ are always of the form $y^2 = x^3 + b$. A convenient solution to Equation (6.3) for $k = 2^i \cdot 3$ is $m = (x-1)^2/3$ and $f(x) = (x-1)(2x^4-1)/3$, for which we can take $r = \Phi_k(x)$. Taking $i = 3$, we give a cryptographically useful example of a BLS (Barreto-Lynn-Scott) curve with $k = 24$.

Example 6.3.2 (Magma script). Following the above description, the BLS family with $k = 24$ is parameterised as $q(x) = (x-1)^2(x^8-x^4+1)/3+x$, $r(x) = \Phi_{24}(x) = x^8-x^4+1$, $t(x) = x+1$. The family has $\rho = \frac{\deg(q(x))}{\deg(r(x))} = 10/8 = 1.25$ and therefore $\rho \cdot k = 30$. Referring back to Figure 6.1, we see that such a curve gives a nice balance between the sizes of r and q^k (the ECDLP and DLP) for pairings at the 256-bit security level. Indeed, at present this family remains the front-runner for this particular security level [Sco11, CLN11]. To find a curve suitable for this level, we need r to be about 512 bits, and since $\deg(r(x)) = 8$, we will start the search for q, r both prime with a 64-bit value; note that $x \equiv 1 \bmod 3$ makes $q(x)$ an integer, so the first such value is $x = 2^{63} + 2$. After testing a number of incremental $x \leftarrow x + 3$ values, $x = 9223372036854782449$ gives $q(x)$ and $r(x)$ as 629 and 505 bit primes respectively. Since $D = 3$ and $E/\mathbb{F}_q : y^2 = x^3 + b$, i.e. there is only one curve constant, we do not need to use the CM method. Instead, it is usually quicker to try successive values of b until we find the correct curve. In this case, $b = 1$ gives $E/\mathbb{F}_q : y^2 = x^3 + 1$ as our pairing-friendly $k = 24$ BLS curve.

Barreto *et al.* [BLS02, §3.2] actually give a more general algorithm which, instead of insisting that $t = x + 1$, takes $t = x^i + 1$. Brezing and Weng [BW05]

found even more useful families by searching with more general polynomials for $t(x)$. Several constructions followed by looking for parameterisations that satisfy the following conditions which define a family [FST10, Def. 2.7] (also see [Fre06, Def. 2.5]):

(i) $r(x)$ is nonconstant, irreducible, and integer-valued with a positive leading coefficient.

(ii) $r(x) \mid q(x) + 1 - t(x)$.

(iii) $r(x) \mid \Phi_k(t(x) - 1)$.

(iv) The parameterised CM equation $Df^2 = 4q(x) - t(x)^2$ has infinitely many integer solutions (x, f).

Referring to condition (iv) above, we say that a family parameterised by $(t(x),\ r(x),\ q(x))$ is a *complete* family if there exists $f(x) \in \mathbb{Q}[x]$ such that $Df(x)^2 = 4q(x) - t(x)^2$. Otherwise, we say the family is *sparse*. We have already seen a curve belonging to the popular Barreto-Naehrig (BN) family in Example 6.1.2. In the following example we look at the BN parameterisations in terms of the above conditions.

Example 6.3.3 (Magma script). Barreto and Naehrig [BN05] discovered that, for $k = 12$, setting the trace of Frobenius t to be $t(x) = 6x^2 + 1$ gives $\Phi_{12}(t(x) - 1) = \Phi_{12}(6x^2) = (36x^4 + 36x^3 + 18x^2 + 6x + 1)(36x^4 - 36x^3 + 18x^2 - 6x + 1)$. This facilitates the choice of $r(x)$ as the first factor $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$, from which taking $q(x)$ as $q(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ means not only that $r(x) \mid q(x) + 1 - t(x)$ (condition (ii) above), but in fact that $r(x) = q(x) + 1 - t(x)$. Thus, when x is found that makes $r(x)$ and $q(x)$ simultaneously prime, we have a pairing-friendly curve with $k = 12$ that has prime order. Not only is the ρ-value $\rho = 1$ ideal, but there are many more reasons why BN curves have received a great deal of attention [DSD07, PJNB11, AKL+11]. Notice that $D = 3$ and $f(x) = 6x^2 + 4x + 1$ satisfies the CM equation (condition (iv) above), so the BN family is a complete family and BN curves are always of the form $y^2 = x^3 + b$.

The last point of Example 6.3.3 is a crucial one. Referring back to Section 4.3, we know that $D = 3$ curves of the form $y^2 = x^3 + b$ admit cubic and sextic twists. Thus, in the case of BN curves where $k = 12$, we can make use of a sextic twist to represent points in $\mathbb{G}_2 \in E(\mathbb{F}_{q^{12}})$ as points in a much smaller subfield on the twist, i.e. in $\Psi^{-1}(\mathbb{G}_2) = \mathbb{G}_2' \in E'(\mathbb{F}_{q^2})$. In general then, when k has

the appropriate factor $d \in \{3, 4, 6\}$, we would like to make use of the highest degree twist possible, so we would prefer our pairing-friendly curves to be of the following two forms:

degree d	curve	j-invariant	CM discriminant	field
$3, 6 \mid k$	$y^2 = x^3 + b$	$j(E) = 0$	$D = 3$	$q \equiv 1 \bmod 3$
$4 \mid k$	$y^2 = x^3 + ax$	$j(E) = 1728$	$D = 1$	$q \equiv 1 \bmod 4$

Table 6.2: Pairing-friendly elliptic curves admitting high-degree twists.

See [Sil09][p. 45] for the definition of the j-invariant of an elliptic curve (and the associated calculations); we simply remark that two elliptic curves $E/\mathbb{F}_q$ and $\tilde{E}/\mathbb{F}_q$ are isomorphic over $\overline{\mathbb{F}}_q$ if and only they have the same j-invariant. Due to the preferences in Table 6.2, our discussion will really only be dealing with curves of j-invariants (respectively CM discriminants) $j \in \{0, 1728\}$ (respectively $D \in \{3, 1\}$). In this respect, we are also very fortunate that most of the best constructions of pairing-friendly families have either $D = 1$ or $D = 3$, depending on the embedding degree they target. In general, a severe loss of efficiency is suffered in pairing computations when choosing a curve that does not offer a high-degree twist, so at any particular security level we tend to focus on the curves whose embedding degrees are suitable, both according to Figure 6.1 and which contain $d \in \{3, 4, 6\}$ as a factor [FST10, §8.2]. Besides, as we will see in the next chapter, there are further efficiency reasons that happily coincide with having $d \mid k$ for $d \in \{3, 4, 6\}$. The equivalence conditions on q in Table 6.2 are to ensure E is ordinary, complementing the supersingular cases in Table 6.1.

Our last example in this chapter belongs to another complete family from the more recent work of Kachisa, Schaefer and Scott [KSS08], who present record-breaking (in terms of the lowest ρ-value) curves for embedding degrees $k \in \{16, 18, 36, 40\}$.

Example 6.3.4 (Magma script). We choose a KSS curve with $k = 16$, which is parameterised by $t(x) = (2x^5 + 41x + 35)/35$, $q(x) = (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 + 625x^2 + 2398x + 3125)/980$ and $r(x) = (x^8 + 48x^4 + 625)/61250$. This family has $\rho = 5/4$, so referring back to Figure 6.1 we see that $\rho \cdot k = 20$ is a nice fit for pairings at the 192-bit security level. Thus, r should be around 384 bits, so starting our search with x around 2^{50} should do the trick (we add the extra two bits to account for the 16-bit denominator of $r(x)$). The polynomials for $q(x)$ and $t(x)$ can only take on integers if $x \equiv \pm 25 \bmod 70$, so we start

with $x \equiv 2^{50} + 21 \equiv 25 \bmod 70$ and iterate accordingly. We soon arrive at $x = 1125899907533845$, which gives a 491-bit q as

$$q = 3340194518359587075607904514504348578130581647867654217642899810042867643534741048241225178436682317003010155280705836842596368221341280505964970897,$$

and a 385-bit prime factor r of $\#E(\mathbb{F}_q)$ as

$$r = 4215918189011304280250800671237881596873006793850195934448558095361634092780222932018149564359414764607793390912163 3.$$

Again, we do not need the CM method to find the curve: we simply start with $a = 1$ in $y^2 = x^3 + ax$ and increment until we find $a = 3$ which gives the correct curve as $E/\mathbb{F}_q : y^2 = x^3 + 3x$. E has embedding degree 16 with respect to r, so the full extension field $\mathbb{F}_{q^k}$ is 7842 bits.

We finish this chapter with two important remarks.

Remark 6.3.1 (Curves for ECC vs. curves for PBC). At the highest level, finding curves that are suitable for ECC really imposes only one condition on our search, whilst finding curves that are suitable for PBC imposes two: in ECC we only look for curves with large prime order subgroups, whilst in PBC we have the added stipulation in that we also require a low embedding degree. Whilst one can search for suitable curves for ECC by checking "random" curves until we come across one with almost prime order, in PBC we require very special constructions (like all those discussed in this chapter) that also adhere to the extra criterion – as we have already discussed, we can not expect to find any pairing-friendly curves by choosing curves at random [BK98]. A major consequence is that in ECC we can specify the underlying field $\mathbb{F}_q$ however we like before randomly looking for a suitable curve over that field. In this case fields can therefore be chosen to take advantage of many low-level optimisations; for example, Mersenne primes achieve very fast modular multiplications which blows out the relative cost of inversions. On the other hand, in PBC we are confined to the values taken by the polynomials $q(x)$ and have limited control over the prime fields we find. Thus, we are not afforded the luxury of many low-level optimisations and this drastically affects the ratios between field operations (inversions/multiplications/squarings/additions). For example, whilst $\mathbb{F}_q$-inversions

in ECC are commonly reported to cost more than 80 $\mathbb{F}_q$-multiplications, the ratio in the context of PBC is nowhere near as drastic [LMN10, AKL+11]. This means we often have to rethink trade-offs between field operations that were originally popularised in ECC.

Remark 6.3.2 (Avoiding pairing-friendly curves in ECC). In the previous remark we said that in ECC we only need to satisfy one requirement (the large prime subgroup), but this is not the full story. In fact, in this context we prefer to choose curves that are strictly not pairing-friendly. After all, in ECC there is no need for a low embedding degree, so choosing a curve that (unnecessarily) has one gives an adversary another potential avenue for attack. Indeed, exploiting curves with low embedding degrees in the context of ECC was the first use of pairings in cryptography – the famous Menezes-Okamato-Vanstone (MOV) [MOV93] and Frey-Rück (FR) [FR94] attacks. Thus, so long as we avoid supersingular curves, the heuristic argument [BK98] tells us that the curves we choose at random will have enormous embedding degrees with overwhelmingly high probability, so this is not a restriction in the sense of Remark 6.3.1.

6.4 Chapter summary

We stressed the importance of finding elliptic curves with large prime order subgroups and small embedding degrees, i.e. pairing-friendly curves. We showed that supersingular curves, whilst easy to find, severely limit the efficiency of pairing computations, particularly at moderate to high levels of security, because they are confined to $k \leq 6$ (and $k \leq 2$ over prime fields). Thus, we turned our focus to the more difficult task of constructing ordinary pairing-friendly elliptic curves, and summarised many landmark results that have enhanced this arena over the last decade. In particular, we gave examples of some of the most notable families of pairing-friendly elliptic curves, some of which have already become widespread in real-world implementations of pairings.

Chapter 7

The state-of-the-art

This chapter summarises the evolution of pairing computation over the last decade. We illustrate the landmark achievements that accelerated early implementations of pairings from "a few minutes" [Men93][1] into current implementations that take less than a millisecond [AKL+11].

Initial improvements in pairing computations were spearheaded by evidence that computing the Tate pairing $f_{r,P}(D_Q)^{(q^k-1)/r}$ is more efficient than computing the Weil pairing $f_{r,P}(D_Q)/f_{r,Q}(D_P)$. At first glance it seems that comparing the two computations amounts to comparing an exponentiation by $(q^k-1)/r$ to a (second) run of Miller's algorithm $f_{r,Q}(D_P)$, and indeed, at levels of security up to 128 bits, this comparison does favour the Tate pairing (cf. [SCA06, Tab. 1-5], [Sco07c]). However, as we will see in Section 7.1, exponentiating by $(q^k-1)/r$ actually facilitates many "Tate-specific" optimisations within the associated Miller loop. It is these enhancements that gave the field of pairing computation its first big boost.

7.1 Irrelevant factors (a.k.a. denominator elimination)

In this section we will work our way to a refined version of Miller's algorithm for pairings over large prime fields, which is mostly due to improvements sug-

[1] As Scott says however, this comparison is unfair – in 1993 there was no incentive to try and optimise the computation past what was needed to apply the MOV attack [MOV93].

gested by Barreto, Kim, Lynn and Scott [BKLS02], and also partly due to Galbraith, Harrison and Soldera [GHS02]. Thus, it is often referred to as the BKLS algorithm [Sco05a, WS07], or sometimes as the BKLS-GHS algorithm [Sco05b, BGOS07]. Our exposition will make use of twisted curves, which we discussed in Section 4.3 and the employment of which is originally due to Barreto, Lynn and Scott [BLS03]. The early works that included Barreto, Lynn and Scott are also culminated in [BLS04].

We start with an observation that allows us to conveniently replace the divisor D_Q with the point Q in the Tate pairing definition. Namely, so long as $k > 1$ and P and Q are linearly independent, then $f_{r,P}(D_Q)^{(q^k-1)/r} = f_{r,P}(Q)^{(q^k-1)/r}$ [BKLS02, Th. 1]. This saves the hassle of defining a divisor equivalent to $D_Q = (Q) - (\mathcal{O})$ with support disjoint to $(f_{r,P})$, but more importantly allows us to simply evaluate the intermediate Miller function at the point Q (rather than two points) in each iteration of Algorithm 5.1.

Example 7.1.1 (Magma script). We reuse the parameters from Example 5.3.1 so a comparison between intermediate values is possible. Thus, let $q = 47$, $E/\mathbb{F}_q : y^2 = x^3 + 21x + 15$, $\#E(\mathbb{F}_q) = 51$, $r = 17$, $k = 4$, $\mathbb{F}_{q^4} = \mathbb{F}_q(u)$ with $u^4 - 4u^2 + 5 = 0$, $P = (45, 23) \in \mathbb{G}_1$ and $Q = (31u^2 + 29, 35u^3 + 11u) \in \mathbb{G}_2$. Thus, the Tate pairing is $e(P, Q) = f_{r,P}(Q)^{(q^k-1)/r} = (32u^3 + 17u^2 + 43u + 12)^{287040} = 33u^3 + 43u^2 + 45u + 39$, which is the same value we got when instead computing $f_{r,P}(D_Q) = f_{r,P}([2]Q)/f_{r,P}(Q)$ in Example 5.3.1. When comparing the fifth columns of both tables, one should keep in mind that the numerator and denominator of the fractions in Example 5.3.1 were themselves both computed as fractions. Indeed, updates in this example are just the denominator of the updates in Example 5.3.1, which gives an indication of how advantageous it is to evaluate the pairing functions at one point (e.g. Q), rather than at a divisor consisting of multiple points (e.g. $([2]Q) - (Q)$). Notice that the values $f_{r,P}(D_Q)$ and $f_{r,P}(Q)$ output after the Miller loops in both examples are not the same, but

i/r_i	steps of Alg. 5.1	point R	update ℓ/v	update at Q $\ell(Q)/v(Q)$	paired value f
	1	$(45, 23)$			1
3/0	3-5	$(12, 16)$	$\frac{y+33x+43}{x+35}$	$\frac{35u^3+36u^2+11u+13}{31u^2+17} = 6u^3 + 19u^2 + 36u + 33$	$6u^3 + 19u^2 + 36u + 33$
2/0	3-5	$(27, 14)$	$\frac{y+2x+7}{x+20}$	$\frac{35u^3+15u^2+11u+18}{31u^2+2} = 39u^3 + 8u^2 + 20u + 18$	$11u^3 + 17u^2 + 24u + 4$
1/0	3-5	$(18, 31)$	$\frac{y+42x+27}{x+29}$	$\frac{35u^3+33u^2+11u+23}{31u^2+11} = 18u^3 + 32u^2 + 41u + 30$	$22u^3 + 34u^2 + 5u + 10$
0/1	3-5	$(45, 24)$	$\frac{y+9x+42}{x+2}$	$\frac{35u^3+44u^2+11u+21}{31u^2+31} = 21u^3 + 26u^2 + 25u + 20$	$8u^3 + 22u^2 + 5u + 27$
	6-10	$\mathcal{O}$	$x + 2$	$= 31u^2 + 31$	$32u^3 + 17u^2 + 43u + 12$
	12	$f_{r,P}(Q) \leftarrow 32u^3 + 17u^2 + 43u + 12$			

the *final exponentiation* maps them to the same element in μ_{17}. This is because $f_{r,P}(D_Q)$ and $f_{r,P}(Q)$ lie in the same coset of $(\mathbb{F}_{q^k}^*)^r$ in $\mathbb{F}_{q^k}^*$, i.e. they are the same element in the quotient group $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$.

We are now in a position to describe the important *denominator elimination* optimisation. Barreto *et al.* were the first to notice that $q-1 \mid (q^k-1)/r$ [BKLS02, Lemma 1], since if $r \mid q-1$ then the embedding degree would be $k=1$. This allows us to write the final exponent as $(q^k-1)/r = (q-1)\cdot c$, which gives $f_{r,P}(Q)^{(q^k-1)/r} = (f_{r,P}(Q)^{q-1})^c$, meaning that any elements of $\mathbb{F}_q$ contributing to $f_{r,P}(Q)$ will be mapped to one under the final exponentiation. Thus, one can freely multiply or divide $f_{r,P}(Q)$ by an element of $\mathbb{F}_q$ without affecting the pairing value [BKLS02, Corr. 1]. When working over supersingular curves with $k=2$, the x-coordinate of Q is defined over $\mathbb{F}_q$ (see any of Examples 4.1.4, 4.1.5 ,4.3.1, 5.2.1). Therefore, the vertical lines appearing on the denominators of Miller's algorithm for the Tate pairing are entirely defined over $\mathbb{F}_q$: the line is a function $x - x_R$ that depends on $P \in E(\mathbb{F}_q)[r]$, which is evaluated at $x_Q \in \mathbb{F}_q$. Thus, in this case the contribution of (each of) the denominators to $f_{r,P}(Q)$ ends up being mapped to 1 under the final exponentiation, so these denominators (the v's in the ℓ/v's – see Steps 5 and 9 in Algorithm 5.1) can be removed from the Miller loop.

For ordinary curves with $k > 2$ however, the x-coordinate of Q will no longer be in the base field $\mathbb{F}_q$, but in some proper subfield $\mathbb{F}_{q^e}$ of $\mathbb{F}_{q^k}$, where $e = k/d$ and d is the degree of the twist employed[2] – see Section 4.3. Here it helps to assume that k is even, i.e. $k = 2\ell$, so that (at the very least) we can take $Q = (x_Q, y_Q)$ where $x_Q \in \mathbb{F}_{q^\ell}$ is such that $y_Q \in \mathbb{F}_{q^k} \setminus \mathbb{F}_{q^\ell}$. Thus, when advancing beyond $k=2$ supersingular curves, Barreto *et al.* generalised the original statement to facilitate the same trick. Namely, that $q^e - 1 \mid (q^k-1)/r$ for any proper factor $e \mid k$ [BLS03, Lemma 5, Corr. 2], so denominators can be omitted from computations in general.

Example 7.1.2 (Magma script). Again, we will continue on from Example 7.1.1 for the sake of a convenient comparison. We simply give an updated table that details the intermediate Miller functions and pairing values subject to denominator elimination. Therefore, $e(P,Q) = f_{r,P}(Q)^{(q^k-1)/r} = (9u^3+10u^2+32u+36)^{(q^k-1)/r} = 33u^3+43u^2+45u+39$, which agrees with the Tate pairing value

[2]When $d=3$ cubic twists are able to be employed for odd k, it is the y-coordinate of Q that is in the subfield; we will treat this in Chapter 4.

i/ r_i	steps of Alg. 5.1	point R	update ℓ	update at Q $\ell(Q)$	paired value f
	1	$(45, 23)$			1
3/0	3-5	$(12, 16)$	$y + 33x + 43$	$35u^3 + 36u^2 + 11u + 13$	$35u^3 + 36u^2 + 11u + 13$
2/0	3-5	$(27, 14)$	$y + 2x + 7$	$35u^3 + 15u^2 + 11u + 18$	$44u^3 + 34u^2 + 3u + 44$
1/0	3-5	$(18, 31)$	$y + 42x + 27$	$35u^3 + 33u^2 + 11u + 23$	$5u^3 + 24u^2 + 21u + 24$
0/1	3-5	$(45, 24)$	$y + 9x + 42$	$35u^3 + 44u^2 + 11u + 21$	$21u^3 + 36u^2 + 9u + 25$
	6-10	$\mathcal{O}$	$x + 2$	$31u^2 + 31$	$9u^3 + 10u^2 + 32u + 36$
	12	$f_{r,P}(Q) \leftarrow 9u^3 + 10u^2 + 32u + 36$			

in Examples 5.3.1 and 7.1.1. Notice again that the value output from the Miller loop is not equal to either of the values output in 5.3.1 or 7.1.1, but rather that all three are equivalent under the relation $a = b$ if $a/b \in (\mathbb{F}_{q^k}^*)^r$.

We now refine Miller's algorithm for the Tate pairing computation subject to the BKLS-GHS improvements. Specifically, notice that the denominators that were on lines 5 and 9 have now gone (under the assumption that k is even), and that the second input is now the point Q, rather than a divisor equivalent to D_Q. Further notice that we have necessarily include the final exponentiation in Algorithm 7.1 since this is what facilitates the modifications. We have also assumed a Type 3 pairing so the coordinates of P and Q lie in fields that allow for denominator elimination. Recall from the discussion at the end of Example 5.3.1, or from Example 7.1.2, that the vertical line joining $(r-1)P = -P$ and P in the last iteration can also be omitted. Thus, an optimised Tate pairing computation will execute the main loop from $i = n-2$ to $i = 1$ before performing a "doubling-only" iteration to finish; we left the main loop to $i = 0$ for simplicity.

7.2 Projective coordinates

Although the optimisations described in the previous section removed the denominators in Step 5 and Step 9 of Algorithm 7.1, $\mathbb{F}_q$-inversions are still apparent in the routine since the affine explicit formulas for the elliptic curve group operations (see Eq. (2.4) and (2.5)) require them. The penalty for performing field inversions in PBC is not as bad as it is in ECC (more on this later), but in any case inversions are still much more costly than field multiplications. In this section we employ the same techniques to avoid field inversions as we did in the context of ECC in Example 2.1.9. Namely, we show how Algorithm 7.1 can become inversion-free if we adopt projective coordinates. In the early days the situation for projective coordinates in the context of pairings was perhaps a little

Algorithm 7.1 The BKLS-GHS version of Miller's algorithm for the Tate pairing.

Input: $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ (Type 3 pairing) and $r = (r_{n-1} \dots r_1 r_0)_2$ with $r_{n-1} = 1$.
Output: $f_{r,P}(Q)^{(q^k-1)/r} \leftarrow f$.

1: $R \leftarrow P$, $f \leftarrow 1$.
2: **for** $i = n - 2$ down to 0 **do**
3: Compute the sloped line function $\ell_{R,R}$ for doubling R.
4: $R \leftarrow [2]R$.
5: $f \leftarrow f^2 \cdot \ell_{R,R}(Q)$.
6: **if** $r_i = 1$ **then**
7: Compute the sloped line function $\ell_{R,P}$ for adding R and P.
8: $R \leftarrow R + P$.
9: $f \leftarrow f \cdot \ell_{R,P}(Q)$.
10: **end if**
11: **end for**
12: **return** $f \leftarrow f^{(q^k-1)/r}$.

unclear [Gal05, IX.14], but nowadays all of the record-breaking implementations (at least up to the 128-bit security level) have exploited the savings offered by working in projective space.

The potential of projective coordinates was mentioned in passing in the early landmark papers [BKLS02, §3.2], [GHS02], but the first detailed investigation was by Izu and Tagaki [IT02]. As Galbraith mentions [Gal05, IX.14], the analysis in [IT02] is misleading, however projective coordinates did not wait too long before more accurate expositions that also endorsed their usefulness surfaced [CSB04, Sco05a]. The following example shows how projective coordinates can be used to achieve an inversion-free version of Miller's algorithm.

Example 7.2.1 (Magma script). In the context of standard ECC operations, we gave the (homogeneous) projective point addition formulas in Example 2.1.9. Thus, here we will give the homogeneous doubling formulas for computing $(X_{[2]R} : Y_{[2]R} : Z_{[2]R}) = [2](X_R : Y_R : Z_R)$ on $E/\mathbb{F}_q : Y^2Z = X^3 + aXZ^2 + bZ^3$ in Step 4 of Algorithm 7.1, together with the formulas for computing the line function $\ell_{R,R}(Q)$ in Step 3. The affine doubling formulas in Equation (2.5) are moved into homogeneous projective space via the substitution $x = X/Z$ and $y = Y/Z$,

which gives:

$$\lambda = \frac{3X_R^2 + Z_R^2}{2Y_R Z_R}; \qquad \nu = -\frac{3X_R^3 + X_R Z_R^2 - 2Y_R^2 Z_R}{2Y_R Z_R^2};$$
$$\frac{X_{[2]R}}{Z_{[2]R}} = \frac{-8X_R Y_R^2 Z_R + 6X_R^2 Z_R^2 + 9X_R^4 + Z_R^4}{4Y_R^2 Z_R^2};$$
$$\frac{Y_{[2]R}}{Z_{[2]R}} = -\frac{8Y_R^4 Z_R^2 + Z_R^6 - 12X_R Z_R^3 Y_R^2 - 36X_R^3 Z_R Y_R^2 + 27Z_R^2 X_R^4 + 9Z_R^4 X_R^2 + 27X_R^6}{8Y_R^3 Z_R^3},$$

where $\ell : y - (\lambda x + \nu)$ is still an affine line tangent to E at the point R. It is again the ability to multiply by factors in proper subfields of $\mathbb{F}_{q^k}$ that allows us to arrive at an inversion-free routine. Namely, we clear the denominators of λ and ν through multiplication by $2Y_R Z_R^2$, so the line ℓ becomes

$$\ell : (2Y_R Z_R^2) \cdot y - ((3X_R^2 Z_R + Z_R^3) \cdot x - (3X_R^3 + X_R Z_R^2 - 2Y_R^2 Z_R)),$$

which will be evaluted at $y = y_Q$ and $x = x_Q$. Note that since Q remains fixed throughout the routine, there is no need to cast it into projective space. Finally, setting $Z_{[2]R} = 8Y_R^3 Z_R^3$ and updating the numerator of $X_{[2]R}$ above allows us to compute $(X_{[2]R} : Y_{[2]R} : Z_{[2]R})$ from $(X_R : Y_R : Z_R)$ without any $\mathbb{F}_q$-inversions. Thus, we have an inversion-free way to proceed through the Miller doubling stage (Steps 3-5 of Algorithm 7.1), and performing the analogous procedure for the Miller addition stage (Steps 7-9) will give an inversion-free Miller loop.

7.3 Towered extension fields

This section discusses efficient methods of constructing the full extension field $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$, where the ultimate goal is to minimise the cost of the arithmetic in $\mathbb{F}_{q^k}$. Indeed, the majority of operations within the pairing algorithm take place in the full extension field, which is far more expensive to work in than its proper subfields, so the complexity of Miller's algorithm heavily depends on the complexity of the associated $\mathbb{F}_{q^k}$-arithmetic.

So far we have been using one irreducible degree k polynomial to construct $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$. This has been satisfactory, since our small examples have mostly had embedding degrees $k = 2$ or $k = 3$, where we have no other option but to use polynomials of degree two and three to respectively construct $\mathbb{F}_{q^k}$. However, for large values of k, which will be composite in all cases of interest to us, there is an a natural alternative which turns out to be much faster. This idea was first

put forward by Koblitz and Menezes [KM05], who proposed using embedding degrees of the form $k = 2^i 3^j$ and building up to $\mathbb{F}_{q^k}$ using a series of quadratic and cubic extensions that successively *tower* up the intermediate fields. For such k, they show that if $q \equiv 1 \bmod 12$ and if α is neither a square or cube in $\mathbb{F}_q$, then the polynomial $x^k - \alpha$ is irreducible in $\mathbb{F}_q[x]$ [KM05, Th. 1]. This means that the tower can be constructed by a sequence of Kummer extensions: this involves successively adjoining the square root or cube root α, then the square root or cube root of that, and so on.

Example 7.3.1 (Magma script). Let $q = 97$, and consider constructing $\mathbb{F}_{q^{12}}$ using $\alpha = 5$ which is a non-square and non-cube in $\mathbb{F}_q$, so that $\mathbb{F}_{q^{12}}$ can be constructed directly as $\mathbb{F}_{q^{12}} = \mathbb{F}_q[X]/(X^{12} - \alpha)$. Choosing instead a tower of quadratic and cubic extensions, we could construct $\mathbb{F}_{q^{12}}$ as

$$\mathbb{F}_q \xrightarrow{\beta^2-\alpha} \mathbb{F}_{q^2} \xrightarrow{\gamma^3-\beta} \mathbb{F}_{q^6} \xrightarrow{\delta^2-\gamma} \mathbb{F}_{q^{12}}.$$

We show a random element in $\mathbb{F}_{q^{12}}$:

$$\big((79\beta + 63)\gamma^2 + (29\beta + 63)\gamma + (38\beta + 27)\big)\,\delta + (63\beta + 22)\gamma^2 + (93\beta + 10)\gamma + 75\beta + 10.$$

Observe what happens if, instead of performing multiplications in $\mathbb{F}_{q^{12}}$ over $\mathbb{F}_q$, we start by performing multiplications over $\mathbb{F}_{q^6}$. Writing $a, b \in \mathbb{F}_{q^{12}}$ over $\mathbb{F}_{q^6}$ gives $a = a_0 + a_1\delta$ and $b = b_0 + b_1\delta$, with $a_0, a_1, b_0, b_1 \in \mathbb{F}_{q^6}$. Thus, $a \cdot b = (a_0 b_0 - a_1 b_1 \gamma) + (a_0 b_1 + a_1 b_0)\delta$, where each of the components inside the parentheses are in $\mathbb{F}_{q^6}$. To perform each of the multiplications in $\mathbb{F}_{q^6}$, we then work over $\mathbb{F}_{q^2}$, so for example we would need to compute a multiplication between $a_0 = a_{0,0} + a_{0,1}\gamma + a_{0,2}\gamma^2$ and $b_0 = b_{0,0} + b_{0,1}\gamma + b_{0,2}\gamma^2$, where each component $a_{0,i}$ and $b_{0,i}$ is in $\mathbb{F}_{q^2}$. In this way the operations filter down the tower until we are performing multiplications in $\mathbb{F}_q$.

The computational advantage of adopting a tower of extensions may not be immediately evident. Namely, suppose we were to analyse the complexity of the $\mathbb{F}_{q^{12}}$ multiplication in Example 7.3.1. If we were to employ the naive "schoolbook" method of multiplying two extension field elements, which operates component-wise, then an $\mathbb{F}_{q^{12}}$ multiplication computed directly over $\mathbb{F}_q$ would cost 144 $\mathbb{F}_q$ multiplications. If we instead descend down the tower employing schoolbook multiplication, then an $\mathbb{F}_{q^{12}}$ multiplication would cost 4 $\mathbb{F}_{q^6}$ multiplications, each of which would cost 9 $\mathbb{F}_{q^2}$ multiplications, with each of these costing 4 mul-

tiplications in $\mathbb{F}_q$, giving $4 \cdot 9 \cdot 4 = 144$ base field multiplications in this case too. However, one of the reasons that the towered approach betters a direct extension to $\mathbb{F}_{q^k}$ is because there exist much better (than schoolbook) methods of performing arithmetic in quadratic and cubic extensions. Specifically, the Karatsuba method [KO63] for quadratic extensions allows us to compute multiplications in $\mathbb{F}_{q^{2u}}$ using 3 multiplications in $\mathbb{F}_{q^u}$, or to compute a squaring in $\mathbb{F}_{q^{2u}}$ using only 2 multiplications in $\mathbb{F}_{q^u}$. The same method applied to cubic extensions allows us to compute multiplications in $\mathbb{F}_{q^{3u}}$ using only 6 multiplications in $\mathbb{F}_{q^u}$ (rather than 9), and squarings in $\mathbb{F}_{q^{3u}}$ using 6 $\mathbb{F}_{q^u}$-squarings (which are faster than $\mathbb{F}_{q^u}$-multiplications in general). There are also other methods and variations which are competitive for these small extensions, such as the Toom-Cook method [Too63, CA66], which computes an $\mathbb{F}_{q^{3u}}$ multiplication using only 5 $\mathbb{F}_{q^u}$ multiplications, but this requires a substantially higher number of additions. A helpful report that compares all of these methods in the contexts of pairings is given by Devegili *et al.* [DOSD06]. Referring back to the examples above, and this time descending down the tower using Karatsuba multiplications for the quadratic and cubic extensions gives that $\mathbb{F}_{q^{12}}$ multiplications now cost $3 \cdot 6 \cdot 3 = 54$ $\mathbb{F}_q$ multiplications; a huge improvement over the schoolbook method. We note that a different ordering of the quadratic and cubic towers from $\mathbb{F}_q$ to $\mathbb{F}_{q^{12}}$ could be chosen, and that this would give the same number of $\mathbb{F}_q$ multiplications for a multiplication in $\mathbb{F}_{q^{12}}$, but that there are certainly reasons (other than the twisted curve) that we would prefer one tower over another.

It could potentially be misleading however, to argue that the low number of $\mathbb{F}_q$ multiplications offered by degree 2 and 3 Karatsuba-like methods is what makes the towered extensions preferable to a direct extension. Indeed, the Karatsuba and Toom-Cook algorithms generalise to extensions of any degree [WP06], [Ber01, §6]. In fact, generalised Toom-Cook theoretically guarantees that we will be able to perform the $\mathbb{F}_{q^{12}}$ multiplication from the above example (via a direct extension) using only 23 $\mathbb{F}_q$ multiplications, which is less than half the number of $\mathbb{F}_q$ multiplications used in our towered Karatsuba approach. However, such high-degree generalisations require an enormous number of $\mathbb{F}_q$ additions, and the theoretical number of multiplications they save is nowhere near enough to offset this deficit. Thus, technically speaking, it is in the saving of $\mathbb{F}_q$-additions that the towered approach gains its advantage. Indeed, the additions encountered when performing the highest level multiplications at the

top most sub-extension of the tower filter down linearly to $\mathbb{F}_q$, whilst performing $\mathbb{F}_{q^k}$-arithmetic via a direct extension blows the number of additions out (at the very least) quadratically.

Given the simple test to determine irreducibility of the binomial $x^k - \alpha$ when $q \equiv 1 \bmod 12$ and $k = 2^i 3^j$ above, Koblitz and Menezes defined a *pairing-friendly field* to be a prime field with characteristic q of this form. However, given the number of conditions already imposed on the search for pairing-friendly curves, Benger and Scott argue that this extra restriction is unnecessary [BS10]. They relax this constraint and introduce the notion of *towering-friendly fields*: a field $\mathbb{F}_{q^m}$ is called towering-friendly if all prime divisors of m also divide $q - 1$. For such fields, they invoke Euler's conjectures to give an irreducibility theorem that facilitates all intermediate subextensions to be constructed via a binomial.

Loop shortening has played a major role in the evolution of pairing computation. Indeed, the series of landmark works that are summarised in this section have an impressive evolution of their own. Duursma and Lee [DL03] were the first to show that, in special cases, a bilinear pairing can be obtained without iterating Miller's algorithm as far as the large prime group order r. Barreto *et al.* [BGOS07] generalised this observation to introduce the η_T pairing (the *eta pairing*); a pairing which achieves a much shorter loop length (than r) on any supersingular curve. Hess, Smart and Vercauteren [HSV06] simplified and extended the η_T pairing to ordinary curves, introducing the *ate pairing*, whose loop length is $T = t - 1$, where t is the trace of the Frobenius endomorphism (see Eq. (2.6)), which is much smaller than r in general cases of interest. A number of authors followed this work with observations that in many cases we can do even better than the ate pairing. This included the introduction of the *R-ate pairing* [LLP09], as well as other optimised variants of the ate pairing [MKHO07]. Vercauteren [Ver10] culminated all of these works and introduced the notion of *optimal pairings*, conjecturing a lower bound on the loop length required to obtain a bilinear pairing on any given curve, and showing how to achieve it in many cases of interest. His conjecture was proven soon after by Hess, who drew a line under all the loop-shortening work to date, putting forward a general framework that encompasses all elliptic curve pairings [Hes08].

Our intention in this section is to bring the reader up to speed with optimal pairings, by picking a few examples that illustrate key concepts. For the sake of simplicity, we are forced to skip past some of the key works mentioned in

the last paragraph; in particular, we will not present the η_T pairing that targets supersingular curves, since it is most suited to curves over fields of characteristic 2 and 3. We will also not be giving examples of the works that came between the ate and optimal ate pairing papers (e.g. [MKHO07, LLP09]), in hope that the reader will not have too much trouble following an immediate generalisation.

At a high level, the notion of loop shortening makes use of two observations. Firstly, recall from Chapter 2 (in particular Example 2.2.11), that appropriate endomorphisms on E compute some multiple $[\lambda]P$ from P, which essentially allow us to "skip ahead" in the fundamental computation of $[m]P$ from P. Just as they can be used to shorten the double-and-add loop for scalar multiplications in ECC, efficient endomorphisms can be used to shorten the Miller loop in PBC. The second observation is that, given any two bilinear pairings on E, their product or quotient will also give a bilinear pairing. More generally, we can say that if $e_1, ..., e_n$ are bilinear pairings on E, then $\prod_i e_i^{j_i}$ $(j_i \in \mathbb{Z})$ will also be a bilinear pairing [ZZH08a, Corr. 1].

We start with an example of Scott's idea [Sco05b], which came from the first paper to look at loop shortening on any type of ordinary curve. He looked at a special case of ordinary curves called *not supersingular curves* (NSS). These should not be confused with the more general term non-supersingular, which (by definition) means all ordinary curves. NSS curves are a special type of ordinary curve, but they cover the cases that are most useful in the context of pairings. In fact, we have already seen NSS curves, as they are precisely the curves described in Table 6.2. Essentially, the modularity conditions imposed on the curves $y^2 = x^3 + b$ and $y^2 = x^3 + ax$ in Table 6.1 is what makes them supersingular, because these conditions force the maps ϕ described in that table to be defined over the extension field – i.e. these congruences make ϕ a distortion map. On the other hand, the alternative modularities on the same curves in Table 6.2 mean that the associated ϕ's are defined over $\mathbb{F}_q$. Thus, Scott starts with the motivating question: *under these circumstances, what becomes of these distortion maps?* The rest of his paper responds by showing that they are useful, not as distortion maps, but rather as efficient endomorphisms on E. The following example does not give the details of Scott's algorithm; it merely hints towards it by showing the potential of the endomorphisms ϕ on an NSS curve.

Example 7.3.2 (Magma script). Taking $x = -1$ generates the smallest BN curve (see Example 6.3.3 for the polynomials) with $q = 19$, $E/\mathbb{F}_q : y^2 = x^3 + 2$

and $r = 13$ as the group order. It is clearly an NSS curve (see Table 6.2 or [Sco05b, Eq. 4]). The non-trivial cube roots of unity are defined over $\mathbb{F}_q$, and are $\zeta_3 = 7$ and $\zeta_3^2 = 11$. They both define a different endomorphism on E (e.g. $\zeta_3 : (x, y) \mapsto (\zeta_3 x, y)$) which corresponds to a different scalar multiplication λ, i.e. $(\zeta_3 x, y) = [\lambda](x, y)$. The two different λ's are the solutions of $\lambda^2 + \lambda + 1 = 0 \bmod r$, which comes from $\lambda^3 \equiv 1 \bmod r$ matching $\zeta_3^3 = [1]$ in $\mathrm{End}(E)$, so $\lambda_1 = 9$ and $\lambda_2 = 3$ correspond to ζ_3 and ζ_3^2 respectively. Miller's algorithm would usually double-and-add to compute $[r]P = [\lambda^2 + \lambda + 1]P = [\lambda]([\lambda]P + P) + P$. However, for $P = (x, y)$, the endomorphism allows us to easily calculate the point $[\lambda]P + P = (-(\lambda + 1)x, -y)$. Thus, if we store the values of the points in the $n = \lfloor \log_2 \lambda \rfloor$ doublings that build up to $[\lambda]P$, the values of the points in the second n doublings can be found at the cost of a single multiplication. This is already more efficient, but Scott notices that since the points are related, the lines they contribute in the point doubling phase of Miller's algorithm are similarly related. Namely, the contribution to the pairing value in the first n iterations is $(y_Q - y_i) - m_i(x_Q - x_i)$, where (x_i, y_i) is the point $[2^i]P$, and m_i is the line slope resulting for the point doubling (we use m in this example because λ is already taken). It follows (see [Sco05b, §5]) that the contribution to the pairing value from the final n doublings will be $(-y_Q - y_i) - m_i(\lambda x_Q - x_i)$. This means we only need to loop as far as $n = \lfloor \log_2 \lambda \rfloor$ (rather than $2n = \lfloor \log_2 \lambda^2 \rfloor$) to get all the information we need. See Scott's paper for the algorithm description that ties all this together, where he deals with cases where $\lambda = 2^a + 2^b$. Thus, to finish our example with the algorithm write $\lambda_1 = 2^{a_1} + 2^{b_1}$ and $\lambda_2 = 2^{a_2} + 2^{b_2}$ with $a_1 = 3$, $b_1 = 0$, $a_2 = 1$, $b_2 = 0$.

The ϕ maps on NSS curves clearly offer an advantage, but there is another endomorphism we have already seen that turns out to be much more powerful. Namely, the ate pairing makes use of the Frobenius endomorphism π on E. A key observation is that the Frobenius endomorphism acts trivially on elements in the base field, i.e. $\pi(P) = P$ in $\mathbb{G}_1$, so we instead look at using the trace-zero subgroup $\mathbb{G}_2$ where π acts non-trivially. Here $\pi(Q) = [q](Q)$, but since $[q](Q) = [t - 1](Q)$, we have $\pi(Q) = [T](Q)$ (recall that $T = t - 1$). Hess, Smart and Vercauteren [HSV06] use this endomorphism to derive the ate pairing a_T, which is a map

$$a_T : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_T,$$

defined as

$$a_T(Q, P) = f_{T,Q}(P)^{(q^k-1)/r}.$$

It helps to see a brief sketch of their proof as follows. We show that a_T is bilinear by relating its value $f_{T,Q}(P)^{(q^k-1)/r}$ to the Tate pairing (with Q as the first argument), which we already know is bilinear. Since $q \equiv T \bmod r$, $T^k \equiv 1 \bmod r$ (because k is the embedding degree), so write $mr = T^k - 1$ for some m. Recall the Tate pairing (with Q as the first argument) as $e(Q, P) = f_{r,Q}(P)^{(q^k-1)/r}$, which (under simple properties of divisors) means $e(Q, P)^m = f_{mr,Q}(P)^{(q^k-1)/r} = f_{T^k-1,Q}(P)^{(q^k-1)/r}$. We can then (again using simple properties of divisors) split this into a product of $f_{T,[T^i]Q}(P)$, each of which is raised to an appropriate exponent. Since $Q \in \mathbb{G}_2$, each of these $[T^i]Q$'s is the same as $\pi^i(Q)$, and since π is purely inseparable of degree q, all of the values $f_{T,[T^i]Q}(P)$ in the product become $f^{q^i}_{T,Q}(P)$, so we can clean up the exponent to get $e(Q, P) = a_T(Q, P)^v$. The exponent v does not divide r in general, so the bilinearity of the ate pairing follows from that of the Tate pairing (see [HSV06, Th. 1] for the full details).

Since there is a final exponentiation, the optimisations that transformed Miller's algorithm into the BKLS version still apply, so we only need to update the input definitions in Algorithm 7.1. Namely, r becomes T, P and Q (from $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively) switch roles. For no other reason than for ease of future reference, we write these updates in an ate-specific version below. Note that if $T = t - 1 < 0$, then it is fine to take $T = |T|$ [Ver10, §C]. There is only one trick that was used in the Tate pairing that does not carry across to the ate setting. Namely, we can no longer ignore the last bit in the final iteration like we did in Section 7.1, because if an addition occurs in the final iteration it will now be a sloped line, whilst in the Tate pairing the last addition line joined P and $[r - 1]P = -P$ and was therefore vertical.

Example 7.3.3 (Magma script). It helps to immediately see the difference between the ate and Tate pairing, so we will continue on from Example 7.1.2: $q = 47$, $E/\mathbb{F}_q : y^2 = x^3 + 21x + 15$, $\#E(\mathbb{F}_q) = 51$, $r = 17$, $k = 4$, $\mathbb{F}_{q^4} = \mathbb{F}_q(u)$, $u^4 - 4u^2 + 5 = 0$, $P = (45, 23) \in \mathbb{G}_1$ and $Q = (31u^2 + 29, 35u^3 + 11u) \in \mathbb{G}_2$. The trace of Frobenius is $t = -3$, so take $T = 4$. Thus, we will compute the ate pairing via Algorithm 7.2 with only two doublings. We have combined the indeterminate function ℓ and its evaluation $\ell(P)$ at P into the same column to fit the table in. Thus, the ate pairing a_T is computed as $a_T(Q, P) = f_{r,Q}(P)^{(q^k-1)/r} =$

Algorithm 7.2 The BKLS-GHS version of Miller's algorithm for the ate pairing.

Input: $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ (Type 3 pairing) and $T = (T_{n-1} \dots T_1 T_0)_2$ with $T_{n-1} = 1$.
Output: $f_{T,Q}(P)^{(q^k-1)/r} \leftarrow f$.

1: $R \leftarrow Q$, $f \leftarrow 1$.
2: **for** $i = n-2$ down to 0 **do**
3: Compute the sloped line function $\ell_{R,R}$ for doubling R.
4: $R \leftarrow [2]R$.
5: $f \leftarrow f^2 \cdot \ell_{R,R}(P)$.
6: **if** $r_i = 1$ **then**
7: Compute the sloped line function $\ell_{R,Q}$ for adding R and Q.
8: $R \leftarrow R + Q$.
9: $f \leftarrow f \cdot \ell_{R,Q}(P)$.
10: **end if**
11: **end for**
12: **return** $f \leftarrow f^{(q^k-1)/r}$.

i/ R_i	steps of Alg. 5.1	point R	update (ℓ); update at P ($\ell(P)$)	paired value f
	1	$(31u^2 + 29, 35u^3 + 11u)$		1
1/0	3-5	$(7u^2 + 25, 37u^3 + 28u)$	$y + (u^3 + 32u)x + 42u^3 + 15u$; $40u^3 + 45u + 23$	$40u^3 + 45u + 23$
0/0	3-5	$(16u^2 + 12, 6u^3 + 24u)$	$y + (28u^3 + 22u)x + 17u^3 + 26u$; $8u^3 + 29u + 23$	$44u^3 + 24u^2 + 41u + 31$
	12		$f_{r,Q}(P) \leftarrow 44u^3 + 24u^2 + 41u + 31$	

$$(44u^3 + 24u^2 + 41u + 31)^{287040} = 21u^3 + 37u^2 + 25u + 25.$$

Notice the price we pay for the much shorter loop in the ate pairing, in that it is now the first argument of the pairing (Q) that is defined over the larger field, so the elliptic curve operations (doublings/additions) and line function computations are now taking place in $\mathbb{F}_{q^k}$. For example, compare the second and third columns of the table in Example 7.3.3 to the table in Example 7.1.2. It is here that the power of a high-degree twist really aids our cause. Namely, utilising the twisting isomorphism allows us to move the points in $\mathbb{G}_2$, which is defined over $\mathbb{F}_{q^k}$, to points in $\mathbb{G}_2'$, which is defined over the smaller field $\mathbb{F}_{q^{k/d}}$. In Example 7.3.3 above where $k = 4$, the maximum degree twist permitted by E is $d = 2$, so we could have performed the point operation and line computations in $\mathbb{F}_{q^{k/2}} = \mathbb{F}_{q^2}$. However, if the curve had have been of the form $y^2 = x^3 + ax$, we could have utilised a $d = 4$ quartic twist (see Section 4.3) and performed these operations all the way down in the base field $\mathbb{F}_q$; i.e. in this case we would pay no price for a much smaller loop. In general though, provided we make use of high-degree twists in the ate pairing, then the price we pay in doing more work

(per iteration) in the larger field is nowhere near enough to offset the savings we gain through having a much shorter loop, meaning that the ate pairing (or one of its variants) is much faster than the Tate pairing. We now turn to describing optimal pairings. Vercauteren [Ver10] begins with the observation that the ate pairing a_T corresponding to $T \equiv q \bmod r$ is a special case of the pairing a_{λ_i} that is obtained by taking any power $\lambda_i \equiv q^i \bmod r$; some specific consequences of this observation were previously considered in [MKHO07, ZZH08b]. Since λ_i corresponds to the loop length of the pairing a_{λ_i}, we would like it to be as small as possible. Thus, we would like to find the smallest value of $q^i \bmod r$ ($i \in \mathbb{Z}$), and since $q^k \equiv 1 \bmod r$, finding the smallest a_{λ_i} would only require testing the possibilities up to $k-1$ ($i = k$ clearly gives the trivial degenerate pairing). However, Vercauteren actually does much better than this by observing that since $q^i \bmod r$ induces a bilinear pairing a_{λ_i}, then any linear combination of $\sum_{i=0}^{l} c_i q^i \equiv 0 \bmod r$ gives rise to a bilinear pairing

$$(Q, P) \mapsto \left(\prod_{i=0}^{l} f_{c_i,Q}^{q^i}(P) \cdot \prod_{i=0}^{l-1} \ell_i \right)^{(q^k-1)/r}, \tag{7.1}$$

where the ℓ_i are simple "one-off" line functions (chords) that are needed to make the bilinearity hold – see [Ver10, Eq. 7] for details. Also, the exponentiations of each of the (at most $\ell + 1$) line functions to the power of q^i should not concern us, as these are just repeated applications of the Frobenius endomorphism in $\mathbb{G}_T$, which is essentially cost-free (more on this in Section 7.5). The main point to note is that the loop lengths of the Miller functions $f_{c_i,Q}$ are the c_i. Thus, we would like to find a multiple mr of r with a base-q expansion $mr = \sum_{i=0}^{l} c_i q^i$ that has the smallest c_i coefficients possible. Vercauteren proceeds naturally by posing this search as a lattice problem, i.e. that such small c_i are obtained by solving for short vectors in the following lattice

$$L = \begin{pmatrix} r & 0 & 0 & \dots & 0 \\ -q & 1 & 0 & \dots & 0 \\ -q^2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \\ -q^{\varphi(k)-1} & 0 & \dots & 0 & 1 \end{pmatrix}, \tag{7.2}$$

which is spanned by the rows, and where $\varphi(k)$ is the Euler phi function of k.

He then invokes Minkowski's theorem [Min10] to show that there exists a short vector $(v_1, ..., v_{\varphi(k)-1})$ in L such that $\max_i |v_i| \leq r^{1/\varphi(k)}$. Thus, we have an upper bound on the largest Miller loop length that will be encountered when computing the pairing in (7.1). Vercauteren uses this bound to define an optimal pairing [Ver10, Def. 3]: $e(\cdot, \cdot)$ is called an *optimal pairing* if it can be computed in $\log_2 r/\varphi(k) + \epsilon$ Miller iterations, with $\epsilon \leq \log_2 k$. He subsequently conjectures that any bilinear pairing on an elliptic curve requires at least $\log_2 r/\varphi(k)$ Miller iterations. Following [Ver10, Def. 3], Vercauteren also notes that the reason that the dimension of L is $\varphi(k)$ is because we really only need to consider q^i up to $q^{\varphi(k)-1}$. This is due to that fact that $\Phi_k(q) \equiv 0 \bmod r$ implies that q^j with $j > \varphi(k)$ can be written as linear combinations of the q^i $(i \leq \varphi(k) - 1)$ with small coefficients, which means only these q^i should be considered linearly independent.

Before giving examples, we mention a caveat. Observe that $\max_i |c_i| \leq r^{1/\varphi(k)}$ does not imply that the lower bound is met, since the number of Miller iterations required is given by $\sum_i \log_2 c_i$. However, we will be searching for small vectors in the lattice L, where q and r come from families and are therefore given as polynomials $q(x)$ and $r(x)$. Therefore, the c_i in the short vectors will themselves be polynomial expressions $c_i(x)$, meaning that the Miller functions $f_{c_i(x),Q}$ in (7.1) will typically follow from $f_{x,Q}$.

We will illustrate with three families that were used as examples in Section 6. Vercauteren gives more examples. Magma has a built in algorithm `ShortestVectors`() that serves our purpose, but the code we use in the following three examples was written by Paulo Barreto, and passed on to us by Luis Dominguez Perez.

Example 7.3.4 (Magma script). Recall the parameterisations for $k = 12$ BN curves from Example 6.3.3: $t(x) = 6x^2 + 1$, $q(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ and $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$. These were actually used to generate the curve in Example 6.1.2, with $x = 9453956337761452438$ being 67 bits, which generated a 271-bit q and r. Observe that Miller's algorithm to compute $f_{r,P}(Q)$ in the Tate pairing would therefore require around 270 iterations. Alternatively, $t = t(x)$ is 137 bits, so computing the ate pairing $a_T(Q, P) = f_{T,Q}(P)^{(q^k-1)/r}$ would require around 136 iterations. However, Vercauteren's bound suggests we can do even better: since $\varphi(12) = 4$, our loop can be reduced by a factor of 4, i.e. we should require $\log_2 r/4 \approx 68$ iterations. Following (7.2) then, we seek short

vectors in the lattice

$$L = \begin{pmatrix} 36x^4+36x^3+18x^2+6x+1 & 0 & 0 & 0 \\ -6x^2 & 1 & 0 & 0 \\ 36x^3+18x^2+6x+1 & 0 & 1 & 0 \\ 36x^3+24x^2+12x+3 & 0 & 0 & 1 \end{pmatrix},$$

where the $-q(x)^i$ down the first column were immediately reduced modulo $r(x)$. Some short vectors in L are $V_1(x) = (6x+2, 1, -1, 1)$, $V_2(x) = (6x+1, 6x+3, 1, 0)$, $V_3(x) = (-5x-1, -3x-2, x, 0)$, $V_4(x) = (2x, x+1, -x, x)$. In reference to the point we made before this example, we prefer the short vectors with the minimum number of coefficients of size x, so choosing $V_1(x)$ and computing the optimal ate pairing $a_{V_1(x)}$ following (7.1) gives

$$\begin{aligned} a_{V_1(x)} &= (f_{6x+2,Q}(P) \cdot f_{1,Q}(P) \cdot f_{-1,Q}(P) \cdot f_{1,Q}(P) \cdot M)^{(q^k-1)/r}, \\ &= (f_{6x+2,Q}(P) \cdot M)^{(q^k-1)/r}, \end{aligned}$$

where $f_{1,Q} = 1$ and $f_{-1,Q} = 1/f_{1,Q}v_Q$ (which disappears in the final exponentation) can be discarded, and M is a product of 3 simple line functions that are computed easily – this example is in [Ver10, IV.A], where M is defined. The only Miller loop we need to compute is $f_{6x+2,Q}(P)$, which for our x-value, is 69 bits, meaning the optimal pairing indeed requires $\log_2 r/4 \approx 68$ iterations. Notice then, the difference between the ease of using $V_1(x)$ compared to any of the other short vectors above, which all suggest more than one Miller loop.

Example 7.3.5 (Magma script). Recall the parameterisations for $k = 16$ KSS curves from Example 6.3.4 as $t(x) = (2x^5 + 41x + 35)/35$, $q(x) = (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 + 625x^2 + 2398x + 3125)/980$ and $r(x) = (x^8 + 48x^4 + 625)/61250$. For any x-value, the Tate pairing requires computing the function $f_{x^8+48x^4+625,P}(Q)$, whilst the ate pairing computes the function $f_{(2x^5+41x+35)/35,Q}(P)$. Since $\varphi(k) = 8$, the ate pairing is not optimal, i.e. $\log_2 r/8$ should have an optimal pairing loop length of order $O(x)$, not $O(x^5)$. Thus, we

look for short vectors in the lattice

$$L = \begin{pmatrix} x^8+48x^4+625 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2x^5-41x & 35 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4x^6+117x^2 & 0 & 175 & 0 & 0 & 0 & 0 & 0 \\ 2x^7-29x^3 & 0 & 0 & 875 & 0 & 0 & 0 & 0 \\ 1x^4+24 & 0 & 0 & 0 & 7 & 0 & 0 & 0 \\ -1x^5-38x & 0 & 0 & 0 & 0 & 35 & 0 & 0 \\ -3x^6-44x^2 & 0 & 0 & 0 & 0 & 0 & 175 & 0 \\ 11x^7+278x^3 & 0 & 0 & 0 & 0 & 0 & 0 & 875 \end{pmatrix}.$$

A nice short vector is $V(x) = (x, 1, 0, 0, 0, -2, 0, 0)$, so indeed an optimal pairing is

$$a_{V(x)} = (f_{x,Q}(P) \cdot f_{-2,Q}(P) \cdot M)^{(q^k-1)/r},$$

where M is again a product of simple one-off lines, and we can compute $f_{-2,Q}(P)$ as $1/f_{2,Q}(P)$, since the vertical line that makes two equal evaporates in the final exponentiation. Note that $f_{2,Q}(P)$ is simply the first doubling of Q at P, and that $f_{x,Q}(P)$ is the only Miller loop required.

Example 7.3.6 (Magma script). Recall the parameterisations for a $k = 24$ BLS curve from Example 6.3.2 as $t(x) = x + 1$, $q(x) = (x-1)^2(x^8 - x^4 + 1)/3 + x$ and $r(x) = \Phi_{24}(x) = x^8 - x^4 + 1$. The Tate pairing requires the computation $f_{x^8-x^4+1,P}(Q)$ whilst the ate pairing computes $f_{x,Q}(P)$. Since $\varphi(k) = 8$, the ate pairing is already optimal, i.e. it has a loop length of $\log_2(r)/8$. In cases when the ate pairing is not optimal, like the previous two examples, it is common that other variants like the R-ate pairing of [LLP09] also achieve optimality. For example, Scott uses the R-ate pairing to achieve optimality for $k = 12$ and $k = 18$ implementations targeting the 128 and 192-bit security levels [Sco11, Table 1].

7.4 Low Hamming weight loops

This short section describes a more obvious optimisation to Miller's algorithm. This trick was suggested in the very early papers on pairing computation, but for reasons that will become clear in a moment, we have delayed its introduction in this section until after we described the ate and optimal ate pairings. Regardless of the pairing-based protocol, the loop length of the pairing is known publicly;

therefore, unlike ECC where we try to avoid special choices of scalars that might give attackers unnecessary advantage, in PBC there is no problem in specialising the choice of the loop length. In this light, it is advantageous to use curves where the loop length has a low Hamming weight, thus minimising the number of additions incurred in Miller's algorithm.

For supersingular curves over prime fields, where $\#E(\mathbb{F}_q) = q + 1$, finding a curve whose large prime divisor r has low Hamming weight is relatively easy. Thus, in the early days, facilitating a low Hamming weight Miller loop was not too difficult. However, once the introduction of parameterised families were needed for higher embedding degrees, the polynomial representation for $r(x)$ meant that controlling the loop length (r) of the Tate pairing was a little more difficult. The best we could do in this scenario is search for x values of low Hamming weight, in the hope that the polynomial $r(x)$ wouldn't completely destroy this. Nowadays however, the introduction of the ate and optimal ate pairings makes this optimisation very relevant. Namely, as we saw in the examples in the previous section, the loop length associated with the optimal Miller function is often some small function of x, if not x itself. Thus, choosing x to be of low Hamming weight can be very advantageous for a faster Miller loop, as we show in the following example. In fact, we will see in the next section that a faster Miller loop is only a partial consequence.

Example 7.4.1 (Magma script). Both $x = 258419657403767392$ and $x = 144115188109674496$ are 58-bit values that result in $k = 24$ BLS curves suitable for pairings at the 224-bit security level. The former was found by kick-starting the search at a random value between 2^{57} and 2^{58}, and as such, has a Hamming weight of 28, as we would expect. On the other hand, the second value is actually $2^{57}+2^{25}+2^{18}+2^{11}$, which has Hamming weight 4. Thus, we would much prefer the second value since this would result in 24 less additions through the Miller loop. Another nice alternative that gives similar parameter sizes is $x = 2^{56} + 2^{40} - 2^{20}$, which does not have a low Hamming weight, but rather a low NAF-weight (weight in the signed binary representation), for which Miller's algorithm can be easily updated to take advantage of.

7.5 The final exponentiation

Until now, our optimisations have all applied to the Miller loop. This was a natural place to look for tricks and shortcuts in the early days, since at low levels of security, the Miller loop is by far the bottle-neck of the algorithm. However, as the security level increases, the relative cost of the final exponentiation also increases [DS10]. It appears that, all known high-level optimisations considered, pairings on BN curves at the 128-bit security level is roughly the "crossover point" where the complexities of the Miller loop and the final exponentiation are similar [AKL+11, Table 4], [BGDM+10, Table 3], [NNS10, Table 2]. Thus, at higher levels of security, the final exponentiation is the most time-consuming stage of the pairing computation.

For curves belonging to families, Scott *et al.*'s algorithm [SBC+09a] is the fastest method to date. In this section we illustrate their technique by means of an example, which we take directly from our joint work with Kristin Lauter and Michael Naehrig [CLN11]. This work looked at $k = 24$ BLS curves in detail, since this family is a frontrunner for high-security pairings, particularly when targeting 256-bit security. There are several other examples looked at in [SBC+09a].

We start with a brief description of the general algorithm, before applying it to our particular case. Suppose k is even and write $d = k/2$. We start by splitting the final exponent into three components

$$(q^k - 1)/r = \underbrace{[(q^d - 1)] \cdot [(q^d + 1)/\Phi_k(q)]}_{\text{easy part}} \cdot \underbrace{[\Phi_k(q)/r]}_{\text{hard part}},$$

where the two components on the left are the "easy part" because (the second bracket reduces to powers of q and) raising elements in $\mathbb{F}_{q^k}$ to the power of q involves a simple application of the Frobenius operator π, which almost comes for free. It is the $\Phi_k(q)/r$ term that does not reduce to such a form and which is aptly named the "hard part" of the final exponentiation. Suppose we have already exponentiated through the easy part, and our intermediate value is $m \in \mathbb{F}_{q^k}$. The straightforward way to perform the hard part, i.e. $m^{\Phi_k(q)/r}$, is to write the exponent in base q as $\Phi_k(q)/r = \sum_{i=0}^{n-1} \lambda_i q^i$, and to further exploit repeated applications of π in

$$m^{\Phi_k(q)/r} = (m^{q^{n-1}})^{\lambda_{n-1}} \dots (m^q)^{\lambda_1} \cdot m^{\lambda_0},$$

so that all the m^{q^i} terms essentially come for free, and the hard part becomes the individual exponentiations to the power of the λ_i, which are performed using generic methods. These methods, however, do not take advantage of the polynomial description of q, which is where Scott *et al.*'s work advances beyond the more obvious speed-ups.

Example 7.5.1 (Magma script). Recall the $k = 24$ BLS parameterisations from Example 7.3.6: $t(x) = x + 1$, $q(x) = (x-1)^2(x^8 - x^4 + 1)/3 + x$ and $r(x) = \Phi_{24}(x) = x^8 - x^4 + 1$. To give an idea of the task we are up against, suppose we are targeting the 256-bit security level, as we did with these curves in Example 6.3.2 with $x = 9223372036854782449$. The final exponentiation in this case involves raising a 15082-bit value $f \in \mathbb{F}_{q^{24}}$, to the 14578-bit exponent $(q^{24}-1)/r$, a number far bigger than what we would like to write here (but see the corresponding script). Performing this exponentiation using a naive square-and-multiply with no optimisations would therefore involve 14578 squarings and roughly half as many multiplications in the 15082-bit field, a computation that would blow out the pairing complexity by several orders of magnitude. To take a much faster route, we start by splitting the exponent as

$$(q^{24} - 1)/r = \underbrace{[(q^{12} - 1) \cdot (q^4 + 1)]}_{\text{easy part}} \cdot \underbrace{[(q^8 - q^4 + 1)/r]}_{\text{hard part}}.$$

We compute $f^{(q^k-1)/r} = \left(f^{(q^{12}-1)\cdot(q^4+1)}\right)^{(q^8-q^4+1)/r}$. The exponentiation inside the parentheses is almost free, since $f^{q^{12}}$ is just 12 repeated applications of the Frobenius operation π, and similarly for raising to the power of q^4, so the easy part essentially incurs just a couple of multiplications and maybe an inversion. We are now left with the exponent $(q^8 - q^4 + 1)/r$, for which we can not pull out any more "easy factors". However, a very helpful observation which aids the remaining computations is that, after the first exponentiation to the power $q^{12}-1$, the value $m \in \mathbb{F}_{q^{24}}$ is now such that its norm is $N_{\mathbb{F}_{q^{24}}/\mathbb{F}_{q^{12}}}(m) = 1$. This allows any inversions in $\mathbb{F}_{q^{24}}$ to be computed for free using a simple conjugation [SB04, NBS08, SBC+09a], and any squarings in $\mathbb{F}_{q^{24}}$ to be computed more efficiently than standard $\mathbb{F}_{q^k}$ squarings [GS10, Kar10, AKL+11]. We now make use of the non-trivial part of the algorithm in [SBC+09a], and write the hard part as

$$(q(x)^8 - q(x)^4 + 1)/r(x) = \sum_{i=0}^{7} \lambda_i(x) q(x)^i.$$

In an appendix of her thesis, Benger [Ben10] computed the λ_i for a range of curve families, including BLS curves with $k = 24$, giving $\lambda_i = \nu_i/3$, where

$$\begin{aligned}
\nu_7(x) &= x^2 - 2x + 1, \\
\nu_6(x) &= x^3 - 2x^2 + x = x \cdot \nu_7(x), \\
\nu_5(x) &= x^4 - 2x^3 + x^2 = x \cdot \nu_6(x), \\
\nu_4(x) &= x^5 - 2x^4 + x^3 = x \cdot \nu_5(x), \\
\nu_3(x) &= x^6 - 2x^5 + x^4 - x^2 + 2x - 1 = x \cdot \nu_4(x) - \nu_7(x), \\
\nu_2(x) &= x^7 - 2x^6 + x^5 - x^3 + 2x^2 - x = x \cdot \nu_3(x), \\
\nu_1(x) &= x^8 - 2x^7 + x^6 - x^4 + 2x^3 - x^2 = x \cdot \nu_2(x), \\
\nu_0(x) &= x^9 - 2x^8 + x^7 - x^5 + 2x^4 - x^3 + 3 = x \cdot \nu_1(x) + 3.
\end{aligned}$$

This representation reveals another nice property exhibited by $k = 24$ BLS curves: namely, a very convenient way to compute the ν_i with essentially just multiplications by x. Letting $\mu_i = m^{\nu_i(x)}$, this structure allows us to write the hard part of the final exponentiation as

$$m^{(q^8-q^4+1)/r} = \mu_0 \cdot \mu_1^p \cdot \mu_2^{p^2} \cdot \mu_3^{p^3} \cdot \mu_4^{p^4} \cdot \mu_5^{p^5} \cdot \mu_6^{p^6} \cdot \mu_7^{p^7},$$

where the μ_i can be computed using the following sequence of operations:

$$\begin{aligned}
\mu_7 &= (m^x)^x \cdot (m^x)^{-2} \cdot m,\ \mu_6 = (\mu_7)^x,\ \mu_5 = (\mu_6)^x,\ \mu_4 = (\mu_5)^x, \\
\mu_3 &= (\mu_4)^x \cdot (\mu_7)^{-1},\ \mu_2 = (\mu_3)^x,\ \mu_1 = (\mu_2)^x, \mu_0 = (\mu_1)^x \cdot m^2 \cdot m.
\end{aligned}$$

The computation of $m^{(q^8-q^4+1)/r}$ requires 9 exponentiations by x, 12 multiplications in $\mathbb{F}_{q^{24}}$, 2 special squarings, 2 conjugations to compute the inverses and 7 q-power Frobenius operations. We detail a possible scheduling for the full exponentiation routine in Table 7.1. Note that we can simply forget about the difference between the λ_i and the ν_i; by leaving out the 3 in the denominators, we just compute the third power of the pairing.

7.6 Other optimisations

There are hundreds of papers that have helped accelerate pairing computation to the point it is at today. Of course, we could not delve into the details of

`FinalExp`	Input: $f_{r,Q}(P) \in \mathbb{F}_{q^{24}}$ and loop parameter x

`Initialize` $f \leftarrow f_{r,Q}(P)$,

$t_0 \leftarrow 1/f$, $m \leftarrow \overline{f}$, $m \leftarrow m \cdot t_0, t_0 \leftarrow \pi_q^4(m), m \leftarrow m \cdot t_0$,

$m_1 \leftarrow m^x$, $m_2 \leftarrow m_1^x$ $m_1 \leftarrow m_1^2$, $m_1 \leftarrow \overline{m_1}$, $\mu_7 \leftarrow m_2 \cdot m_1, \mu_7 \leftarrow \mu_7 \cdot m$,
$\mu_6 \leftarrow \mu_7^x$, $\mu_5 \leftarrow \mu_6^x$, $\mu_4 \leftarrow \mu_5^x$, $\mu_7' \leftarrow \overline{\mu_7}$, $\mu_3 \leftarrow \mu_4^x$, $\mu_3 \leftarrow \mu_3 \cdot \mu_7'$,
$\mu_2 \leftarrow \mu_3^x$, $\mu_1 \leftarrow \mu_2^x$, $\mu_0 \leftarrow \mu_1^x$, $m' \leftarrow m^2$, $\mu_0 \leftarrow \mu_0 \cdot m'$, $\mu_0 \leftarrow \mu_0 \cdot m$,
$f \leftarrow \pi_q(\mu_7), f \leftarrow f \cdot \mu_6$, $f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_5$, $f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_4$,
$f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_3$, $f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_2$, $f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_1$,
$f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_0$,

`Return` $f_{r,Q}(P)^{(q^{24}-1)/r} \leftarrow f$.

	Output: $f_{r,Q}(P)^{(q^{24}-1)/r}$

Table 7.1: The final exponentiation for BLS curves with $k = 24$.

all the optimisations and improvements that are available. For example, since our exposition is largely concerned with computational efficiency, we have not covered the work on compressed pairings [SB04,NBS08,Nae09] which targets low bandwidth environments, or the work by Galbraith and Lin [GL09] which looks at computing pairings using x-coordinates only.

In addition, a number of papers have looked at operations in a pairing-based protocol that are not the pairing computation itself, the most important of which are point multiplications in the pairing-specific groups $\mathbb{G}_1$ and $\mathbb{G}_2$. In Section 6.3 (and Table 6.2 in particular) we saw that the pairing-friendly curves that are most useful in practice are those of the form $E : y^2 = x^3 + b$ or $E : y^2 = x^3 + ax$. In both of these cases there is a non-trivial endomorphism $\phi \in \mathrm{End}(E)$ that facilitates faster point multiplications via GLV/GLS scalar decompositions (refer to Example 2.2.11). For point multiplications in $\mathbb{G}_1$ that take place over the base field, the standard GLV decomposition can make use of ϕ to decompose the scalar. For the more expensive point multiplications in $\mathbb{G}_2$ that take place over extension fields, the GLS technique (which additionally exploits the non-trivial action of the Frobenius endomorphism π) can be used for higher dimensional decompositions. We particularly make mention of the work of Scott *et al.* [SBC+09b] and Fuentes-Castaeda *et al.* [FCKRH11], who consider fast hashing to the group $\mathbb{G}_2$, the bottleneck of which is the expensive cofactor scalar multiplication in $\mathbb{G}_2$. For pairings to become widespread in the industry, efficient off-the-shelf solutions to

all the operations involved in pairing-based protocols need to be available.

Finally, we mention that some recent work has revived the potential of the Weil pairing in practice [AKMRH11, AFCK+12]. Indeed, since the complexity of the final exponentiation in the Tate pairing (and its ate-like variants) overtakes that of the Miller loop at higher security levels, it is natural to reconsider the Weil pairing for these scenarios. Although several of the Tate-specific optimisations do not translate across, loop shortening is available in the Weil pairing. Indeed, Hess presented a general framework for loop shortening in both the Tate and Weil pairing methodologies [Hes08]. Aranha *et al.* used this idea to derive Weil pairing variants that are particularly suited to the parallel environment [AKMRH11], and actually showed that their new Weil pairing is substantially faster than the optimal ate pairing when 8 cores are used in parallel.

Chapter 8

Summary

The fundamental computation in ECC is the scalar multiplication which, in the most straightforward case, computes $[m]P$ from $m \in \mathbb{Z}$ and $P \in E$ via a double-and-add routine. Computing the Miller loop in the Tate pairing $e(P, Q)$ can be thought of as an extension of this computation by stipulating that the line functions used in the scalar multiplication of P are evaluated at Q and accumulated as we proceed to compute $[m]P$. Thus, those who understand ECC related computations should find a relatively easy transition to the basics of pairing computation. This is why we started with a general overview of ECC in Chapter 2, which included an elementary description of the group law, as well as many optimisations like that of adopting projective coordinates or the GLV technique which exploits endomorphisms to accelerate the computation of $[m]P$. Carrying many ECC related improvements over to the context of PBC is straightforward, whilst translating other optimisations requires a firm knowledge of the functions involved in the pairing computation. For example, one could not hope to thoroughly understand how or why the (optimal) ate pairing works without knowing the basics of divisor theory. In Chapter 3 we presented all the divisor theory that is necessary in preparation for the description of the Weil, Tate and ate-like pairings. We gave a very detailed description of the r-torsion group on E in Chapter 4, and illustrated that the availability of different (efficiently computable) maps between order r subgroups give rise to different pairing types. We adopted the widely accepted argument that Type 3 pairings are most commonly the preferred setting, thereby defining $\mathbb{G}_1$ and $\mathbb{G}_2$ as the base field subgroup and

trace-zero subgroup respectively. We finished that chapter by detailing an efficient method of working in $\mathbb{G}_2$, namely by exploiting the isomorphism between the trace-zero subgroup $\mathbb{G}_2$ on E and the trace-zero subgroup $\mathbb{G}_2'$ on the twisted curve E', which is defined over a smaller field. In Chapter 5 we defined the Weil and Tate pairings and described Miller's algorithm which makes cryptographic pairing computations practical. Having described an efficient algorithm to compute pairings, Chapter 6 looked at the complementary arena of generating pairing-friendly curves. We discussed that pairing-friendly curves are very special in general, and cannot be found by searching at random, before giving a general overview of the many clever methods that have been developed in the last decade to facilitate their construction. We finished in Chapter 7 by bringing the reader up to speed with some of the major milestones in efficient pairing computation, most notably the BKLS-GHS algorithm for the Tate pairing, and the impressive work on loop shortened versions of the Tate pairing which was pinnacled by the optimal ate pairing.

Bibliography

[AB10] M. Abdalla and P. S. L. M. Barreto, editors. *Progress in Cryptology - LATINCRYPT 2010, First International Conference on Cryptology and Information Security in Latin America, Puebla, Mexico, August 8-11, 2010, Proceedings*, volume 6212 of *Lecture Notes in Computer Science*. Springer, 2010.

[ACD+05] R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *The Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC, 2005.

[AFCK+12] D. F. Aranha, L. Fuentes-Castañeda, E. Knapp, A. J. Menezes, and F. Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. Cryptology ePrint Archive, Report 2012/232, 2012. `http://eprint.iacr.org/`.

[AKL+11] D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. López. Faster explicit formulas for computing pairings over ordinary curves. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 48–68. Springer, 2011.

[AKMRH11] D. F. Aranha, E. Knapp, A. Menezes, and F. Rodríguez-Henríquez. Parallelizing the Weil and Tate pairings. In Chen [Che11], pages 275–295.

[AM93] A.O.L. Atkin and F. Morain. Elliptic curves and primality proving. *Mathematics of computation*, 61:29–29, 1993.

[BBC+09] J. Balakrishnan, J. Belding, S. Chisholm, K. Eisenträger, K.E. Stange, and E. Teske. Pairings on hyperelliptic curves. *WIN-*

Women in Numbers: Research Directions in Number Theory, Fields Institute Communications, 60:87–120, 2009.

[BCP97] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[Ben10] N. Benger. *Cryptographic Pairings: Efficiency and DLP Security.* PhD thesis, Dublin City University, May 2010.

[Ber01] D.J. Bernstein. Multidigit multiplication for mathematicians. *Advances in Applied Mathematics*, 2001.

[BGDM+10] J. Beuchat, J. E. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya. High-speed software implementation of the optimal ate pairing over Barreto-Naehrig curves. In Joye et al. [JMO10], pages 21–39.

[BGN05] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[BGOS07] P. S. L. M. Barreto, S. D. Galbraith, C. O'Eigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography*, 42(3):239–271, 2007.

[BJ03] O. Billet and M. Joye. The Jacobi model of an elliptic curve and side-channel analysis. In M. P. C. Fossorier, T. Hoholdt, and A. Poli, editors, *AAECC*, volume 2643 of *Lecture Notes in Computer Science*, pages 34–42. Springer, 2003.

[BK98] R. Balasubramanian and N. Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the Menezes - Okamoto - Vanstone algorithm. *J. Cryptology*, 11(2):141–145, 1998.

[BKLS02] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In Yung [Yun02], pages 354–368.

[BL07a] D. J. Bernstein and T. Lange. Explicit-formulas database. `http://www.hyperelliptic.org/EFD`, 2007.

[BL07b] D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007.

[BLS02] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In S. Cimato, C. Galdi, and G. Persiano, editors, *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 257–267. Springer, 2002.

[BLS03] P. S. L. M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In M. Matsui and R. J. Zuccherato, editors, *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 17–25. Springer, 2003.

[BLS04] P. S. L. M. Barreto, B. Lynn, and M. Scott. Efficient implementation of pairing-based cryptosystems. *J. Cryptology*, 17(4):321–334, 2004.

[BN05] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[BRS11] D. Boneh, K. Rubin, and A. Silverberg. Finding composite order ordinary elliptic curves using the Cocks-Pinch method. *Journal of Number Theory*, 131(5):832–841, 2011.

[BS10] N. Benger and M. Scott. Constructing tower extensions of finite fields for implementation of pairing-based cryptography. In Hasan and Helleseth [HH10], pages 180–195.

[BW05] F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. *Des. Codes Cryptography*, 37(1):133–141, 2005.

[CA66] S.A. Cook and S.O. Aanderaa. *On the minimum computation time of functions.* PhD thesis, Harvard., 1966.

[CCS07] L. Chen, Z. Cheng, and N. P. Smart. Identity-based key agreement protocols from pairings. *Int. J. Inf. Sec.*, 6(4):213–241, 2007.

[Che11] L. Chen, editor. *Cryptography and Coding - 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*, volume 7089 of *Lecture Notes in Computer Science*. Springer, 2011.

[CLN11] C. Costello, K. Lauter, and M. Naehrig. Attractive subfamilies of BLS curves for implementing high-security pairings. In D. J. Bernstein and S. Chatterjee, editors, *INDOCRYPT*, volume 7107 of *Lecture Notes in Computer Science*, pages 320–342. Springer, 2011.

[CM09] S. Chatterjee and A. J. Menezes. On cryptographic protocols employing asymmetric pairings - the role of psi revisited. *IACR Cryptology ePrint Archive*, 2009:480, 2009.

[Coh96] H. Cohen. *A course in computational algebraic number theory*, volume 138. Springer-Verlag, 3rd printing, 1996.

[CP01] C. Cocks and R.G.E. Pinch. Id-based cryptosystems based on the Weil pairing. Unpublished manuscript, 2001.

[CSB04] S. Chatterjee, P. Sarkar, and R. Barua. Efficient computation of Tate pairing in projective coordinate over general characteristic fields. In C. Park and S. Chee, editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 168–181. Springer, 2004.

[CV11] W. Castryck and F. Vercauteren. Toric forms of elliptic curves and their arithmetic. *J. Symb. Comput.*, 46(8):943–966, 2011.

[DEM05] R. Dupont, A. Enge, and F. Morain. Building curves with arbitrary small MOV degree over finite prime fields. *J. Cryptology*, 18(2):79–89, 2005.

[Deu41] M. Deuring. Die typen der multiplikatorenringe elliptischer funktionenkörper. *Abh. Math. Sem. Hansischen Univ.*, 14:197–242, 1941.

[Die12] C. Diem. What on earth is "index calculus"? The ECC blog: http://ellipticnews.wordpress.com/2012/05/07/246/, May 2012.

[DKS09] L. J. Dominguez Perez, E. J. Kachisa, and M. Scott. Implementing cryptographic pairings: a magma tutorial. Cryptology ePrint Archive, Report 2009/072, 2009. http://eprint.iacr.org/.

[DL03] I. M. Duursma and H. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In C. Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 111–123. Springer, 2003.

[DOSD06] A. J. Devegili, C. O'Eigeartaigh, M. Scott, and R. Dahab. Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Report 2006/471, 2006. http://eprint.iacr.org/.

[DS10] L. J. Dominguez Perez and M. Scott. Private communication, November 2010.

[DSD07] A. J. Devegili, M. Scott, and R. Dahab. Implementing cryptographic pairings over Barreto-Naehrig curves. In Takagi et al. [TOOO07], pages 197–207.

[Edw07] H.M. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44(3):393–422, 2007.

[FCKRH11] L. Fuentes-Castañeda, E. Knapp, and F. Rodríguez-Henríquez. Faster hashing to G_2. In Miri and Vaudenay [MV12], pages 412–430.

[FR94] 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*, 62(206):865–874, 1994.

[Fre06] D. Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. In Hess et al. [HPP06], pages 452–465.

[Fre10] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Gilbert [Gil10], pages 44–61.

[Fri05] S. Friedl. An elementary proof of the group law for elliptic curves. Personal webpage: `http://math.rice.edu/~friedl/papers/AAELLIPTIC.PDF`, August 2005.

[FST10] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–280, 2010.

[Ful08] W. Fulton. *Algebraic curves: an introduction to algebraic geometry (3rd edition).* `http://www.math.lsa.umich.edu/~wfulton/CurveBook.pdf`, 2008.

[Gal01] S. D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2001.

[Gal05] S. D. Galbraith. *Pairings*, volume 317 of *London Mathematical Society Lecture Notes*, chapter IX, pages 183–213. Cambridge University Press, 2005.

[Gal12] S. D. Galbraith. *Mathematics of Public Key Cryptography.* Cambridge University Press, March 2012.

[GHS02] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In C. Fieker and D. R. Kohel, editors, *ANTS*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 2002.

[Gil10] H. Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

[GL09] S. D. Galbraith and X. Lin. Computing pairings using x-coordinates only. *Designs, Codes and Cryptography*, 50(3):305–324, 2009.

[GLS11] S. D. Galbraith, X. Lin, and M. Scott. Endomorphisms for faster elliptic curve cryptography on a large class of curves. *J. Cryptology*, 24(3):446–469, 2011.

[GLV01] R. P. Gallant, R. J. Lambert, and S. A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In J. Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 2001.

[GMV07] S. D. Galbraith, J. F. McKee, and P. C. Valença. Ordinary abelian varieties having small embedding degree. *Finite Fields and Their Applications*, 13(4):800–814, 2007.

[GP08] S. D. Galbraith and K. G. Paterson, editors. *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*, volume 5209 of *Lecture Notes in Computer Science*. Springer, 2008.

[GPS08] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[GS10] R. Granger and M. Scott. Faster squaring in the cyclotomic subgroup of sixth degree extensions. In Nguyen and Pointcheval [NP10], pages 209–223.

[Har77] R. Hartshorne. *Algebraic Geometry*, volume 52 of *Graduate texts in mathematics*. Springer-Verlag, 1977.

[Hes08] F. Hess. Pairing lattices. In Galbraith and Paterson [GP08], pages 18–38.

[HH10] M. A. Hasan and T. Helleseth, editors. *Arithmetic of Finite Fields, Third International Workshop, WAIFI 2010, Istanbul, Turkey, June 27-30, 2010. Proceedings*, volume 6087 of *Lecture Notes in Computer Science*. Springer, 2010.

[His10] H. Hisil. *Elliptic curves, group law, and efficient computation*. PhD thesis, Queensland University of Technology, 2010.

[HLX12] Z. Hu, P. Longa, and M. Xu. Implementing the 4-dimensional GLV method on GLS elliptic curves with j-invariant 0. *Des. Codes Cryptography*, 63(3):331–343, 2012.

[HPP06] F. Hess, S. Pauli, and M. E. Pohst, editors. *Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings*, volume 4076 of *Lecture Notes in Computer Science*. Springer, 2006.

[HSV06] F. Hess, N. P. Smart, and F. Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

[HWCD08] H. Hisil, K. Koon-Ho Wong, G. Carter, and E. Dawson. Twisted Edwards curves revisited. In J. Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2008.

[HWCD09] H. Hisil, K. Koon-Ho Wong, G. Carter, and E. Dawson. Jacobi quartic curves revisited. In C. Boyd and J. M. González Nieto, editors, *ACISP*, volume 5594 of *Lecture Notes in Computer Science*, pages 452–468. Springer, 2009.

[IT02] T. Izu and T. Takagi. Efficient computations of the Tate pairing for the large MOV degrees. In P. J. Lee and C. H. Lim, editors, *ICISC*, volume 2587 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2002.

[JMO10] M. Joye, A. Miyaji, and A. Otsuka, editors. *Pairing-Based Cryptography - Pairing 2010 - 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings*, volume 6487 of *Lecture Notes in Computer Science*. Springer, 2010.

[JQ01] M. Joye and J.J. Quisquater. Hessian elliptic curves and side-channel attacks. In *Cryptographic Hardware and Embedded Systems—CHES 2001*, pages 402–410. Springer, 2001.

[Kar10] K. Karabina. Squaring in cyclotomic subgroups. *IACR Cryptology ePrint Archive*, 2010:542, 2010.

[KM05] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In N. P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 13–36. Springer, 2005.

[KO63] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. In *Soviet physics doklady*, volume 7, page 595, 1963.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[Koh11] D. Kohel. Addition law structure of elliptic curves. *Journal of Number Theory*, 2011.

[KSS08] E. J. Kachisa, E. F. Schaefer, and M. Scott. Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. In Galbraith and Paterson [GP08], pages 126–135.

[Lew12] A. B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 318–335. Springer, 2012.

[Lic69] S. Lichtenbaum. Duality theorems for curves over P-adic fields. *Inventiones mathematicae*, 7(2):120–136, 1969.

[LLP09] E. Lee, H.-S. Lee, and C.-M. Park. Efficient and generalized pairing computation on abelian varieties. *IEEE Transactions on Information Theory*, 55(4):1793–1803, 2009.

[LMN10] K. Lauter, P. L. Montgomery, and M. Naehrig. An analysis of affine coordinates for pairing computation. In Joye et al. [JMO10], pages 1–20.

[Lyn07] B. Lynn. *On the Efficient Implementation of Pairing-Based Cryptosystems*. PhD thesis, Stanford University, June 2007.

[Men93] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.

[Men05] A. J. Menezes, editor. *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*. Springer, 2005.

[Men09] A. J. Menezes. Asymmetric Pairings. Talk at ECC 2009, University of Calgary, Canada., August 2009.

[Mil85] V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *CRYPTO*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.

[Mil04] V. S. Miller. The Weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004.

[Min10] H. Minkowski. *Geometrie der zahlen*, volume 1896. Teubner, 1910.

[MKHO07] S. Matsuda, N. Kanayama, F. Hess, and E. Okamoto. Optimised versions of the ate and twisted ate pairings. In S. D. Galbraith, editor, *IMA Int. Conf.*, volume 4887 of *Lecture Notes in Computer Science*, pages 302–312. Springer, 2007.

[MNT01] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 2001.

[Mon87] P.L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.

[MOV93] A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[MS07] R. Murty and I. Shparlinski. Group structure of elliptic curves over finite fields and applications. *Topics in Geometry, Coding Theory and Cryptography*, pages 167–194, 2007.

[MV12] A. Miri and S. Vaudenay, editors. *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*. Springer, 2012.

[Nae09] M. Naehrig. *Constructive and computational aspects of cryptographic pairings*. PhD thesis, Eindhoven University of Technology, May 2009.

[NBS08] M. Naehrig, P. S. L. M. Barreto, and P. Schwabe. On compressible pairings and their computation. In S. Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2008.

[NIS99] NIST. Recommended elliptic curves for Federal Government Use. Technical report, National Institute of Standards and Technology, July 1999.

[NNS10] M. Naehrig, R. Niederhagen, and P. Schwabe. New software speed records for cryptographic pairings. In Abdalla and Barreto [AB10], pages 109–123.

[NP10] P. Q. Nguyen and D. Pointcheval, editors. *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, volume 6056 of *Lecture Notes in Computer Science*. Springer, 2010.

[PJNB11] G. C. C. F. Pereira, M. A. Simplício Jr., M. Naehrig, and P. S. L. M. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, 2011.

[Pol78] J.M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.

[RS02] K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In Yung [Yun02], pages 336–353.

[SB04] M. Scott and P. S. L. M. Barreto. Compressed pairings. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156. Springer, 2004.

[SB06] M. Scott and P. S. L. M. Barreto. Generating more MNT elliptic curves. *Des. Codes Cryptography*, 38(2):209–217, 2006.

[SBC+09a] M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez, and Ezekiel J. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. In Shacham and Waters [SW09], pages 78–88.

[SBC+09b] M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez, and Ezekiel J. Kachisa. Fast hashing to G_2 on pairing-friendly curves. In Shacham and Waters [SW09], pages 102–113.

[SCA06] M. Scott, N. Costigan, and W. Abdulwahab. Implementing cryptographic pairings on smartcards. In L. Goubin and M. Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 134–147. Springer, 2006.

[Sch85] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp*, 44(170):483–494, 1985.

[Sco04] M. Scott. Understanding the Tate pairing. Personal webpage: `http://www.computing.dcu.ie/~mike/tate.html`, 2004.

[Sco05a] M. Scott. Computing the Tate pairing. In Menezes [Men05], pages 293–304.

[Sco05b] M. Scott. Faster pairings using an elliptic curve with an efficient endomorphism. In S. Maitra, C. E. V. Madhavan, and R. Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 258–269. Springer, 2005.

[Sco07a] M. Scott. An introduction to pairings. Talk at ICE-EM RNSA 2007 Cryptography Workshop, Queensland University of Technology, Australia, June 2007.

[Sco07b] M. Scott. Efficient implementation of cryptographic pairings. Talk at ICE-EM RNSA 2007 Cryptography Workshop, Queensland University of Technology, Australia, June 2007.

[Sco07c] M. Scott. Implementing cryptographic pairings. In Tsuyoshi Takagi, Tatsuaki Okamoto, and Eiji Okamoto, editors, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 177–196. Springer, 2007.

[Sco11] M. Scott. On the efficient implementation of pairing-based protocols. In Chen [Che11], pages 296–308.

[Sha05] H. Shacham. *New Paradigms in Signature Schemes*. PhD thesis, Stanford University, December 2005.

[Sil09] J. H. Silverman. *The Arithmetic of Elliptic Curves (2nd Edition)*. Number 106 in Graduate texts in mathematics. Springer-Verlag, 2009.

[Sil10] J. H. Silverman. A survey of local and global pairings on elliptic curves and abelian varieties. In Joye et al. [JMO10], pages 377–396.

[Sma01] N. P. Smart. The Hessian form of an elliptic curve. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 118–125. Springer, 2001.

[Sma10] N. P. Smart. ECRYPT II yearly report on algorithms and keysizes (2009-2010). Technical report, ECRYPT II – European Network of Excellence in Cryptology, EU FP7, ICT-2007-216676, 2010. Published as deliverable D.SPA.13, `http://www.ecrypt.eu.org/documents/D.SPA.13.pdf`.

[Sta07] K. E. Stange. The Tate pairing via elliptic nets. In Takagi et al. [TOOO07], pages 329–348.

[Sut12] A. V. Sutherland. Accelerating the CM method. *LMS Journal of Computation and Mathematics*, 15:172–204, 2012.

[SV07] N. P. Smart and F. Vercauteren. On computable isomorphisms in efficient asymmetric pairing-based systems. *Discrete Applied Mathematics*, 155(4):538–547, 2007.

[SW09] H. Shacham and B. Waters, editors. *Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings*, volume 5671 of *Lecture Notes in Computer Science*. Springer, 2009.

[Too63] A.L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics Doklady*, volume 3, pages 714–716, 1963.

[TOOO07] T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors. *Pairing-Based Cryptography - Pairing 2007, First International Conference, Tokyo, Japan, July 2-4, 2007, Proceedings*, volume 4575 of *Lecture Notes in Computer Science*. Springer, 2007.

[Ver01] E. R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In B. Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2001.

[Ver06a] F. Vercauteren. Mathematics of Pairings: Part II. Talk at Pairing-Based Cryptography Workshop, 2006.

[Ver06b] F. Vercauteren. Mathematics of Pairings: Part I. Talk at Pairing-Based Cryptography Workshop, 2006.

[Ver10] F. Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2010.

[WP06] A. Weimerskirch and C. Paar. Generalizations of the Karatsuba algorithm for efficient implementations. Cryptology ePrint Archive, Report 2006/224, 2006. `http://eprint.iacr.org/`.

[WS07] C. Whelan and M. Scott. The importance of the final exponentiation in pairings when considering fault attacks. In Takagi et al. [TOOO07], pages 225–246.

[Yun02] M. Yung, editor. *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.

[ZZH08a] C. Zhao, F. Zhang, and J. Huang. All pairings are in a group. Cryptology ePrint Archive, Report 2008/085, 2008. http://eprint.iacr.org/.

[ZZH08b] C. Zhao, F. Zhang, and J. Huang. A note on the ate pairing. *Int. J. Inf. Sec.*, 7(6):379–382, 2008.

Index