Efficient Pairings on Various Platforms

by

Gurleen Grewal

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Pairings have found a range of applications in many areas of cryptography. As such, to utilize the enormous potential of pairing-based protocols one needs to efficiently compute pairings across various computing platforms. In this thesis, we give an introduction to pairing-based cryptography and describe the Tate pairing and its variants. We then describe some recent work to realize efficient computation of pairings. We further extend these optimizations and implement the O-Ate pairing on BN-curves on ARM and x86-64 platforms. Specifically, we extend the idea of lazy reduction to field inversion, optimize curve arithmetic, and construct efficient tower extensions to optimize field arithmetic. We also analyze the use of affine coordinates for pairing computation leading us to the conclusion that they are a competitive choice for fast pairing computation on ARM processors, especially at high security levels [25]. Our resulting implementation is more than three times faster than any previously reported implementation on ARM processors.

Acknowledgements

I would like to thank my supervisor David Jao for his invaluable guidance throughout the process of creating this thesis. I would like to thank my readers Edlyn Teske and Alfred Menezes for their helpful feedback. Finally, I would like to thank my fellow graduate students Ben, Marco, Dale, and Andrew for their friendship and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the field of cryptography, pairings were initially used to break elliptic curve protocols [29]. However, in the past decade they have found a range of constructive applications in areas such as identity-based encryption, key establishment, and short signatures. The utility of pairings in constructing protocols which cannot be otherwise implemented has led to a surge of interest in pairing-based protocols. Naturally, implementing any such protocol requires efficient computation of the pairing function. Considerable work has been done to compute fast pairings on PCs [19, 9, 33, 4]. Recently, Arahna et al. [4] computed the O-Ate pairing at the 128-bit security level in under 2 million cycles on several 64-bit personal computers.

Less emphasis has been placed on computing pairings on non-PC platforms such as mobile devices. Mobile devices like smartphones and tablets are widely predicted to become the dominant computing platform in the near future. In February 2011, the number of smartphone shipments surpassed PCs [14]. As a result, for pairing-based protocols to have practical relevance, it is important to be able to efficiently compute pairings on both mobile devices and PCs.

In this work, we examine the problem of computing efficient pairings at multiple security levels across different platforms. We extend the work of Arahna et al. [4] to incorporate different BN-curves and security levels and use it to obtain fast pairings on the ARM platform. In the process, we make several optimizations and analyze different options available for implementation at various levels of the pairing computation:

- Firstly, we extend the concept of lazy reduction employed in Arahna et al. [4] to inversion in extension fields. We also present an improved doubling formula using Jacobian coordinates, and optimize the sparse multiplication algorithm in the degree 12 extension field.

- We examine different choices of towers for extension field arithmetic over various prime fields. We determine the most efficient implementation of extension fields in the context of pairing computation over BN-curves from the various choices available.

- The M-type sextic twist [39] has been largely ignored for use in pairing computations, most likely due to the inefficient untwisting map. We demonstrate that by computing the pairing on the twisted curve, we can bypass the inefficient untwisting. As a result, it does not matter from an optimization perspective whether we use M-type or D-type twists.

Using the results from our analysis and optimizations, we develop software to compute the O-Ate pairing on BN-curves. Our software computes the O-Ate pairing on BN-curves, and is over three times faster than previously reported pairing implementations on ARM processors. We also present explicit operation counts for the pairing computation on three security levels, which are faster than any previously known implementation.

Affine coordinates were first suggested for pairing computation by Lauter et al. in [25]. Acar et al. [3] noted that low extension field inversion to multiplication ratios make affine coordinates a better choice for pairing computation than homogeneous coordinates. We examine their claim on both x86-64 and ARM platforms. We find that inversion to multiplication ratios of below 10-12 at different security levels make affine coordinates a better choice than homogeneous. This implies that more often than not, affine coordinates are faster on the ARM platform, but homogeneous coordinates are the better choice for x86-64 processors. A detailed comparison between their measurements and ours is given in Chapter 5.

The remainder of this work is organized as follows. The rest of this chapter gives background required to understand the O-Ate pairing. In Chapter 2, we define the Tate pairing and describe optimizations building up to the O-Ate pairing. Chapter 3 discusses our implementation of the field arithmetic, and Chapter 4 describes our optimizations to the curve arithmetic. Finally, in Chapter 5 we present our implementation results including pairing operation counts and timings on the x86-64 and ARM platforms.

1.1 Introduction to Pairing Based Cryptography

1.1.1 Bilinear Pairings and their Applications

Research in pairing-based cryptography has exploded in the past decade. Although initially used for destructive purposes, pairings have been used in clever ways to construct novel cryptographic schemes. In this section, we first define bilinear pairings and then present a few selected applications. Due to the influx of interest in this topic in recent years, it is unrealistic to touch on all applications of pairing-based cryptography. Instead, we focus on a few well-known schemes. For a more comprehensive review, see [10, Ch. 10]. Familiarity with concepts such as public-key cryptography, digital signatures, and key exchange schemes is assumed. For a survey of these, see [28].

Definition 1.1.1. Let G_1 and G_2 be cyclic groups of prime order n written in additive notation with the identity element 0. Suppose G_3 is a cyclic group of order n written in multiplicative notation with identity element 1. A pairing is a function

$$e : G_1 \times G_2 \to G_3$$

that satisfies the following additional properties:

Bilinearity: For all $P, P' \in G_1$ and all $Q, Q' \in G_2$ we have

$$e(P + P', Q) = e(P, Q)\, e(P', Q) \text{ and}$$

$$e(P, Q + Q') = e(P, Q)\, e(P, Q')$$

Non-degeneracy:

- For all $P \in G_1$, with $P \neq 0$, there is some $Q \in G_2$ such that $e(P, Q) \neq 1$.
- For all $Q \in G_2$, with $Q \neq 0$, there is some $P \in G_1$ such that $e(P, Q) \neq 1$.

Applications

Key Distribution Schemes. Key distribution is a fundamental problem in cryptography. In 2000, Joux [22] devised a key agreement scheme based on pairings.

Tripartite one-round key exchange.

- The three parties pick a base point $P \in G_1$.
- Each party chooses a secret integer α, β, γ, and broadcasts respectively $\alpha P, \beta P, \gamma P$.
- Bilinearity implies:

$$e(P,P)^{\alpha\beta\gamma} = e(\alpha P, \beta P)^{\gamma} = e(\alpha P, \gamma P)^{\beta} = e(\beta P, \gamma P)^{\alpha}.$$

  Hence, $e(P,P)^{\alpha\beta\gamma}$ can computed by anyone who has knowledge of one of the secret exponents. The three parties can use this quantity to derive a secret key.

The key cannot be computed by an adversary, provided the bilinear Diffie-Hellman problem is hard. This problem can be stated as follows:

Definition 1.1.2. Bilinear Diffie-Hellman Problem: given P, aP, bP, $cP \in G_1$, with a, b, $c \in_R \mathbb{Z}_n$, compute $e(P,P)^{abc}$.

One can give a formal security proof relating the above problem to the security of the protocol against a passive adversary, i.e., an adversary who is not allowed to request any additional information.

Identity-based Encryption. Identity-based encryption was first introduced by Shamir at Crypto'84 [40]. An identity-based cryptosystem is a public-key system with the special property that the public keys of users are derived from their identities. This does away with the problem of obtaining and verifying a user's public key. The first practical and provably secure identity-based encryption protocols were based on pairings. The scheme proposed by Boneh and Franklin in 2001 [12] is the most well known of these. Their protocol uses ideas from an earlier work of Sakai et al. [37] that constructs a non-interactive key distribution scheme using pairings.

Boneh-Franklin IBE.

Public parameters: A Trusted third party(TTP) publishes the following parameters:

- A bilinear pairing $e\colon G_1 \times G_2 \to G_3$ between groups of prime order n.

- A hash function $H\colon \{0,1\}^* \to G_2$.
- A generator $P \in G_1$ and a point $\alpha P \in G_1$ where $\alpha \in_R \mathbb{Z}_n^*$ is a random (secret) integer chosen by the TTP.

Key Generation:

- To generate the public key for user A, compute $Q = H(ID_A)$.
- The public key is ID_A and the private key is αQ given to the user A by the TTP.

Encryption:

- Given a public key ID_A and a message m (of length n), let $Q = H(ID_A)$.
- Choose $r \in_R \mathbb{Z}_n^*$ and compute rP and $c = m \oplus e(\alpha P, rQ)$. The ciphertext is (c, rP).

Decryption:

- Given ciphertext (c_1, c_2) compute $m' = c_1 \oplus e(c_2, \alpha Q)$.
- For a valid encryption, $c_1 = m \oplus e(\alpha P, rQ)$ and $c_2 = rP$.
- Bilinearity implies:

$$m' = m \oplus e(\alpha P, rQ) \oplus e(rP, \alpha Q) = m.$$

Other Applications of Pairings In addition to key distribution and identity-based encryption, pairings have found many other applications in cryptography — short signatures, non-repudiable signatures, and escrowable public key encryption, etc.

1.1.2 Elliptic Curves

All known cryptographic pairings are defined on objects known as *abelian varieties* in algebraic geometry. An elliptic curve is a special abelian variety. Because of the simplicity of the group law, and their relative ease of implementation, elliptic curve groups provide a good choice for use in pairing-based cryptography. In fact, almost all implementations of pairings for cryptographic purposes use elliptic curve groups. In this thesis, we will only consider pairings on elliptic curve groups. We start by giving some background on elliptic

curves and outlining some of their properties that are used in this thesis. The following material is based on class notes from a course taught on Algebraic Curves in the Winter of 2010 at the University of Waterloo [32]. More background material can be found in [42].

Background in Algebraic Geometry. We fix some notation:

- $\mathbb{F}[x, y]$ and $\mathbb{F}[x, y, z]$ represent the ring of polynomials over a field $\mathbb{F}$.
- $\bar{\mathbb{F}}$ denotes the closure of the field $\mathbb{F}$.
- f_x is the partial derivative of f with respect to x.
- $\mathbb{F}^* = \mathbb{F} \setminus \{0\}$.
- $\mathbb{F}_q$ denotes the finite field of size q.
- Let $\gamma \in \mathbb{F}_{q^n}$. The *Norm* of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ of γ is denoted $N_{\mathbb{F}_{q^n}/\mathbb{F}_q(\gamma)}$ and is given by the product of all its conjugates:

$$N_{\mathbb{F}_{q^n}/\mathbb{F}_q(\gamma)} = \prod_{i=0}^{n-1} (\gamma)^{q^i} \in \mathbb{F}_q.$$

Intersection multiplicity.

Definition 1.1.3. Let $f \in \overline{\mathbb{F}}[x, y]$ be a polynomial. Then $C : f = 0$ is called an affine plane curve which is irreducible if f is an irreducible polynomial. Let $p = (a, b)$ be a point on C. The point p is said to be smooth if $\nabla f(a, b) \neq 0$, where $\nabla f = (f_x, f_y)$. If all points on C are smooth, then C is called a smooth curve.

Definition 1.1.4. Given an irreducible plane curve $C : f = 0$, define the *ideal* of C as follows:

$$I(C) := \{f \in \overline{\mathbb{F}}[x, y] \mid f(P) = 0 \text{ for all } P \in C\}$$

Definition 1.1.5. Let C be an irreducible plane curve. The coordinate ring of C is

$$\mathbb{K}[C] = \overline{\mathbb{F}}[x, y]/I(C)$$

Definition 1.1.6. The function field $\mathbb{K}(C)$ of C consists of rational functions of the form

$$g(x,y) = \frac{g_1(x,y)}{g_2(x,y)}$$

where g_1 and $g_2 \in \mathbb{K}[C]$ are relatively prime and $g_2 \neq 0$ in $\mathbb{K}[x,y]$.

Theorem 1.1.7. *Let $C : f = 0$ be an irreducible plane curve. Let $p = (a,b)$ be a smooth point on C. Let $t = 0$ be a line passing through p that is not tangent to C at p. Then t is called a uniformization parameter at p and any $g \in \mathbb{K}[C]$ can be written as $g = t^n z$, where $n \in \mathbb{Z}_{\geq 0}$, $z \in \mathbb{K}(C)$ and $z(p) \neq 0$. Moreover, n is independent of the choice of t.*

Definition 1.1.8. Given any non-zero $g \in \mathbb{K}[C]$ and its decomposition $g = t^n z$, we define $\mathrm{ord}_p(g) := n$. This number is called the intersection multiplicity of C and g at p, and denoted $I(p, C \cap g)$. We extend this definition to any rational function in $\mathbb{K}(C)$: if $g = \frac{g_1}{g_2} \in \mathbb{K}(C)$, then define $\mathrm{ord}_p(g) = \mathrm{ord}_p(g_1) - \mathrm{ord}_p(g_2)$.

Properties of Intersection Multiplicity. Let C be an irreducible plane curve. Let $p \in C$ be a smooth point, let t be a uniformization parameter, and let $g, h \in \mathbb{K}(C)$.

- Note that since $t = 0$ intersects p, $t(p) = 0$. This in turn implies that if $g(p) \neq 0$, then $\mathrm{ord}_p(g) = 0$. In fact, $\mathrm{ord}_p(g) = 0$ iff $g(p) \neq 0$.
- $\mathrm{ord}_p(g) > 0$ iff f has a zero at p and $\mathrm{ord}_p(g) < 0$ iff f has a pole at p.
- Additivity: $I(p, C \cap hg) = I(p, C \cap h) + I(p, C \cap g)$

Coordinate Systems in Algebraic Geometry. Let $\mathbb{A}^n(\mathbb{F}) = \{x_1, x_2, ..., x_n \mid x_1, ..., x_n \in \mathbb{F}\}$. This is called affine n-space. Consider $\mathbb{A}^3(\mathbb{F})$. The set of all lines passing through the origin $\mathbf{0} = (0,0,0)$ is called the 2-dimensional projective space (or the projective plane) and is denoted $\mathbb{P}^2(\mathbb{F})$, or simply $\mathbb{P}^2$. Points in $\mathbb{P}^2$ can be written as 3-tuples (called *homogeneous* or *projective coordinates*):

$$\mathbb{P}^2 := (\mathbb{A}^3 \setminus \{0\})/\mathbb{F}^*$$

where $(X : Y : Z) \sim (\lambda X : \lambda Y : \lambda Z)$ for all $\lambda \in \mathbb{F}^*$.
That is, two points in $(\mathbb{A}^3 - 0)$ are equivalent if they lie on the same line through the origin.

Set

$$\begin{aligned} U_z &= \{\text{points in } \mathbb{P}^2 \text{ such that } Z \neq 0\} \\ &= \{[X : Y : Z] \mid Z \neq 0\} \\ &= \{[\tfrac{X}{Z} : \tfrac{Y}{Z} : 1]\} \\ &= \{[x : y : 1]\}. \end{aligned}$$

In this way, $\mathbb{A}^2$ is a subset of $\mathbb{P}^2$. If we restrict our attention to points on the projective plane where $Z \neq 0$, then we can write $P = [X : Y : Z]$ as $p = \{x, y\}$ (as computed above) with the understanding that $z = 1$. This way of denoting points on the projective plane is called *affine coordinates*. It is easy to switch between affine and projective coordinates using the bijection:

$$\begin{gathered} [X : Y : Z] \mapsto [X/Z : Y/Z : 1] \mapsto \{X/Z : Y/Z\} \text{ and} \\ \{x, y\} \mapsto [X : Y : 1]. \end{gathered} \tag{1.1}$$

Example 1.1.9. A line on the projective plane is given by a homogenous polynomial of degree 1:

$$aX + bY + cZ = 0$$

Restricting to U_z, we get the equation of a line in the {x-y} affine plane:

$$ax + by + c = 0 \tag{1.2}$$

We "homogenize" equation (1.2) by replacing x with X/Z, y with Y/Z, and eliminating the denominator to get back the homogeneous polynomial:

$$\begin{aligned} ax + by + c &= 0 \\ a(X/Z) + b(Y/Z) + c &= 0 \\ Z(a(X/Z) + b(Y/Z) + c) &= 0 \\ aX + bY + cZ &= 0 \end{aligned}$$

In this way, one can easily switch between affine and projective coordinates.

Similarly, we get U_x and U_y by setting $x \neq 0$ and $y \neq 0$ respectively. U_x, U_y, and U_z are called the affine pieces of the projective plane.

If we set $Z = 0$, we get the following subset of $\mathbb{P}^2$:

$$H_\infty := \{[x : y : 0] \mid x, y \in \mathbb{F}\}$$

which is known as the hyperplane at infinity. So:

$$\mathbb{P}^2 = U_z \cup H_\infty = \mathbb{A}^2 \cup H_\infty$$

Points on the projective plane can also be represented using another coordinate system known as *jacobian coordinates*. Similar to projective coordinates, points in jacobian coordinates can be written as 3-tuples:

$$\mathbb{P}^2 := (\mathbb{A}^3 - 0)/\mathbb{F}^*$$

where $(X : Y : Z) \sim (\lambda^2 X : \lambda^3 Y : \lambda Z)$ for all $\lambda \in \mathbb{F}^*$. Restricting our attention to U_z, the jacobian point $(X : Y : Z)$ corresponds to the affine point $(\frac{X}{Z^2}, \frac{Y}{Z^3})$.

Elliptic Curves in Algebraic Geometry. We now give a technical definition of an elliptic curve. The work presented in this thesis uses only BN-elliptic curves (to be defined later in Chapter 2), which are defined over fields having characteristic not equal to 2 or 3. Hence, we restrict our attention to fields having characteristic not equal to 2 or 3.

Definition 1.1.10. Let $\mathbb{F}$ be a field with characteristic $\neq 2, 3$. An elliptic curve E over $\mathbb{F}$ is the set of solutions to the Weierstrass equation of the form:

$$E : Y^2 Z = X^3 + aXZ^2 + bZ^3 \tag{1.3}$$

with $a, b \in \mathbb{F}$.

The discriminant Δ of the curve E is

$$\Delta = 4a^3 + 27b^2.$$

An elliptic curve is called non-singular if and only if $\Delta \neq 0$. For the remainder of this thesis, we will assume that $\Delta \neq 0$.

If we set $Z = 0$, Equation (1.3) becomes $X^3 = 0$, giving us a solution $[0 : 1 : 0]$. This point is called the point at infinity and is denoted ∞.

Restricting our attention to $Z \neq 0$, and using bijection (1.1) to switch to affine coordinates, we get

$$\frac{Y^2}{Z^2} = \frac{X^3}{Z^3} + a\frac{X}{Z} + b.$$

Replacing X/Z with x, and Y/Z with y, we get the following equation:

$$y^2 = x^3 + ax + b. \tag{1.4}$$

Hence we may look at an elliptic curve as the set of solutions to Equation (1.4) together with the point at infinity.

If $\mathbb{K}$ is an extension of $\mathbb{F}$ then the set of all $\mathbb{K}$-rational points on E is:

$$E(\mathbb{K}) : \{(x, y) \in \mathbb{K} \times \mathbb{K} : y^2 = x^3 + ax + b\} \cup \{\infty\}.$$

The Group Law. The set of points on an elliptic curve forms an abelian group. The group law can be described using the chord and tangent process illustrated in Figure 1.1 and described below:
Let P and Q be two distinct points on E.

- If $P = \infty$, then $P + \infty = P$ and $\infty + P = P$.
- If $P \neq \infty$ and $Q \neq \infty$, let l be the straight line joining P and Q. If $P = Q$, then let l be the tangent line to the curve at P. If l does not interesect the curve at any other point then $P + Q = \infty$.
- In all other cases, l must intersect the curve at exactly one additional point, since we are intersecting a line with a cubic. Call this point R. We define $P + Q$ to be the reflection of R in the y-axis. (Note that if (x, y) is a solution to Equation (1.4), then $(x, -y)$ is also a solution, hence $P + Q$ lies on the curve).

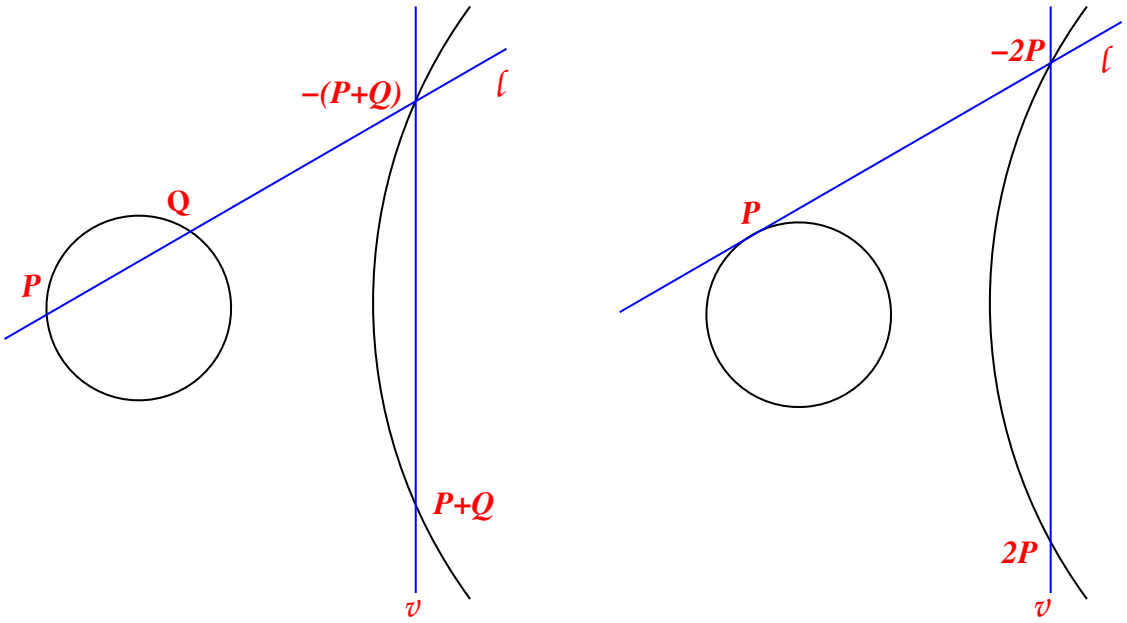

Figure 1.1: Adding points P and Q, and doubling the point P using the chord and tangent process.

One can use the above definition to derive the following explicit formulas for the group law:

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. If $x_1 = x_2$, then $P + Q = \infty$. Otherwise, let $P + Q = (x_3, y_3)$ where:

$$\begin{aligned} \lambda &= \begin{cases} \frac{3x_1^2+a}{2y_1} \text{ if } P = Q \\ \frac{y_2-y_1}{x_2-x_1} \text{ otherwise,} \end{cases} \\ x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= -(\lambda(x_3 - x_1) + y_1). \end{aligned} \tag{1.5}$$

For a proof that the points on an elliptic curve indeed form a group under the above operation, see [42, §III.2].

Point Counting and j-invariants

Definition 1.1.11. Given an elliptic curve E, define the ideal of E as follows:

$$I(E) := \{f \in \mathbb{F}[x, y, z] \mid f(P) = 0 \text{ for all } P \in E\}.$$

Theorem 1.1.12. *The ideal of* $E : Y^2Z = X^3 + aXZ^2 + bZ^3$ *is the prime ideal generated by* $Y^2Z - X^3 - aXZ^2 - bZ^3$.

Definition 1.1.13. Let E be an elliptic curve. The homogeneous coordinate ring of E is

$$\Gamma_H(E) = \mathbb{F}[x, y, z]/I(E).$$

Definition 1.1.14. The function field $\mathbb{F}(E)$ of E consists of rational functions of the form

$$f(x, y, z) = \frac{f_1(x, y, z)}{f_2(x, y, z)}$$

where f_1 and f_2 are homogeneous polynomials of the same degree and $f_2 \neq 0$ in $\Gamma_H(E)$.

Note: $\mathbb{F}(E)$ is indeed a field of functions on $\mathbb{P}^2$. If f_1 and f_2 are two homogeneous polynomials of the same degree d then $\frac{f_1}{f_2} \in \mathbb{F}(E)$ and

$$\frac{f_1}{f_2}(\lambda x, \lambda y, \lambda z) = \frac{f_1(\lambda x, \lambda y, \lambda z)}{f_2(\lambda x, \lambda y, \lambda z)} = \frac{\lambda^d}{\lambda^d}\frac{f_1(x, y, z)}{f_2(x, y, z)} = \frac{f_1}{f_2}(x, y, z)$$

given that $f_2(x, y, z) \neq 0$. Hence, $\frac{f_1}{f_2}$ is well-defined on points in $\mathbb{P}^2$.

Intersection multiplicity on elliptic curves. Let f be a non-zero function in $\mathbb{F}(E)$. The intersection multiplicity of f and E at a point P is defined to be the intersection multiplicity of E and f restricted to an affine piece containing P. For example, if $z \neq 0$, then we can write $P = [x_0 : y_0 : 1] \in \{E \cap U_z\}$. If $E \cap U_z$ is smooth at P, then the intersection multiplicity of E and f at P is defined as:

$$I(E \cap f) = I(E \cap f \cap U_z) = \text{ord}_P(f).$$

If $z = 0$ then we consider the intersection of E and f with U_x or U_y.

Divisors

All known pairings on elliptic curves use the concept of divisors. In this section, we give a brief introduction to that portion of the theory of divisors which is required to discuss cryptographic pairings.

Definition 1.1.15. A divisor D on an elliptic curve E is a formal sum of points

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

where $n_P \in \mathbb{Z}$ and only finitely many of them are non-zero.

The divisor with all $n_P = 0$ is called the empty divisor and denoted $\emptyset$. Two divisors $D_1 = \sum_{P \in E} n_P(P)$ and $D_2 = \sum_{P \in E} m_P(P)$ are equal if and only if $n_P = m_P$ for all $P \in E$. The set of all divisors on E is denoted $\mathrm{Div}(E)$ and has a natural group structure of addition.

The support of a divisor $D = \sum_{P \in E} n_P(P)$ is all n_P such that $n_P \neq 0$.

The degree of a divisor $D = \sum_{P \in E} n_P(P)$ is $\sum_{P \in E} n_P \in \mathbb{Z}$. The empty divisor has degree zero.

The degree zero divisors form a subgroup of $\mathrm{Div}(E)$ denoted $\mathrm{Div}^0(E)$.

Let f be a non-zero function in $\mathbb{F}(E)$. The divisor of f, written $\mathrm{div}(f)$, is the divisor $\sum_{P \in E} \mathrm{ord}_P(f)(P)$. From the properties of multiplicity above, the only points appearing in $\mathrm{div}(f)$ are the zeroes and poles of f. It follows from these properties that $\mathrm{div}(fg) = \mathrm{div}(f) + \mathrm{div}(g)$ and $\mathrm{div}(\frac{1}{f}) = -\mathrm{div}(f)$.

Theorem 1.1.16 (Bezout's Theorem)**.** *Let f(x,y) be a homogeneous polynomial with degree d, and E an elliptic curve. Then E and f intersect at 3d points counting multiplicity.*

Definition 1.1.17. A divisor D is called principal if $D = \mathrm{div}(f)$ for some function $f \in \overline{\mathbb{F}}(E)$.

Bezout's Theorem implies that $\mathrm{div}(f) = \emptyset$ iff f is a constant. Hence, if $\mathrm{div}(f) = \mathrm{div}(g)$, then $\mathrm{div}(f/g) = \emptyset$ and $\frac{f}{g}$ must be a constant.

Bezout's Theorem also implies that if D is principal then the degree of D is zero. That is, $\deg(\mathrm{div}(f)) = 0$ for all non-zero $f \in \mathbb{F}(E)$. The principal divisors form a subgroup of $\mathrm{Div}^0(E)$ and are denoted $P(E)$. We can then define:

$$Cl^0(E) := \mathrm{Div}^0(E)/P(E).$$

This group is known as the divisor class group of E. We say that $D, D' \in \mathrm{Div}^0(E)$ are linearly equivalent, denoted $D \sim D'$ if $(D - D') \in P(E)$.

We now state two important results on divisors on elliptic curves.

Theorem 1.1.18. *Let E be an elliptic curve over a field $\mathbb{F}$. Let*

$$D = \sum_P n_P(P)$$

be a degree 0 divisor on E. Then $D \sim \emptyset$ (i.e. $D = \mathrm{div}(f)$ for some $f \in \overline{\mathbb{F}}(E)$) if and only if $\sum_P [n_P]P = \infty$ on E.

The proof of Theorem 4 can be found in [10, Ch. 9]. Let f be a function and let $D = \sum_P n_P(P)$ be a divisor of degree zero such that $\mathrm{div}(f)$ and D have disjoint support. Define

$$f(D) = \prod_P f(P)^{n_P}.$$

Theorem 1.1.19 (Weil Reciprocity)**.** *Let f and g be non-zero functions on $\mathbb{F}(E)$. Suppose that $\mathrm{div}(f)$ and $\mathrm{div}(g)$ have disjoint support. Then $f(\mathrm{div}(g)) = g(\mathrm{div}(f))$.*

The proof of Weil Reciprocity can be found in [10, Appendix].

1.1.3 The Tate Pairing

The Tate Pairing. Let E be an elliptic curve over a finite field $\mathbb{F}_q$. Let the order of E be hn, where n is a prime not equal to the characteristic of $\mathbb{F}_q$ and h and n are coprime.

The set of n-th roots of unity is defined to be $\mu_n = \{u \in \overline{\mathbb{F}}_q : u^n = 1\}$.

Definition 1.1.20. Define the field $\mathbb{F} = \mathbb{F}_q(\mu_n)$ to be the extension of $\mathbb{F}_q$ generated by the nth roots of unity. Let k be the degree of this extension. Then k is called the embedding degree of E with respect to n. An equivalent condition is that the embedding degree is the smallest integer k such that $n \mid q^k - 1$.

The set of all points $P \in E(\bar{\mathbb{F}})$ satisfying $nP = \infty$ is denoted by $E[n]$. It is known that $E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n$ and $E[n] \subseteq E(\overline{\mathbb{F}}) = E(\mathbb{F}_{q^k})$ [5]. We further assume that $n \nmid (\#E(\mathbb{F})/n^2)$.

The Tate pairing is a map

$$e\colon E[n] \times E[n] \to \mu_n$$

which is defined as follows. Let P, $Q \in E[n]$. Since $nP = \infty$, the divisor $D_1 = n(P) - n(\infty)$ is principal. Let $f_{n,P}$ be a function with divisor D_1. Let $R \in E[n]$ such that $R \notin \{\infty, P, -Q, P - Q\}$, and let $D_Q = (Q + R) - (R)$. Our choice of R ensures that D_Q and D_1 have disjoint support. We now define

$$\boldsymbol{e}(P, Q) = f_{n,P}(D_Q)^{\frac{q^k-1}{n}} = \left(\frac{f_{n,P}(Q + R)}{f_{n,P}(R)}\right)^{\frac{q^k-1}{n}}.$$

The value of the Tate pairing does not depend on the choice of $f_{n,P}$ or D_Q, hence it is well defined. Moreover, it is bilinear and non-degenerate [6].

In general, a function with divisor $r(S) - ([r]S) - (r - 1)(\infty)$ is denoted by $f_{r,S}$. Such functions are called Miller functions.

It can be shown [6, 44] that for $k > 1$, D_Q in the computation of the Tate pairing can be replaced by the point Q. That is, the Tate pairing can be written as $e(P, Q) = f_{n,P}(Q)^{\frac{q^k-1}{n}}$. Hence, we assume from now on that $k > 1$.

1.1.4 Miller's Algorithm

The usefulness of the Tate pairing in practical cryptography relies on its efficient calculation. The key component in computing the Tate pairing is to compute Miller functions. In 1986, Victor Miller discovered an algorithm to compute these functions in polynomial time [30]. To compute the pairing, one needs to construct a function $f_{n,P}$ such that $\mathrm{div}(f_{n,P}) = n(P) - n(\infty)$. The key idea in Miller's algorithm is to construct this function using a double and add algorithm.

Theorem 1.1.21. *Recall the elliptic curve group law. Consider the computation* $[i]P + [j]P = [i+j]P$. *Let* l *be the line through* $[i]P$ *and* $[j]P$. *Let* v *be the vertical line through* $-[i+j]P$ *and* $[i+j]P$. *Then* $f_{i+j,P} = f_{i,P} f_{j,P} \frac{l}{v}$ *(See Figure 1.1).*

Proof. A line in $\mathbb{P}^2$ is given by a homogeneous polynomial of degree 1. By Bezout's theorem each of l and v have three zeroes on E. It is easy to see that the zeroes of l on E are $[i]P$, $[j]P$, and $-[i+j]P$. Let v be given by the equation $0 = x - a$ in the affine plane. Homogenizing, we get $0 = X - Za$. The point at infinity, $\infty = [0:1:0]$, is a solution to this equation. Hence the three zeroes of v on E are $\infty, -[i+j]P$, and $[i+j]P$. Thus,

$$\operatorname{div}\left(\frac{l}{v}\right) = ([i]P) + ([j]P) - ([i+j]P) - (\infty)$$

Therefore,

$$\begin{aligned}\operatorname{div}(f_{i,P} f_{j,P} \tfrac{l}{v}) &= i(P) - ([i]P) - (i-1)(\infty) + j(P) - ([j]P) - (j-1)(\infty) + \\ &\quad ([i]P) + ([j]P) - ([i+j]P) - (\infty) \\ &= (i+j)P - ([i+j]P) - (i+j-1)(\infty) \\ &= \operatorname{div}(f_{i+j,P})\end{aligned}$$

as required. □

Miller's algorithm uses the above fact to construct $f_{n,P}$ from $f_{1,P} = 1$. Indeed, $\operatorname{div}(1) = \emptyset = 1(P) - ([1]P) - (0)\infty$. The function $f_{n,P}$ has divisor $n(P) - ([n]P) - (n-1)\infty = n(P) - n(\infty)$ as required. Algorithm 1.1 presents Miller's algorithm to compute the Tate pairing.

Steps 2 to 9 are called the Miller loop. Since we only require the values $f_{n,P}(Q)$, instead of computing all intermediate functions $f_{i,P}$, Miller's algorithm simply computes their values at the point Q.

Algorithm 1.1 Miller's Algorithm to compute the Tate pairing [30]

Input: $P, Q \in E[n]$ and $n = (n_{l-1}n_{l-2}...n_1n_0)_2 \in \mathbb{N}$

Output: $f_{n,P}(Q)^{\frac{q^k-1}{n}}$

1: $T \leftarrow P$, $f \leftarrow 1$
2: **for** $i = l-2$ to 0 **do**
3: $\quad f \leftarrow f^2 \cdot \frac{l_{T,T}(Q)}{v_{2T}(Q)}$
4: $\quad T \leftarrow 2T$
5: $\quad$ **if** $l_i \neq 0$ **then**
6: $\quad\quad f \leftarrow f \cdot \frac{l_{T,P}(Q)}{v_{T+P}(Q)}$
7: $\quad\quad T \leftarrow T + P$
8: $\quad$ **end if**
9: **end for**
10: $f \leftarrow f^{\frac{q^k-1}{n}}$
11: **return** f

Chapter 2

Optimal Pairings

2.1 Pairing Friendly Curves

From the definition of the Tate pairing, one may suspect that a randomly chosen elliptic curve may not be suitable for implementing pairing-based protocols. For example, the coordinates of the point Q lie in the field $\mathbb{F}_{q^k}$. Hence, if the embedding degree of E is very large, it is unrealistic to implement arithmetic in a field of size q^k. The embedding degree with respect to n should be small enough so that arithmetic in the extension field can be implemented, yet large enough so that the DLP is intractible in $\mathbb{F}_{q^k}$. Another desirable property is that the elliptic curve E should have a large prime order subgroup since larger subgroups provide a higher level of security.

Example 2.1.1. Consider the following elliptic curve which was generated randomly using the software package magma:

$$E : y^2 = x^3 + 15762226x + 26554729 \text{ over } \mathbb{F}_q$$

where $q = 33554467$. The order of E is 33557039 which has a prime factor $n = 2393$. The embedding degree of E with respect to n is 1196. Given the limitations of today's computers, it is impractical to implement field extensions of degree 1196, even on a base field size as small as 25-bits.

In general, elliptic curves with suitably low embedding degrees are rare. Balasubramaniam and Koblitz [5] showed that one can expect $k \approx q$ for a randomly selected prime-order

elliptic curve over a randomly selected prime-order field. In fact, the probability that $k \leq \log^2 q$ is vanishingly small. Luca, Mireles, and Shparlinski [27] have obtained similar results when $\mathbb{F}_q$ is fixed. As a result, one needs a systematic way of generating pairing-friendly curves.

2.1.1 Barreto-Naehrig Curves

There are many methods available in the literature for generating elliptic curves for implementing pairings. For a comprehensive review, see [15]. We focus here on Barreto-Naehrig (BN) curves, discovered by Barreto and Naehrig in 2005 [7]. BN-curves have prime order and embedding degree 12 which is appropriate for implementation at the 128-bit security level. As we will see, BN-curves are ideal for implementing the O-Ate pairing, which is an optimized variant of the Tate pairing.

Let $Q(x)$ and $N(x)$ denote the polynomials

$$Q(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1,$$
$$N(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1.$$

Choose an integer x such that both $Q(x)$ and $N(x)$ evaluate to prime numbers and let $q = Q(x)$. This can be done easily by randomly choosing an x until both $Q(x)$ and $N(x)$ are prime. Now, choose $b \in \mathbb{F}_q^*$ such that $b+1$ is a quadratic residue. For the appropriately chosen b, the curve given by $E : y^2 = x^3 + b$ will have order $N(x)$ over $\mathbb{F}_q$ with embedding degree equal to 12 with respect to $N(x)$. To find a BN-curve, test values for b until the curve has the correct order. Additionally, $P = (1, \sqrt{b+1})$ is a point on E, which can be used as a generator for $E(\mathbb{F}_q)$.

2.2 Deriving the O-Ate Pairing

Starting with Miller's algorithm, we successively apply optimizations to obtain the O-Ate pairing and a variant of Miller's algorithm to compute it.

2.2.1 Choice of n

We choose n to have low Hamming weight if possible, so the number of addition steps in Miller's algorithm is minimized.

2.2.2 Choice of P

Note that in the Miller loop, we are implicitly computing $nP = \infty$ using a double-and-add algorithm. To make this computation simpler, we should choose P so that it's coordinates lie in a subfield of F_{q^k}. This also simplifies the computation of the lines $l_{T,T}$ and $l_{T,P}$. The points T and P are in a subfield, so the equation of the line through these points has coefficients in the subfield as well. We will choose P so it's coordinates lie in the base field $\mathbb{F}_q$. As mentioned above, one can use the point $P = (1, \sqrt{b+1})$ as a generator of this subgroup.

2.2.3 Final Exponentiation

For this part, we restrict ourselves to BN-curves. They have embedding degree $k = 12$, so we can write $k = 2d$. We can now implement $\mathbb{F}_{q^k}$ as a quadratic extension on top of $\mathbb{F}_{q^d}$. An element in $\mathbb{F}_{q^k}$ can then be represented as $a = \alpha + w\beta$ where $\alpha, \beta \in \mathbb{F}_{q^d}$ and w is an adjoined square root.

It is well known (Frobenius) that

$$(\alpha + w\beta)^{q^d} = (\alpha - w\beta). \tag{2.1}$$

The output of the Miller loop is raised to the exponent $\frac{q^k-1}{n}$. We can write this as–

$$\frac{q^k - 1}{n} = \frac{q^{2d} - 1}{n} = \frac{(q^d - 1)(q^d + 1)}{n}.$$

By the definition of embedding degree, $n \nmid q^d - 1$. Therefore, $n \mid q^d + 1$. Therefore, we can split the final exponentiation into two parts — exponention by $q^d - 1$ and $\frac{q^d+1}{n}$. The second part of the exponentiation will be dealt with in Chapter 3.

2.2.4 Denominator Elimination

We now describe an important optimization, known as denominator elimination [6], which speeds up computation time by almost 50% [38]. For this section, we will assume that we are working with BN-curves.

Theorem 2.2.1. *Let* $x \in \mathbb{F}_{q^d} \subset \mathbb{F}_{q^k}$. *Then* $x^{\frac{q^k-1}{n}} = 1$.

Proof. We know by an extension of Fermat's last theorem that $x^{q^d-1} = 1$. Then, we have:

$$x^{\frac{q^k-1}{n}} = x^{(q^d-1)(\frac{q^d+1}{n})} = 1^{\frac{q^d+1}{n}}.$$ □

Theorem 2.2.1 has a few implications for pairing computations. Firstly, we have already chosen P so that the coefficients of the equations of the lines $l_{T,T}$ and $l_{T,P}$ lie in the subfield $\mathbb{F}_q \subset \mathbb{F}_{q^d}$. If Q is also chosen to have coordinates in $\mathbb{F}_{q^d}$ then the values $l_{T,T}(Q)$ and $l_{T,P}(Q)$ also lie in the subfield $\mathbb{F}_{q^d}$. Using the same argument, we can reach the same conclusion for the values $v_{T,T}(Q)$ and $v_{T,P}(Q)$. Therefore, the output of the Miller loop lies in a subfield. Then, by Theorem 2.2.1, the pairing value is 1.

This implies that in order for the pairing to be useful, we must choose Q so that $Q \in E(\mathbb{F}_{q^k})$, and $Q \notin E(\mathbb{F}_{q^d})$.

Definition 2.2.2. Frobenius Endomorphism. The Frobenius endomorphism is the mapping $\Phi\colon E(\mathbb{F}_{q^k}) \to E(\mathbb{F}_{q^k})$ given by $(x, y) \mapsto (x^q, y^q)$.

One can check that above map is indeed an endomorphism over the field of definition of E or any extension of it. It is well known that the Frobenius endomorphism has two eigenspaces in $E(\mathbb{F}_{q^k})[n]$ with eigenvalues 1 and q.

The 1-*eigenspace* consists of $E(\mathbb{F}_q)$ (because $x^q = x$ for $x \in \mathbb{F}_q$). These are precisely the elements from which we choose the parameter P for the pairing computation.

The *q-eigenspace* consists of points of the form $(\alpha, w\beta)$ where $\alpha, \beta \in \mathbb{F}_{q^d}$ and w is an adjoined square root [5]. In order to be able to define the Ate pairing in the next section, we restrict Q to be a point from this set.

Both eigenspaces when restricted to $E[n]$ form subgroups of $E(\mathbb{F}_{q^k})[n]$ of order n. From now on, we will use G_1 to denote the 1-*eigenspace* and G_2 to denote the *q-eigenspace*

restricted to $E[n]$. Since $E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n$, and the two eigenspaces are independent of each other, one can conclude that a set of the form $\{P, Q\}$, $P \in G_1, Q \in G_2$ forms a basis for $E[n]$.

In summary, we choose $P \in G_1$ and $Q \in G_2$ for our pairing computation on BN-curves. Therefore, the values $v_{T,T}(Q)$ and $v_{T,P}(Q)$ lie in $\mathbb{F}_{q^d}$ — as the x-coordinate of Q lies in $\mathbb{F}_{q^d}$. So by Theorem 2.2.1, these values vanish after the final exponentiation and do not affect the pairing value. As a result, we can remove the computation of $v_{T,T}(Q)$ and $v_{T,P}(Q)$ from Miller's loop. This is the denominator elimination optimization.

2.2.5 The Ate Pairing

The Ate Pairing was discovered in 2006 by Hess, Smart and Vercauteren [21]. It is similar to the Tate pairing with its parameters restricted to the Frobenius eigenspaces, and the Miller loop is shorter, hence the name Ate (Tate with the T removed). Here we derive the Ate pairing using an approach presented in [44], which will lead to the construction of the Optimal Ate pairing. First, we will switch the order of the parameters of the pairing, i.e. we will choose the first parameter $Q \in G_2$ and the second parameter $P \in G_1$. This may seem peculiar, but all of the optimizations described above are still valid, since $P \in E(\mathbb{F}_q) \subset E(\mathbb{F}_{q^d})$. The reason for this choice will soon become clear as we construct the Ate pairing.

We start with the following lemma:

Lemma 2.2.3. *$f_{ab,Q} = f_{a,Q}^b \cdot f_{b,aQ}$ for all $a, b \in \mathbb{Z}$.*

Proof. Observe that

$$\begin{aligned} \operatorname{div}(f_{a,Q}^b) &= b(a(Q) - ([a]Q) - (a-1)(\infty)) \\ &= ba(Q) - b([a]Q) - b(a-1)(\infty). \text{ Also,} \\ \operatorname{div}(f_{b,aQ}) &= b(aQ) - ([ab]Q) - (b-1)(\infty). \end{aligned}$$

Therefore,

$$\begin{aligned} \operatorname{div}(f_{a,Q}^b\, f_{b,aQ}) &= ba(Q) - b([a]Q) - b(a-1)(\infty) + b([a]Q) - ([ab]Q) - (b-1)(\infty) \\ &= ba(Q) - ([ab]Q) - (ba-1)(Q) \\ &= \operatorname{div}(f_{ab,Q}). \end{aligned}$$

□

This leads to the following lemma from [44].

Lemma 2.2.4. *$e(Q,P)^m = f_{mn,Q}(P)^{\frac{q^k-1}{n}}$ where $e(Q,P)$ is the Tate pairing with $Q \in G_2$, $P \in G_1$ and $m \in \mathbb{Z}$.*

Proof. We have

$$\begin{aligned} e(Q,P)^m &= f_{n,Q}(P)^{\frac{q^k-1}{n}m} \\ &= \frac{f_{nm,Q}(P)^{\frac{q^k-1}{n}}}{f_{m,nQ}(P)} \\ &= \frac{f_{nm,Q}(P)^{\frac{q^k-1}{n}}}{f_{m,\infty}(P)} \\ &= f_{nm,Q}(P)^{\frac{q^k-1}{n}}. \end{aligned}$$

□

The next step is to find a special multiple mn of n such that the computation of $f_{nm,Q}$ can be performed faster than the computation of $f_{n,Q}$. This is done by writing $f_{nm,Q}$ as the power of a Miller function with a shorter loop. We fix $\lambda \equiv q \bmod n$. This implies that

$n|\lambda^k - 1$. So we take $m = \frac{\lambda^k - 1}{n}$. By the above lemma, we have:

$$\begin{aligned}
e(Q,P)^m &= f_{nm,Q}(P)^{\frac{q^k-1}{n}} \\
&= f_{\lambda^k-1,Q}(P)^{\frac{q^k-1}{n}} \\
&= \left(f_{\lambda^k,Q}(P) f_{-1,Q}(P) \frac{l_{\lambda^k Q,-Q}(P)}{v_{(\lambda^k-1)Q}(P)} \right)^{\frac{q^k-1}{n}} \\
&= \left(f_{\lambda^k,Q}(P) \cdot 1 \cdot \frac{l_{Q,-Q}(P)}{v_{(\lambda^k-1)Q}(P)} \right)^{\frac{q^k-1}{n}} \\
&= f_{\lambda^k,Q}(P)^{\frac{q^k-1}{n}} && \text{(by denominator elimination)} \\
&= f_{\lambda,Q}^{\lambda^{k-1}}(P) f_{\lambda^{k-1},\lambda Q}(P))^{\frac{q^k-1}{n}} && \text{(by Lemma 2.2.3)} \\
&\vdots \\
&= (f_{\lambda,Q}^{\lambda^{k-1}}(P) f_{\lambda,\lambda Q}^{\lambda^{k-2}}(P) ... f_{\lambda,\lambda^{k-1}Q}(P))^{\frac{q^k-1}{n}} && \text{(repeatedly applying Lemma 2.2.3)} \\
&= \left(\prod_{i=0}^{k-1} f_{\lambda,q^i Q}^{\lambda^{k-1-i}} \right)^{\frac{q^k-1}{n}} . && (\textit{Since } \lambda \equiv q \bmod n)
\end{aligned} \tag{2.2}$$

At this point, we need the following fact:

Fact 2.2.5. *$f_{a,\Pi_q(Q)}(P) = f_{a,Q}(P)^q$ for all $a \in \mathbb{Z}$ and $Q \in G_2$.*

Proof. We will use induction on a. Clearly, $1 = f_{1,qQ}(P) = (f_{1,Q}(P))^q$. Suppose the fact holds for all $a = 1, 2, ..., k$. Let $kQ = (x_{kQ}, y_{kQ})$, $Q = (x_Q, y_Q)$, and $P = (x, y)$. Then:

$$(f_{k+1,Q}(P))^q = \left(f_{k,Q}(P) f_{1,Q}(P) \frac{l_{kQ,Q}(P)}{v_{[k+1]Q}(P)} \right)^q$$

We also have that:

$$\begin{aligned}
(l_{kQ,Q}(P))^q &= \left(\frac{y-y_Q}{x-x_Q} - \frac{y_{kQ}-y_Q}{x_{kQ}-x_Q}\right)^q \\
&= \left(\frac{y^q-y_Q^q}{x^q-x_Q^q} - \frac{y_{kQ}^q-y_Q^q}{x_{kQ}^q-x_Q^q}\right) && \text{(since } q \text{ is a prime power)} \\
&= \left(\frac{y-y_{\Pi_q(Q)}}{x-x_{\Pi_q(Q)}} - \frac{y_{\Pi_q(kQ)}-y_{\Pi_q(Q)}}{x_{\Pi_q(kQ)}-x_{\Pi_q(Q)}}\right) \\
&= l_{\Pi_q(kQ),\Pi_q(Q)}(P).
\end{aligned}$$

We can similarly show that $(v_{[k+1]Q(P)})^q = v_{[k+1]\Pi_q(Q)}(P)$. It now follows that $(f_{a+1,Q}(P))^q = f_{a+1,\Pi_q(Q)}(P) = f_{a+1,qQ}(P)$ for all $a \in \mathbb{Z}$, since Q is in the q-eigenspace of Frobenius. □

It also follows using the same technique that $f_{a,q^iQ}(P) = (f_{a,Q}(P))^{q^i}$. Hence, we obtain:

$$\prod_{i=0}^{k-1} f_{\lambda,q^iQ}^{\lambda^{k-1-i}}(P) = \prod_{i=0}^{k-1} f_{\lambda,Q}^{q^{k-1-i}q^i}(P) = f_{\lambda,Q}(P)^{\sum_{i=0}^{k-1} q^{k-1}}.$$

Substituting the above into Equation (2.2), we get $e(Q,P)^m = f_{\lambda,Q}(P)^{\frac{q^k-1}{n}kq^{k-1}}$.

Since n and q are prime, we have that $n \nmid q$ and $n \nmid k$. Thus we can define:

$$a(Q,P) = e(Q,P)^{m((k^{-1}q^{-(k-1)}) \bmod n)} = f_{\lambda,Q}(P)^{\frac{q^k-1}{n}}.$$

We call this the reduced Ate pairing [44]. It is bilinear and non-degenerate because $m((k^{-1}q^{-(k-1)}) \bmod n)$ is relatively prime to n. In the case of BN-curves, $q(x) \cong 6x^2 \pmod{n(x)}$, so we take $\lambda = 6x^2$. This reduces the Miller loop length from $\log(36x^4 + 36x^3 + 18x^2 + 6x + 1)$ to $\log(6x^2)$.

Suppose we take $\lambda = q^i \bmod n$. Then, $n \mid \lambda^{i\gcd(i,k)} - 1$. After performing an analysis similar to above, we get more Ate pairings with loop length λ.

It is possible to further reduce the length of the Miller loop, as shown by Vercauteren [44]. The pairing obtained through his work is called the Optimal Ate or the O-Ate pairing.

2.2.6 The Optimal Ate Pairing

Consider the m^{th} power of the Tate pairing. Let $\sigma = mn$ and suppose $\sigma = \sum_{i=0}^{l} c_i q^i$ is the base-q expansion of σ. Define $s_i = \sum_{j=i}^{l} c_j q^j$ We have:

$$\begin{aligned}
e(Q,P)^m &= f_{n,Q}(P)^{m\frac{q^k-1}{n}} \\
&= f_{\sigma,Q}(P)^{\frac{q^k-1}{n}} && \text{(By Lemma 2.2.4)} \\
&= f_{s_0,Q}(P)^{\frac{q^k-1}{n}} \\
&= \left(f_{c_0,Q}(P) f_{s_1,Q}(P) \frac{l_{c_0Q,s_1Q}(P)}{v_{s_0,Q}(P)} \right)^{\frac{q^k-1}{n}} \\
&\vdots \\
&= \left(\prod_{i=0}^{l} f_{c_i q^i,Q}(P) \prod_{i=0}^{l-1} \frac{l_{[c_i q^i]Q,s_{i+1}Q}(P)}{v_{s_i,Q}(P)} \right)^{\frac{q^k-1}{n}} \\
&= \left(\prod_{i=0}^{l} f_{q^i,Q}^{c_i}(P) f_{c_i,[q^i]Q}(P) \prod_{i=0}^{l-1} \frac{l_{[c_i q^i]Q,s_{i+1}Q}(P)}{v_{s_i,Q}(P)} \right)^{\frac{q^k-1}{n}} && \text{(By Lemma 2.2.3)} \\
&= \left(\prod_{i=0}^{l} f_{q^i,Q}^{c_i}(P)^{\frac{q^k-1}{n}} \right) \left(\prod_{i=0}^{l} f_{c_i,Q}^{q^i}(P) \prod_{i=0}^{l-1} \frac{l_{[c_i q^i]Q,s_{i+1}Q}(P)}{v_{s_i,Q}(P)} \right)^{\frac{q^k-1}{n}}. && \text{(By Fact 2.2.5)}
\end{aligned}$$

The left hand side of the above equation is a bilinear pairing. The factor between the first set of brackets on the right hand side is a product of powers of Ate pairings, so it is also bilinear. Hence, the factor between the second set of brackets must also be a bilinear pairing. We now have the following result.

Theorem 2.2.6 ([44]). *Let $\sigma = mn$ with $n \nmid m$ and define c_i and s_i as above. Then*

$a_O \colon G_2 \times G_1 \to \mu_n$ *given by:*

$$(Q, P) \mapsto \left(\prod_{i=0}^{l} f_{c_i,Q}^{q^i}(P) \prod_{i=0}^{l-1} \frac{l_{[c_i q^i]Q, s_{i+1}Q}(P)}{v_{s_i,Q}(P)} \right)^{\frac{q^k-1}{n}}$$

defines a bilinear pairing called the O-Ate pairing. Furthermore, if

$$mkq^{-1} \not\equiv \frac{q^k - 1}{r} \sum_{i=0}^{l} i c_i q^{i-1} \bmod n,$$

then the pairing is non-degenerate.

Proof. We have just shown the bilinearity of the O-Ate pairing. For non-degeneracy, see [44]. □

Since we are using BN-curves, denominator elimination applies and we can ignore the vertical lines in the pairing.

Recall that on BN-curves q and n are given by polynomials in x:

$$\begin{aligned} Q(x) &= 36x^4 + 36x^3 + 24x^2 + 6x + 1, \\ N(x) &= 36x^4 + 36x^3 + 18x^2 + 6x + 1. \end{aligned}$$

We then have:

$$\begin{aligned} Q(x) &= 6x^2 \bmod N(x), \\ Q(x)^2 &= 36x^3 - 18x^2 - 6x - 1 \bmod N(x), \\ Q(x)^3 &= 36x^3 - 24x^2 - 12x - 3 \bmod N(x). \end{aligned}$$

Therefore,

$$6x + 2 + Q(x) - Q(x)^2 + Q(x)^3 = 0 \bmod N(x). \tag{2.3}$$

Take $\sigma = 6x + 2 + Q(x) - Q(x)^2 + Q(x)^3$, which gives the following O-Ate pairing on BN-curves:

$$(Q, P) \mapsto f_{6x+2,Q}(P) f_{1,Q}^{q}(P) f_{-1,Q}^{q^2}(P) f_{1,Q}^{q^3}(P) g(P).$$

where $g(P)$ is given by:

$$g(P) = l_{[6x+2]Q,[q-q^2+q^3]Q}(P) l_{[q]Q,[-q^2+q^3]Q}(P) l_{[-q^2]Q,[q^3]Q}(P).$$

Now, $f_{1,Q} = f_{-1,Q} = 1$, so we can ignore these in the pairing computation. We make the computation of $g(P)$ easier using the following lemma discovered in [33]:

Lemma 2.2.7. *$g(P)^{\frac{q^k-1}{n}} = h(P)^{\frac{q^k-1}{n}}$ where $h(P) = l_{[6x+2]Q,qQ}(P) l_{[6x+2]Q+qQ,-q^2Q}(P)$.*

Proof. Fix $Q_1 = qQ$, $Q_2 = q^2Q$, $Q_3 = q^3Q$, and $Q_x = [6x+2]Q$. Then

$$\begin{aligned}
\mathrm{div}(g) &= \mathrm{div}(l_{Q_x,Q_1-Q_2+Q_3}) + \mathrm{div}(l_{Q_1,-Q_2+Q_3}) + \mathrm{div}(l_{-Q_2,Q_3}) \\
&= (Q_x) + (Q_1 - Q_2 + Q_3) + (-[Q_x + Q_1 - Q_2 + Q_3]) + \\
&\quad (Q_1) + (Q_3 - Q_2) + ([Q_2 - Q_3 - Q_1]) + (-Q_2) + (Q_3) + \\
&\quad (Q_2 - Q_3) - 9(\infty) \\
&= (Q_x) + (Q_1 - Q_2 + Q_3) + (Q_1) + (Q_3 - Q_2) + \\
&\quad ([Q_2 - Q_3 - Q_1]) + (-Q_2) + (Q_3) + (Q_2 - Q_3) - 8(\infty) \qquad \text{(by equation (2.3))} \\
&= (Q_x) + (Q_1) + (-Q_2) + (Q_3) - 4(\infty) + \mathrm{div}(v_{Q_1-Q_2+Q_3}) + \mathrm{div}(v_{Q_2-Q_3}).
\end{aligned}$$

Also,

$$\begin{aligned}
\mathrm{div}(h) &= \mathrm{div}(l_{Q_x,Q_1}) + \mathrm{div}(l_{Q_x+Q_1,-Q_2}) \\
&= (Q_x) + (Q_1) + (-Q_x - Q_1) + (Q_x + Q_1) + (-Q_2) + (Q_2 - Q_1 - Q_x) - 6(\infty) \\
&= (Q_x) + (Q_1) + (-Q_2) + (Q_2 - Q_1 - Q_x) - 4(\infty) + \mathrm{div}(v_{Q_x+Q_1}) \\
&= (Q_x) + (Q_1) + (-Q_2) + (Q_3) - 4(\infty) + \mathrm{div}(v_{Q_x+Q_1}) \qquad \text{(by Equation (2.3))}.
\end{aligned}$$

We now see that the divisors of g and h differ only by divisors of vertical lines. Hence, g and h must also differ only by vertical lines. Since the denominator elimination optimization applies, we can ignore these vertical lines in the computation of $g(P)^{\frac{q^k-1}{n}}$ and $h(P)^{\frac{q^k-1}{n}}$. □

By the above lemma, $g(P)$ can be replaced by $h(P)$ when computing the O-Ate pairing on BN-curves. In this way we obtain the following O-Ate pairing on BN-curves:

$$(Q, P) \mapsto f_{6x+2,Q}(P) \cdot h(P).$$

2.3 Twists on Elliptic Curves

We now show how a pairing computation can be accelerated using elliptic curve twists.

Definition 2.3.1. Let E be an elliptic curve defined over $\mathbb{F}_q$. An elliptic curve E' is called a twist of E if there exists an isomorphism $\Psi\colon E'(\mathbb{F}_{q^r}) \to E(\mathbb{F}_{q^r})$ defined over the extension field $\mathbb{F}_{q^r}$. The minimum extension degree for which there exists such an isomorphism is called the degree of the twist.

BN-curves have two twists of degree six when defined over $\mathbb{F}_{q^2}$ [42, Prop.X.5.4]. Let $E : y^2 = x^3 + b$ be a BN-curve defined over $\mathbb{F}_q$ and let ξ be a cubic and quadratic non-residue over $\mathbb{F}_{q^2}$. Then the following curve is a degree 6 twist of E:

$$E' : y^2 = x^3 + \frac{b}{\xi}. \tag{2.4}$$

We will call the twist given by the above equation a D-type twist (because ξ is being divided). The twisting isomorphism is given by:

$$\Psi\colon E' \to E$$

$$\Psi\colon (x, y) \mapsto (\xi^{\frac{1}{3}}x, \xi^{\frac{1}{2}}y).$$

The second sextic twist of E is given by the equation:

$$E'' : y^2 = x^3 + \xi b. \tag{2.5}$$

The above is known as an M-type twist because ξ is being multiplied. In this case, the twisting isomorphism is given by:

$$\Psi\colon E'' \to E$$

$$\Psi\colon (x, y) \mapsto (\xi^{-\frac{2}{3}}x, \xi^{-\frac{1}{2}}y).$$

Of the two sextic twists of BN-curves, exactly one of them maps points in G_2 to points on the twisted curve over $\mathbb{F}_{q^2}$ [7]. Hence, we can represent the group G_2 using points on $E'[n](\mathbb{F}_{q^2})$ or $E''[n](\mathbb{F}_{q^2})$. We will call this group of points G_2'. This gives us the Twisted O-Ate pairing on BN-curves:

$$\hat{a_O}\colon \mathbb{G}_2' \times \mathbb{G}_1 \to \mu_n$$

$$\hat{a_O}\colon (Q', P) \mapsto f_{6x+2,\Psi(Q')}(P)h'(P)$$

where $h'(P) = l_{[6x+2]\Psi(Q'),q\Psi(Q')}(P)l_{[6x+2]\Psi(Q')+q\Psi(Q'),-q^2\Psi(Q')}(P)$.

To determine which of the two sextic twists is the correct one to use in pairing computation, we compute the order of the twisted curve over $\mathbb{F}_{q^2}$. The curve with order dividing n is the

correct twist. Instead of using a degree 12 extension, the point Q can now be represented using only elements in a quadratic extension field. In addition to the benefit of saving space, $E'(\mathbb{F}_{q^2})[n]$ is also much faster to manipulate. When doubling and adding points and computing the line function in the Miller loop, we can perform the arithmetic in G_2', and then map the result to G_2. This considerably speeds up operations in the Miller loop. We will elaborate on this in Chapter 3.

Chapter 3

Field Arithmetic

3.1 Representation of Extension Fields

BN-curves are defined over prime fields, which means the computation of a pairing over a BN-curve relies on arithmetic over finite fields. Hence, efficient implementation of the underlying extension fields is crucial for fast pairing computation. Arithmetic over $\mathbb{F}_{q^2}$ is required for manipulating points on the twisted curve, and the computation of the Miller function. Moreover, accumulating and multiplying values to compute $f_{n,P}$ and the final exponentiation involves arithmetic over $\mathbb{F}_{q^{12}}$. IEEE P1363.3 [1] recommends using towers to represent $\mathbb{F}_{q^k}$. Construction of tower extensions for the purpose of pairing computation has been explored in [8, 17, 24, 7]. Next we outline and analyze two approaches for constructing tower fields.

Note that if $q \equiv 3 \bmod 4$, then both -1 and -2 are quadratic non-residues and we can represent $\mathbb{F}_{q^2}$ by $\mathbb{F}_q[i]/(i^2 - \beta)$ where $\beta = -1$ or -2. Multiplications by i are required throughout the pairing computation, for instance when multiplying two extension field elements. Representing $\mathbb{F}_{q^2}$ as above, multiplications by i are very cheap, requiring either only a simple negation or a negation and an addition. Having the choice of 2 elements for β also leaves us some choice of representation for implementing higher extensions. When x is odd, we get $Q(x) \equiv 3 \bmod 4$, and when x is even, we get $Q(x) \equiv 1 \bmod 4$. When x is even, neither -1 nor -2 is guaranteed to be a quadratic non-residue so multiplication by i can end up being relatively costly. Therefore, when computing BN-curves using the polynomial $Q(x)$, we restrict ourselves to choosing odd x, so that $q \equiv 3 \bmod 4$.

Geovandro et al. [17] recommend a family of implementation friendly BN-curves which has a very natural choice for the suitable representation of extension fields. We give a description of this sub-family and outline its benefits.

Definition 3.1.1. A BN-curve $E_b : y^2 = x^3 + b$ over $\mathbb{F}_q$ is called friendly if $q \equiv 3 \bmod 4$ and there exist $c, d \in \mathbb{F}_q^*$ such that either $b = c^4 + d^6$ or $b = c^6 + 4d^4$.

One can use the following properties of friendly BN-curves to implement the pairing computation in an efficient manner:

1. Let $\xi = c^2 + d^3 i$ if $b = c^4 + d^6$, or $\xi = c^3 + 2d^2 i$ if $b = c^6 + 4d^4$. Then, $b = \xi\overline{\xi}$. Lemma 2 of [17] says that ξ is neither a square nor a cube in $\mathbb{F}_{p^2}$. Thus, we can use ξ to construct $\mathbb{F}_{q^{12}}$ as follows:
$$\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[v]/(v^3 - \xi).$$
$$\mathbb{F}_{q^{12}} = \mathbb{F}_{q^6}[w]/(w^2 - v).$$

2. Theorem 1 in [17] says that the curve E'_b given by:
$$E'_b : y^2 = x^3 + \frac{b}{\xi} = x^3 + \overline{\xi}$$
gives a D-type sextic twist of E_b.

3. Generators for $E'(\mathbb{F}_{p^2})[n]$ can be found as $[h]G$, where $h = 2p - n$ and $G = (-di, c)$ or $G = (-c, d(1 - i))$, respectively.

Using the above sub-family, square or cube detection is not necessary to build field extensions or generate a twist. Moreover, one does not have to compute the order of the curve which may generate the sextic twist, as the correct twist is immediately revealed.

Another approach to finding appropriate irreducible polynomials for constructing tower extensions of fields is to use the following theorem of Benger and Scott [8]:

Theorem 3.1.2 ([8])**.** *Let $m > 1, n > 0$ be integers, q an odd prime and $\alpha \in \mathbb{F}_{q^n}^*$. The binomial $x^m - \alpha$ is irreducible in $\mathbb{F}_{q^n}[x]$ if the following two conditions are satisfied:*

- *Each prime factor p of m divides $q - 1$ and $N_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\alpha) \in \mathbb{F}_q$ is not a p-th residue in $\mathbb{F}_q$;*

- *If* $m \equiv 0 \bmod 4$ *then* $q^4 \equiv 1 \bmod 4$.

Based on the above theorem, Benger and Scott [8] give the following construction for BN primes congruent to 3 mod 8. The same conclusion is also drawn in Shirase [41]:

Construction 3.1.3. *Let* $q = q(x)$ *be the prime characteristic of the field over which a BN-curve is defined. If* $x \equiv 7$ *or* $11 \bmod 12$ *then* $y^6 - (1 + \sqrt{-1})$ *is irreducible over* $\mathbb{F}_{q^2} = \mathbb{F}_q(\sqrt{-1})$.

We are able to use the above construction in 2/3rds of the cases when $q \equiv 3 \bmod 8$; i.e. $x \equiv 3 \bmod 4$. It only fails when $x \equiv 2 \bmod 3$. Since we are restricting ourselves to choosing odd x, we also need to consider the case when $x \equiv 1 \bmod 4$. Following [8], we give the following construction for BN-primes congruent to 7 mod 8:

Construction 3.1.4. *Let* $q = q(x)$ *be the prime characteristic of the field over which a BN-curve is defined. If* $x \equiv 2, 3, 4, 6, 7$ *or* $8 \bmod 9$ *then* $y^6 - (1 + \sqrt{-2})$ *is irreducible over* $\mathbb{F}_{q^2} = \mathbb{F}_q(\sqrt{-2})$.

Proof. We will show that the conditions in Theorem 3.1.2 are satisfied two-thirds of the time for $m = 6, n = 2$, when q a BN-prime congruent to 7 mod (8), and $\alpha = 1 + \sqrt{-2}$. To satisfy condition (2), it suffices to show that $q^4 \equiv 1 \bmod 4$. This is trivial because $q \equiv 3 \bmod 4$. Now, $N_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\alpha) = (1 + \sqrt{-2})(1 - \sqrt{-2}) = 3$; and the prime factors of 12 are 2 and 3. To satisfy condition (1) we need to show the following:

- $2 \mid q - 1$ and $3 \mid q - 1$;
- 3 is not a cubic or a quadratic residue in $\mathbb{F}_q$.

Recall that q is given by the polynomial $q(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ for some $x \equiv 1 \bmod 4$. As a result $2 \mid q - 1$ and $3 \mid q - 1$. Morevoer, $x \equiv 1 \bmod 4$ implies that $x \equiv 1, 5$ or $9 \bmod 12$, which in turn implies that $q \equiv 7 \bmod 12$. As a result, 3 is not a quadratic residue in $\mathbb{F}_q$. We now need to determine when 3 is a cubic residue in $\mathbb{F}_q$. A prime $q \equiv 1 \bmod 3$ can be written as $q = a^2 + 3b^2$ for some integers a, b. It was conjectured by Euler and proven by Gauss that 3 is a cubic residue if and only if $9 \mid b$, or $9 \mid (a \pm b)$ [26]. For BN-primes we can write $q(x) = a(x) + 3x^2$, where $a(x) = 6x^2 + 3x + 1$ [41]. Hence, 3 is a cubic residue if $9 \mid x$, or $9 \mid 6x^2 + 4x + 1$, or $9 \mid 6x^2 + 2x + 1$. This occurs when $x \equiv 0, 1,$ or $5 \bmod 9$ which happens with probability 1/3. Thus 3 is a cubic non-residue modulo q for approximately 2/3rds of the values $q \equiv 7 \bmod 8$. □

When deciding on the above construction for BN-primes congruent to 7 mod 8, we tried to choose α so that $N_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\alpha)$ is minimized. This means that the polynomials used to construct tower extensions will have small coefficents, so arithmetic will be efficient.

3.1.1 Towering Scheme for Primes Congruent to 3 mod 8

Aranha et al. [4] use $E : y^2 = x^3 + 2$ for the BN-curve, and $x = -(2^{62} + 2^{55} + 1)$ to generate the 254-bit prime $Q(x)$. As a result, both methods outlined above yield the same towering scheme for the prime field over which this curve is defined:

- $\mathbb{F}_{q^2} = \mathbb{F}_q[i]/(i^2 - \beta)$, where $\beta = -1$.
- $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[v]/(v^3 - \xi)$, where $\xi = 1 + i$.
- $\mathbb{F}_{q^{12}} = \mathbb{F}_{q^6}[w]/(w^2 - v)$.

This towering scheme is ideal since it keeps the coefficents of the irreducible polynomials as small as possible. At some points during the pairing computation, it is required that finite field elements be multiplied by ξ (for example, when multiplying two elements over $\mathbb{F}_{q^{12}}$). Using the above towering scheme, multiplication by i requires one negation over $\mathbb{F}_q$, and multiplication by ξ requires only one addition over $\mathbb{F}_{q^2}$. For primes congruent to 3 mod 8, we use the above towering scheme.

We represent all prime and extension fields using a towering scheme as above, varying the choice of ξ and v to suit the prime q in question.

3.1.2 Towering Scheme for Primes Congruent to 7 mod 8

We now illustrate explicit towering schemes for primes congruent to 7 mod 8 using the 446-bit prime given by $Q(x)$, where $x = 2^{110} + 2^{36} + 1$. This prime was recommended in [17] and used in [3] to implement the O-Ate pairing. In this case, the above scheme does not work because $1 + i$ is not a cubic non-residue in $\mathbb{F}_{p^2}$. Following the recommendation in [3], the BN-curve used is:

$$E_{257} : y^2 = x^3 + 257$$

We then get the following towering scheme which we refer to as **Scheme 1**:

- $\mathbb{F}_{q^2} = \mathbb{F}_q[i]/(i^2 - \beta)$, where $\beta = -1$.
- $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[v]/(v^3 - \xi)$, where $\xi = 16 + i$.
- $\mathbb{F}_{q^{12}} = \mathbb{F}_{q^6}[w]/(w^2 - v)$.

Here β is minimal, however ξ is slightly large. Minimizing β might be beneficial because a large chunk of the arithmetic during pairing computation is performed over $\mathbb{F}_{q^2}$. Multiplication by i requires a negation; multiplication by ξ requires five additions in $\mathbb{F}_{q^2}$.

To increase the efficiency of the pairing computation we can try a towering scheme as dictated by Construction 2. This will make multiplication by ξ cheaper:

- $\mathbb{F}_{q^2} = \mathbb{F}_q[i]/(i^2 - \beta)$, where $\beta = -2$.
- $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[v]/(v^3 - \xi)$, where $\xi = 1 + i$.
- $\mathbb{F}_{q^{12}} = \mathbb{F}_{q^6}[w]/(w^2 - v)$.

We refer to the above scheme as **Scheme 2**. Here, multiplication by i requires one addition and one negation in $\mathbb{F}_q$, and multiplication by ξ requires two additions in $\mathbb{F}_{q^2}$. Note that here we have taken the opposite approach to the one suggested in [17]. Instead of choosing a curve that gives the right twist, and then letting these choices dictate the towering scheme, we first choose a towering scheme that optimizes extension field arithmetic and deal with the curve later. As illustrated in Appendix A using the curve BN446, using Scheme 2 results in a faster pairing than using Scheme 1. Since we did not follow the recommendations of [17], we lose the benefit of being able to generate towering schemes and twists without performing additional mathematical operations. However, in pairing-based protocols, these operations need only be performed once, whereas there may be thousands of pairing computations required. As the bulk of the pairing computation requires extension field arithmetic, optimizing the arithmetic leads to better performance overall.

3.2 Field Arithmetic

3.2.1 Lazy Reduction

Before proceeding, we fix some notation regarding field operation algorithms and costs. Lower case variables denote single-precision integers, and upper case variables denote double-precision integers. $\times$ represents multiplication without reduction, and $\otimes$ represents multiplication with reduction. The letters m, s, a, and i denote a multiplication, squaring, addition, and inversion in $\mathbb{F}_q$ respectively. Likewise $\tilde{m}, \tilde{s}, \tilde{a}$, and $\tilde{\imath}$ denote multiplication, squaring, addition and inversion in $\mathbb{F}_{q^2}$ respectively. $m_u, \tilde{m}_u, s_u$ and $\tilde{s}_u$ denote unreduced multiplications and squarings in the respective field. We write m_b, m_i, m_ξ, and m_v for multiplication by b, i, ξ, and v respectively. To perform arithmetic over finite fields, we use Karatsuba multiplication and squaring with lazy reduction as in [4]. We extend their idea of lazy reduction to field inversion. By applying lazy reduction, we are able to save one $\mathbb{F}_q$ reduction per $\mathbb{F}_{q^2}$ inversion, and 13 $\mathbb{F}_q$ reductions per $\mathbb{F}_{q^{12}}$ inversion. This speeds up the inversion routine in $\mathbb{F}_{q^2}$ by 4%, and in $\mathbb{F}_{q^{12}}$ by 10%. Algorithms 3.1, 3.2, and 3.3 present our routines for inversion in the extension fields using lazy reduction. All costs for algorithms presented in this chapter are for the 254-bit BN prime used in [4].

Algorithm 3.1 Inversion in $\mathbb{F}_{q^2}$ (Cost $= i + 4m + 3r + 2a$)

Input: $a = a_0 + a_1 i;\ a_0, a_1 \in \mathbb{F}_q$
Output: $c = a^{-1} \in \mathbb{F}_{q^2}$
$T_0 \leftarrow a_0 \times a_0$
$T_1 \leftarrow -\beta \cdot (a_1 \times a_1)$
$T_0 \leftarrow T_0 + T_1$
$t_0 \leftarrow T_0 \bmod p$
$t \leftarrow t_0^{-1} \bmod p$
$c_0 \leftarrow a_0 \otimes t$
$c_1 \leftarrow -(a_1 \otimes t)$
return $c = c_0 + c_1 i$

3.2.2 Multiplication of Sparse Elements

Using the above towering scheme, the elements $\{1, v, v^2, w, vw, v^2 w\}$ form a basis for $\mathbb{F}_{q^{12}}$ over $\mathbb{F}_{q^2}$. When using projective and jacobian coordinates, the line function in the Miller

Algorithm 3.2 Inversion in $\mathbb{F}_{q^6}$ (Cost $= \tilde{i} + 9\tilde{m} + 3\tilde{s} + 9\tilde{r} + 14\tilde{a}$)

Input: $a = a_0 + a_1 v + a_2 v^2;\ a_0, a_1, a_2 \in \mathbb{F}_{q^2}$
Output: $c = a^{-1} \in \mathbb{F}_{q^6}$

$T_0 \leftarrow a_0 \times a_0$
$V_0 \leftarrow a_1 \times a_2$
$V_0 \leftarrow \xi V_0$
$V_0 \leftarrow T_0 - V_0$
$v_0 \leftarrow V_0 \bmod p$
$T_0 \leftarrow a_2 \times a_2$
$T_0 \leftarrow \xi T_0$
$V_1 \leftarrow a_1 \times a_0$
$V_1 \leftarrow T_0 - V_1$
$v_1 \leftarrow V_1 \bmod p$
$T_0 \leftarrow a_1 \times a_1$
$V_2 \leftarrow a_2 \times a_0$
$V_2 \leftarrow T_0 - V_2$
$v_2 \leftarrow V_2 \bmod p$
$c_1 \leftarrow a_1 \otimes v_2$
$c_1 \leftarrow \xi c_1$
$c_0 \leftarrow a_0 \otimes v_0$
$c_2 \leftarrow a_2 \otimes v_1$
$c_2 \leftarrow \xi c_2$
$t_0 \leftarrow c_0 + c_1$
$t_0 \leftarrow t_0 + c_2$
$t_0 \leftarrow t_0^{-1} \bmod p$
$c_0 \leftarrow v_0 \otimes t_0$
$c_1 \leftarrow v_1 \otimes t_0$
$c_2 \leftarrow v_2 \otimes t_0$
return $c = c_0 + c_1 v + c_2 v^2$

Algorithm 3.3 Inversion in $\mathbb{F}_{q^{12}}$ (Cost $= \tilde{\imath} + 15\tilde{m} + 9\tilde{s} + 18\tilde{r} + 69\tilde{a}$)

Input: $a = a_0 + a_1 w;\ a_0, a_1 \in \mathbb{F}_{q^6}$
Output: $c = a^{-1} \in \mathbb{F}_{q^{12}}$
$T_0 \leftarrow a_0 \times a_0$
$T_1 \leftarrow v \cdot (a_1 \times a_1)$
$T_0 \leftarrow T_0 - T_1$
$t_0 \leftarrow T_0 \bmod p$
$t_0 \leftarrow t_0^{-1} \bmod p$
$c_0 \leftarrow a_0 \otimes t_0$
$c_1 \leftarrow -a_1 \otimes t_0$
return $c = c_0 + c_1 w$

loop evaluates to a sparse $\mathbb{F}_{q^{12}}$ element containing only three of the six basis elements. In the case of a D-type twist, the line function evaluates to an element of the form

$$a_0 + a_1 w + a_2 v w, \qquad a_0, a_1, a_2 \in \mathbb{F}_{q^2}.$$

In the case of an M-type twist, it evaluates to an element of the form -

$$a_0 + a_1 v + a_2 v w, \qquad a_0, a_1, a_2 \in \mathbb{F}_{q^2}.$$

In both cases, when multiplying the line function evaluation with $f_{i,Q}(P)$, one can utilize its sparseness to avoid full $\mathbb{F}_{q^{12}}$ arithmetic. We use Algorithm 3.4 to multiply a sparse $\mathbb{F}_{q^{12}}$ element with a non-sparse $\mathbb{F}_{q^{12}}$ element. In this case, the sparse element arises as the evaluation of a line function when a D-type twist is involved. Note that multiplication by v involves a multiplication by ξ, which in turn is equal to one $\mathbb{F}_{q^2}$ addition. The dense-sparse multiplication algorithm presented in Algorithms 3.4 and 3.5 requires 17 fewer $\mathbb{F}_{q^2}$ additions than in [4]. The dense-sparse multiplication algorithm is similar when we are using an M-type twist, and requires an extra multiplication by v.

3.2.3 Mapping from the Twisted Curve to the Original Curve

Suppose we take ξ (as used in the towering scheme) to be the cubic and quadratic non-residue to generate the sextic twist of the BN-curve E. In the case of a D-type twist, the untwisting isomorphism is given by:

$$\Psi \colon (x, y) \mapsto (\xi^{\frac{1}{3}} x, \xi^{\frac{1}{2}} y) = (w^2 x, w^3 y).$$

Algorithm 3.4 D-type sparse-dense Multiplication in $\mathbb{F}_{q^{12}}$ (Cost = $13\tilde{m} + 6\tilde{r} + 44\tilde{a}$)

Input: $a = a_0 + a_1 w + a_2 vw,\ a_0, a_1, a_2 \in \mathbb{F}_{q^2}; b = b_0 + b_1 w,\ b_0, b_1 \in \mathbb{F}_{q^6}$
Output: $ab \in \mathbb{F}_{q^{12}}$
$A_0 \leftarrow a_0 \times b_0[0],\ A_1 \leftarrow a_0 \times b_0[1],\ A_2 \leftarrow a_0 \times b_0[2]$
$A \leftarrow A_0 + A_1 v + A_2 v^2$
$B \leftarrow \text{Fq6SparseMul}(a_1 w + a_2 vw, b_1)$
$c_0 \leftarrow a_0 + a_1, c_1 \leftarrow a_2, c_2 \leftarrow 0$
$c \leftarrow c_0 + c_1 v + c_2 v^2$
$d \leftarrow b_0 + b_1$
$E \leftarrow \text{Fq6SparseMul}(c, d)$
$F \leftarrow E - (A + B)$
$G \leftarrow Bv$
$H \leftarrow A + G$
$c_0 \leftarrow H \bmod p$
$c_1 \leftarrow F \bmod p$
return $c = c_0 + c_1 w$

Algorithm 3.5 Fq6SparseMul (Cost = $5\tilde{m} + 12\tilde{a}$)

Input: $a = a_0 + a_1 v,\ a_0, a_1 \in \mathbb{F}_{q^2}; b = b_0 + b_1 v + b_2 v^2,\ b_0, b_1, b_2 \in \mathbb{F}_{q^2}$
Output: $ab \in \mathbb{F}_{q^6}$
$A \leftarrow a_0 \times b_0,\ B \leftarrow a_1 \times b_1$
$C \leftarrow a_1 \times b_2 \xi$
$D \leftarrow A + C$
$e \leftarrow a_0 + a_1, f \leftarrow b_0 + b_1$
$E \leftarrow e \times f$
$G \leftarrow E - (A + B)$
$H \leftarrow a_0 \times b_2$
$I \leftarrow H + B$
return $D + Gv + Iv^2$

Following the construction of the tower extensions, both w^3 and w^2 are basis elements used to represent an element in $\mathbb{F}_{p^{12}}$. Therefore, the untwisting map is almost free.

The efficient untwisting described above is lost if we use a M-type twist where the untwisting isomorphism is given by:

$$\Psi\colon (x, y) \mapsto (\xi^{-\frac{2}{3}}x, \xi^{-\frac{1}{2}}y) = (\xi^{-1}w^4x, \xi^{-1}w^3y).$$

The cost of the untwisting in this case is 2 multiplications by ξ. However, if we compute the pairing value on the twisted curve instead of the original curve, then we do not need to use the untwisting map. Instead, we require the twisting map which is given by

$$\Psi^{-1}\colon (x, y) \mapsto (w^2x, w^3y).$$

Therefore, in order to make the pairing computation as efficient as possible, we compute the pairing on the original curve E when a D-type twist is involved, and on the twisted curve E' when an M-type twist is involved.

3.3 Final Exponentiation

As discussed earlier, the hard part of the final exponentiation is raising to the exponent $\frac{q^6+1}{n}$. In this section we focus on computing this for BN-curves. We can further split the remaining exponent into two additional parts:

$$\frac{q^6+1}{n} = (q^2+1)\frac{q^4-q^2+1}{n}.$$

Raising to q^2+1 is two applications of the Frobenius operator, which is considered a cheap operation (details to follow). Again, we are left with a hard to compute exponent — $\frac{q^4-q^2+1}{n}$. We outline the fastest way currently known to compute this exponent, described by Fuentes-Castañeda et al. [16].

We observe that if the Tate pairing is raised to some power, then the new function given by $e(P, Q)^m$ also gives a bilinear pairing. This pairing is non-degenerate as long as $n \nmid m$ since $e(P, Q)$ evaluates to an element in μ_n. Hence, instead of using $\frac{q^4-q^2+1}{n}$, we use a multiple of it which still gives a valid pairing.

Recall that for BN-curves, q and n are polynomials in x. Therefore, $\frac{q^4-q^2+1}{n}$ is also a polynomial in x. We call this polynomial $d(x)$. Fuentes-Castañeda et al. [16] showed that

$$\begin{aligned} 2x(6x^2+3x+1)d(x) = 1 &+ 6x + 12x^2 + 12x^3 \\ &+ (4x + 6x^2 + 12x^3)p(x) \\ &+ (6x + 6x^2 + 12x^3)p(x)^2 \\ &+ (-1 + 4x + 6x^2 + 12x^3)p(x)^3. \end{aligned}$$

The above value can be computed as follows. First, the following exponentiations are computed

$$f \mapsto f^x \mapsto f^{2x} \mapsto f^{4x} \mapsto f^{6x} \mapsto f^{6x^2} \mapsto f^{12x^2} \mapsto f^{12x^3}$$

which requires three exponentiations by x, three squarings and one multiplication. Then we compute the terms $a = f^{12x^3} f^{6x^2} f^{6x}$ and $b = a(f^{2x})^{-1}$ which require 3 multiplications. Finally, the final pairing value is obtained as

$$af^{6x^2} fb^p a^{p^2} (bf^-1)^{p^3}$$

which requires 6 multiplications and 6 Frobenius operations. In total, this part of the final exponentiation requires three exponentiations by x, three squarings, ten multiplications, and three Frobenius operations. In comparison, the previous fastest known method requires three additional multiplications and an additional squaring [4].

3.3.1 Exponentiation by x

The final exponentiation requires three exponentiations by x. This is traditionally done using a square-and-multiply method. Before we raise the output of the Miller loop to the power x, we exponentiate it to $(q^6-1)(q^2+1)$. This ensures that the value we need to exponentiate to the power x lies in the cyclotomic subgroup $\mathbb{G}_{\phi_6}(\mathbb{F}_{q^2})$.

Definition 3.3.1. We denote by $\mathbb{G}_{\phi_{12}}(\mathbb{F}_q)$ the cyclotomic subgroup of $\mathbb{F}^*_{q^{12}}$. This is the subgroup of all elements $\alpha \in \mathbb{F}_{q^{12}}$ such that $\alpha^{q^4-q^2+1} = 1$.

For more details on $\mathbb{G}_{\phi_{12}}(\mathbb{F}_q)$, refer to [18]. Now, we have

$$(q^6-1)(q^2+1) = (q^6-1)\frac{(q^6+1)}{q^4-q^2+1} = \frac{q^{12}-1}{q^4-q^2+1}.$$

Thus, an element raised to $(q^6-1)(q^2+1)$ lies in $\mathbb{G}_{\phi_{12}}(\mathbb{F}_q)$. Fast formulas for computing squarings in $\mathbb{G}_{\phi_{12}}(\mathbb{F}_q)$ are given in [4] which we use in our implementation. To compute a square, an element is first compressed, then squared in compressed form, and then decompressed. It is not known how to perform multiplication of compressed elements. Hence, when raising an element to the exponent x, one may keep squaring in compressed form, but when multiplication is required, one needs to decompress the elements. A compressed squaring requires $6\tilde{s}$, $28\tilde{a}$, and $3m_\xi$. A decompression requires $1\tilde{i}, 2\tilde{m}, 3\tilde{s}$, $9\tilde{a}$, and $2m_\xi$. Let h be the Hamming weight of x and l be the bit-length of x. Using Montogomery's simultaneous inversion trick, an exponentiation by x requires l compressed squarings, $l-1$ multiplications in $\mathbb{F}_{q^{12}}$, and $h(3\tilde{m}+3\tilde{s}+9\tilde{a}+2m_\xi)+3(h-1)\tilde{m}+\tilde{i}$ additional operations.

3.4 The Frobenius Operator

Let $a=\alpha+i\beta \in \mathbb{F}_{q^2}$ where $i=\sqrt{-1}$ is an adjoined square root. Then

$$\begin{aligned} a^q &= (\alpha+i\beta)^q \\ &= \alpha^q + i^q\beta^q \\ &= \alpha + i^3\beta && (\text{ since } q \equiv 3 \bmod 4) \\ &= \alpha - i\beta. \end{aligned}$$

Thus, computing a^q requires one base field addition.

Now, suppose $A=\sum_{i=0}^{5} a_i w^i \in \mathbb{F}_{q^{12}}$, with each $a_i \in \mathbb{F}_{q^2}$ and w is defined as in the towering schemes given in subsections 3.1.1 and 3.1.2. By examining the polynomial $q(x)$, we note that $q \equiv 1 \bmod 6$. Then,

$$
\begin{aligned}
A^q &= \left(\sum_{i=0}^{5} a_i w^i\right)^q \\
&= \sum_{i=0}^{5} a_i^q w^{i\cdot q} \\
&= \sum_{i=0}^{5} a_i^q w^{i\cdot(q-1)} \cdot w^i \\
&= \sum_{i=0}^{5} (a_i^q \xi^{i\cdot\frac{q-1}{6}}) \cdot w^i \qquad\qquad (\text{ since } q \equiv 1 \bmod 6).
\end{aligned}
$$

$\xi^0 = 1$, and we precompute $\xi^{i\frac{q-1}{6}}$ for $i = \{1, 2, ..., 5\}$. Hence, applying the Frobenius operator in $\mathbb{F}_{q^{12}}$ costs $5\tilde{m} + 6a$.

Chapter 4

Curve Arithmetic

4.1 Doubling and Addition Formulas

From now on, we refer to the BN-curve defined in [17] over the 254-bit prime field as BN254, the curve defined over the 446-bit prime field as BN446, and so on. Points lying on the twisted curve are manipulated inside the Miller loop. In this section, we present and compare doubling and addition formulas for points on the twisted curve obtained using M-type and D-type twists. In order to speed up the pairing computation, the line functions $l_{2\Psi(T)}(P)$ and $l_{\Psi(T),\Psi(Q)}(P)$, are also computed at the same time as the doubling of point T and the addition of T and Q respectively.

The line computations are optimized to allow us to utilize the special dense-sparse multiplication mentioned in Chapter 3, and to make the twisting/untwisting as simple as possible. In the case of a D-type twist, we compute the pairing on points on the original curve E. Thus, we need to untwist the point T using the map $(x, y) \rightarrow (xw^2, yw^3)$ during the line computation. In the case of an M-type twist, we compute the pairing on points on the twisted curve E'. Hence, we need to twist the point P using the map $(x_P, y_P) \rightarrow (x_P w^2, y_P w^3)$ during the line computation. Making this distinction allows the pairing to be computed optimally regardless of the type of twist involved. All operation counts presented in this chapter are for the curve BN254, but similar results hold for the other curves. Also, to remain consistent with the previous literature, we do not distinguish between single and double-precision operations for doubling and addition costs presented in this chapter.

4.1.1 Point Doubling

Affine Coordinates

Let the point $T = (x, y) \in E'(\mathbb{F}_q)$ be in affine coordinates. To compute $2T = (x_3, y_3)$ and the tangent line, we use the following formulae:

$$\begin{aligned} m &= \frac{3x^2}{2y}, \\ x_3 &= m^2 - 2x, \\ y_3 &= m(x - x_3) - y. \end{aligned}$$

If we are working with a D-type twist then the tangent line evaluated at $P = (x_p, y_p)$ is given by the following equation:

$$l_{2\Psi(T)}(P) = y_p - mx_p w - (y - mx)w^3.$$

To compute the above, we use the following sequence of operations which requires $1\tilde{\imath}$, $3\tilde{m}$, $2\tilde{s}$, $8\tilde{a}$, and $2m$.

$$A = \frac{1}{2y,} \qquad B = 3x^2, \qquad C = AB, \qquad D = 2x, \qquad x_3 = C^2 - D,$$

$$E = C(x - x_3), \qquad y_3 = E - y, \qquad F = Cx_{,}p \qquad G = y - Cx,$$

$$l_{2\Psi(T)}(P) = y_p - Fw - Gw^3.$$

If we are working with an M-type twist then the the tangent line evaluated at $\Psi^{-1}(P) = (x_p w^2, y_p w^3)$ is given by the following equation:

$$l_{2T}(\Psi^{-1}(P)) = y_p w^3 - mx_p w^2 - (y - mx).$$

This can be computed in a manner similar to above, requiring the same sequence of operations.

Jacobian Coordinates

The point T on the twisted curve is traditionally stored and manipulated using Jacobian coordinates. The formula presented here is derived from [4], and is revised to minimize the number of $\mathbb{F}_{q^2}$ squarings. Let $T = (X, Y, Z) \in E'(\mathbb{F}_{q^2})$ be in Jacobian coordinates. Then $2T = (X_3, Y_3, Z_3)$ is given by:

$$\begin{aligned} X_3 &= X(\frac{9}{4}X^3 - 2Y^2), \\ Y_3 &= 3X^3(-Y^2 - \frac{9}{4}X^3) - Y^4, \\ Z_3 &= YZ. \end{aligned}$$

In the case of a D-type twist, the corresponding line function evaluated at $P = (x_P, y_P)$ is given by:

$$l_{2\Psi(T)}(P) = 2Z_3Z^2y_P - 3X^2Z^2x_Pw + (3X^3 - 2Y^2)w^3.$$

Compared to [4], the line evaluation has been multiplied by 2 to save an addition in $\mathbb{F}_{q^2}$. This extra factor is eliminated by the final exponentiation. We use the following sequence of operations to compute the point doubling and line evaluation in $6\tilde{m}$, $4\tilde{s}$, $13\tilde{a}$, and $4m$:

$$A = 3X^2, \quad E = 3X^3, \quad B = \frac{9X^3}{4}, \quad C = Y^2, \quad D = 2Y^2, \quad X_3 = X(B - D),$$

$$F = C^2, \quad Y_3 = E(-C-B), \quad Z_3 = YZ, \quad G = Z^2, \quad H = 2Z_3G, \quad I = -AGx_p, \quad J = E-D,$$

$$l_{2\Psi(T)}(P) = Hy_P + Iw + Jw^3.$$

In the case of an M-type twist the corresponding line computation can be computed using the same sequence of operations as above. The equation for the tangent line computed at $\Psi^{-1}(P) = (x_pw^2, y_pw^3)$ is:

$$l_{2T}(\Psi^{-1}(P)) = 2Z_3Z^2y_Pw^3 - 3X^2Z^2x_Pw^2 + (3X^3 - 2Y^2).$$

Homogeneous Coordinates

Homogeneous coordinates were used in [4] to compute the O-Ate pairing. We present their formulae here. Note that in [4] the O-Ate pairing was computed on the twisted curve E'

obtained through a D-type twist. By computing the pairing on the original curve E, we are able to save a $\mathbb{F}_{q^2}$ addition when doubling a point. Let $T = (X, Y, Z) \in E'(\mathbb{F}_{q^2})$ be in homogeneous coordinates. Then $2T = (X_3, Y_3, Z_3)$ is given by:

$$\begin{aligned} X_3 &= \frac{XY}{2}(y^2 - 9b'Z^2), \\ Y_3 &= \left[\frac{1}{2}(Y^2 + 9'bZ^2)\right]^2 - 27b'^2Z^4, \\ Z_3 &= 2Y^3Z. \end{aligned}$$

In the case of a D-type twist, the corresponding line function evaluated at $P = (x_P, y_P)$ is given by:

$$l_{2\Psi(T)}(P) = -2YZy_P + 3X^2x_Pw + (3b'Z^2 - Y^2)w^3.$$

The above can be computed using the following sequence of operations giving a cost of $3\tilde{m}$, $6\tilde{s}$, $17\tilde{a}$, $4m$, and $1m_b$. In the case of BN254, multiplication by $b' = 1 - i$ can be performed using one addition and the cost is $3\tilde{m}$, $6\tilde{s}$, $18\tilde{a}$, and $4m$.

$$A = \frac{XY}{2}, \quad B = Y^2, \quad C = Z^2, \quad E = 3bC, \quad F = 3E, \quad X_3 = A \cdot (B - F),$$

$$G = \frac{B + f}{2}, \quad Y_3 = G^2 - 3E^2, \quad H = (Y + Z)^2 - (B + C), \quad Z_3 = B \cdot H,$$

$$l_{2\Psi(T)}(P) = -Hy_P + 3X^2x_Pw + (E - B)w^3.$$

The authors in [4] point out that $\tilde{m} - \tilde{s} \approx 3\tilde{a}$. Hence, it is faster to compute XY directly rather than using $(X + Y)^2$, X^2 and Y^2 on a desktop machine. However, in the case of ARM processors, $\tilde{m} - \tilde{s} \approx 6\tilde{a}$. Thus, it is more efficient to use the latter technique when computing pairings on ARM processors, at a cost of $2\tilde{m}$, $7\tilde{s}$, $22\tilde{a}$, and $4m$. In the case of an M-type twist, the corresponding line function evaluated at $\Psi^{-1}(P) = (x_Pw^2, y_Pw^3)$ is given by:

$$l_{2T}(\Psi^{-1}(P)) = -2YZy_Pw^3 + 3X^2x_Pw^2 + (3bZ^2 - Y^2).$$

Again, this can be computed in the same manner as for the D-type twist. As in [4], we also use lazy reduction techniques to optimize the above formulae. Next we discuss point addition on the twisted curve E'.

4.1.2 Point Addition

Affine Coordinates

Let the points $T = (x, y)$ and $Q = (x_2, y_2) \in E'(\mathbb{F}_q)$ be in affine coordinates. To compute $T + Q = (x_3, y_3)$ and the line passing through them, we use the following formulae:

$$\begin{aligned} m &= \frac{x_2 - x}{y_2 - y}, \\ x_3 &= m^2 - x - x_2, \\ y_3 &= m(x - x_3) - y. \end{aligned}$$

If we are working with a D-type twist then the secant line evaluated at $P = (x_p, y_p)$ is given by the following equation:

$$l_{\Psi(T+Q)}(P) = y_p - mx_p w - (y - mx)w^3.$$

To compute the above, we use the following sequence of operations which requires $1\tilde{\imath}$, $3\tilde{m}$, $1\tilde{s}$, $7\tilde{a}$, and $2m$.

$$\begin{gathered} A = \frac{1}{y_2 - y}, \qquad B = x_2 - x, \qquad C = AB, \qquad D = 2x, \qquad x_3 = C^2 - D, \\ E = C(x - x_3), \qquad y_3 = E - y, \qquad F = Cx_p, \qquad G = y - Cx, \\ l_{\Psi(T+Q)}(P) = y_p - Fw - Gw^3. \end{gathered}$$

If we are working with an M-type twist then the secant line evaluated at $\Psi^{-1}(P) = (x_p w^2, y_p w^3)$ is given by the following equation:

$$l_{T+Q}(\Psi^{-1}(P)) = y_p w^3 - mx_p w^2 - (y - mx).$$

This can be computed in a manner similar to above, requiring the same sequence of operations.

Jacobian Coordinates

Let $T = (X, Y, Z)$ and $Q = (X_2, Y_2, 1) \in E'(\mathbb{F}_{q^2})$ be in Jacobian coordinates with $T \neq Q$. Then $T + Q = (X_3, Y_3, Z_3)$ is given by:

$$\begin{aligned} \theta = Y_2 Z^3 - Y, \qquad \lambda = X_2 Z^2 - X, \\ X_3 = \theta^2 - 2X\lambda^2 - \lambda^3, \\ Y_3 = \theta(X\lambda^2 - X_3) - Y\lambda^3, \\ Z_3 = Z\lambda. \end{aligned}$$

In the case of a D-type twist, the corresponding line function evaluated at $P = (x_P, y_P)$ is given by:

$$l_{\Psi(T+Q)}(P) = Z_3 y_P - \theta x_P w + (\theta X_2 - Y_2 Z_3) w^3.$$

In [4], computing the above formula requires $10\tilde{m}$, $3\tilde{s}$, $8\tilde{a}$, and $4m$. We use the following sequence of operations which uses two fewer $\mathbb{F}_{q^2}$ additions.

$$\begin{gathered} A = Z^2, \qquad B = Z^3, \qquad \theta = Y_2 B - Y, \qquad \lambda = X_2 A - X, \qquad C = \theta^2, \\ D = \lambda^2, \qquad E = \lambda^3, \qquad F = C - E, \qquad G = XD, \qquad X_3 = F - 2G, \\ Y_3 = \theta(G - X_3) - YE, \qquad Z_3 = Z\lambda, \qquad J = \theta X_2 - Y_2 Z_3, \\ l_{\Psi(T+Q)}(P) = Z_3 y_P - \theta x_P w + J w^3. \end{gathered}$$

In the case of an M-type twist the corresponding line computation can be computed using the same sequence of operations as above. The equation for the tangent line computed at $\Psi^{-1}(P) = (x_p w^2, y_p w^3)$ is:

$$l_{T+Q}(\Psi^{-1}(P)) = Z_3 y_P w^3 - \theta x_P w^2 + (\theta X_2 - Y_2 Z_3).$$

Homogeneous Coordinates

Let $T = (X, Y, Z)$ and $Q = (X_2, Y_2, 1) \in E'(\mathbb{F}_{q^2})$ be in homogeneous coordinates with $T \neq Q$. Then $T + Q = (X_3, Y_3, Z_3)$ is given by:

$$\begin{aligned} \theta = Y - Y_2 Z, \qquad \lambda = X - X_2 Z, \\ X_3 = \lambda(\lambda^3 + Z\theta^2 - 2X\lambda^2), \\ Y_3 = \theta(3X\lambda^2 - \lambda^3 - Z\theta^2) - Y\lambda^3, \\ Z_3 = Z\lambda^3. \end{aligned}$$

In the case of a D-type twist, the corresponding line function evaluated at $P = (x_P, y_P)$ is given by:

$$l_{\Psi(T+Q)}(P) = -\lambda y_P - \theta x_P w + (\theta X_2 - \lambda Y_2)w^3.$$

In [4] computing the above formula requires $11\tilde{m}$, $2\tilde{s}$, $12\tilde{a}$, and $4m$. We use the following sequence of operations which uses 4 fewer $\mathbb{F}_{q^2}$ additions.

$$\begin{gathered} A = Y_2 Z, \quad B = X_2 Z, \quad \theta = Y - A, \quad \lambda = X - B, \quad C = \theta^2, \\ D = \lambda^2, \quad E = \lambda^3, \quad F = ZC, \quad G = XD, \quad H = E + F - 2G, \\ X_3 = \lambda H, \quad I = YE, \quad Y_3 = \theta(G - H) - I, \quad Z_3 = ZE, \quad J = \theta X_2 - \lambda Y_2, \\ l_{\Psi(T+Q)}(P) = -\lambda y_P - \theta x_P w + Jw^3. \end{gathered}$$

In the case of an M-type twist the corresponding line computation can be computed using the same sequence of operations as above. The equation for the tangent line computed at $\Psi^{-1}(P) = (x_p w^2, y_p w^3)$ is:

$$l_{T+Q}\Psi^{-1}(P) = -\lambda y_P - \theta x_P w + Jw^3.$$

Table 4.1 lists the operation counts for each of the above operations for easier comparison. Between Jacobian and homogeneous coordinates, homogeneous coordinates are the faster choice for the doubling operation. Jacobian coordinates are faster for addition, but the gain is not sufficient to warrant the use of Jacobian coordinates over homogeneous coordinates for pairing computation.

Coordinates	Doubling Cost	Addition Cost
Affine	$\tilde{\imath} + 3\tilde{m} + 2\tilde{s} + 8\tilde{a} + 2m$	$\tilde{\imath} + 3\tilde{m} + 1\tilde{s} + 7\tilde{a} + 2m$
Jacobian	$6\tilde{m} + 4\tilde{s} + 13\tilde{a} + 4m$	$10\tilde{m} + 3\tilde{s} + 6\tilde{a} + 4m$
Homogeneous	$2\tilde{m} + 7\tilde{s} + 22\tilde{a} + 4m$	$11\tilde{m} + 2\tilde{s} + 8\tilde{a} + 4m$

Table 4.1: Doubling and Addition costs comparison on ARM architecture

4.1.3 Affine Coordinates on ARM

Acar et al. [3] argue that on ARM processors, the inversion to multiplication ratios are small enough that it is more efficient to compute a pairing using affine coordinates. Refering to addition and doubling costs above, we see that if inversions are cheap enough, then the addition and doubling steps using affine coordinates will be faster. Using affine coordinates also leads to a faster dense-sparse multiplication algorithm. Tables 5.1 and 5.2 list the cost of the addition and doubling steps and dense-sparse multiplication taking into account the distinction between double and single precision operations. Using this, we compute the crossover point for the inversion-to-multiplication ratio at which a pairing computation becomes faster using affine coordinates.

If we are using a prime congruent to 3 mod 8, then compared to a projective doubling, an affine doubling costs an extra $\mathbb{F}_{q^2}$ inversion and unreduced multiplication, and saves $5\tilde{s}_u + 3\tilde{r} + 16.5\tilde{a} + 2m$. Compared to a first doubling, it costs an extra $\mathbb{F}_{q^2}$ inversion and saves $2\tilde{s}_u + 2\tilde{r} + 6.5\tilde{a} + 2m$. An addition costs an extra $\mathbb{F}_{q^2}$ inversion and saves $8\tilde{m}_u + \tilde{s}_u + 7\tilde{r} + 3\tilde{a} + 2m$, and a dense-sparse multiplication needs 6 additional base field multiplications and saves $3\tilde{m}_u + 3\tilde{r} + 0.5\tilde{a}$. Computing a pairing on BN254 requires 1 first doubling, 63 doublings, 6 additions, and 66 dense-sparse multiplications. Thus, the difference between an affine and projective pairing is:

$$\begin{aligned}
254A - 254P &= (\tilde{\imath} - 2\tilde{s}_u - 2\tilde{r} - 6.5\tilde{a} - 2m) + 63(\tilde{\imath} + \tilde{m}_u - 5\tilde{s}_u - 3\tilde{r} - 16.5\tilde{a} - 2m) + \\
&\quad 6(\tilde{\imath} - 8\tilde{m}_u - \tilde{s}_u - 7\tilde{r} - 3\tilde{a} - 2m) + 66(6m - 3\tilde{m}_u - 3\tilde{r} - 0.5\tilde{a}) \\
&= 70\tilde{\imath} + 256m - 183\tilde{m}_u - 323\tilde{s}_u - 431\tilde{r} - 1097\tilde{a} \\
&= 70(i + 4m_u + 3r + 3a) + 256(m_u + r) - 183(3m_u + 8a) - \\
&\quad 323(2m_u + 3a) - 431(2r) - 1097(2a) \\
&= 70i - 659m_u - 396r - 4417a.
\end{aligned}$$

On an ARM Feroceon, where we measure $m \approx 2r$ and $15a \approx m$, this gives a crossover inversion-to-multiplication(I/M) ratio of 11.5. On an iPad, we found that $m \approx 1.5r$ which

yields a crossover I/M ratio of 10.8. This means that if the I/M ratio is below 10.8, using affine coordinates will be faster for pairing computation. Note that the equivalent crossover I/M ratios in $\mathbb{F}_{q^2}$ are half as small as the ratios in the base field. In general, actual and crossover I/M ratios getter smaller as the degree of the extension field increases because of the inversion formula for higher extensions. As a result, as the degree of the extension field used for the bulk of the arithmetic in a pairing computation increases, the case for using affine coordinates gets stronger.

Similarly, we computed the crossover I/M ratios for BN446 and BN638 – details are in Appendix B and the crossover ratios are given in Table 4.2. The crossover ratios over different size finite fields (where the binary or NAF representation of $6x + 2$ is sparse), seem to be very close to each other.

Table 4.3 lists the observed inversion-to-multiplication(I/M) ratios in various fields across different platforms for our work and [3]. In general, the ratios get smaller as the size of the finite field increases. We can see that according to these ratios, projective coordinates are faster than affine on x86-64. On the other hand, the choice for ARM processors is not so clear cut. For BN256, the actual I/M ratios are fairly close to the crossover points. For BN446 and BN638, the actual ratios are below the crossover point. As a result, affine coordinates should in theory be a better choice for our implementation. Further optimizations such as hand-optimizing the low-level field arithmetic operations in assembly language may affect the I/M ratios. However, the I/M ratios in the 446-bit and 638-bit prime fields are low enough that we can confidently say that affine coordinates will be the better choice in these fields.

4.2 NAF Miller Loop

Instead of using the regular binary representation of a number, one may choose to represent it using a non-adjacent form (NAF). A NAF is a unique signed digit representation using digits $\{1, 0, -1\}$ with the property that no two consecutive digits are nonzero. For example, the decimal digit 7 can be expressed as $1, 0, 0, -1 = 2^3 - 1$ using NAF. The advantage of using NAF is that it minimizes the Hamming weight of an integer. On average, one-third of all digits are non-zero when using NAF representation.

To reduce the number of addition steps in the Miller loop, we can use a NAF repre-

[1] ARM Feroceon 88FR131 @ 1.2GHz

Curve	**Cross-over I/M ratio**	
	Feroceon[1]	iPad
ratios in $\mathbb{F}_q$		
BN254	11.5	10.8
BN446	11.0	10.2
BN638	11.0	10.2
ratios in $\mathbb{F}_{q^2}$		
BN254	4.5	4.1
BN446	4.2	3.8
BN638	4.3	3.8

Table 4.2: Cross-over I/M ratios for various curves

Curve	**x86-64**		**ARM**		
	Acar et al.	our work	Acar et al.	Feroceon	iPad
BN254	25	30	10.67	10.2	10.5
BN446	22.33	28.1	8.94	8.9	9.2
BN638	18.59	24.5	6.82	7.7	7.9

Table 4.3: Actual $\mathbb{F}_q$ inversion-to-multiplication ratios in various fields

sentation of u and execute a double, add and subtract version of the algorithm as is done traditionally for elliptic curve scalar multiplication. Taking the inverse of a point on an elliptic curve is almost free, hence the subtraction step in the Miller loop costs almost the same as the addition step. In our implementation, the NAF version of Miller's algorithm reduces the number of addition steps when computing a pairing on BN638 from 88 to 6.

Chapter 5

Implementation Results

In this chapter, we present detailed operation counts for the number of operations required for a pairing computation described in previous sections. Counts are presented for the curves BN254, BN446, and BN638 which are also used in Acar et al. [3]. We also present timings on several platforms for our implementation of the pairing computation at various security levels.

5.1 Operation Counts

5.1.1 Miller Loop Operation Count

Table 5.1 shows the operation count for all the computations in a pairing for the curve using the 254-bit prime field, and Table 5.2 gives the operation count for the curve using the 446-bit prime field.

For BN256, using the techniques described in previous chapters, the Miller loop executes one negation in $\mathbb{F}_q$ to precompute $\overline{y_p}$, one first doubling with line evaluation, 63 point doublings with line evaluations, 6 point additions with line evaluations, one p-power Frobenius in $E'(\mathbb{F}_{p^2})$, one p^2-power Frobenius in $E'(\mathbb{F}_{p^2})$, 66 sparse multiplications, 63 squarings in $\mathbb{F}_{p^2}$, one negation in $E'(\mathbb{F}_{p^2})$, two sparser multiplications, and one multiplication in $\mathbb{F}_{p^{12}}$.

$E'(\mathbb{F}_{p^2})$ - **Arithmetic**	**Operation Count**
Doubling/Eval. (x86=64)	$3\tilde{m}_u + 6\tilde{s}_u + 8\tilde{r} + 21\tilde{a} + 4m$
Doubling/Eval. (ARM)	$2\tilde{m}_u + 7\tilde{s}_u + 8\tilde{r} + 25\tilde{a} + 4m$
Doubling/Eval. (Affine)	$\tilde{\imath} + 3\tilde{m}_u + 2\tilde{s}_u + 5\tilde{r} + 8\tilde{a} + a + 2m$
Addition/Eval.	$11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 11\tilde{a} + 4m$
Addition/Eval. (Affine)	$\tilde{\imath} + 3\tilde{m}_u + \tilde{s}_u + 4\tilde{r} + 8\tilde{a} + 2m$
First Doubling/Eval	$3\tilde{m}_u + 4\tilde{s}_u + 7\tilde{r} + 14\tilde{a} + 4m$
p-power Frobenius	$2\tilde{m} + 2a$
p^2- power Frobenius	$4m$
Negation	$\tilde{a}$
$\mathbb{F}_{p^2}$ - **Arithmetic**	**Operation Count**
Add./Sub./Neg.	$\tilde{a} = 2a$
Conjugation	a
Multiplication	$\tilde{m} = \tilde{m}_u + \tilde{r} = 3m_u + 2r + 8a$
Squaring	$\tilde{s} = \tilde{s}_u + \tilde{r} = 2m_u + 2r + 3a$
Multiplication by i	a
Multiplication by ξ	$2a$
Inversion	$\tilde{\imath} = i + 4m + 3r + 3a$
$\mathbb{F}_{p^{12}}$ - **Arithmetic**	**Operation Count**
Add./Sub	$6\tilde{a}$
Conjugation	$3\tilde{a}$
Multiplication	$18\tilde{m}_u + 110\tilde{a} + 6\tilde{r}$
Sparse Multiplication	$13\tilde{m}_u + 6\tilde{r} + 48\tilde{a}$
Sparser Multiplication	$6\tilde{m}_u + 6\tilde{r} + 13\tilde{a}$
Affine Sparse Multiplication	$10\tilde{m}_u + 6\tilde{r} + 47\tilde{a} + 6m_u + a$
Squaring	$12\tilde{m}_u + 6\tilde{r} + 73\tilde{a}$
Cyclotomic Squaring	$9\tilde{s}_u + 46\tilde{a} + 6\tilde{r}$
Compressed Squaring	$6\tilde{s}_u + 31\tilde{a} + 4\tilde{r}$
Simultaneous Decompression	$9\tilde{m} + 6\tilde{s} + 22\tilde{a} + \tilde{\imath}$
p-power Frobenius	$5\tilde{m} + 6a$
p^2-power Frobenius	$10m + 2\tilde{a}$
Exponentiation by x	$45\tilde{m}_u + 378\tilde{s}_u + 275\tilde{r} + 2164\tilde{a} + \tilde{\imath}$
Inversion	$25\tilde{m}_u + 9\tilde{s}_u + 18\tilde{r} + 123\tilde{a} + \tilde{\imath}$

Table 5.1: Operation counts for a 254-bit prime field , refer to Section 3.2.1 for notation.

$E'(\mathbb{F}_{p^2})$ - **Arithmetic**	**Operation Count**
Doubling/Eval. (x86-64)	$3\tilde{m}_u + 6\tilde{s}_u + 8\tilde{r} + 30\tilde{a} + a + 4m$
Doubling/Eval. (ARM)	$2\tilde{m}_u + 7\tilde{s}_u + 8\tilde{r} + 34\tilde{a} + a + 4m$
Doubling/Eval. (Affine)	$\tilde{\imath} + 3\tilde{m}_u + 2\tilde{s}_u + 5\tilde{r} + 8\tilde{a} + a + 2m$
Addition/Eval.	$11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 11\tilde{a} + 4m$
First Doubling/Eval	$3\tilde{m}_u + 4\tilde{s}_u + 7\tilde{r} + 23\tilde{a} + a + 4m$
Addition/Eval. (Affine)	$\tilde{\imath} + 3\tilde{m}_u + \tilde{s}_u + 4\tilde{r} + 8\tilde{a} + 2m$
p-power Frobenius	$8\tilde{m} + 2a$
Negation	$\tilde{a}$
$\mathbb{F}_{p^2}$ - **Arithmetic**	**Operation Count**
Add./Sub./Neg.	$\tilde{a} = 2a$
Conjugation	a
Multiplication	$\tilde{m} = \tilde{m}_u + \tilde{r} = 3m_u + 2r + 10a$
Squaring	$\tilde{s} = \tilde{s}_u + \tilde{r} = 2m_u + 2r + 5a$
Multiplication by i	$2a$
Multiplication by ξ	$3a$
Inversion	$\tilde{\imath} = i + 4m + 3r + 5a$
$\mathbb{F}_{p^{12}}$ - **Arithmetic**	**Operation Count**
Add./Sub	$6\tilde{a}$
Conjugation	$3\tilde{a}$
Multiplication	$18\tilde{m}_u + 117\tilde{a} + 6\tilde{r}$
Sparse Multiplication	$13\tilde{m}_u + 6\tilde{r} + 54\tilde{a}$
Sparser Multiplication	$6\tilde{m}_u + 6\tilde{r} + 14\tilde{a}$
Affine Sparse Multiplication	$10\tilde{m}_u + 6\tilde{r} + 53\tilde{a} + 6m_u + a$
Squaring	$12\tilde{m}_u + 6\tilde{r} + 78\tilde{a}$
Cyclotomic Squaring	$9\tilde{s}_u + 49\tilde{a} + a + 6\tilde{r}$
Compressed Squaring	$6\tilde{s}_u + 33\tilde{a} + a + 4\tilde{r}$
Simultaneous Decompression	$9\tilde{m} + 6\tilde{s} + 24\tilde{a} + \tilde{\imath}$
p-power Frobenius	$5\tilde{m} + 6a$
p^2-power Frobenius	$10m + 2\tilde{a}$
Exponentiation by x	$45\tilde{m}_u + 666\tilde{s}_u + 467\tilde{r}_u + 3888\tilde{a} + 110a + \tilde{\imath}$
Inversion	$25\tilde{m}_u + 9\tilde{s}_u + 18\tilde{r} + 138\tilde{a} + \tilde{\imath}$

Table 5.2: Operation counts for a 446-bit prime field, refer to Section 3.2.1 for notation.

Thus, the cost of the Miller loop on an ARM processor using projective coordinates is:

$$\begin{aligned}\text{ML256P} &= a + 3\tilde{m}_u + 4\tilde{s}_u + 7\tilde{r} + 14\tilde{a} + 4m + 63(2\tilde{m}_u + 7\tilde{s}_u + 8\tilde{r} + 25\tilde{a} + 4m) + \\ &\quad 6(11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 11\tilde{a} + 4m) + 2\tilde{m} + 2a + 4m + 66(13\tilde{m}_u + 6\tilde{r} + 48\tilde{a}) + \\ &\quad 63(12\tilde{m}_u + 6\tilde{r} + 73\tilde{a}) + \tilde{a} + 2(6\tilde{m}_u + 6\tilde{r} + 13\tilde{a}) + 18\tilde{m}_u + 110\tilde{a} + 6\tilde{r} \\ &= 1841\tilde{m}_u + 457\tilde{s}_u + 1371\tilde{r} + 9522\tilde{a} + 284m + 3a.\end{aligned}$$

For BN446, the Miller loop executes one negation in $\mathbb{F}_q$ to precompute $\overline{y_p}$, one first doubling with line evaluation, 111 point doublings with line evaluations, 6 point additions with line evaluations, two p-power Frobenius operations in $E'(\mathbb{F}_{p^2})$, 114 sparse multiplications, 111 squarings in $\mathbb{F}_{p^2}$, one negation in $E'(\mathbb{F}_{p^2})$, two sparser multiplications, and one multiplication in $\mathbb{F}_{p^{12}}$. Thus, the cost of the Miller loop on an ARM processor using projective coordinates is:

$$\begin{aligned}\text{ML446P} &= a + 3\tilde{m}_u + 4\tilde{s}_u + 7\tilde{r} + 23\tilde{a} + a + 4m + 111(2\tilde{m}_u + 7\tilde{s}_u + 8\tilde{r} + 34\tilde{a} + a + 4m) \\ &\quad + 6(11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 11\tilde{a} + 4m) + 2(8\tilde{m} + 2a) + 114(13\tilde{m}_u + 6\tilde{r} + 54\tilde{a}) \\ &\quad + 111(12\tilde{m}_u + 6\tilde{r} + 78\tilde{a}) + \tilde{a} + 2(6\tilde{m}_u + 6\tilde{r} + 14\tilde{a}) + 18\tilde{m}_u + 117\tilde{a} + 6\tilde{r} \\ &= 3151\tilde{m}_u + 793\tilde{s}_u + 2345\tilde{r} + 18601\tilde{a} + 472m + 117a.\end{aligned}$$

Lastly, for BN638, we use the NAF Miller algorithm. Hence, the Miller loop executes one negation in $\mathbb{F}_q$ to precompute $\overline{y_p}$, one first doubling with line evaluation, 160 point doublings with line evaluations, 8 point additions with line evaluations, two p-power Frobenius operations in $E'(\mathbb{F}_{p^2})$, 167 sparse multiplications, 160 squarings in $\mathbb{F}_{p^2}$, two negations in $E'(\mathbb{F}_{p^2})$, two sparser multiplications, and one multiplication in $\mathbb{F}_{p^{12}}$. Thus, the cost of the Miller loop is:

$$\begin{aligned}\text{ML638P} &= a + 3\tilde{m}_u + 4\tilde{s}_u + 7\tilde{r} + 23\tilde{a} + a + 4m + 160(2\tilde{m}_u + 7\tilde{s}_u + 8\tilde{r} + 34\tilde{a} + a + 4m) \\ &\quad + 8(11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 11\tilde{a} + 4m) + 2(8\tilde{m} + 2a) + 167(13\tilde{m}_u + 6\tilde{r} + 54\tilde{a}) \\ &\quad + 160(12\tilde{m}_u + 6\tilde{r} + 78\tilde{a}) + 2\tilde{a} + 2(6\tilde{m}_u + 6\tilde{r} + 14\tilde{a}) + 18\tilde{m}_u + 117\tilde{a} + 6\tilde{r} \\ &= 4548\tilde{m}_u + 1140\tilde{s}_u + 3557\tilde{r} + 27206\tilde{a} + 676m + 166a.\end{aligned}$$

Similarly, we can compute the cost of the Miller loop on x86-64 using projective coordinates, and also using affine coordinates. Table 5.3 summarizes these costs for all three curves.

5.1.2 Final Exponentiation

We now count the number of operations required for the final exponentiation part of the pairing computation, for each of the three curves.

BN254	
Projective Coordinates on ARM	$1841\tilde{m}_u + 457\tilde{s}_u + 1371\tilde{r} + 9522\tilde{a} + 284m + 3a$
Projective Coordinates on x86-64	$1904\tilde{m}_u + 394\tilde{s}_u + 1371\tilde{r} + 9306\tilde{a} + 284m + 3a$
Affine Coordinates	$70\tilde{\imath} + 1658\tilde{m}_u + 134\tilde{s}_u + 942\tilde{r} + 8398\tilde{a} + 540m + 133a$
BN446	
Projective Coordinates on ARM	$3151\tilde{m}_u + 793\tilde{s}_u + 2345\tilde{r} + 18601\tilde{a} + 472m + 117a$
Projective Coordinates on x86-64	$3262\tilde{m}_u + 682\tilde{s}_u + 2345\tilde{r} + 18268\tilde{a} + 472m + 117a$
Affine Coordinates	$118\tilde{\imath} + 2872\tilde{m}_u + 230\tilde{s}_u + 1610\tilde{r} + 15790\tilde{a} + 920m + 231a$
BN638	
Projective Coordinates on ARM	$4548\tilde{m}_u + 1140\tilde{s}_u + 3557\tilde{r} + 27206\tilde{a} + 676m + 166a$
Projective Coordinates on x86-64	$4708\tilde{m}_u + 980\tilde{s}_u + 3557\tilde{r} + 26556\tilde{a} + 676m + 166a$
Affine Coordinates	$169\tilde{\imath} + 4143\tilde{m}_u + 330\tilde{s}_u + 2324\tilde{r} + 22830\tilde{a} + 1340m + 333a$

Table 5.3: Cost of the Miller Loop using various coordinates/processors, refer to Section 3.2.1 for notation.

In the case of BN256, since we are working with a negative r, the final exponentiation requires 6 conjugations in $\mathbb{F}_{p^{12}}$, one negation in $E'(\mathbb{F}_{p^2})$, one inversion in $\mathbb{F}_{p^{12}}$, 12 multiplications in $\mathbb{F}_{p^{12}}$, two p-power Frobenius operations in $\mathbb{F}_{p^{12}}$, three p^2-power Frobenius operations in $\mathbb{F}_{p^{12}}$, three exponentiations by x, and three cyclotomic squarings. Hence, the total cost of the final exponentiation is:

$$\begin{aligned}\text{FE256} &= 6(3\tilde{a}) + \tilde{a} + 25\tilde{m}_u + 9\tilde{s}_u + 18\tilde{r} + 123\tilde{a} + \tilde{\imath} + 12(18\tilde{m}_u + 110\tilde{a} + 6\tilde{r}) \\ &\quad + 2(5\tilde{m} + 6a) + 3(10m + 2\tilde{a}) + 3(45\tilde{m}_u + 378\tilde{s}_u + 275\tilde{r} + 2164\tilde{a} + \tilde{\imath}) \\ &\quad + 3(9\tilde{s}_u + 49\tilde{a} + a + 6\tilde{r}) \\ &= 386\tilde{m}_u + 1164\tilde{s}_u + 943\tilde{r} + 4\tilde{\imath} + 7989\tilde{a} + 30m + 15a.\end{aligned}$$

In the case of BN446, the final exponentiation requires five conjugations in $\mathbb{F}_{p^{12}}$, one inversion in $\mathbb{F}_{p^{12}}$, twelve multiplications in $\mathbb{F}_{p^{12}}$, two p-power Frobenius operations in $\mathbb{F}_{p^{12}}$, three p^2-power Frobenius operations in $\mathbb{F}_{p^{12}}$, three exponentiations by x, and three cyclotomic squarings. Hence, the total cost of the final exponentiation is:

$$\begin{aligned}\text{FE446} &= 5(3\tilde{a}) + 25\tilde{m}_u + 9\tilde{s}_u + 18\tilde{r} + 138\tilde{a} + \tilde{\imath} + 12(18\tilde{m}_u + 117\tilde{a} + 6\tilde{r}) \\ &\quad + 2(5\tilde{m} + 6a) + 3(10m + 2\tilde{a}) + 3(45\tilde{m}_u + 666\tilde{s}_u + 467\tilde{r} + 3888\tilde{a} + 110a + \tilde{\imath}) \\ &\quad + 3(9\tilde{s}_u + 49\tilde{a} + a + 6\tilde{r}) \\ &= 386\tilde{m}_u + 2034\tilde{s}_u + 1519\tilde{r} + 4\tilde{\imath} + 13374\tilde{a} + 30m + 345a.\end{aligned}$$

Curve		Cost
BN256	ML256P + FE256	$2227\tilde{m}_u + 1621\tilde{s}_u + 2314\tilde{r} + 4\tilde{i} + 17484\tilde{a} + 314m + 18a$
BN446	ML446P + FE446	$3537\tilde{m}_u + 2827\tilde{s}_u + 3872\tilde{r} + 4\tilde{i} + 31975\tilde{a} + 506m + 463a$
BN638	ML638P + FE638	$4984\tilde{m}_u + 4020\tilde{s}_u + 5700\tilde{r} + 4\tilde{i} + 45734\tilde{a} + 606m + 655a$

Table 5.4: Operation count for pairing computations at various security levels

In the case of BN638, the high level operations are the same as the previous case. The Hamming weight of x is 4, and hence simultaneous decompression requires $16\tilde{m}+9\tilde{s}+35\tilde{a}+\tilde{i}$. Thus exponentiation by x requires $70\tilde{m} + 948\tilde{s} + 675\tilde{r} + 5606\tilde{a} + 158a + \tilde{i}$. As a result, the total cost of the final exponentiation is:

$$\begin{aligned}\text{FE638} &= 5(3\tilde{a}) + 25\tilde{m}_u + 9\tilde{s}_u + 18\tilde{r} + 138\tilde{a} + \tilde{i} + 12(18\tilde{m}_u + 117\tilde{a} + 6\tilde{r}) \\ &\quad + 2(5\tilde{m} + 6a) + 3(10m + 2\tilde{a}) + 3(70\tilde{m} + 948\tilde{s} + 675\tilde{r} + 5606\tilde{a} + 158a + \tilde{i}) \\ &\quad + 3(9\tilde{s}_u + 49\tilde{a} + a + 6\tilde{r}) \\ &= 436\tilde{m}_u + 2880\tilde{s}_u + 2143\tilde{r} + 4\tilde{i} + 18528\tilde{a} + 30m + 489a.\end{aligned}$$

Table 5.4 gives the total cost of the pairing computation on each of the three curves using projective coordinates on the ARM processor. The cost for our work is computed using the results in Table 5.4 and expressing them in terms of base field arithmetic operations. The cost for Arahna et al. [4] is computed with the same method using the results given in Section 6 in their work.

Table 5.5 gives a comparison of the operation count between our implementation and the previous best known implementation of the Optimal Ate pairing [4] using the curve BN256 on the x86-64 platform. Our implementation improves the work in [4] by the lowering the cost of the pairing computation by $249m + 22r + 2875a$.

5.2 Implementation Times

We now present timings of our C implementation of the pairing computation on various platforms. We used the same C code on all platforms. As stated previously, the base field

Work	Phase	Pairing Cost
Arahna et al. [4]	ML	$6796m + 2736r + 20436a$
	FE	$3753m + 1926r + 17025a$
	Pairing	$10549m + 4662r + 37461a$
Our Implementation	ML	$6784m + 2742r + 18615a$
	FE	$3516m + 1882r + 15997a$
	Pairing	$10300m + 4624r + 34612a$

Table 5.5: Comparison of Operation count for pairing computation on the BN254 curve.

arithmetic largely follows that of [4], using the gmp library for low level routines. Tables 5.6, 5.7, and 5.8 compare our implementation using the curves BN254, BN446, and BN638 respectively with related work. In the work of Arahna et al. [4], similar numbers are also provided for the Core i7, Opteron, and the Core 2 Duo. In our comparison we use their results for the Phenom II because we feel it is closest to our Athlon II processor. The times presented for the Miller Loop are using the coordinate system which is the fastest available for that work.

As expected, projective pairings are faster than affine pairings using our code on x86-64 but not on ARM processors. Comparing [3] to our iPad numbers, which were obtained on the same microarchitecture and clock speed, we find that our implementation is 73%, 73%, and 82% faster on BN254, BN446, and BN638 respectively. The largest improvement is in the case of the curve BN638, which is attributed to the implementation of the NAF-Miller algorithm. For BN638, using the NAF version of the Miller algorithm saved 82 curve additions and line computations.

Although we have a lower operation count, our O-Ate pairing on BN254 is slower than that of [4] on x86-64, since we did not implement $\mathbb{F}_{p^2}$ arithmetic in hand-optimized assembly.

Work/ Platform	**Operation**								
	$\tilde{a}$	$\tilde{m}$	$\tilde{s}$	$\tilde{i}$	$\tilde{r}$	ML	FE	O-Ate(a)	O-Ate(p)
our work, ARM Feroceon 88FR131 @ 1.2 GHz, μs	.35	4.96	4.01	24.01	2.1	11,877	7,550	19,427	19,880
Acar et al., nVidia Tegra 2 Cortex A9 @ 1GHz, μs	1.42	8.18	5.20	26.61	n/a	26,320	24,690	51,010	55,190
our work, Apple iPad 2 Cortex A9 @ 1GHz, μs	.25	3.48	2.88	19.19	1.8	8,339	5,483	14,605	13,822
our work, x86-64 AMD Athlon II @ 2.1GHz, cycles	61	699	640	7,104	n/a	2,367	1,498,159	4,220,661	3,865,529
Acar et al., x86-64 Core2 E6600 @ 2.4 GHz, cycles	336	2,131	1,318	12,774	n/a	7,491,593	6,633,846	14,125,439	14,989,039
Aranha et al., x86-64 Phenom II, cycles	n/a	368	288	n/a	n/a	898,000	664,000	n/a	1,562,000

Table 5.6: Field arithmetic timings in a 254-bit prime field, and O-Ate pairing timings on BN254.

Work/ Platform	**Operation**								
	$\tilde{a}$	$\tilde{m}$	$\tilde{s}$	$\tilde{i}$	$\tilde{r}$	ML	FE	O-Ate(a)	O-Ate(p)
our work, ARM Feroceon 88FR131 @ 1.2 GHz, μs	0.38	10.8	8.6	47.9	4.5	41,723	23,089	64,812	65,958
Acar el al., nVidia Tegra 2 Cortex A9 @ 1GHz, μs	2.37	17.24	10.84	54.23	n/a	97,530	86,750	184,280	195,560
our work, Apple iPad 2 Cortex A9 @ 1GHz, μs	.26	8.03	6.46	37.95	3.2	32,087	17,181	49,652	49,267
our work, x86-64 AMD Athlon II @ 2.1GHz, cycles	91	1,415	1,259	14,728	n/a	6,641,288	3,927,155	11,896,956	10,568,443
Acar et al., x86-64 Core2 E6600 @ 2.4 GHz, cycles	493	3,821	2,445	22,957	n/a	23,995,542	20,156,765	44,152,307	45,515,035

Table 5.7: Field arithmetic timings in a 446-bit prime field, and O-Ate pairing timings on BN446.

Work/ Platform	**Operation**								
	$\tilde{a}$	$\tilde{m}$	$\tilde{s}$	$\tilde{i}$	$\tilde{r}$	ML	FE	O-Ate(a)	O-Ate(p)
our work, ARM Feroceon 88FR131 @ 1.2 GHz, μs	.51	18.23	14.93	77.11	7.6	98,044	51,351	149,395	153,713
Acar el al., nVidia Tegra 2 Cortex A9 @ 1GHz, μs	3.48	31.81	20.55	91.92	n/a	236,480	413,370	649,850	768,060
our work, Apple iPad 2 Cortex A9 @ 1GHz, μs	.34	15.07	12.09	64.68	6	79,056	40,572	119,628	123,410
our work, x86-64 AMD Athlon II @ 2.1GHz, cycles	109	2,350	1,997	22,266	n/a	14,811,009	8,169,431	25,998,299	22,980,440
Acar et al., x86-64 Core2 E6600 @ 2.4 GHz, cycles	659	6,176	3,961	34,935	n/a	51,497,159	85,037,269	136,534,428	157,309,156

Table 5.8: Field arithmetic timings in a 638-bit prime field, and O-Ate pairing timings on BN638.

References

[1] IEEE P1363.3: Standard for identity-based cryptographic techniques using pairings. draft 3:section 5.3.2. `http://grouper.ieee.org/groups/1363/IBC/index.html`.

[2] Michel Abdalla and Paulo 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.

[3] Tolga Acar, Kristin Lauter, Michael Naehrig, and Daniel Shumow. Affine pairings on arm. *IACR Cryptology ePrint Archive*, 2011:243, 2011.

[4] Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, and Julio López. Faster explicit formulas for computing pairings over ordinary curves. In Paterson [35], pages 48–68.

[5] R. Balasubramanian and Neal 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.

[6] Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In Yung [45], pages 354–368.

[7] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Preneel and Tavares [36], pages 319–331.

[8] Naomi Benger and Michael Scott. Constructing tower extensions of finite fields for implementation of pairing-based cryptography. In Hasan and Helleseth [20], pages 180–195.

[9] Jean-Luc Beuchat, Jorge Enrique González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-speed software implementation of the optimal ate pairing over Barreto-Naehrig curves. In Joye et al. [23], pages 21–39.

[10] I.F. Blake, G. Seroussi, and N.P. Smart. *Advances in Elliptic Curve Cryptography*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2005.

[11] G. R. Blakley and David Chaum, editors. *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*. Springer, 1985.

[12] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[13] Wieb Bosma, editor. *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*, volume 1838 of *Lecture Notes in Computer Science*. Springer, 2000.

[14] Canalys. Smart phones overtake client PCs in 2011. Press Release, February 2011. `http://www.canalys.com/static/press_release/2012/canalys-press-release-030212-smart-phones-overtake-client-pcs-2011_0.pdf`.

[15] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–280, 2010.

[16] Laura Fuentes-Castañeda, Edward Knapp, and Francisco Rodríguez-Henríquez. Faster hashing to G_2. In Miri and Vaudenay [31], pages 412–430.

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

[18] Robert Granger and Michael Scott. Faster squaring in the cyclotomic subgroup of sixth degree extensions. [34], pages 209–223.

[19] D. Hankerson, A. Menezes, and M. Scott. Software implementation of pairings. In M. Joye and G. Neven, editors, *Identity-Based Cryptography*, volume 2, pages 188 –206. IOS Press, 2008.

[20] M. Anwar Hasan and Tor 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.

[21] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The Eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

[22] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In Bosma [13], pages 385–394.

[23] Marc Joye, Atsuko Miyaji, and Akira 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.

[24] Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In Smart [43], pages 13–36.

[25] Kristin Lauter, Peter L. Montgomery, and Michael Naehrig. An analysis of affine coordinates for pairing computation. In Joye et al. [23], pages 1–20.

[26] F. Lemmermeyer. *Reciprocity laws: from Euler to Eisenstein*. Springer monographs in mathematics. Springer, 2000.

[27] F. Luca, D. M. Morales, and I. Shparlinski. MOV attack in various subgroups on elliptic curves. *Illinois Journal of Mathematics*, 48(3):1041–1052, 2004.

[28] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

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

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

[31] Ali Miri and Serge 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.

[32] R Moraru. Algebraic curves. University Course, 2010.

[33] Michael Naehrig, Ruben Niederhagen, and Peter Schwabe. New software speed records for cryptographic pairings. In Abdalla and Barreto [2], pages 109–123.

[34] Phong Q. Nguyen and David 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.

[35] Kenneth G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.

[36] Bart Preneel and Stafford E. Tavares, editors. *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, volume 3897 of *Lecture Notes in Computer Science*. Springer, 2006.

[37] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairings. In *Proceedings of Symposium on Crytography and Information Security*, 2000.

[38] M. Scott. Faster identity based encryption. *Electronics Letters*, 40(14):861 – 862, July 2004.

[39] M. Scott. A note on twists for pairing friendly curves. Personal webpage, February 2009. `ftp://ftp.computing.dcu.ie/pub/resources/crypto/twists.pdf`.

[40] Adi Shamir. Identity-based cryptosystems and signature schemes. In Blakley and Chaum [11], pages 47–53.

[41] M. Shirase. Universal construction of a 12th degree extension field for asymmetric pairing. *IEICE Transactions*, 94-A(1):156–164, 2011.

[42] Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, Dordrecht, second edition, 2009.

[43] Nigel P. Smart, editor. *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*. Springer, 2005.

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

[45] Moti 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.

Appendix A

Pairing Operation Counts using Scheme 2

Table A.1 lists various field and elliptic curve operation counts using Scheme 1. Using this scheme, the cost of the Miller loop using projective coordinates on BN446 is:

$$\begin{aligned}\text{ML446} &= a + 3\tilde{m}_u + 4\tilde{s}_u + 7\tilde{r} + 19\tilde{a} + 4m + 111(2\tilde{m}_u + 7\tilde{s}_u + 8\tilde{r} + 30\tilde{a} + 4m) \\ &\quad + 6(11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 11\tilde{a} + 4m) + 2(8\tilde{m} + 2a) + 114(13\tilde{m}_u + 6\tilde{r} + 110\tilde{a}) \\ &\quad + 111(12\tilde{m}_u + 6\tilde{r} + 113\tilde{a}) + \tilde{a} + 2(6\tilde{m}_u + 6\tilde{r} + 21\tilde{a}) + 18\tilde{m}_u + 173\tilde{a} + 6\tilde{r} \\ &= 3151\tilde{m}_u + 793\tilde{s}_u + 2345\tilde{r} + 28714\tilde{a} + 472m + 5a \\ &= 11511m_u + 5162r + 85020a.\end{aligned}$$

On the other hand the cost of the Miller loop using Scheme 2 is:

$$\text{ML446A} = 11511m_u + 5162r + 64906a.$$

The cost of the final exponentiation using Scheme 1 is:

$$\begin{aligned}\text{FE446} &= 5(3\tilde{a}) + 25\tilde{m}_u + 9\tilde{s}_u + 24\tilde{r} + 200\tilde{a} + \tilde{\imath} + 12(18\tilde{m}_u + 173\tilde{a} + 6\tilde{r}) \\ &\quad + 2(5\tilde{m} + 6a) + 3(10m + 2\tilde{a}) + 3(45\tilde{m}_u + 666\tilde{s}_u + 467\tilde{r} + 5939\tilde{a} + 110a + \tilde{\imath}) \\ &\quad + 3(9\tilde{s}_u + 74\tilde{a} + 6\tilde{r}) \\ &= 386\tilde{m}_u + 2034\tilde{s}_u + 1525\tilde{r} + 4\tilde{\imath} + 20336\tilde{a} + 30m + 342a.\end{aligned}$$

Using Scheme 2 requires 9111 fewer base field additions.

$E'(\mathbb{F}_{p^2})$ - **Arithmetic**	**Operation Count**
Doubling/Eval. (x86-64)	$3\tilde{m}_u + 6\tilde{s}_u + 8\tilde{r} + 36\tilde{a} + 4m$
Doubling/Eval. (ARM)	$2\tilde{m}_u + 7\tilde{s}_u + 8\tilde{r} + 30\tilde{a} + 4m$
Doubling/Eval. (Affine)	$\tilde{\imath} + 3\tilde{m}_u + 2\tilde{s}_u + 5\tilde{r} + 8\tilde{a} + a + 2m$
Addition/Eval.	$11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 11\tilde{a} + 4m$
First Doubling/Eval	$3\tilde{m}_u + 4\tilde{s}_u + 7\tilde{r} + 19\tilde{a} + 4m$
Addition/Eval. (Affine)	$\tilde{\imath} + 3\tilde{m}_u + \tilde{s}_u + 4\tilde{r} + 8\tilde{a} + 2m$
p-power Frobenius	$8\tilde{m} + 2a$
Negation	$\tilde{a}$
$\mathbb{F}_{p^2}$ - **Arithmetic**	**Operation Count**
Add./Sub./Neg.	$\tilde{a} = 2a$
Conjugation	a
Multiplication	$\tilde{m} = \tilde{m}_u + \tilde{r} = 3m_u + 2r + 8a$
Squaring	$\tilde{s} = \tilde{s}_u + \tilde{r} = 2m_u + 2r + 3a$
Multiplication by i	$2a$
Multiplication by ξ	$3a$
Inversion	$\tilde{\imath}$
$\mathbb{F}_{p^{12}}$ - **Arithmetic**	**Operation Count**
Add./Sub	$6\tilde{a}$
Conjugation	$3\tilde{a}$
Multiplication	$18\tilde{m}_u + 173\tilde{a} + 6\tilde{r}$
Sparse Multiplication	$13\tilde{m}_u + 6\tilde{r} + 110\tilde{a}$
Sparser Multiplication	$6\tilde{m}_u + 6\tilde{r} + 21\tilde{a}$
Affine Sparse Multiplication	$10\tilde{m}_u + 6\tilde{r} + 109\tilde{a} + 6m_u + a$
Squaring	$12\tilde{m}_u + 6\tilde{r} + 113\tilde{a}$
Cyclotomic Squaring	$9\tilde{s}_u + 74\tilde{a} + 6\tilde{r}$
Compressed Squaring	$6\tilde{s}_u + 51\tilde{a} + 4\tilde{r}$
Simultaneous Decompression	$9\tilde{m} + 6\tilde{s} + 38\tilde{a} + \tilde{\imath}$
p-power Frobenius	$5\tilde{m} + 6a$
p^2-power Frobenius	$10m + 2\tilde{a}$
Exponentiation by x	$45\tilde{m}_u + 666\tilde{s}_u + 467\tilde{r}_u + 5939\tilde{a} + 110a + \tilde{\imath}$
Inversion	$25\tilde{m}_u + 9\tilde{s}_u + 24\tilde{r} + 200\tilde{a} + \tilde{\imath}$

Table A.1: Operation counts for 446-bit prime field using Scheme 1, for notation refer to Section 3.2.1.

Appendix B

Cross-over I/M ratios on BN446 and BN638

When using a prime congruent to 7 mod 8, an affine doubling costs an extra $\mathbb{F}_{q^2}$ inversion and unreduced multiplication, and saves $5\tilde{s}_u + 3\tilde{r} + 26\tilde{a} + 2m$ compared to a projective doubling. Compared to a first doubling, it costs an extra $\mathbb{F}_{q^2}$ inversion and saves $2\tilde{s}_u + 2\tilde{r} + 15\tilde{a} + 2m$. An addition costs an extra $\mathbb{F}_{q^2}$ inversion and saves $8\tilde{m}_u + \tilde{s}_u + 7\tilde{r} + 3\tilde{a} + 2m$, and a dense-sparse multiplication needs six additional base field multiplications and saves $3\tilde{m}_u + 3\tilde{r} + 0.5\tilde{a}$.

B.1 BN446

Computing a pairing on BN446 requires one first doubling, 111 doublings, six additions, and 114 dense-sparse multiplications. Thus, the difference between an affine and projective

pairing is:

$$\begin{aligned}
446A - 446P &= (\tilde{\imath} - 2\tilde{s}_u - 2\tilde{r} - 15\tilde{a} - 2m) + 111(\tilde{\imath} + \tilde{m}_u - 5\tilde{s}_u - 3\tilde{r} - 26\tilde{a} - 2m) + \\
&\quad 6(\tilde{\imath} - 8\tilde{m}_u - \tilde{s}_u - 7\tilde{r} - 3\tilde{a} - 2m) + 114(6m - 3\tilde{m}_u - 3\tilde{r} - 0.5\tilde{a}) \\
&= 118\tilde{\imath} + 448m - 279\tilde{m}_u - 551\tilde{s}_u - 719\tilde{r} - 2976\tilde{a} \\
&= 118(i + 4m_u + 3r + 5a) + 448(m_u + r) - 279(3m_u + 10a) - \\
&\quad 551(2m_u + 5a) - 719(2r) - 2976(2a) \\
&= 118i - 1019m_u - 636r - 10907a.
\end{aligned}$$

On an ARM Feroceon, where we can assume that $m \approx 2r$ and $23a \approx m$, this gives a crossover inversion-to-multiplication(I/M) ratio of 11. On an iPad, we see that $m \approx 1.5r$ and we get a crossover I/M ratio of 10.2.

B.2 BN638

Computing a pairing on BN638 requires one first doubling, 160 doublings, eight additions, and 167 dense-sparse multiplications. Thus, the difference between an affine and projective pairing is:

$$\begin{aligned}
638A - 638P &= (\tilde{\imath} - 2\tilde{s}_u - 2\tilde{r} - 15\tilde{a} - 2m) + 160(\tilde{\imath} + \tilde{m}_u - 5\tilde{s}_u - 3\tilde{r} - 26\tilde{a} - 2m) + \\
&\quad 8(\tilde{\imath} - 8\tilde{m}_u - \tilde{s}_u - 7\tilde{r} - 3\tilde{a} - 2m) + 167(6m - 3\tilde{m}_u - 3\tilde{r} - 0.5\tilde{a}) \\
&= 169\tilde{\imath} + 664m - 405\tilde{m}_u - 810\tilde{s}_u - 1039\tilde{r} - 4282.5\tilde{a} \\
&= 169(i + 4m_u + 3r + 5a) + 664(m_u + r) - 405(3m_u + 10a) - \\
&\quad 810(2m_u + 5a) - 1039(2r) - 4282.5(2a) \\
&= 169i - 1531m_u - 934r - 15865a.
\end{aligned}$$

On an ARM Feroceon, where we can assume that $m \approx 2r$ and $25a \approx m$, this gives a crossover inversion-to-multiplication(I/M) ratio of 11. On an iPad, we see that $m \approx 1.5r$ and we get a crossover I/M ratio of 10.2. These ratios are the same as the ones for BN446.