ON THE IMPLEMENTATION OF PAIRING-BASED CRYPTOSYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ben Lynn

June 2007

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dan Boneh Principal Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

John Mitchell

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Xavier Boyen

Approved for the University Committee on Graduate Studies.

Abstract

Pairing-based cryptography has become a highly active research area. We define bilinear maps, or pairings, and show how they give rise to cryptosystems with new functionality.

There is only one known mathematical setting where desirable pairings exist: hyperelliptic curves. We focus on elliptic curves, which are the simplest case, and also the only curves used in practice. All existing implementations of pairing-based cryptosystems are built with elliptic curves. Accordingly, we provide a brief overview of elliptic curves, and functions known as the Tate and Weil pairings from which cryptographic pairings are derived.

We describe several methods for obtaining curves that yield Tate and Weil pairings that are efficiently computable yet are still cryptographically secure.

We discuss many optimizations that greatly reduce the running time of a naive implementation of any pairing-based cryptosystem. These techniques were used to reduce the cost of a pairing from several minutes to several milliseconds on a modern consumer-level machine.

Applications of pairings are largely beyond our scope, but we do show how pairings allow the construction of a digital signature scheme with the shortest known signature lengths at typical security levels.

ON THE IMPLEMENTATION OF PAIRING-BASED CRYPTOSYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ben Lynn

June 2007

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dan Boneh Principal Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

John Mitchell

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Xavier Boyen

Approved for the University Committee on Graduate Studies.

Preface

Pairing-based cryptography is a relatively young area of cryptography that revolves around a particular function with interesting propreties. It allows the construction of novel cryptosystems that are otherwise difficult or impossible to assemble using standard primitives.

We first define a cryptographic pairing abstractly and show how it can be used to build a signature scheme that is simple yet has many desirable properties. Next we examine how a pairing can be implemented in practice. The only known mathematical setting where suitable pairings exist are groups on certain families of curves. We focus on the simplest and most well-understood case: elliptic curves.

We discuss methods for finding curves that yield cryptographic pairings, and outline various optimizations that can be applied. We shall see that pairing-based cryptosystems are quite practical and compare well against traditional cryptosystems.

It is hoped that this work will be a useful guide to implementing pairing-based cryptography to a programmer who has experience with conventional cryptosystems. While not comprehensive, the information contained herein should be enough to allow one to build practical pairing-based applications from scratch. Notably absent are in-depth discussion of the characteristic 3 case, pairings where the subgroup size is necessarily significantly smaller than the field size, and hyperelliptic curves,

This text is intended to be self-contained. Aside from arithmetic at the lowest level, algorithms not likely to be found in a basic course in number theory or cryptography are quoted here, enabling the reader to implement pairings without referring to any other sources.

Some background is required by the reader as we do not review basic abstract algebra. On the other hand, we introduce enough elliptic curve theory for cryptographic purposes.

The following publications form the foundation of this thesis:

- D.Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(1):297–319, 2004.

- P. S. L. M. Barreto, H. Y. Kim, B. Lynn and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO 2002*, pages 354–368.

- P. S. L. M. Barreto, B. Lynn and M. Scott. Constructing elliptic curves with prescribed embedding degree. In *Third Conference on Security in Communication Networks 2002*.

- P. S. L. M. Barreto, B. Lynn and M. Scott. On the selection of pairing-friendly groups. In *Selected Areas of Cryptography 2003*.

More polished versions of the above works can be found in the Journal of Cryptology [18, 7].

Most of the described algorithms and optimizations feature in the PBC (Pairing-Based Cryptography) library, maintained by the author, and available under the GNU Public License at `http://crypto.stanford.edu/pbc/`.

Contents

List of Tables

List of Figures

Chapter 1

Bilinear Maps

Asymmetric ciphers, or public-key cryptosystems, are perhaps the most celebrated contribution of modern cryptography. They certainly have had the most impact. It is hard to imagine what the world would be like without their revolutionary approach to key distribution.

All public-key cryptosystems in wide use today can trace their roots to the Diffie-Hellman key exchange protocol [27] or the RSA cryptosystem [54]. The former depends on cyclic groups with particular properties. The latter, though using similar arithmetic operations, relies on different principles. For example, RSA uses groups that are not cyclic and requires that the order of the group to be unknown to the attacker.

Roughly speaking, bilinear maps, or pairings, give cyclic groups additional properties. Initially, in the 1990s, these additional properties were seen as detrimental as they could be exploited to break cryptosystems [46, 32, 31], but it was later discovered that they could also be exploited to build cryptosystems. Rather than avoiding pairings, one can seek them out to construct new schemes.

Boneh and Franklin's identity-based encryption scheme [13] is perhaps the most famous early example of what could be achieved using bilinear maps, though not the first [57, 40]. Shamir first discussed identity-based encryption in 1984 [61], but researchers were unable to build a practical scheme by conventional means for approximately twenty years. Boneh and Franklin found an elegant solution using bilinear maps [13]. Extending the basic idea leads to identity-based schemes with additional useful properties such as authenticated or hierarchical identity-based encryption [45, 39]. More generally, so many cryptographic applications of the pairing have been identified that this area of research is sometimes considered its own field called pairing-based cryptography [3].

We note that an identity-based scheme based on quadratic residues and not bilinear maps has since been proposed [20, 14], albeit one that is signifcantly less practical. However, this is the exception rather than the rule. In general it is not known how to find conventional equivalents of a given pairing-based cryptosystem.

1.1 Cyclic Groups

Let $G = \langle g \rangle$ of prime order r. Let g be a generator of G and let x, y, z be integers in $[0, r - 1]$. Consider the following problems.

Discrete Log Problem. Given g, g^x, compute x.

Computational Diffie-Hellman Problem. Given g, g^x, g^y, compute g^{xy}.

Decisional Diffie-Hellman Problem. Given g, g^x, g^y, g^z, determine if $xy = z$.

A vast array of cryptosystems including the Diffie-Hellman key exchange protocol can be built assuming one of these problems, or one of many other related problems, is difficult to solve.

The discrete log problem is the most important, for if it could be solved, all other related problems could also be solved.

1.2 Formal Security Definitions

For completeness we include formal definitions of difficult problems commonly used by cryptographers, even though we do not need them as we do not provide any security proofs. Our focus is on the implementation of pairing-based cryptosystems, not their correctness.

There are two flavours of definitions. One is asymptotic and the other uses fixed security parameters.

We first describe the asymptotic definition. Roughly speaking, a problem is hard if there is no efficient algorithm that can solve it. We say a function $f : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is negligible if for all $c > 0$ there exists a $\lambda_0 > 0$ such that for all $\lambda \geq \lambda_0$, $f(\lambda) < 1/\lambda^c$. Intuitively a function is negligible if it is smaller than any polynomial function for sufficiently large inputs. Examples of negligible functions are $2^{-\lambda}$ and $\lambda^{-\log \lambda}$, while $\lambda^{-10000000}$ is nonnegligible.

Let $\mathcal{G} = \{G_1, G_2, ...\}$ be a family of cyclic groups where the order of G_i is an i-bit number. For an algorithm A define the *advantage* of A to be

$$Adv_A(t) = \Pr[A(g, g^x) = x]$$

is negligible, where g is randomly chosen (uniformly) from G_t and x from $[0, |G_t| - 1]$.

Then we say that the discrete log problem is hard in $\mathcal{G}$ if Adv_A is negligible for all probabilistic expected polynomial-time algorithms A.

Decisional variants of problems must be handled differently. For example, we say the decisional Diffie-Hellman problem is hard if for all probabilistic expected polynomial-time algorithms A

$$Adv_A(t) = |\Pr[A(g, g^x, g^y, g^z) = B(x, y, z)] - 1/2|$$

is negligible, where g is randomly chosen from G_t, x, y from $[0, r - 1]$, and z from $[0, r - 1]$ with probability $1/2$ and set to xy otherwise, and where B returns 1 if $xy = z$ and 0 otherwise.

The other kind of definition on the other hand is couched explicitly in terms of running times and probabilities. It is useful when examining groups of a particular size, and similarly in cases where there are no families of infinite size. A classic example is the DES cipher, where an asymptotic definition cannot be used since DES keys are always 56 bits long.

When using fixed security parameters, we say the discrete log problem is (t, ε)-hard in a given group G if for all probabilistic t-time algorithms A we have

$$\Pr[A(g, g^x) = x] < \varepsilon$$

where g is uniformly chosen from G and x from $[0, |G| - 1]$.

The security definitions for other problems in either flavour are similarly constructed and are omitted here.

Of course, there is always another requirement for cyclic groups used in cryptography. Efficient algorithms for multiplying group elements, inverting group elements, and hashing to a group element must exist. Exponentiating and choosing a random element should be efficient as well, but this is implied by fast multiplication: exponentiation by repeated squaring is efficient, and for random element generation one can simply choose a random integer and raise a generator by this power.

In practice, “efficient” means fast enough so that on the platform that the system is deployed the user will not notice any slowdown. With pairing-based cryptography, the bilinear map must also be efficient.

1.3 Concrete Cyclic Groups

There are two mathematical settings to choose from when implementing cyclic groups: finite fields and elliptic curves. We give a short summary here but will expound on elliptic curves later.

For finite fields, one picks a prime n and uses a subgroup G of $\mathbb{Z}_n^*$ of prime order r, so the group operation is field multiplication. Note an RSA cryptosystem arises when n is instead chosen to be the product of two large primes P, Q, in which case computations are still possible in $\mathbb{Z}_n^*$ even if its order is unknown.

For elliptic curves, one takes an elliptic curve E over some finite field K and takes some subgroup G of the group of points $E(K)$ with prime order r, so the group operation is point addition. There is no elliptic curve version of RSA, but we shall encounter a case which may seem confusingly similar to RSA: we sometimes choose a curve whose group of points has order $n = PQ$.

Finite fields were proposed first, and much has been published on implementing finite fields and their potential weaknesses. Of particular interest are subexponential discrete log algorithms such as index calculus, resulting in recommendations that field sizes be at least 1024 bits.

Elliptic curves were investigated later, and their use in cryptography was at first viewed as experimental. However, they are now much less controversial thanks to the efforts of many researchers. A myriad of optimizations and algorithms were discovered for elliptic curve cryptography, and encouragingly, a specialized elliptic curve discrete log algorithm has yet to be found. In other words, the fastest known way to break discrete log for a general elliptic curve is to use a algorithm for a generic cyclic group, such as the Pollard rho and lambda methods, implying that the security of a 160-bit elliptic curve group and a 1024-bit finite field are roughly equivalent.

Thus currently elliptic curves present many advantages over finite fields. Despite more complex algorithms for basic operations, they are much faster and more compact due to a smaller group size.

In Section 2.8 we find that finite fields can be viewed as a special case of elliptic curves due to a correspondence between certain singular curves and multiplicative groups of finite fields.

1.4 The Symmetric Pairing

We begin with the original and simplest abstract definition of the pairing. Let G, G_T be cyclic groups of prime order r. Let g be a generator of G. A *bilinear pairing* or *bilinear map* e is an efficiently computable function

$$e : G \times G \to G_T$$

such that

1. *(Nondegeneracy)* $e(g, g) \neq 1$
2. *(Bilinearity)* $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}$

A symmetric bilinear map is completely defined by the value it takes at $e(g, g)$. Essentially there is always exactly one bilinear map for any given cyclic group: we have the degenerate case when $e(g, g) = 1$ while the other $r - 1$ maps are equivalent up to a constant, in the sense that if e_1, e_2 are nondegenerate bilinear maps then for some constant c we have then $e_1(g, h) = e_2(g, h)^c$ for all $g, h \in G$. The difficulty lies in finding such a map that is efficiently computable.

We can immediately demonstrate power of such a map. Given g, g^x, g^y, g^z, by bilinearity and nondegeneracy, $z = xy$ if and only if $e(g, g^z) = e(g^x, g^y)$. In other words we can solve the decisional Diffie-Hellman problem. Thus cryptosystems relying on the intractability of this problem cannot be constructed on a cyclic group with a symmetric pairing, though later we present a more general pairing definition where DDH can still be difficult despite the presence of a bilinear map.

Note it is not known how to solve the computational Diffie-Hellman or discrete log problem using a pairing.

1.5 The BLS Signature Scheme

With the abstract definition and some assumptions, we can readily construct cryptosystems. We have just noted that if cyclic group with such a bilinear map exists, then we have a group where the computational Diffie-Hellman problem is thought to be hard, yet the decisional variant is easy to solve. Such groups are sometimes called Gap Diffie-Hellman groups [51], and imply a signature scheme, often referred to as the BLS Signature Scheme [17]:

Setup. Choose a Gap Diffie-Hellman group G of prime order r. Publish a generator $g \in G$.

Key Generation. Choose a random x in $\{1, ..., r-1\}$. Output the public key g^x and the private key x.

Signing. Given a message $h \in G$, output h^x.

Verify. Given a message-signature pair h, σ and public key g^x, check that $\langle g, h, g^x, \sigma \rangle$ is a Diffie-Hellman tuple.

It can be shown that under the computational Diffie-Hellman assumption, this signature scheme is secure against existential forgery under a chosen-message attack in the random oracle model [17].

Hence an abstract view of the pairing is all that is needed to build a cryptosystem, and frequently this suffices. However, we shall see that it can be useful to understand pairings in more detail. For example, we need to know more to show the above signature scheme is short in length, which in fact was the original motivation for its proposal.

This signature scheme in fact has other useful properties, including batched verification and allowing the the simple construction of aggregate, ring and verifiably-encrypted signatures[15].

1.6 New Hardness Assumptions

Classic problems have natural counterparts in pairing-based cryptography. In the examples below, let g be a generator for a group G of prime order r, and let e be a bilinear map on G.

Bilinear Diffie-Hellman Problem [13]. Given g, g^x, g^y, g^z compute $e(g, g)^{xyz}$. This assumption was implicitly used before it was formally defined [57, 40].

Decisional Bilinear Diffie-Hellman Problem [10]. Given $g, g^x, g^y, g^z, e(g,g)^w$, determine if $w = xyz$. This has also been referred to as BDDH [13].

q-Strong Diffie-Hellman Problem [11]. Given $g, g^x, ..., g^{(x^q)}$, compute $c, g^{1/(x_c)}$ for any $c \in \{1, ..., r-1\}$.

Just as an algorithm that solves discrete log can be used to solve the computational and decisional Diffie-Hellman problems, security reductions exist between some of these new problems [41].

The BLS signature scheme does not use any of these new assumptions, and merely employs the pairing to solve the decisional Diffie-Hellman problem. In contrast, other pairing-based cryptosystems such as the identity-based encryption scheme of Boneh and Franklin [13] do rely on the hardness of one or more problems involving pairings.

Sometimes a conventional cryptosystem suggests a pairing-based counterpart. For example, the above pairing-based signature scheme and a ring signature scheme due to Rivest et al. [55] were presented in adjacent sessions at a conference. Someone taking the ideas from both talks would not have much difficulty in synthesizing their results. Indeed, a pairing-based ring signature scheme was presented about one year later[15].

As before, both asymptotic and fixed-security-parameter formal security definitions can be constructed for each of the above problems.

1.7 Loosening the Pairing Definition

In practice, symmetric pairings can be instantiated by using suitable supersingular elliptic curves. However, in order to allow a wider range of curves to be used, the definition must be modified. One early pairing-based cryptography publication by Boneh, Lynn and Shacham suggested the following [17]:

Let G_1, G_2, G_T be cyclic groups of prime order r. Assume the Diffie-Hellman problem is hard in G_1. Let $\phi : G_2 \to G_1$ be an efficiently computable group isomorphism. Let g_2 be a generator of G_2. Set $g_1 = \phi(g_2)$ (so g_1 generates G_1). A bilinear pairing e is an efficiently computable function

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

such that $e(g_1, g_2) \neq 1$ and $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}$.

This version of the bilinear map is sometimes called the *asymmetric pairing*.

This definition allows a greater variety of pairings to be used, notably those constructed on ordinary curves. Additionally, because of the map ϕ security proofs require only minimal changes. For example, a scheme based on the CDH assumption in G under the symmetric pairing definition may now be based on the CDH assumption in G_2.

However, it is advantageous to further loosen the definition.

1.7.1 A Problem With Hashing

Suppose we have chosen an ordinary curve that does not yield a symmetric pairing, and suppose that we wish to redesign the cryptosystem to instead use an asymmetric pairing defined above.

It turns out that there is no known method to hash to an element of G_2 such that its discrete log to some fixed base is unknown. We can hash to elements of G_1 and G_T, and perform all other cryptographically useful operations in G_2 such as picking a random element, multiplication and inversion. We can map an element of G_2 to G_1. We just cannot hash a string to an element of G_2.

For some cryptosystems, this issue can complicate their design, and in extreme cases we may have to give up and stick to symmetric pairings only. However, with a more general pairing definition, we can use ordinary curves and still be able to hash to G_2.

1.8 The General Bilinear Pairing

To enable certain optimizations (see Section 6.5) and to permit hashing to any element of any group including G_2, we prefer the following definition. Additionally, its more flexible nature allows a greater number of assumptions which in turn allow more cryptosystems to be built.

Let r be a prime. Let G_1, G_T be cyclic groups of order r. Let G_2 be a group where each element has order dividing r. In particular G_2 is not necessarily cyclic. Again we use multiplicative group notation. A bilinear pairing e is an efficiently computable function

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

such that

1. *(Nondegeneracy)* $e(g_1, g_2) = 1_{G_T}$ for all $g_2 \in G_2$ if and only if $g_1 = 1_{G_1}$, and similarly

$e(g_1, g_2) = 1_{G_T}$ for all $g_1 \in G_1$ if and only if $g_2 = 1_{G_2}$.

2. *(Bilinearity)* for all $g_1 \in G_1$ and $g_2 \in G_2$: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}$

In this setting the hardness assumptions must be altered further. Depending on the scheme, we may have to assume certain problems are hard in both G_1 and G_2, or a combination of the two. For example, we may need to assume that given $g_1, g_1^x \in G_1$ and $g_2 \in G_2$, there is no efficient algorithm to compute g_2^x. This last example is sometimes referred to as the computational co-Diffie-Hellman assumption [17].

This definition also allows assumptions that were previously impossible. For example, some schemes such as the group signature scheme of Boneh, Boyen and Shacham [12] require discrete log, CDH and co-CDH to be hard in G_1, G_2 and also DDH to be hard in G_1. Other cryptosystems still require in addition DDH to be hard in G_2, require G_2 to be cyclic, or require an efficiently computable group isomorphism $\phi : G_2 \to G_1$, all of which are variations on this formal definition, and in Section 6.4.2 we learn how to construct pairings for each case.

We have not required r to be prime. A composite group order is useful for some cryptosystems[16], but we must be aware that in this case even if g_1 and g_2 have order r, $e(g_1, g_2)$ may not be a generator of G_T, but rather a generator of some subgroup of G_T whose order is a factor of r.

When r is composite, it is incorrect to say all nondegenerate pairings are equivalent up to a constant. Indeed, if $d > 1$ is a divisor of r and g_1, g_2 are generators of G_1, G_2 respectively, then a bilinear map that takes (g_1, g_2) to a dth root of unity is still nondegenerate. Since there is more than one choice for d for a composite r, these maps cannot be equivalent up to a constant. They are however still completely determined by $e(g_1, g_2)$ where g_1, g_2 are generators. We present a concrete example later in Section 4.2.1.

This also means that $e(g_1, g_2) = 1$ does not imply at least one of g_1, g_2 is the identity element, in contrast to the case when r is prime.

Thus some care is needed when dealing with composite r. Facts about the pairing that are true for prime r do not always carry over. We shall encounter a concrete example of this phenomenon later. Note when r is chosen to be a product of two primes, the resulting scheme is still substantially different to RSA as the groups are cyclic and of known order.

1.9 Exponentiation as a Bilinear Pairing

Given some integer $r > 1$, let us take $G_1 = G_T = \mathbb{Z}_r^*$ and $G_2 = \mathbb{Z}_{r-1}^+$. Define $e : G_1 \times G_2 \to G_T$ by $e(g, a) = g^a$.

Then it is easily seen that if we relax the condition that the groups are cyclic, then exponentiation satisfies the definition of a bilinear pairing. Thus we may view all discrete log and RSA cryptosystems as pairing-based cryptosystems, though there are significant differences in this setting. For example, the discrete log problem is easy in G_2, and $G_1 = G_T$.

Classic problems can be restated in terms of pairings. For example:

Discrete Log Problem. Given $g \in G_1, c \in G_T$, find $a \in G_2$ such that $e(g, a) = c$.

RSA Problem. Given $a \in G_2, c \in G_T$, find $g \in G_1$ with $e(g, a) = c$.

Strong RSA Problem. Given $c \in G_T$, find $g \in G_1, a \in G_2$ with $e(g, a) = c$

Thinking of conventional cryptosystems as pairing-based cryptosystems can inspire new ideas.

1.10 Choosing a Definition

The symmetric pairing with cyclic groups into which strings can be hashed is often selected as it allows simpler and briefer mathematical statements and definitions, and the symmetry can be useful. However, there are few families of elliptic curves that allow such a pairing.

In practice, one can usually trivially modify symmetric-pairing-based schemes and their corresponding security proofs to use a more general pairing definition, so that more families of curves can be used. For example, one may have to replace the the Computation Diffie-Hellman assumption with the co-Computation Diffie-Hellman problem. Or perhaps a minor reworking allows the cryptosystem to function when G_2 is not cyclic.

Ideally a cryptosystem should use the most general definition when first proposed, allowing maximum choice for the pairing in an implementation. If a more restrictive definition is necessary, one must pick the extra assumptions carefully and be aware that perhaps only a few pairing families may satisfy them. For example, there is no known pairing such that

1. Both input groups G_1, G_2 are cyclic.

2. One can hash strings to G_1 and G_2 without knowing the discrete log of the output to some fixed base.

3. There is an efficiently computable isomorphism $\phi : G_2 \to G_1$ whose inverse is difficult to compute.

yet for any two of the three conditions, we can find pairings that satisfy both.

Familiarizing oneself with the mathematics behind pairings has several advantages over merely memorizing valid assumptions and properties of the pairings that correspond to them. Being ignorant of the inner workings of available pairings means we do not know when we have made one assumption too many. For instance, it is not obvious that we can only stipulate up to two of the three conditions in the above list. Knowledge of the computations being performed also enables us to estimate the time and space taken by cryptographic operations, which is affected by the pairing choice in subtle ways. For example, if we use a symmetric pairing, then typically elements will be 512 bits long. If we forgo symmetry and use two groups G_1, G_2, then it turns out we may achieve a 160-bit G_1 and a 320-bit G_2, though pairings are slower.

1.11 Concrete Bilinear Maps

There is only one known setting for cryptographically useful bilinear maps, namely elliptic curves with an efficiently computable Tate pairing.

Strictly speaking, one can use hyperelliptic curves where much of the theory generalizes, but this is beyond our scope. We can find hyperelliptic curves [33, 56, 29] suitable for pairings, and although hyperelliptic curve cryptography appears to be a promising avenue for future research, there are no compelling reasons to use them yet. To the author's knowledge, there are no implementations of pairing-based cryptosystems based on hyperelliptic curves, and in practical terms, G, or G_1 and G_2 are always groups of points on an elliptic curve, and G_T is always a subgroup of a multiplicative group of a finite field.

Mathematicians have traditionally employed additive notation for the group of points on an elliptic curve, since it is Abelian. As a result, publications in the area often use additive group notation along with uppercase letters for elements for the input groups.

We have used multiplicative notation in this chapter to highlight the Diffie-Hellman heritage of pairing-based cryptography. This notation also suggests the right way to compare elliptic curve and finite field operations: point multiplications should be compared with exponentiations, not integer multiplications. Later we will switch to additive notation when discussing the underlying mathematics.

When deploying a pairing-based cryptosystem one must select a type of curve to use. The correct choice depends on several factors, which we discuss in later chapters.

We conclude with a toy example of the BLS signature scheme. Let $q = 427211363219$. Consider the curve $E : y^2 = x^3 + x$ over $\mathbb{F}_q$. Then $E(\mathbb{F}_q)$ contains a subgroup G of order $r = 524353$ (which is a prime factor of $427211363219 + 1$). Take a random generator

$$g = (359473638793, 293998693014)$$

for the system parameter, and a random secret key

$$x = 66995.$$

Then the corresponding public key is

$$g^x = (166505572345, 357692656519).$$

Suppose a message hashes to

$$h = (370499138522, 78458612837).$$

Then the corresponding signature is

$$\sigma = h^x = (278647014375, 78919786700).$$

To verify, we must check that $e(g, \sigma) = e(g^x, h)$ which turns out to be

$$85791756805 + 99975533880i,$$

an element of $K[i]$ that has order r. The exact value of the last result may vary from implementation to implementation since the pairing is only unique up to a constant and different implementations may result in different constants, but both sides of the equation always match.

The order r in the above example is roughly 20 bits in length, while the order q of the field, and the space required by compressed elements of G_1 and G_2 is roughly 40 bits. In real life, one would require a 160-bit r and 512-bit q for secure signatures.

Chapter 2

Elliptic Curves

We review basic facts about elliptic curves. Aside from the problem of generating suitable elliptic curves and counting the size of the resulting group, we will cover enough theory to replace finite fields with elliptic curves in cryptosystems based on cyclic groups.

We discuss pairings and algorithms to find pairing-friendly curves in future chapters. We will not discuss curve-finding and point-counting algorithms geared towards standard elliptic curve cryptography which necessarily requires curves that are not pairing-friendly, but instead direct the interested reader to Blake, Seroussi and Smart [9].

2.1 Informal Overview of Elliptic Curves

We present a highly informal overview to elliptic curve theory, which may aid those that have not encountered them before. This section can be safely skipped.

Consider a polynomial C in two variables X, Y. We are interested in the solutions to $C = 0$ which describe a curve on a two-dimensional plane.

We first observe that if C' is another curve that is an affine transformation of C, that is, if we can linearly transform (e.g. rotate, scale, shear) and then translate C to obtain C' then a correspondence exists between the solutions to $C = 0$ and the solutions to $C' = 0$. Knowing the solutions of one allows us to easily compute the solutions of the other, For this reason we consider such curves C and C' to be equivalent.

If every term in C has combined degree of at most 1, that is, if $C = aX + bY + c$ then C describes a line. The geometry of lines is too simple to yield anything cryptographically useful.

If every term in C has combined degree at most 2, then C describes a single line, a pair of lines, an ellipse, a parabola or a hyperbola. The first two possiblities can be viewed as special cases, occuring when C is reducible or degenerate in some sense.

By adding *points of infinity* to the plane, we can find affine transformations that change any ellipse, parabola or hyperbola into the unit circle centred at the origin. Intuitively, the two ends of the parabola can be thought of as meeting at a point at infinity, forming a circle, and similarly opposite ends of hyperbolas connect at infinite points.

Thus to study degree 2 curves is essentially to study the unit circle, whose geometry is again is too simple for our purposes.

However, degree 3 curves, called elliptic curves, are nontrivial (for instance, unlike the previous two cases we cannot transform any elliptic curve into any other) and have a rich structure well-suited for cryptography.

For any irreducible elliptic curve C, applying appropriate affine transformations produces an equation C' of a certain form known as the *Weierstrass form*. In this form, C' always contains exactly one point at infinity. Transforming C so that no points are infinite is possible but leads to more complicated equations. Using an equation that has exactly one point at infinity simplifies some of the expressions.

2.2 Points on Elliptic Curves

Let $\mathbb{F}_q$ be a field for some prime $q > 3$. Unless otherwise specified we shall always define curves over a field of prime order and of characteristic greater than three. Elliptic curves can be implemented over fields of characteristic 2 and 3 and enjoy many optimizations, but suffer from specialized discrete log attacks [24] and should generally be avoided. We do consider a certain family of characteristic three curves in Section 4.16, where we also outline their trade-offs.

An elliptic curve E over such a field $\mathbb{F}_q$ is an equation of the form

$$E : Y^2 = X^3 + aX + b$$

where $a, b \in \mathbb{F}_q$. Let $\Delta = 4a^3 + 27b^2$, the discriminant of the cubic in x. Then E is *singular* if $\Delta = 0$, i.e. the cubic has a repeated root, and *nonsingular* otherwise, i.e. the cubic has distinct roots.

Unless otherwise stated we always consider nonsingular elliptic curves. Later we will encounter *supersingular* curves, a particular breed of nonsingular curves that are not singular despite their name. The antonym of supersingular is *ordinary* or *nonsupersingular.*

For any field $\mathbb{F}_{q^k}$ define $E(\mathbb{F}_{q^k})$ to be the set of all solutions of E over $\mathbb{F}_{q^k}$, called the *finite points* along with a special point denoted O, that is called the *point at infinity.* We write $\#E(\mathbb{F}_{q^k})$ or $|E(\mathbb{F}_{q^k})|$ for the number of elements of $E(\mathbb{F}_{q^k})$. Intuitively, the point O can be thought of as the point where all lines parallel to the Y-axis meet. Mathematically, we solve the curve equation using *projective coordinates* [25] and one can show that $O = (0, 1, 0)$ is always a unique infinite solution to the equation. We shall never need this fact.

We quote two well-known theorems.

Theorem (Hasse)**.** *Let* $t = q^k + 1 - \#E(\mathbb{F}_{q^k})$*. Then* $|t| \leq 2\sqrt{q^k}$*.*

Thus the number of points on an elliptic curve in a given field is on the same order as the size of the field. The quantity t is called the *trace of Frobenius.*

Theorem (Weil)**.** *Let* $t = q + 1 - \#E(\mathbb{F}_q)$ *where* q *is a prime power. Factor the polynomial* $x^2 - tx + q$ *as* $(x - \alpha)(x - \beta)$ *over* $\mathbb{C}[x]$*. Then*

$$\#E(\mathbb{F}_{q^k}) = q^k + 1 - (\alpha^k + \beta^k).$$

This last theorem is more practical in the following form. Let $t_0 = 2$, Let $t_1 = q + 1 - \#E(\mathbb{F}_q)$. Define t_n recursively by

$$t_n = t_1 t_{n-1} - q t_{n-2}.$$

Then $\#E(\mathbb{F}_{q^k}) = q^k + 1 - t_k$.

For example, consider the curve E given by

$$Y^2 = X^3 + X + 6$$

over $\mathbb{F}_{19}$, an example used by Balasubramanian and Koblitz [1]. There are 18 points

$$\begin{array}{llllll}
(0,5), & (4,6), & (2,4), & (3,6), & (14,3), & (12,13), \\
(18,2), & (10,3), & (6,0), & (10,16), & (18,17), & (12,6), \\
(14,16), & (3,13), & (2,15), & (4,13), & (0,14), & O
\end{array}$$

thus the trace of Frobenius $t = 2$.

Over $\mathbb{F}_{19^2}$, we have

$$\#E(\mathbb{F}_{19^2}) = 19^2 + 1 - t_2$$

where $t_2 = 2 \cdot 2 - 19 \cdot 2 = -34$, thus $\#E(\mathbb{F}_{19^2}) = 396$.

2.3 Finding Points

Let $E : Y^2 = X^3 + aX + b$ be an elliptic curve over a field K. There always exists an unique infinite solution, namely O. We describe a simple method for finding the finite points of E.

For any $x \in K$, we may attempt to solve $Y^2 = x^3 + ax + b$ for Y by finding a square root of the right-hand side. We momentarily postpone describing the details of square root algorithms. For now, assume we can find square roots.

When solutions for Y do exist for a given x, we have found exactly two points, one for each square root, except in the rare case when the point lies on the X-axis, which can happen in at most three places.

Also, recall from an above theorem that the size of K is roughly the same as the number of points on $E(K)$.

Combining these two facts shows that for approximately half of the choices for $x \in K$, a square root exists and we can solve E to find a point. Thus we have a fast method of finding random points on E:

1. Choose $x \in K$ at random.

2. Solve $Y^2 = x^3 + ax + b$ for Y. If there are no solutions then go to the previous step.

3. Flip a coin to decide which solution of Y to use.

Of course, it is impossible to choose the point at infinity with this method, and points that lie on the X-axis have a slightly higher probability of been picked than other points. For cryptography this is of no concern since the point of infinity is usually unwanted, and the probability of finishing at a point with zero Y-coordinate is negligible since there are at most three of them. Moreover, it is often unimportant which square root is chosen.

If one insists on choosing all points of $E(K)$ uniformly, one could simply add a step before choosing x. Let $n = \#E(K)$. Then with with $1/n$ probability, choose O or one of the points lying on the X-axis, otherwise proceed with the above algorithm, except in the second step, we also go back to the first step if the only solution is $Y = 0$.

Before attempting to find a square root of a given element $x \in K$, we can check that one actually exists first. When K has prime order, one can compute the Legendre symbol before attempting to square root x. More generally it can be checked that $X^2 - x$ is reducible.

Alternatively, one can omit the check, proceed with a square root algorithm, and compare the square of the output with x: if there is a mismatch then x is not a square after all.

It remains to describe how to take square roots. For a field of prime order one can use the Tonelli-Shanks algorithm to compute square roots [9, 47], which we quote below.

For a general finite field, one must use a more complex algorithm. Perhaps the simplest of these Legendre's method which can be viewed as using the Cantor-Zassenhaus algorithm of Section 5.8 to factor $X^2 - x$. Faster algorithms exist, though sometimes require precomputation [8].

2.3.1 Tonelli-Shanks Algorithm

Suppose we wish to compute $b = \sqrt{a}$ in a field $\mathbb{F}_q$ for some prime q. (These paragraphs should be viewed as self-contained; the notation from earlier paragraphs does not apply here.)

Algorithm 1 (Tonelli-Shanks) Find $b = \sqrt{a}$ in a prime field

```
Find an element $g \in \mathbb{F}_q$ that is not a square.
Since $q$ is odd (unless $q = 2$ in which case square roots are trivial), we may write
  $q - 1 = 2^s t$ for some odd $t$.
$e \leftarrow 0$
for $i \leftarrow 2$ to $s$ do
    if $(ag^{-e})^{(q-1)/2^i} \neq 1$ then
        $e \leftarrow 2^{i-1} + e$
    end if
end for
$h \leftarrow ag^{-e}$
$b \leftarrow g^{e/2} h^{(t+1)/2}$
Return $b$
```

The first step can be accomplished by choosing random elements $g \in \mathbb{F}_q$ until $g^{(q-1)/2} = -1$. Clearly this g can be stored for use in future square root computations in the same field.

We can briefly explain the Tonelli-Shanks algorithm as follows. Observe square roots in a cyclic group of order t where t is odd can be computed by exponentiating by $(t+1)/2$. Then using $\mathbb{F}_q^* \cong \mathbb{Z}_{2^s}^+ \times \mathbb{Z}_t^+$ for some odd t leads to the above.

2.3.2 Hashing to Points

Finding points by choosing an X-coordinate and solving for Y suggests an efficient algorithm for hashing to a point in E. The input is hashed to some $x \in K$, and then a corresponding y is sought. On failure, a new x-coordinate is deterministically generated from x, and again we attempt to solve E for y. Repeating this process as many times as necessary eventually yields a valid point $(x, y) \in E(K)$.

2.4 Point Compression and Reduction

Another implication of taking square roots to find a Y corresponding to a particular X value is that a point (x, y) can be represented by x along with a single bit indicating which solution of y to take. This technique is known as *point compression.*

Alternatively, for some cases we can simply use x alone, which is sometimes called *point reduction.* For example, suppose Alice needs to send Bob a point in some cryptosystem. She sends Bob x, who guesses the solution y. Bob attempts to proceed normally. If the protocol fails e.g. the signature does not verify, then Bob tries again with $-y$. This does not cost much more since the solutions are related by negation, and in Section 6.13 we show how to check both possibilities with only one operation.

Trivially we can take this principle further. Alice can omit k bits of x, and leave Bob to try all 2^k possibilities.

2.5 The Chord-Tangent Law of Composition

We define an operation $+$ on $E(\mathbb{F}_{q^k})$. Let $P = (a, b), Q = (c, d) \in E(\mathbb{F}_{q^k})$ be finite points.

If $P \neq Q$, then it is not hard to show that if $a \neq c$ then the line through P and Q must intersect E at another point (x, y) where $x, y \in \mathbb{F}_{q^k}$. Note that $(x, -y)$ also is a solution of E. Define $P + Q = (x, -y)$ for $a \neq c$. If $a = c$ (in which case we must have $b = -d$), then define $P + Q = O$.

Now suppose $Q = P$. In this case, consider the tangent line going through P. It turns out it must intersect E at another point (x, y) where $x, y \in \mathbb{F}_{q^k}$ unless $b = 0$. Define $P + P = (x, -y)$ for $b \neq 0$. For $b = 0$ define $P + P = O$.

Lastly define $P + O = P$, $O + O = O$.

This operation turns $E(\mathbb{F}_{q^k})$ into a group. The point O is the identity, and the inverse of a point $P = (x, y)$ is the point $-P = (x, -y)$.

As usual, define $0P = O$, $1P = P$, $nP = (n-1)P + P$ for integers $n > 1$ and $nP = -(-n)P$ for integers $n < 0$. This operation is termed *point multiplication*. Point multiplication can be performed efficiently via carefully chosen point additions, in a process that mirrors the repeated squaring technique for exponentiation in finite fields.

Recall the previous chapter used multiplicative group notation to emphasize the connection between discrete log cryptosystems and pairing-based cryptosystems, as we are using elliptic curve groups where one would use the multiplicative group of a finite field.

Mathematicians use additive group notation for the elliptic curve group since the group is Abelian, and we will adhere to this convention in this section.

2.6 Explicit Formulas

Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}_{q^k}$. Let $P_1, P_2 \in E(\mathbb{F}_{q^k})$ and suppose we wish to find $P_3 = P_1 + P_2$. Note the case where at least one of P_1, P_2 is O is trivial, so assume both P_1 and P_2 are finite. Then write $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$.

If $x_1 = x_2$ and $y_1 = -y_2$ then the line V through P_1 and P_2 is vertical and can be given by

$$V : X - x_1 = 0.$$

In this case $P_3 = O$ (and $P_1 = -P_2$).

Otherwise P_3 also must be finite, hence write $P_3 = (x_3, y_3)$. We have two cases. If $P_1 = P_2$ then the tangent line T at P_1 has slope

$$\lambda = \left[\frac{\partial E/\partial X}{\partial E/\partial Y}\right]_{(x_1,y_1)} = \frac{3x_1^2 + a}{2y_1}.$$

and if $P_1 \neq P_2$ then the line L through P_1 and P_2 has slope

$$\lambda = (y_2 - y_1)/(x_2 - x_1).$$

If we set $\mu = y_1 - \lambda x_1$ then the equation of L or T is given by:

$$L, T : Y - (\lambda X + \mu) = 0$$

We can find the x-coordinate of the other point of intersection $(x_3, -y_3)$ by substituting $Y = \lambda X + \mu$ into E. We find

$$E(X, \lambda X + \mu) = X^3 - \lambda^2 X^2 + a_1 X + a_0 = 0$$

for some $a_1, a_0 \in \mathbb{F}_{q^k}$, thus the sum of the roots $x_1 + x_2 + x_3 = \lambda^2$.

This allows us to compute x_3, y_3. Explicitly we have for $P_1 \neq P_2$:

$$\begin{aligned} \lambda &\leftarrow (y_2 - y_1)/(x_2 - x_1) \\ x_3 &\leftarrow \lambda^2 - x_1 - x_2 \\ y_3 &\leftarrow (x_1 - x_3)\lambda - y_1 \end{aligned}$$

and for $P_1 = P_2$:

$$\begin{aligned} \lambda &\leftarrow (3x_1^2 + a)/(2y_1) \\ x_3 &\leftarrow \lambda^2 - 2x_1 \\ y_3 &\leftarrow (x_1 - x_3)\lambda - y_1 \end{aligned}$$

The most expensive step is the division in the computation of λ.

The expressions for the lines L, T, V will be used later in the computation of a pairing.

For the curve $Y^2 = X^3 + X + 6$ over $\mathbb{F}_{19}$, using these formulas, it can be verified for instance that $(0, 5) + (0, 5) = (4, 6)$, and $(0, 5) + (2, 15) = (4, 13)$.

2.7 Elliptic Curve Cryptography

We now possess enough theory to show how elliptic curves may be used in cryptography. Let E be an elliptic curve over a field K. The group operation described above means that every point on E generates a cyclic group G. Then we can use G for cyclic group cryptography provided that its order is prime, that the basic operations, namely group operation, inversion, hashing, are efficient, and that problems such as discrete log are difficult.

The formulas above show that only a few operations in K are required for point addition and negation. We previously saw how to hash to points in $E(K)$. Thus it seems elliptic curve cryptography can replace cryptography in finite fields by using points in $E(K)$ instead of elements of some F^* for some finite field F, and the group operation is point addition instead of modular multiplication. The only obstacle is ensuring that randomly chosen points and hashed points lie in G, and not all of $E(K)$.

Recall that cryptographic schemes in F^* for some finite field F often operate within a subgroup G of a particular order r, so elements chosen at random and hashed to must have order r, or a factor of r. But an element of F^* in general does not have such an order. Thus after using some algorithm to choose or hash to some element $x \in F^*$, to obtain an element of a suitable order one simply exponentiates x by n/r where $n = \#F^*$. On elliptic curves, the construction of a point of order r, or a factor of r, from some given point $P \in E(K)$ can be accomplished similarly by multiplying P by n/r where $n = \#E(K)$.

Let $n = \#E(K)$. Then from Abelian group theory for any prime r dividing n, there exists a point $P \in E(K)$ of order r. and furthermore, if r^2 does not divide n then there is exactly one subgroup G of $E(K)$ of order r.

This suggests the following procedure for implementing any cryptographic scheme based on cyclic groups of prime order:

1. Choose any curve $E(K)$ and somehow work out $n = \#E(K)$.

2. Find a prime r dividing n, such that $r^2 \nmid n$. We shall work in the unique cyclic subgroup $G \subset E(K)$ of points of order r.

3. When a random group element of G is required, first choose a random point of $E(K)$ and then multiply by n/r. Similarly, when hashing to a point of G, first hash to a point in $E(K)$ and then multiply by n/r.

4. Other operations are straightforward: every time a group operation is required, we perform a point addition. To find an inverse of a group element, we negate the y-coordinate of a point. When an exponentiation is called for we carry out a point multiplication

This easily generalizes to squarefree r.

In some applications it does not matter if the group is not cylic. For example, many cryptosystems function equally well in a group isomorphic to $\mathbb{Z}_r^+ \times \mathbb{Z}_r^+$. In these cases we allow r^2 to divide n.

We quote another well-known theorem which implies that we never have more than two copies of $\mathbb{Z}_r^+$ in $E(K)$. Note our most general abstract definition of the pairing in fact permits groups with at least three copies of a cyclic group, even though this theorem shows this is impossible for elliptic curves.

Theorem (See Silverman [62])**.**

$$E(K) = \mathbb{Z}_r^+ \times \mathbb{Z}_s^+$$

for some integers r, s with $r \mid s$.

In the running example, $E : Y^2 = X^3 + X + 6$ over $\mathbb{F}_{19}$, the order of the group is $n = 18 = 3^2 \times 2$, and it so happens that the group is cyclic: it can be checked that $(0, 5)$ is a generator. Thus this curve can be used for cryptography on a cyclic group of order $r = 3$. However, in general, if $r^2|n$ there may be more than one subgroup of order r and the above procedure cannot be followed.

We have yet to describe how to compute $\#E(K)$ for a given curve. Fast algorithms exist for this [9] but it turns out that if a pairing is desired, we must seek out elliptic curves whose orders satisfy various conditions. As a result, instead of choosing a curve first and counting the number of points n and hoping for a large prime factor r of n, we must use families of curves where the size of the group is known in advance and has the requisite properties, a subject of a later chapter.

For now, we exhibit one case where $\#E(K)$ is always easy to determine and furthermore $E(K)$ is always cyclic.

2.8 Singular Elliptic Curves

Using singular elliptic curves is equivalent to conventional cryptography. It can be shown that the set E_{ns} of nonsingular points (all but one) of a singular elliptic curve over a field K also form a group under the chord-tangent law, and $E_{ns} \cong K^*$ or $E_{ns} \cong K^+$ [62, Proposition 2.5]. Furthermore these isomorphisms are efficiently computable in either direction. Note the latter case is useless for cryptographically as discrete log is easy in K^+.

For example, consider the curve

$$E : y^2 = x^3 + x^2$$

This has a singular point at $(0, 0)$. Then the other solutions to this curve (including the point at infinity) form a group $E_{ns}(\mathbb{F}_{q^k})$ for any finite field $\mathbb{F}_{q^k}$.

Furthermore, there is an isomorphism $E_{ns}(\mathbb{F}_{q^k}) \rightarrow \mathbb{F}_{q^k}^*$ given by

$$(x, y) \mapsto \frac{y - x}{y + x}$$

(and maps O to 1) which is efficienty computable in either direction. A little algebra shows the reverse map is

$$z \mapsto \left(x', \frac{1+z}{1-z} x'\right)$$

where $x' = 4z/(1-z)^2$ for $z \neq 1$, and $1 \mapsto O$.

This example has little practical value as point additions are slower than modular multiplications, and keeping both coordinates of a point in memory takes twice as much space. However, it does suggest that cryptography in finite fields can be viewed as a special case of elliptic curve cryptography, and strictly speaking, those that complain that elliptic curve cryptography is controversial, too experimental and untested must qualify their remarks by making exceptions for singular curves!

Needless to say, it is entirely possible that researchers may one day discover special discrete log algorithms that work only for singular curves, in which case we are left only with finite fields for cryptography. Similarly, if a subexponential discrete log algorithm applicable to any elliptic curve were found, then again finite fields would be the only reasonable choice because elliptic curves are less efficient for the same field size.

It is clear that if an efficiently bilinear nondegenerate pairing were found for singular curves then the Decisional Diffie-Hellman problem could be broken in finite fields.

2.9 Security

We have not yet shown any benefits to using elliptic curves instead of finite fields. On the contrary, although a group inversion is almost free, the group operation, hashing, and random element generation are considerably more expensive for the same field.

Their strength derives from the fact that no discrete log algorithm for general elliptic curves that outperforms generic methods has ever been discovered.

The best generic methods are based on the birthday paradox, and have $O(\sqrt{n})$ expected running time, where n is the group order. A group order around 160 bits in length is sufficient to defeat such attacks. In contrast, subexponential discrete log algorithms for finite fields mean that one must use at least 1024-bit finite fields for security.

Hence one may work with fields over six times smaller in length if elliptic curves are used. The increase in speed in the underlying field arithmetic easily compensates for the more complex group operations. Thus choosing elliptic curves instead of finite fields results in savings in both space and time.

The most important attribute of elliptic curves for our purposes is that without them, we cannot construct a cryptographically useful bilinear map. That elliptic curves happen to be more efficient than finite fields is a happy coincidence and makes pairings even more attractive.

2.10 Short Signatures

At this point we can see why the BLS signature scheme introduced in the first chapter features a short signature length. Recall in abstract terms, a signature is a single element of a cyclic group G of prime order r and we need a bilinear map to exist on G.

In Section 4.16 we give examples of pairings where one input group is a subgroup of points on an elliptic curve E over some field $\mathbb{F}_q$ where q is about 160 bits long.

A point has two coordinates, both taking about 160 bits to represent. We can use point reduction, that is, discard the y-coordinate entirely, and represent signatures by their x-coordinate only if we modify verification as follows:

1. Given an x-coordinate of a signature, find any solution y in the curve equation E.

2. If the signature (x, y) does not verify, check if the signature $(x, -y)$ verifies.

3. If the signature still does not verify then reject it.

Thus signatures are elements of $\mathbb{F}_q$ and hence roughly 160 bits in length.

Verification of the signature involves computing $e(P, Q)$ for some points P, Q. By the bilinearity of the pairing, if we have guessed the point Q wrongly, we can obtain the value of $e(P, -Q) = e(P, Q)^{-1}$ by performing an inversion rather than recompute another pairing.

In Section 6.12 we will see we can do better than this by using a technique known as *pairing compression*, and in effect check both cases at once.

Instead of discarding the y-coordinate, we can replace it with a single bit signifying which solution of y to take, a technique known as *point compression* [9, Section IV.4].

Chapter 3

The Weil and Tate Pairings

The Weil and Tate pairings take *r-torsion points* as input, and in the case of the Weil pairing, both inputs are r-torsion points. They can be defined using *rational functions*. They output an element of a finite field that is an rth root of unity.

We shall state facts whose proofs can be found in many textbooks on elliptic curves such as Silverman [62].

3.1 Torsion Points

Let K be a finite field of characteristic q, so that $K = \mathbb{F}_{q^m}$ for some natural number m. Let E be an elliptic curve defined over K. Let $n = \#E(K)$ Suppose $P \in E(K)$ satisfies $rP = O$, so that P has order r or a factor of r. We call P an *r-torsion point*. We denote the set of r-torsion points in $E(K)$ by $E(K)[r]$.

For the curves we shall consider, n and q are always coprime thus r is also coprime to q (since $r \mid n$). The case when r is not coprime to q leads to *the anomalous attack* [9, Section V.3] which breaks discrete log in linear time.

It can be proved that for some integer $k \geq 1$, $E(\mathbb{F}_{q^k})[r]$ contains exactly r^2 points and is in fact the direct product of two cyclic groups of order r, that is, isomorphic to $\mathbb{Z}_r \times \mathbb{Z}_r$. Furthermore, for all $k' \geq k$ we have $E(\mathbb{F}_{q^{k'}})[r] = E(\mathbb{F}_{q^k})[r]$. Roughly speaking, there are no other r-torsion points beyond the first r^2 points found, no matter how many times we extend the field. (That is, if $\bar{K}$ is the algebraic closure of K then $\#E(\bar{K})[r] = r^2$.) We define $E[r] = E(\mathbb{F}_{q^k})[r]$, the set of *$r$-torsion points* of E.

Thus the above isomorphism may be written

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

When $r = 2$ this is easy to see: a point has order 2 if and only if it has a zero y-coordinate. Since E is nonsingular, $x^3 + ax + b = 0$ has three distinct solutions, thus we can always find some field $\mathbb{F}_{q^k}$ where E has four points of order 2: the point O and three points of the form $(\alpha, 0)$ where α is a root of the cubic. Since the line through any two of the finite points is simply the line $Y = 0$, which certainly intersects E at the other finite point, we have $E[2] \cong \mathbb{Z}_2 \times \mathbb{Z}_2$. The proof is much less trivial for general r.

The Weil pairing is a bilinear map takes pairs of elements from $E[r]$ and outputs an rth root of unity in $\mathbb{F}_{q^k}$. The Tate pairing is similar but only the first input is from $E(\mathbb{F}_q)[r]$.

3.2 Rational Functions

We study the behaviour of quotients of polynomials in two variables on points on an elliptic curve. Denote the ring of polynomials in two variables X, Y with coefficients in $\mathbb{F}_{q^k}$ by $\mathbb{F}_{q^k}[X, Y]$.

Let E be an elliptic curve $Y^2 = X^3 + aX + b$ over $\mathbb{F}_{q^k}$. Write $E(X, Y)$ for the polynomial $Y^2 - X^3 - aX - b$. It can be shown that a polynomial f satisfies $f(X, Y) = 0$ whenever $E(X, Y) = 0$ if and only if $f(X, Y)$ is a multiple of $E(X, Y)$.

Thus define $\mathbb{F}_{q^k}[E] = \mathbb{F}_{q^k}[X, Y]_{E(X,Y)}$. In other words we look at all possible polynomials and consider two of them to be equivalent if they take the same values on all points of E. Every element $f(X, Y)$ in $\mathbb{F}_{q^k}[E]$ can be written in the form $f_x(X) + Y f_y(X)$ for polynomials $f_x, f_y \in \mathbb{F}_{q^k}[X]$ because we may replace Y^2 by $X^3 + aX + b$.

Define $\mathbb{F}_{q^k}(E)$ to be the field of fractions of $\mathbb{F}_{q^k}[E]$. Elements of $\mathbb{F}_{q^k}(E)$ can be written in the form $f(X, Y)/g(X, Y)$ where f, g are polynomials in two variables X, Y and g is not a multiple of E.

3.3 Curve Endomorphisms

Given a point P and integer m consider the "*multiplication-by-m*" map $[m]$ given by

$$P \mapsto mP.$$

If P is viewed as a pair of variables (X, Y), this map can be written as $(f(X, Y), g(X, Y))$ for some rational functions f, g. Using the explicit formulas for point addition and doubling given in the previous chapter, we can write down explicit formulas for f and g for any m, but even for relatively small m the expressions are unwieldy.

This is an example of a *curve endomorphism*, because points of $E(\mathbb{F}_{q^k})$ are mapped to points of $E(\mathbb{F}_{q^k})$, and furthermore the map preserves point addition. Another important example is the qth power *Frobenius map* Φ, given by $(X, Y) \mapsto (X^q, Y^q)$. In general one defines q^kth power Frobenius maps but we only define the qth power map, and if we need higher powers k we shall write Φ^k.

We shall also need the *trace map*, which is defined by

$$\operatorname{tr} P = P + \Phi(P) + ... + \Phi^{k-1}(P)$$

which takes points of $E(\mathbb{F}_{q^k})$ to $E(\mathbb{F}_q)$.

We state another well-known theorem from the theory of elliptic curves. Below, $f \circ g$ denotes the map defined by $P \mapsto f(g(P))$.

Theorem (Hasse)**.**

$$\Phi^k \circ \Phi^k - [t] \circ \Phi^k + [q^k] = [0]$$

where $t = q^k + 1 - \#E(\mathbb{F}_{q^k})$.

Since $E[r] \cong \mathbb{Z}_r \times \mathbb{Z}_r$ any curve endomorphism that maps $E[r]$ to $E[r]$ (which is true for Frobenius and multiplication-by-m maps) can be viewed as a 2×2 matrix when restricted to $E[r]$.

In particular for a prime r dividing $\#E(\mathbb{F}_q)$ the eigenvalues of Φ are the solutions to $x^2 - tx + q \pmod r$. The polynomial factorizes as $(x - q)(x - 1)$ thus any eigenvalues must be q or 1. We can describe the corresponding eigenspaces.

Clearly $E(\mathbb{F}_q)[r]$ is a 1-eigenspace, hence if $E[r] \subset E(\mathbb{F}_q)$ then Φ is simply the identity.

Otherwise consider the set $G = \{P \mid \operatorname{tr} P = O\}$.

We know this set contains more than just the point at infinity since for any $P \in E[r] \setminus E(\mathbb{F}_q)[r]$, the point $P - \Phi(P)$ is nontrivial and has trace zero. Straightforward computations show that G is closed under the group laws, and also show that $\Phi(G) = G$. Hence G is an eigenspace.

As G cannot be the 1-eigenspace (which has already been accounted for; $E(\mathbb{F}_q)[r]$ is the

1-eigenspace), G must be the q-eigenspace, which implies $|G| = r$.

This result is important in Section 6.4, so we restate it:

Theorem. *The set of points of trace zero $G = \{P \mid \operatorname{tr} P = O\}$ is a cyclic group of order r, and every $P \in G$ satisfies $\Phi(P) = qP$.*

3.4 Zeroes and Poles

When we study a polynomial $f(X)$, the roots, or zeroes, of $f(X)$ play an important role. Given the zeroes of a polynomial f, we can instantly write down an expression for f that is unique up to a constant multiple. Extending this idea, when considering the quotient of two polynomials $f(X)/g(X)$, we see the zeroes are the roots of f and the poles are the roots of g, and f/g is completely determined up to a constant by its zeroes and poles and their multiplicities. Indeed, suppose the zeroes and poles are $\alpha_1, ..., \alpha_n$ with multiplicites $a_1, ..., a_n$ where α_i is a zero of multiplcity a_i. We consider a zero of negative multiplicity a_i to be a pole of multiplicity $-a_i$. Then we may write $f(X)/g(X)$ immediately by multiplying equations of appropriate vertical lines:

$$f(X)/g(X) = (X - \alpha_i)^{a_i}$$

One can view roots of a polynomial f as places where the curve $Y = f(X)$ intersects with the curve $Y = 0$. We now generalize by replacing $Y = 0$ with an elliptic curve. When studying an element $f(X,Y) \in \mathbb{F}_{q^k}[E]$, we take note of points of intersection between $f(X,Y)$ and E. Again, we call these points *zeroes* of f.

Complications occur due to the point at infinity O. We omit the details since we do not wish to discuss projective geometry, but it turns out that every line $\alpha X + \beta Y + \gamma$ that intersects a given elliptic curve E has a pole at O of some multiplicity, though this is not immediately suggested from its equation. The following two facts are all that we will say on this topic, and they need not be retained, nor even read. Firstly, if a vertical line $X - \alpha$ intersects the curve E then it has a pole at O of multiplicity 2. Secondly, if a non-vertical line $\alpha X + \beta Y + \gamma$ intersects E then it has a pole of multiplicity 3 at O. From now we will ignore the behaviour of rational functions at O.

Recall an element of $\mathbb{F}_{q^k}(E)$ can be written as $f(X,Y)/g(X,Y)$. In this case the zeroes are the zeroes of f and the poles of g, and the poles are the poles of f and the zeroes of

g. Just as expressions of the form $X - \alpha$ can be viewed as building blocks of quotients of polynomials, we view equations of lines $\alpha X + \beta Y + \gamma$ as building blocks of rational functions on elliptic curves. As with quotients of polynomials, knowing the zeroes and poles of a rational function on E allows us to instantly write its equation, up to multiplication by a constant. We also build the function by multiplying together equations of lines, but need more than just vertical ones.

In what follows, by abuse of notation we often refer to a single rational function f when in fact we are considering a whole family of rational functions cf for some nonzero constant c. To add to the confusion, having arbitrarily chosen a representative of a family, we will sometimes subsequently insist on using that same representative. The reasons for doing so will become clear.

3.5 Divisors

Divisors are the standard way to notate zeroes and poles and their multiplicities, or orders. If we wish to record zeroes and poles $P_1, ..., P_n$ of orders $a_1, ..., a_n$ (where P_i is a zero of order a_i if $a_i > 0$ and is a pole of order $-a_i$ otherwise), then we may write:

$$D = a_1\langle P_1\rangle + ... + a_n\langle P_n\rangle.$$

With this notation, if we add the divisors of two functions together, then the resulting divisor reflects the zeroes and poles of the product of the two functions.

For this chapter we break with tradition and use multiplicative notation for divisors instead. In addition, we will omit the number of poles or zeroes at O, as we are ignoring the behaviour of rational functions at that point. For divisors of rational functions, we actually do not lose any information because if we cared, we could always recover the order of (O) simply by negating of the sum of the orders of the finite points.

Let D be the divisor with zeroes and poles $P_1, ..., P_n$ of multiplicities $a_1, ..., a_n$. We shall write D as

$$D = (P_1)^{a_1}...(P_n)^{a_n}.$$

Our unusual notation has several advantages. Using additive group notation for the elliptic curve and multiplicative notation for divisors emphasizes the difference between them and reduces the likelihood of confusing the two. When rational functions are multiplied together, their divisors can be multiplied together, which is arguably more natural than adding them. Leaving out the (O) term allows us to focus less on bookkeeping and more on the task at hand, and the order of (O) can be readily calculated anyway.

We use (f) to denote the divisor of f, thus in our notation for any rational functions f, g we have $(f)(g) = (fg)$.

On an elliptic curve, in our unorthodox notation:

Lines: a line L through the points P, Q intersects E at a third point $-P-Q$ by the chord-tangent composition law, and so has divisor

$$(L) = (P)(Q)(-P-Q).$$

Tangents: when $Q = P$ we have a tangent line T at P with divisor

$$(T) = (P)^2(-2P).$$

Verticals: when $Q = -P$, we have a vertical line V through the point P with divisor

$$(V) = (P)(-P).$$

3.6 The Weil Pairing

Earlier papers on pairing-based cryptosystems advocated the Weil pairing. We shall see that the related Tate pairing is preferable for our purposes. Nonetheless we first describe the Weil pairing for historical reasons, and also because its simpler description serves well as an introduction. The inputs to the Tate pairing come from different groups, and the definition involves quotient groups, which can obscure the basic idea. However, the actual computation of the Tate pairing is simpler.

Let E be an elliptic curve containing n points over a field $\mathbb{F}_q$. Let G be a cyclic subgroup of $E(\mathbb{F}_q)$ of order r with r, q coprime. Let k be the smallest positive integer such that $E(\mathbb{F}_{q^k})$ contains all of $E[r]$.

We define the Weil pairing $f : E[r] \times E[r] \to \mathbb{F}_{q^k}$ as follows.

For a pair of points $P, Q \in E[r]$, choose any $R, S \in E[\mathbb{F}_{q^k}]$ such that $S \neq R, P+R, P+R-Q, R-Q$. Let f_P be a rational function with divisor $(f_P) = (P+R)^r/(R)^r$, (In other words, we have r zeroes at $P + R$ and r poles at R. In standard notation this is $r\langle P + R\rangle - r\langle R\rangle$.) and similarly let f_Q be a rational function with divisor $(f_Q) = (Q + S)^r/(S)^r$.

Define

$$f(P, Q) = \frac{f_P(Q + S)/f_P(S)}{f_Q(P + R)/f_Q(R)}$$

It can be shown that this value is independent of the choices for R, S. The conditions for choosing R, S ensure we never evaluate f_P nor f_Q at a zero or pole. Also both f_P, f_Q are only unique up to a constant, but this has no effect on the final answer since quotients of these functions are computed and the constants cancel out. This is an example of the grey area mentioned earlier: at first we are free to choose any function as long as they have certain zeroes and poles, but we cannot alter it once we have made a choice.

Finding explicit expressions for these functions is infeasible for cryptographically useful pairings. Instead, we will soon describe an algorithm, known as *Miller's algorithm* [48, 49] that evaluates these functions in a lazy fashion at the required points. This algorithm resembles a typical exponentiation routine that performs repeating squaring.

One can prove that

1. $f(aP, bQ) = f(P, Q)^{ab}$ for all $P, Q \in E[r]$ and all integers a, b.

2. $f(P, P) = 1$ for all $P \in E[r]$.

3. $f(P, Q) = 1$ for all $P \in E[r]$ if and only if $Q = O$.

4. $f(P, Q) = 1$ for all $Q \in E[r]$ if and only if $P = O$.

5. $f(P, Q) = f(Q, P)^{-1}$ for all $P, Q \in E[r]$.

6. $f(\Phi(P), \Phi(Q)) = f(P, Q)^q$ for all $P, Q \in E[r]$. where Φ denotes the Frobenius map.

(The last property is usually stated more generally: $f(\alpha(P), \alpha(Q)) = f(P, Q)^{\deg \alpha}$ for any nonzero endomorphism α.)

The second to last property, known as antisymmetry, is a property the Tate pairing does not have. However it does not yet have any use in cryptography.

3.6.1 Weil Reciprocity

Many of the previous statements, including the fact that the pairing is well-defined (in other words, the choices of R, S above do not matter) and several of the listed properties, can be proved using a fact known as Weil reciprocity [13].

First, we define evaluation of a rational function f at a divisor $\prod_P P^{a_P}$ by

$$f(\prod_P (P)^{a_P}) = \prod_P f(P)^{a_P}$$

Weil reciprocity states that for any rational functions f, g,

$$f((g)) = g((f)),$$

that is, evaluating g at the divisor of f produces the same result as evaluating f at the divsor of g.

Intuitively, Weil reciprocity can be thought of as a generalization of the much simpler version of this statement for polynomials: given two polynomials p, q, the result of evaluating p at the zeroes of q and multiplying has the same absolute value as evaluating q at the zeroes of p and multiplying.

3.7 The Tate Pairing

Let E be an elliptic curve containing n points over a field $\mathbb{F}_q$. Let G be a cyclic subgroup of $E(\mathbb{F}_q)$ of order r with r, q coprime. Let k be the smallest positive integer such that $r \mid q^k - 1$. For brevity write $K = \mathbb{F}_{q^k}$. An equivalent characterization is that $K = \mathbb{F}_{q^k}$ is the smallest extension of $\mathbb{F}_q$ containing the rth roots of unity.

The Tate (or Tate-Lichtenbaum) pairing

$$e : E[r] \cap E(K) \times E(K)/rE(K) \to K^*/K^{*r}$$

is defined as follows.

Let f_P be a rational function with divisor $(f_P) = (P)^r$. Choose an $R \in E(K)$ such that $R \neq P, P - Q, O, -Q$. The define

$$f(P, Q) = f_P(Q + R)/f_P(R)$$

It can be shown that the above value is independent of the choice of R, and:

1. $f(aP, bQ) = e(P, Q)^{ab}$ for all P, Q, a, b.

2. $f(P, Q) = 1$ for all P if and only if $Q = O$.

3. $f(P, Q) = 1$ for all Q if and only if $P = O$.

4. $f(\Phi(P), \Phi(Q)) = f(P, Q)^q$ for all $P, Q \in E[r]$, where Φ denotes the Frobenius map.

The output of this pairing is some $x \in K^*$ that represents the coset xK^{*r}. To standardize the coset representative, we exponentiate the output of the Tate pairing by $(q^k - 1)/r$, which can take a substantial amount of time.

On the positive side, the second input to the Tate pairing is also a coset representative. This means it can be any point of $E(K)$ and may be of any order. (For example, if the order of a point Q is not a multiple of r then Q represents the coset $O + rE(K)$. Otherwise Q represents some nonidentity element.)

In contrast, the Weil pairing requires that the second input Q satisfies $rQ = O$.

3.8 Merging Theory with Practice

We have now described the Weil and Tate pairings. It remains to show how they fit the abstract definitions given in Chapter 1. Let E be some elliptic curve defined over $\mathbb{F}_q$ and let G be some cyclic subgroup of points in $E(\mathbb{F}_q)$ of order r. Suppose the embedding degree of G is $k > 1$, which is usually the case in practice.

Then by defining $G_1 = G$, $G_2 = E[r]$, and G_T to be the rth roots of unity in $\mathbb{F}_{q^k}$, the Weil pairing satisfies the abstract definition of a bilinear map given in Section 1.8. Similarly, the Tate pairing fits the definition.

With high probability, a random point $Q \in G_2$, does not lie in G_1, does not have trace zero, and generates a subgroup H in G_2 of order r. Then if we replace G_2 with H, we have now satisfied the asymmetric pairing definition of Section 1.7 because the trace map nontrivially maps G_2 to G_1.

We see now why we cannot hash into G_2 in general if it is required to be cyclic. In the case of the Weil pairing, we can easily hash to a point of $E[r]$, but it is not known how to hash to a point of some designated cyclic subgroup of $E[r]$ of order r that is not G. For the Tate pairing we have a similar problem. There is one exception that we discuss in Section 6.4.2: we can hash to the trace zero group. (In fact, the proof in the last section contains a statement that suggests how we might do this.)

Cryptosystems that require the symmetric pairings of Section 1.4 and require input groups to which strings can be hashed can only be implemented using supersingular curves and distortion maps as in Section 6.6.

Some kinds of symmetric pairings can be obtained without resorting to supersingular curves. We could treat G_2 as the first and second input groups and apply the map from G_2 to G_1 on the first input before a pairing computation. However, one must relinquish the ability to hash, or the requirement that the input groups be cyclic, or a combination of the two. Additionally, one must compute more in larger fields for the same level of security. The details can be found in Section 6.4.2.

3.9 Miller's Algorithm

For the Weil pairing we need to evaluate some rational function f_P with divisor $(P + R)^r/(R)^r$. For the Tate pairing we need to evaluate some rational function f_P with divisor $(P)^r$. In both cases, we can do this using Miller's algorithm [48, 49].

For any points U, V, let $L_{U,V}(X, Y)$ be an equation for a line through U and V, let $T_U(X, Y)$ be an equation for the tangent through U. let $V_U(X, Y)$ be an equation for a vertical line through U (e.g. $X - x$, where $U = (x, y)$).

Since $T_U = L_{U,U}$ and $V_U = L_{U,-U}$, defining T_U and V_U is unnecessary but they aid comprehension. Note if $U = -U$ then T_U is the same as V_U.

3.9.1 The Weil Pairing

For an integer k, let f_k be a rational function with divisor

$$(f_k) = \frac{(P + R)^k}{(R)^k(kP)}.$$

Observe $f_r = f_P$, since $rR = O$. It can be checked that

$$\left(f_a \cdot f_b \cdot \frac{L_{aP,bP}}{V_{(a+b)P}}\right) = (f_{a+b})$$

via a simple computation on the divisors:

$$\frac{(P + R)^a}{(R)^a(aP)} \cdot \frac{(P + R)^b}{(R)^b(bP)} \cdot \frac{(aP)(bP)(-(a + b)P)}{(a + b)P(-(a + b)P)} = \frac{(P + R)^{a+b}}{(R)^{a+b}((a + b)P)}.$$

This leads to the formula:

$$\left(f_k^2 \cdot \frac{T_{kP}}{V_{2kP}}\right) = (f_{2k})$$

A straightforward manipulation of divisors also shows $(V_{P+R}/L_{P,R}) = (f_1)$.

Thus we may compute $f_P(Q) = f_r(Q)$ using Algorithm 2, where the binary representation of r is $r_t...r_0$.

Algorithm 2 Miller's algorithm for Weil pairing. $x = f_P(Q)$

```
 1: $x_1 \leftarrow V_{P+R}(Q)/L_{P,R}(Q)$
 2: $x \leftarrow x_1$
 3: $Z \leftarrow P$
 4: for $i \leftarrow t-1, ..., 0$ do
 5:    $x \leftarrow x^2 \cdot T_Z(Q)/V_{2Z}(Q)$
 6:    $Z \leftarrow 2Z$
 7:    if $r_i = 1$ then
 8:       $x \leftarrow x \cdot x_1 \cdot L_{Z,P}(Q)/V_{Z+P}(Q)$
 9:       $Z \leftarrow Z + P$
10:    end if
11: end for
```

On termination, we have $x = f_P(Q)$ (and $Z = rP = O$).

Throughout the algorithm we may multipy the equation of any line L, T, V by an arbitrary constant, thus more precisely, this algorithm really computes $f_P(Q)$ for one possible choice of f_P. In the Weil pairing, this causes no problems provided the same equations are chosen the second time this algorithm is run, so that the exact same f_P is evaluated.

Note that

$$\left(f_k \cdot \frac{L_{P+R,kP}}{L_{R,(k+1)P}}\right) = (f_{k+1}).$$

so we could replace step 8 with $x \leftarrow x \cdot L_{P+R,Z}(Q)/L_{R,Z+P}(Q)$ and avoid computing and storing x_1, but unless r has very low Hamming weight this is not a good trade because vertical lines are much cheaper to compute than general lines.

3.9.2 The Tate Pairing

We now consider the Tate pairing. Recall we wish to compute $f_P(Q)$ where f_P has divisor $(P)^r$. We can view our current situation as a special case of the above with $R = O$. Thus define the intermediate functions f_k by

$$(f_k) = (P)^k/(kP).$$

We have $(f_r) = (f_P)$. The following identities can be obtained by simplifying earlier formulas using $R = O$, but it is easy enough to check them directly.

For example, we can show

$$(f_k) = \left(\prod_{i=1}^{k-1} \frac{L_{iP}}{V_{(i+1)P}} \right)$$

via

$$(f_k) = \frac{(P)(P)}{(2P)} \cdot \frac{(P)(2P)}{(3P)} \cdots \frac{(P)((k-1)P)}{(kP)}.$$

We could compute $f_r(Q)$ using this formula, but this is clearly impractical for large r.

We find

$$(f_{2k}) = (f_k^2 T_{kP}/V_{2kP})$$

which can be shown with direct calculation:

$$\frac{(P)^{2k}}{(2kP)} = \frac{(P)^{2k}}{(kP)^2} \cdot \frac{(kP)^2(-2kP)}{(2kP)(-2kP)}$$

and similarly we can show

$$(f_{k+1}) = (f_k L_{kP,P}/V_{(k+1)P}).$$

leading to the following algorithm that computes $f_P(Q)$ given points P, Q (where P has order r). Let the binary representation of r be $r_t ... r_0$.

Algorithm 3 Miller's algorithm for Tate pairing. $x = f_P(Q)$

```
$x \leftarrow 1$
$Z \leftarrow P$
for $i \leftarrow t-1, ..., 0$ do
    $x \leftarrow x^2 \cdot T_Z(Q)/V_{2Z}(Q)$
    $Z \leftarrow 2Z$
    if $r_i = 1$ then
        $x \leftarrow x \cdot L_{Z,P}(Q)/V_{Z+P}(Q)$
        $Z \leftarrow Z + P$
    end if
end for
```

When the algorithm finishes we have $x = f_r(Q)$ (and $Z = rP = O$). Note this algorithm can be viewed as the previous algorithm with $x_1 = 1$.

3.9.3 Intermediate Poles and Zeroes

We have in fact glossed over some complications. The intermediate functions evaluated during Miller's algorithm have zeroes and poles that eventually cancel out. If we were dealing with algebraic expressions for the rational function then we could manipulate them to remove these zeroes and poles at some stage. But since we are evaluating the function iteratively, this cannot be done, and attempting to evaluate an intermediate function at a zero or pole will cause problems.

Recall we have an almost unrestricted choice for the points R, S in the Weil pairing and also for the point R in the Tate pairing. To use Miller's algorithm, we choose them to avoid the zeroes and poles of the intermediate functions. We can simply take points at random since there are only $O(t)$ bad choices.

Alternatively we may ensure their coordinates do not lie in the field that the functions are defined in, but rather in some field extension, so that in the case of a Weil pairing, $P+R$, R, $Q+S$, S cannot possibly be a zero or pole in the intermediate functions though we must still avoid choices such as $R=S$, and in the case of a Tate pairing, neither R nor $Q+R$ can be a zero or pole.

We shall see that under some conditions there is a much simpler fix for the Tate pairing.

3.10 Worked Examples

Consider the curve $E: Y^2 = X^3 + X$ over $\mathbb{F}_{59}$. This has 60 points. Let G be the subgroup of size $r=5$. One generator of G is $P=(25,30)$:

$$G = E(\mathbb{F}_{59})[5] = \{P=(25,30), 2P=(35,31), 3P=(35,28), 4P=(25,29), 5P=O\}$$

Feeding any two of these points as inputs to the Weil pairing is pointless as the output will be 1 since they are linearly dependent. We must first find other 5-torsion points. The Tate pairing also does not function at the moment, because $5 \nmid 59-1$,

But $5 \mid 59^2-1$, and we will see that $E(\mathbb{F}_{59^2})$ contains all 25 points of $E[5]$, so in the field extension $\mathbb{F}_{59^2}$ both the Tate pairing and Weil pairing exist and are nontrivial. Note -1 is a quadratic nonresidue in $\mathbb{F}_{59}$ so we may write $\mathbb{F}_{59^2} = \mathbb{F}_{59}[i]$ where $i=\sqrt{-1}$.

It turns out that the *distortion map* given by $\phi(x,y) = (-x, iy)$ enables us to write down all of $E[5]$ easily. This map ϕ takes G to another subgroup G' in $E[5]$:

$$G' = \{Q = (-25, 30i), ..., 4Q = (-25, 29i), 5Q = O\}$$

where $Q = \phi(P)$, and every element of $E[5]$ can be written as a sum of an element in G and an element in G'.

3.10.1 The Weil Pairing

Let us first consider the Weil pairing, which we denote by f. We walk through the computation of $f((25, 30), (-25, 30i))$. The two inputs points are linearly independent thus the output is not 1.

Computing $f(P, Q)$ entails evaluating several rational functions at various points that depend on two picked points R and S. These points R and S must be selected so that we never encounter a zero or pole while evaluating these functions.

In practice, the probability that randomly chosen R, S are unsuitable is negligible. This is not the case for our toy example, but it turns out that choosing $R = (40, 54)$ and $S = (48 + 55i, 28 + 51i)$ will work. Note that R, S do not need to lie in $E[5]$.

We describe in detail how Miller's algorithm arrives at $f_P(Q + S) = f_5(Q + S)$:

1. We compute $f_1 = V_{P+R}/L_{P,R}$. Using the point addition formulas we find $P + R = (6, 24)$ and $Q + S = (19 + 17i, 46 + 18i)$. Then from the explicit formulas given earlier we have

$$V_{P+R} : X + 53$$

$$L_{P,R} : 35X + 15Y + 32$$

   Evaluating both equations at $Q + S$ and taking their quotient gives $x_1 = f_1(Q + S) = 25 + 31i$.

2. We now enter the main loop. Firstly, $f_2 = f_1^2 T_P/V_{2P}$ where

$$T_P : 12X + Y + 24$$

$$V_{2P} : X + 24$$

   . Then we assign $x \leftarrow x_1^2 T_P(Q + S)/V_{2P}(Q + S)$ which turns out to be $(18 + 16i)(3 +$

$45i)/(43 + 17i) = 33 + 22i$.

3. Since the second bit of $5 = 101_2$ is zero, the condition for the if statement is false and the next iteration begins. Now $f_4 = f_2^2 T_2/V_4$, where

$$T_{2P} : 53X + Y + 2$$

$$V_{4P} : X + 34$$

and we compute $x \leftarrow x^2 T_{2P}(Q+S)/V_{4P}(Q+S)$. It can be checked $x = 58 + 23i$ now.

4. The last bit is 1, so this time we do execute the contents of the if block, namely $x =\leftarrow x \cdot x_1 \cdot L_{4P,P}(Q + S)/V_{5P}(Q + S)$. Since $5P = O$ we discard the denominator in this case, and since $P = -4P$, $L_{4P,P} = V_P$ which is given by

$$V_P : X + 34$$

thus $x = (58 + 23i)(25 + 31i)(53 + 17i) = 18 + 2i$, and the algorithm stops as we have exhausted all the bits.

Thus we have found that $f_P(Q + S) = 18 + 2i$, and yet we never derived an algebraic expression for the rational function f_P. We can repeat the above to find $f_P(S) = 10 + 30i$, but a better way is to compute $f_P(S)$ concurrently to avoid repeating the same operations. In other words, we simultaneously calculate $f_P(Q+S)$ and $f_P(S)$, and as each equation for a certain line is found, we evaluate at $Q + S$ and also at S.

The computation of $f_Q(P + R)$ and $f_Q(R)$ is similar (note since Q does not lie in $\mathbb{F}_q$ the L, T, V equations will involve imaginary numbers), and it can be verified that $f_Q(P+R) = 27 + 25i$ and $f_Q(R) = 10 + 41i$. At last we have

$$f(P, Q) = \frac{f_P(Q + S)/f_P(S)}{f_Q(P + R)/f_Q(R)} = 46 + 56i.$$

Thankfully $(46 + 56i)^5 = 1$ as expected.

3.10.2 The Tate Pairing

Let f denote the Tate pairing. Again we walk through the computation of $f((25, 30), (-25, 30i))$.

In a later chapter we shall show that in this case, for any R we have $f_P(Q) = f_P(Q + R)/f_P(R)$ thus we only walk through the calculation of $f_P(Q)$. Nonetheless, calculating $f_P(Q + R)$ and $f_P(R)$ can be done in a similar fashion.

1. $f_2(Q) = f_1(Q)^2 T_P(Q)/V_{2P}(Q)$. Note $f_1 = 1$, unlike the Weil pairing. From the formulas given before we can find equations describing the tangent $T_P = 0$ at P and the vertical line $V_{2P} = 0$ at $2P$:

   $$T_1 : 12X + Y + 24$$

   $$V_2 : X - 35$$

   (We may write $X + 34$ instead, it doesn't matter.)

   As stated before there are other choices for the equations. If we were computing $f_P(Q + R)$ and $f_P(R)$ we would need to use the same choices both times. However, for our current case it can be shown there are no restrictions save that the coefficients of the functions lie in $\mathbb{F}_q$.

   From $T_1(34, 30i) = 12 \times 34 + 30i + 24 = 19 + 30i$ and $V_2(34, 30i) = 34 - 35 = 58$ we obtain $f_2(Q) = (19 + 30i)/58 = 40 + 29i$.

2. $f_4(Q) = f_2(Q)^2 T_2(Q)/V_4(Q)$ where

   $$T_2 : 53X + Y + 2$$

   $$V_4 : X - 25$$

   We have $T_2(34, 30i) = 53 \times 34 + 30i + 2 = 34 + 30i$ and $V_4(34, 30i) = 34 - 25 = 9$ Thus $f_4 = (40 + 29i)^2 \times (34 + 30i)/9 = 31 + 32i$.

3. $f_5 = f_4 L_4(Q)/V_5(Q)$ We discard V_5 since $5P = O$, and L_4, the line between P and $4P = -P$ is in fact a vertical line:

   $$L_4 : X - 25$$

   Since $L_4(34, 30i) = 34 - 25 = 9$, we have $f_5 = 9(31 + 32i) = 43 + 52i$.

   In a larger example we would come across lines L that are not vertical (nor tangents).

4. $f_P(Q)$ can be considered the output of the Tate pairing, but it is a coset representative, and we standardize the representative by raising it to the power of $(q^k - 1)/r$:

$$f_P(Q)^{(q^k-1)/r} = (43 + 52i)^{(59^2-1)/5} = 42 + 40i$$

As expected, $(42 + 40i)^5 = 1$.

3.10.3 Remarks

We point out a few peculiarities in the above examples that will have significance later.

In both examples, degenerate cases appear in the last iteration of the main loop.

The vertical line V_P appears in the second-last iteration as well as last iteration, effectively canceling itself out. After some thought it can be seen this will always happen for odd r.

Whenever a vertical line is evaluated in the Tate pairing example, the result is an element of $\mathbb{F}_q$.

The Weil pairing requires Miller's algorithm to be run twice, once for $f_P(Q+S)/f_P(S)$ and once for $f_Q(P+R)/f_Q(R)$. In the computation of f_P every line equation has coefficients lying in $\mathbb{F}_q$, where as to compute the equations of the lines, tangents and verticals of f_Q we had to perform arithmetic in the larger field $\mathbb{F}_q[i]$.

The Tate pairing needs but one call to Miller's algorithm, but requires an exponentiation by $(q^k - 1)/r$. Like the computation of f_P in the Weil pairing, to determine the equations of the lines, tangents and verticals during Miller's algorithm only requires arithmetic in the base field $\mathbb{F}_q$.

3.11 Restricting Inputs

We shall have much to say on this topic in Section 6.6, but the above example serves as a good introduction. Recall we have the Weil pairing f on the curve $E : Y^2 = X^3 + X$ over some field $\mathbb{F}_q$ where $q = 3 \bmod 4$. We have a distortion map $\phi(x, y) = (-x, iy)$, and we have a group G of r-torsion points in $E(\mathbb{F}_q)$ for some r.

In real applications, we wrap the Weil pairing f in another map $e : G \times G \to \mathbb{F}_{q^2}$:

$$e(U, V) = f(U, \phi(V)).$$

We use e, not f, as our pairing. There are at least two reasons for doing this.

The input points to e have coordinates in $\mathbb{F}_q$ and lie on the same curve (that is, e is symmetrical) and

Both input groups of e are cyclic. Thus the construction of cryptosystems and their security proofs are simpler. We could work with the Weil pairing directly but then both input groups would be $E[r]$, which is not cyclic.

As $U, \phi(V)$ are always linearly independent, the wrapper function e guarantees a nontrivial pairing output whenever U and V are nontrivial. This is not the case with the Weil pairing, where one must be mindful of linearly dependent inputs.

Lastly, the input points have coordinates in the field $\mathbb{F}_q$ which is smaller and hence more efficient to compute on than $\mathbb{F}_{q^2}$.

Similarly, if f now denotes the Tate pairing, we wrap f in the map $e : G \times G \to \mathbb{F}_{q^2}$:

$$e(U, V) = f(U, \phi(V))$$

and use e instead of f in our cryptosystems. Note in the above formula, $\phi(V)$ is actually a coset representative of $\phi(V) + rG$.

3.12 The Shipsey-Stange Algorithm

There is another procedure for computing Tate pairings that takes a completely different approach. We quote the algorithm as described by Stange [63] but omit explanations of the mathematical underpinnings.

Let $E : Y^2 = X^3 + aX + b$ be an elliptic curve over a field K. Let $P = (x, y) \in E(K)[r]$, $Q = (x_2, y_2) \in E(K)$. (Q represents the coset $Q + rE(K)$.) Define constants

$$A = (x - x_2)^{-1}, B = \left((2x + x_2)(x - x_2)^2 - (y + y_2)^2\right)^{-1}, C = (2y)^{-1}$$

Define the sequence c_k by

$$c_{-2} = -2y, c_{-1} = -1, c_0 = 0, c_1 = 1, c_2 = 2y,$$

$$\begin{aligned}
c_3 &= 3x^4 + 6ax^2 + 12bx - a^2, \\
c_4 &= 4y(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - 8b^2 - a^3), \\
c_{2k-1} &= c_{k+1}c_{k-1}^3 - c_{k-2}c_k^3, \\
c_{2k} &= C(c_k c_{k+2} c_{k-1}^2 - c_k c_{k-2} c_{k+1}^2).
\end{aligned}$$

and the sequence d_k by

$$d_0 = 1, d_1 = 1, d_2 = (2x + x_2) - \left(\frac{y_2 - y}{x_2 - x}\right)^2$$

$$\begin{array}{rclclcl}
d_{2k-1} & = & & d_{k+1}d_{k-1}c_{k-1}^2 & - & d_k^2 c_{k-2}c_k & \\
d_{2k} & = & & d_{k+1}d_{k-1}c_k^2 & - & d_k^2 c_{k-1}c_{k+1} & \\
d_{2k+1} & = & A(& d_{k+1}d_{k-1}c_{k+1}^2 & - & d_k^2 c_k c_{k+2} &) \\
d_{2k+2} & = & B(& d_{k+1}d_{k-1}c_{k+2}^2 & - & d_k^2 c_{k+1}c_{k+3} &)
\end{array}$$

Then the Tate pairing $e : E[r] \cap E(K) \times E(K)/rE(K) \to K^*/K^{*r}$ is given by

$$e(P, Q) = d_{r+1}/c_{r+1}.$$

We describe a repeated-squaring-like algorithm that uses these sequences to compute the Tate pairing. Let $r_t ... r_0$ be the binary representation of r.

Algorithm 4 (Shipsey-Stange) Tate pairing via elliptic nets. Outputs $e(P, Q)$.

```
$k \leftarrow 1$
Compute $A, B, C$ and $c_{k-3}, ..., c_{k+4}$ and $d_{k-1}, d_k, d_{k+1}$ ($c_{-2}, ..., c_5$ and $d_0, d_1, d_2$).
for $j = t - 1$ to 0 do
  if $r_j = 0$ then
    (Double) Compute $c_{2k-3}, ..., c_{2k+4}$, and $d_{2k-1}, d_{2k}, d_{2k+1}$ (using the above recurrence relations)
    $k \leftarrow 2k$
  else $\{r_j = 1\}$
    (Double and add) Compute $c_{2k-2}, ..., c_{2k+5}$ and $d_{2k}, d_{2k+1}, d_{2k+2}$ (using the above recurrence relations)
    $k \leftarrow 2k + 1$
  end if
end for
return $d_{r+1}/c_{r+1}$
```

3.12.1 Example

We take the same curve and points from the other worked examples, that is $E : Y^2 = X^3+X$ over $\mathbb{F}_{59}$, $r = 5$, $P = (25, 30)$, $Q = (-25, 30i)$.

1. We find the constants are $A = 13, B = 25 + 55i, C = 1$, start of the sequence $c_{-2}, ..., c_5 = 56, 58, 0, 1, 3, 31, 37, 0$ and also $d_0, d_1, d_2 = 1, 1, 43 + 31i$. We have $k = 1$. Note c_5 was computed using $c_5 = c_2^3 c_4 - c_3^3$.
2. We enter the main loop. As the second bit of the binary representation is zero, we use the recurrence relations to find $c_{-1}, ..., c_6$. As this is one of the first few iterations, there is much overlap (which could be optimized away in a real application), and the only new value is $c_6 = 53$. Similarly we compute d_1, d_2, d_3 and the only new value is $d_3 = 2 + 43i$. We have $k = 2$.
3. The last bit is one, hence this time we find $c_2, ..., c_9$. Once again there is overlap and the only new values are $c_7 = 32, c_8 = 37, c_9 = 11$. We also compute d_4, d_5, d_6 which turns out to be $55 + 13i, 51 + 26i, 26 + 4i$. We exit the loop with $k = 5 = r$.
4. The algorithm returns $d_6/c_6 = 37 + 42i$.

In practice we power the result by $(p^2 - 1)/r = 696$ (to standardize the coset representative), giving the final answer of $42 + 40i$.

3.12.2 Remarks

The above example is artificially small, and we do not in fact need to bother computing c_7, c_8, c_9. Also, many sequence values are calculated more than once. For realistic sizes, this only happens in the first few iterations, and the time taken by unnecssary computations is a small fraction of the total running time. Nonetheless, checks to avoid superfluous operations can be easily implemented for a slight efficiency gain.

There are many common subexpressions in the above formulas which would be exploited in an implementation but were omitted above for clarity. Setting

$$S_k = c_k^2, T_k = c_{k-1}c_{k+1}, U_k = d_k^2, V_k = d_{k-1}d_{k+1}$$

yields

$$\begin{aligned}
c_{2k-1} &= T_k S_{k-1} - T_{k-1} S_k, \\
c_{2k} &= C(T_{k+1} S_{k-1} - T_{k-1} S_{k+1}) \\
d_{2k-1} &= V_k S_{k-1} - U_k T_{k-1} \\
d_{2k} &= V_k S_k - U_k T_k \\
d_{2k+1} &= A(V_k S_{k+1} - U_k T_{k+1}) \\
d_{2k+2} &= B(V_k S_{k+2} - U_k T_{k+2})
\end{aligned}$$

though we note sometimes it is better to compute c_{2k} via

$$c_{2k} = C c_k (c_{k+2} c_{k-1}^2 - c_{k-2} c_{k+1}^2).$$

as in some cases, certain T_k subexpressions are only used once.

Chapter 4

Curve Selection

Let E be an elliptic curve defined over a finite field K. Let $P \in E(K)$ be a point of prime order r. Let $G = \langle P \rangle$. We intend to build cryptosystems that operate in G. Hence we perform field arithmetic in K during cryptographic operations, while the security of the resulting systems partly depends on the size of r, which is a factor of $\#E(K)$. We focus on the cases when $\#E(K)$ is a small multiple of r. Relaxing this condition permits us to use a wider variety of pairings, and we discuss one such family later.

Recall the size of K is roughly the size of $\#E(K)$, thus if both r and $\#E(K)$ are about l-bits in length, we obtain a pairing-based cryptosystem with l-bit security in G whose running times depend on operations on l-bit numbers. A pairing with this feature is needed by BLS signatures to achieve short signatures.

Happily, it turns out that among the pairings we cover are symmetric pairings with cyclic input groups that can be hashed to, and pairings where r can be specified, both of which can be useful features.

Recall the Tate pairing must be computed in some field extension L of K that contains the rth roots of unity (so r divides $|L| - 1$). On the other hand, the Weil pairing is defined on $E[r]$, the r-torsion points of E, which lie in $E(L')$ for some field extension L' of K.

In order for the Tate pairing to be efficiently computable, operations must be efficient in L. Similarly, for the Weil pairing to be efficiently computable, operations must be efficient in L'. Thus we seek fields L or L' that are small enough so that field operations are still fast.

It turns out that such a field L' will always contain the nth roots of unity, thus we always have $L \subseteq L'$. We will define the *embedding degree* which can be thought of as measuring the size of L compared to the field K. The converse is not true: it is possible for the subset inclusion to be strict. In other words, for a field L to contain the rth roots of unity but at the same time have $E[r]$ contained in $E(L')$ and not $E(L)$, for some L' strictly bigger than L. However, as Balasubramanian and Koblitz have shown [1], the converse is almost true because this event occurs only in special circumstances. We present a slighly different proof:

4.1 The Embedding Degree

Theorem (Balasubramanian-Koblitz). *Let E be an elliptic curve defined over $\mathbb{F}_q$. Let G be a subgroup of $E(\mathbb{F}_q)$ of order r with $r \nmid q-1$. Then for any positive integer k, $E(\mathbb{F}_{q^k})$ contains all r^2 points of order r if and only if $r \mid q^k - 1$.*

Proof. It is well-known that if $E(\mathbb{F}_{q^k})$ contains $E[r]$ then $r \mid q^k - 1$, even without assuming $r \mid N$ or $r \nmid q - 1$ [1].

Conversely, suppose $k > 1$ and $r \mid q^k - 1$. Let Φ denote the Frobenius map. Consider the subgroup T of $E[r]$ consisting of all points of trace zero, that is

$$T = \{Q \in E[r] : \operatorname{tr} Q = Q + \Phi(Q) + ... + \Phi^{k-1}(Q) = O\}$$

(The group T may be explicitly constructed by using the map $P \mapsto P - \Phi(P)$ on points of $E(\mathbb{F}_{q^k})$.) Now we have $\Phi(T) = T$, and also T is not contained in $E(\mathbb{F}_q)$ by assumption.

Hence T is an eigenspace of Φ, but not the 1-eigenspace. Since the eigenvalues of Φ must be 1 and q, we see that T must be the q-eigenspace of Φ and hence

$$\Phi^k(Q) = q^k Q = Q$$

since $r \mid q^k - 1$. Thus T, like $E(\mathbb{F}_q)$ is fixed under Φ^k, and since these groups are linearly independent they generate all of $E[r]$, implying that all of $E[r]$ is fixed under Φ^k. Hence $E[r] \subset E(\mathbb{F}_{q^k})$. □

Definition. *Let E be an elliptic curve defined over $K = \mathbb{F}_q$. Let $G \subseteq E(\mathbb{F}_q)$ be a cyclic group of order r. Let k be the smallest positive integer such that $r \mid q^k - 1$. Then we say that the embedding degree of G is k.*

As already mentioned, we mostly focus on curves where r and $\#E(K)$ are as close as possible, and we can abuse the terminology a little when talking about embedding degrees. When we speak of an embedding degree of a curve $E(\mathbb{F}_q)$, we mean the embedding degree of the subgroup of $E(\mathbb{F}_q)$ of order r where r is the largest prime dividing $\#E(\mathbb{F}_q)$ (or in some applications, the largest factor of $\#E(\mathbb{F}_q)$ that cannot be efficiently factored).

There is a related subtlety we must mention. Let $P \in E(K)$ be a point of order r. Let $G = \langle P \rangle$. Let $n = \#E(\mathbb{F}_q)$. We shall often determine the embedding degree of G by finding the smallest integer k such that $n \mid q^k - 1$ (a process that does not involve r is any way). If r is a proper factor of n, then it could be that for some $j < k$, although $n \nmid q^j - 1$, we have $r \mid q^j - 1$, that is, $\mathbb{F}_{q^j}$ contains all the rth roots of unity and this subfield of $\mathbb{F}_{q^k}$ suffices for the Tate pairing computation on G.

However, in practice, when we do use such a group G and r is a large prime factor of $\#E(\mathbb{F}_q)$ it is almost always the case that $\mathbb{F}_{q^k}$ is the smallest field extension that allows the computation of the pairing. In other words, in general, for a cyclic subgroup $G \subset \#E(\mathbb{F}_q)$ the embedding degree k of $E(\mathbb{F}_q)$ may not be the smallest positive integer such that $r \mid q^k - 1$, but in practice, when r is a large prime factor of $\#E(\mathbb{F}_q)$, the probability that k is not the integer we seek is negligible. Below, we shall make statements about embedding degrees ignoring the unlikely event that the real embedding degree could be smaller. (Of course, for small toy examples one must be wary of the existence of some $j < k$ with $r \mid q^j - 1$.)

At this point, our goal seems to be minimizing this embedding degree, so that L is as close as possible to K, and if possible, to have $K = L$ and have all computations in a small a field as possible. Unfortunately, things are not quite this simple, because of pairing security issues discussed in Section 4.3.

4.2 Weil and Tate Pairing Comparison

The above theorem shows that if the embedding degree k is greater than 1, then both the Tate pairing and the Weil pairing may be computed by performing field arithmetic in $\mathbb{F}_{q^k}$.

On the other hand, if the embedding degree k is 1, the Tate pairing is always computable in $\mathbb{F}_q$, but the Weil pairing sometimes cannot be, as we are about to show. Usually this is not significant because we shall see $k > 1$ is preferred. Nonetheless, we shall find the Tate pairing to be the best choice in any case for other reasons.

Consider the curve $E : Y^2 = X^3 + X + 6$ which has 18 points over $\mathbb{F}_{19}$, which we borrow from a publication by Balasubramanian and Koblitz [1]. The point $R = (0, 5)$ generates a cyclic group of order 18, so the point $P = 6R = (12, 13)$ generates a cyclic group of order 3.

We see $E[3]$ cannot be contained in $\langle R\rangle = E(\mathbb{F}_{19})$ since $E[3]$ is not cyclic, thus the Weil pairing cannot be computed. (In fact, we must move to $\mathbb{F}_{19^3}$ to do so.)

In contrast, $3 \mid 19 - 1$ hence the Tate pairing can be computed in $\mathbb{F}_{19}$. In fact, in one program by the author, it was found that $e(P, R + 3G) = 11$, where e denotes the Tate pairing. Recall that pairings are only unique up to a constant, so different implementations may give different results. Since the cube roots of unity in $\mathbb{F}_{19}$ are $1, 7, 11$, we expect 7 or 11.

4.2.1 Composite Group Orders

We can also use this curve to exhibit interesting behaviour that can occur with pairings on cyclic groups with composite order mentioned in Section 1.8.

It can be checked that $e(R, R) = 17$, a ninth root of unity. If we were to strictly adhere to the definition of the Tate pairing the second input should be written as $R + 18G$ since it is supposed to be a coset, but this is superfluous since $18G$ is the trivial group. Thus we have a pairing that maps two groups of order 18 to a group that has order 9. This implies, for example, $e(R, 9R) = 1$.

In contrast, if the order of R were some prime r, nondegeneracy implies that $e(P, Q)$ must also be of order r for any $P, Q \in \langle R\rangle \setminus \{O\}$. In fact, $e(P, Q) = 1$ would imply at least one of P, Q is O in this case. Because most pairing-based cryptosystems use groups of prime order, one may become so accustomed to facts like these that one may erroneously assume them to be true when working with groups of composite order.

4.3 Pairing Security

Consider a nonsingular elliptic curve E over finie field $\mathbb{F}_q$ containing some cyclic group G of order r with embedding degree k. In some sense, the pairing is a double-edged sword. Though it bestows additional cryptographically useful properties upon cyclic groups, it also allows one to break the discrete log problem in $E(\mathbb{F}_q)$ by first breaking the discrete log problem in $\mathbb{F}_{q^k}$, using an algorithm known as the MOV or Frey-Ruck attack [46, 32] which we now describe.

Suppose we are given $P, nP \in G$ and asked to recover n. Let $e : G \times G \to \mathbb{F}_{q^k}$

be a bilinear, nondegenerate map, such as the Weil or Tate pairing. By bilinearity and nondegeneracy,

$$\mathrm{Dlog}(e(P,P), e(P,nP)) = \mathrm{Dlog}(e(P,P), e(P,P)^n) = n = \mathrm{Dlog}(P, nP)$$

for $n \in \{0, ..., r-1\}$.

Since $e(P,P), e(P,nP) \in \mathbb{F}_{q^k}$, we can use a method like index calculus to recover n, the solution to the discrete log problem in the elliptic curve group G.

Interestingly, these attacks were the first application of pairings in cryptography to appear in the literature, predating all publications describing constructive uses of the pairing. To avoid them, we must ensure q^k is large enough so that finite field discrete logarithm algorithms such as index calculus are infeasible in $\mathbb{F}_{q^k}$.

It may seem we have contradicted ourselves, for in Section 2.9 we touted elliptic curves over finite fields because we claimed they are only susceptible to generic discrete log attacks, yet we have just seen we must guard against index calculus.

But although it is true that a pairing exists for every nonsingular elliptic curve, in general the embedding degree k is on the order of q [1], that is, so large that it is futile to mount a MOV or Frey-Ruck attack on a typical elliptic curve because $\mathbb{F}_{q^k}$ will be too big for any practical calculations. We shun curves with small k in standard elliptic curve cryptography and thus can ignore these attacks, which is why they were not mentioned before.

The newfound utility of pairings has meant we now actively seek out curves with low embedding degrees so that pairings are efficiently computable, hence we must take these attacks into account.

Are there any other attacks that concern us? It turns out the curves used in pairing-based cryptography are special in other senses. They are supersingular, or have complex multiplication. Fortunately, no specific attacks for either case is known.

4.4 Lower Bounds on Field and Group Sizes

Constructing a pairing is a delicate balancing act. Using the notation of the previous section, $\mathbb{F}_q$ must be large enough so that $E(\mathbb{F}_q)$ can foil generic discrete log attacks, while $\mathbb{F}_{q^k}$ must be large enough to resist finite field discrete log attacks. At the same time, $\mathbb{F}_q$ and $\mathbb{F}_{q^k}$ should be as small as possible to minimize time and space usage. More precisely:

1. r must be a large enough prime so that generic discrete logarithm attacks in a group of order r are ineffective. Since $q \approx \#E(\mathbb{F}_q)$, this places a similar lower bound on q.

2. q ought to be as small as possible, so that computations in $\mathbb{F}_q$ are as fast as possible.

3. q^k must be large enough so that finite field discrete logarithm attacks in $\mathbb{F}_{q^k}$ are ineffective. Also q should not have low Hamming weight [64], nor be a power of a small prime [24].

4. q^k must be small enough so that operations in $\mathbb{F}_{q^k}$ are efficient. All other things being equal, q^k should be small as possible so that operations are as fast as possible.

Observe the first three statements are true for any cryptographically useful elliptic curve, not just for pairing-based cryptography. The last condition, requiring $\mathbb{F}_{q^k}$ to be small enough to compute on, is responsible for much of the difficulty in pairing-based cryptography research, as finding curves with small k is nontrivial.

Currently it is acceptable to have an 160-bit r. As for q^k, 1024 bits is adequate for many applications, and calculations in fields of this size can certainly be performed. Ideally we have $r \approx \#E(\mathbb{F}_q) \approx q$ and hence q is also about 160 bits. This gives an embedding degree k around $1024/160 = 6.4$.

We later describe how to construct curves of embedding degree 6, 10 and 12. For the near future, embedding degree 6 should be reasonable if used along with an r that is at least 170 bits long, while embedding degree 10 and 12 curves which currently produce excessively large finite fields may be more desirable as time passes and index calculus improvements are discovered.

4.5 Approaches to Finding Curves

A randomly-chosen elliptic curve will have a large embedding degree, which endows it with resistance to the MOV and Frey-Ruck attacks, but also renders it useless for pairing-based cryptography. We must take one of two approaches to find curves suitable for pairings:

1. Supersingular curves are guaranteed to have a small embedding degree, and are easy to construct. They have been completely classified. Operations on some of them can be highly optimized.

2. By carefully tailoring the complex multiplication method of constructing elliptic curves, we can produce curves of a certain embedding degree.

We note that supersingular hyperelliptic curves have also been considered [33, 56], and as well as the hyperelliptic equivalent of the complex multiplication method [29].

Below we survey some types of curves used in practice for pairing-based cryptography. To make them easier to refer to, we label them with letters. The labelling is arbitrary, though we try to arrange them in the order they appeared in cryptography publications.

4.6 Supersingular Curves

There are six families of supersingular curves, with embedding degree at most six [46]. Let $q = p^m$. Let t denote the trace of Frobenius. Then the six classes can be described as follows.

1. $k = 2$: $t = 0$ and $E(\mathbb{F}_q) \cong \mathbb{Z}_{q+1}$.
2. $k = 2$: $t = 0$ and $E(\mathbb{F}_q) \cong \mathbb{Z}_{(q+1)/2} \oplus \mathbb{Z}_2$ and $q = 3 \pmod 4$.
3. $k = 3$: $t^2 = q$ and m is even.
4. $k = 4$: $t^2 = 2q$ and $p = 2$ and m is odd.
5. $k = 6$: $t^2 = 3q$ and $p = 3$ and m is odd.
6. $k = 1$: $t^2 = 4q$ and m is even.

Some curves in the first two classes are easy to describe, and are extremely useful, as it is easy to find curves containing a subgroup of any desired order.

The $k = 4$ case requires $p = 2$. Many specialized optimizations exist for operations in characteristic two fields, but unfortunately at the same time specialized discrete logarithm attacks exist [24], and we must use bigger fields to compensate for this.

The $k = 6$ case requires $p = 3$. Again, we may apply a host of specialized optimizations, but we must also be wary of low-characteristic discrete logarithm algorithms.

4.7 Type A Curves

Let q be a prime satisfying $q = 3 \pmod 4$. Let E be the curve $y^2 = x^3 + ax$ for any a. In Section 5.9 we will find $a = -3$ is a good choice (though $a = \pm 1$ is better for the first step of the Shipsey-Stange algorithm). Then $E(\mathbb{F}_q)$ is supersingular, $\#E(\mathbb{F}_q) = q + 1$, and $\#E(\mathbb{F}_{q^2}) = (q + 1)^2$ [30, §3.2]. Furthermore, for any odd r dividing $q + 1$ we have that $G = E(\mathbb{F}_q)[r]$ is cyclic and has embedding degree $k = 2$.

Note -1 is a quadratic nonresidue in $\mathbb{F}_q$ since $q = 3 \pmod 4$. Let i be a square root of -1. Then $\mathbb{F}_q[i]$ is a degree 2 extension of $\mathbb{F}_q$. Consider the following map, sometimes referred to as a *distortion map* [65]:

$$\Psi(x, y) = (-x, iy)$$

Then Ψ maps points of $E(\mathbb{F}_q)$ to points of $E(\mathbb{F}_{q^2}) \setminus E(\mathbb{F}_q)$. Thus if f denotes the Tate or Weil pairing, then defining $e : G \times G \to \mathbb{F}_{q^2}$ by

$$e(P, Q) = f(P, \Psi(Q))$$

gives a bilinear nondegenerate map.

Setup for this type of pairing for a cryptosystem can be done as follows.

1. An order r is chosen, large enough to avoid generic discrete logarithm attacks. Other properties may be desired. For some cryptosystems r is an RSA modulus. As we will see, careful choices of r will speed up Miller's algorithm substantially.

2. Recall we require finite field discrete logarithm attacks on $\mathbb{F}_{q^2}$ to be impractical. Thus we randomly generate h where where h is a multiple of four and sufficiently large to guarantee $(hr)^2$ is big enough to resist finite field attacks. For example, if r is 160 bits long, and we want q^2 about 1024 bits long, then h must be about 352 bits long.

3. Next it is checked that $q = hr - 1$ is prime. We have $q = 3 \bmod 4$ by choice of h. If q is not prime, we go back to the previous step and choose another h.

4. For some cryptosystems problems may arise if $r \mid h$, but this occurs with negligible probability for realistic parameters. Nonetheless, when toy examples are constructed, this may need to be checked.

If h is constrained to be a multiple of 3 as well, then cube roots are extremely easy to compute in $\mathbb{F}_q$: for all $x \in \mathbb{F}_q$ we see $x^{-(q-2)/3}$ is the cube root of x. Observe cube roots are unique since each element is a cube. This may be desirable in some situations, and hardly affects the setup algorithm.

4.8 Type B Curves

Let q be a prime satisfying $q = 2 \bmod 3$. Let E be the curve $y^2 = x^3 + b$ for any b. Typically $b = \pm 1$ is chosen.

Then as before, $E(\mathbb{F}_q)$ is supersingular, $\#E(\mathbb{F}_q) = q + 1$, and $\#E(\mathbb{F}_{q^2}) = (q + 1)^2$ [30, §3.2]. Again, for any odd r dividing $q + 1$ we have that $G = E(\mathbb{F}_q)[r]$ is cyclic and has embedding degree $k = 2$.

Consider the distortion map $\Psi : E(\mathbb{F}_q) \to E(\mathbb{F}_{q^2})$ given by

$$\Psi(x, y) \mapsto (\zeta x, y)$$

where ζ is a primitive cube root of unity.

Then Ψ maps points of $E(\mathbb{F}_q)$ to points of $E(\mathbb{F}_{q^2}) \setminus E(\mathbb{F}_q)$. Thus if f denotes the Tate or Weil pairing, then defining $e : G \times G \to \mathbb{F}_{q^2}$ by

$$e(P, Q) = f(P, \Psi(Q))$$

gives a bilinear nondegenerate map.

We shall see cube roots are easy to find since $q = 2 \pmod 3$. For this particular curve, this means we may quickly generate points from a y-coordinate, yielding simple and efficient random point generation and hashing-to-point routines. Additionally, points can be represented by their y-coordinate alone since there is a unique x for each value of y.

Type B have one drawback which we discuss in Section 6.4.2. Unlike the previous type, there is a certain optimization that only applies if we forgo symmetry of the pairing. One must choose between symmetry and efficiency in this case.

4.9 Other Embedding Degree 2 Curves

Type A and B curves require a prime $q > 3$ satisfying $q = 2 \bmod 3$ or $q = 3 \bmod 4$.

We note the remaining case $q = 1 \bmod 12$ can also lead to supersingular embedding degree 2 curves [30, §3.2] but we do not bother with it here. The construction is more involved as it uses the CM method of Section 4.11, and these pairings are usually less desirable as no distortion maps are available and the resulting curves are harder to optimize.

4.10 Type C Curves

Define the curves $E^+ : y^2 = x^3 + 2x + 1$ over $\mathbb{F}_{3^l}$ and $E^- : y^2 = x^3 + 2x - 1$, also over $\mathbb{F}_{3^l}$. Unlike all the other curves we consider, we are working in a low characteristic field. Thus these pairings are susceptible to certain discrete log attacks due to Coppersmith [24] and are perhaps better avoided. We describe them here for completeness.

One can show

$$\#E^+(\mathbb{F}_{3^l}) = \begin{cases} 3^l + 1 + 3^{(l+1)/2} \text{ when } l = \pm 1 \bmod 12 \\ 3^l + 1 - 3^{(l+1)/2} \text{ when } l = \pm 5 \bmod 12 \end{cases}$$

and $\#E^-(\mathbb{F}_{3^l}) + \#E^+(\mathbb{F}_{3^l}) = 2(3^l + 1)$, that is

$$\#E^-(\mathbb{F}_{3^l}) = \begin{cases} 3^l + 1 - 3^{(l+1)/2} \text{ when } l = \pm 1 \bmod 12 \\ 3^l + 1 + 3^{(l+1)/2} \text{ when } l = \pm 5 \bmod 12 \end{cases}$$

and it is easily checked that both types of curve have embedding degree 6 (that is, the order divides $3^{6l} - 1$ in all cases).

For an $l = \pm 1, \pm 5$ and a curve $E : y^2 = x^3 + 2x \pm 1$ over $\mathbb{F}_{3^l}$ let t be a root of $t^3 + 2t \pm 2 = 0$, and let i be a square root of -1. These both exist in $\mathbb{F}_{3^{6l}}$. Define the distortion map Ψ by

$$\Psi(x, y) = (-x + t, iy)$$

where x, y are elements of $\mathbb{F}_{3^{6l}}$. Then Ψ maps points of $E(\mathbb{F}_{3^l})$ to points of $E(\mathbb{F}_{3^{6l}})$.

We arrive at a curve selection algorithm as described by Boneh, Lynn and Shacham [17]:

1. Choose $l = \pm 1, \pm 5 \bmod 12$. To avoid algorithms known as *Weil descent attacks* [34, 35], l should not have any small prime factors (say $3, 5, 7$).

2. Compute $3^l + l \pm 3^{(l+1)/2}$ and check if either has a large prime factor r.

3. If so, set E to the corresponding curve equation.

Curve	l	Field Size $\lceil \lg_2 q \rceil$	Group Size $\lceil \lg_2 r \rceil$	Finite Field Security $\lceil \lg_2 q^6 \rceil$
E^-	79	126	126	752
E^+	97	154	151	923
E^+	121	192	155	1151
E^+	149	237	220	1417
E^+	163	259	256	1551
E^-	163	259	259	1551
E^+	167	265	262	1589

Table 4.1: Some Type C pairings [17].

Table 4.1 shows the first seven cryptographically useful curves found by this algorithm. Evidently they are few and far between. Note the third entry has $l = 11^2$. Although this is a fairly low prime, there has been some evidence suggesting that Weil descent attacks are still ineffectual for this case [26].

4.11 Complex Multiplication

Blake, Seroussi and Smart outline the complex multiplication, or CM, method and provide a brief explanation and many references [9]. We only quote the algorithm itself. Suppose we have integers D, V, q, t satisfying the *CM equation*

$$DV^2 = 4q - t^2$$

such that q is prime and we require $D > 0$, no square of an odd prime divides D and $D = 0, 3 \bmod 4$. The conditions on D are necessary because $-D$ represents a *fundamental discriminant*, the formal definition of which is beyond our scope.

For every such D, there exists a polynomial $H_D(x) \in \mathbb{Z}[x]$ called the Hilbert class field polynomial, the computation of which will be described in Section 4.19. View this Hilbert polynomial $H_D(x)$ as a polynomial in $\mathbb{F}_q[x]$, and let $j \in \mathbb{F}_q$ be any root. For now assume $j \neq 0, 1728$, and set $k = j/(1728 - j)$. Then consider the elliptic curves given by

$$E : y^2 = x^3 + 3kc^2x + 2kc^3$$

for any nonzero $c \in \mathbb{F}_q$.

Set $c = 1$. Then E has order $q + t + 1$ or $q - t + 1$. Generate a random point P of E and check if $(q - t + 1)P = O$. If not, then the order must be $q + t + 1$, and if we set c to be some quadratic nonresidue in $\mathbb{F}_q$ the curve will have order $q - t + 1$. In Section 6.1 we find that the two cases are *quadratic twists* of the same curve.

For $j = 0$ the curve has the form $y^2 = x^3 + k$ for some k. For $j = 1728$ the curve has the form $y^2 = x^3 + kx$ for some k. In these cases one can try different values of k until a curve with the correct order is found. In Section 6.17 we enumerate all possible orders that could arise, and learn that these are caused by the existence of *cubic, quartic and sextic twists.*

Thus given a solution the above equation, we may easily write down a curve of order $q + t - 1$ or $q - t - 1$.

4.12 Type D Curves

Miyaji et al. describe a method for constructing ordinary elliptic curves with embedding degree 3, 4 or 6 [50]. Moreover, they show that in a sense, there are no other parametrizations that lead to curves with these embedding degrees.

We first examine the embedding degree six case, which is the most useful. The embedding degree three and four cases may also be useful in special circumstances and we briefly describe them.

Consider the polynomials $q = x^2 + 1, t = \pm x + 1$. It can be checked that $q(x) + 1 - t(x) \mid q(x)^6 - 1$.

The CM equation becomes $DV^2 = 3x^2 \pm 2x + 3$. If we make the substitution $U = 3x \pm 1$ then we have the *generalized Pell equation*, or a *Pell-type equation*, the solution of which is given in Section 4.17.

$$U^2 - 3DV^2 = -8$$

We can now give a procedure for constructing embedding degree 6 curves.

1. Choose D and solve
$$U^2 - 3DV^2 = -8$$

2. We have $U = \pm 1 \pmod 3$. Set $x = (-1 \pm U)/3$

3. Check that $q = x^2 + 1$ is prime

4. Recall $t = \pm x + 1$. Check that $q - t + 1 = q \mp x$ has a large prime factor r. Ideally r is also prime.

5. Check that r does not divide $q^j = 1$ for any positive integer $j < k$. This can be omitted in practice as it is extremely unlikely this will occur.

6. Use the CM method to construct a curve E with order $q \mp x$.

The resulting curve E will have embedding degree 6.

Though not how the method was discovered, we can explain the MNT algorithm via cyclotomic polynomials $\Phi_k(x)$. Suppose $t = x + 1$. Recall $\Phi_6(x) = x^2 - x + 1$, thus $q = x + \Phi_6(x)$. Then the order n of the resulting curve E is $n = \Phi_6(x)$, thus

$$q^6 - 1 = x^6 - 1 \bmod n$$

Recall $\Phi_6(x) \mid x^6 - 1$, hence n must also divide $q^6 - 1$ giving an embedding degree of 6.

Now suppose $t = -x + 1$. Recall $\Phi_3(x) = x^2 + x + 1$, thus $q = -x + \Phi_3(x)$. The order n of E is $n = \Phi_3(x)$, hence

$$q^6 - 1 = (-x)^6 - 1 = x^6 - 1 \bmod n$$

Since $\Phi_3(x) \mid x^6 - 1$ the embedding degree of E is 6. Of course $\Phi_3(x) \mid x^3 - 1$ but $x^3 \neq (-x)^3$ in general thus the embedding degree is not 3 (with high probability).

For embedding degree $k = 4$, we may also use cyclotomic polynomials. Recall $\Phi_4(x) = x^2 + 1$. Then set $q = \pm x + \Phi_4(x)$ so that $t = \pm x + 1$. Since $n = \Phi_4(x)$ we have

$$q^4 - x = (\pm x)^4 - 1 \bmod n$$

thus $n \mid q^4 - 1$. When the minus sign is chosen and x replaced by $x + 1$ we match the notation of Miyaji et al.[50].

For $k = 3$ Miyaji et al. give $q = 3x^2 - 1$, $t = -1 \pm 3x$ where x is even.

Scott and Barreto discovered a procedure that finds more curves of these types than the original MNT algorithm [60] which can be viewed as a generalization of a method due to Barreto, Lynn and Scott to be described in Section 4.20.

We quote their algorithm without proof. The CM method is used to construct curves from the algorithm's output. The embedding degree k must be chosen to be 3, 4 or 6. The choices $h_{max} = 4$ and $D_{max} \approx 10^8$ or so are reasonable.

Algorithm 5 Scott-Barreto generalization of MNT method.

```
for $h \leftarrow 1$ to $h_{max}$ do
   for $m \leftarrow 1$ to $4h - 1$ do
      for all fundamental discriminants $D \leq D_{max}$ do
         solve $DV^2 = 4h\Phi_k(x)/m - (x-1)^2$ for $x, V$ (generalized Pell equation)
         $r \leftarrow \Phi_k(x)/m$
         $n \leftarrow hr$
         $q \leftarrow n + x$
         if $q, r$ are both primes then
            output $q, r, h, D$
         end if
      end for
   end for
end for
```

Scott and Barreto also concisely state how to transform the CM equation into a generalized Pell equation:

If $k = 3$, set $a \leftarrow 2h + m$.

If $k = 4$, set $a \leftarrow m$.

If $k = 6$, set $a \leftarrow -2h + m$.

Note these three statements can be condensed into $a \leftarrow (-2\lfloor k/2 \rfloor + 4)h + m$.

Set $b \leftarrow 4h - m$. Then the variable substitution defined by $x = (y - a)/b$ transforms the CM equation into the generalized Pell equation

$$y^2 - (mbD)V^2 = (a^2 + b^2).$$

After solving this equation, we must have $b|y - a$ otherwise x will not be an integer.

4.13 Type E Curves

A cursory inspection of the CM method reveals that curves of embedding degree 1 are easily constructed. For example, Let $t = 2$, $D = 7$. Let r be a positive integer and suppose $q = 28r^2h^2 + 1$ is prime for some h. Then the CM equation

$$7V^2 = 4(28(rh)^2 + 1) - 4$$

is satisfied when we take $V = 4rh$.

As $H_7(x) = x + 3375$, if we take $k = -3375/(1728 - 3375)$ in $\mathbb{F}_q$ then the curve

$$y^2 = x^3 + 3kx + 2k$$

has order $q - 1$ or $q + 3$. If it has order $q + 3$, choose some quadratic nonresidue $c \in \mathbb{F}_q$, and use the following curve instead:

$$y^2 = x^3 + 3kc^2x + 2kc^3$$

This is the quadratic twist of Section 6.1.

Note $r \mid q - 1$ hence the Tate pairing on this curve can be computed in $\mathbb{F}_q$. For most applications we choose r to be prime but composite orders can be chosen instead.

We can do better by using a result due to Koblitz and Menezes [43]. If $q = n^2 + 1$ is a prime for some n, then define the elliptic curve E over $\mathbb{F}_q$ by

$$E : y^2 = x^3 - x$$

if $4 \mid n$ and

$$E : y^2 = x^3 - 4x$$

otherwise (that is $n = 2 \bmod 4$).

Then $E(\mathbb{F}_q) = \mathbb{Z}_n^+ \times \mathbb{Z}_n^+$ and the map $\Psi : E(\mathbb{F}_q) \to E(\mathbb{F}_q)$ defined by

$$(x, y) \mapsto (-x, ny)$$

is a distortion map.

4.14 Type F Curves

By considering cyclotomic polynomials, Barreto and Naehrig discovered a parametrization yielding embedding degree 12 curves.

Let $q(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$. Let $t(x) = 6x^2 + 1$. Then if $D = 3$, the CM equation always has the solution $V = 6x^2 + 4x + 1$. Furthermore, it turns out $q(x) + 1 - t(x) \mid q(x)^{12} - 1$. This suggests the following algorithm to generate curves:

1. Pick an integer x of a desired magnitude. It may be negative.

2. Check if $q(x)$ is prime.

3. Check if $n = q(x) - t(x) + 1$ has a large prime factor r. (Ideally it should be prime.)

4. Try different values of k until a random point of $y^2 = x^3 + k$ has order n

Barreto and Naehrig recommend the last step be done as follows: starting from $k = 1$, keep incrementing k until $k + 1$ is a quadratic residue and $n(1, g) = O$, where g is a square root of $k + 1$.

4.15 Type G Curves

Freeman unified many approaches for finding curves in a single framework [28], and also showed how to construct curves with embedding degree 10, with r and q around the same size. Freeman also notes that it seems unlikely that curves with embedding degrees other than the ones described in this chapter can be constructed using these techniques.

We summarize an algorithm due to Freeman and Scott:

Set $q(x) = 25x^4 + 25x^3 + 25x^2 + 10x + 3$ and $t(x) = 10x^2 + 5x + 3$.

1. Choose D such that D is squarefree and $D = 43, 67 \bmod 120$.

2. Find solutions (u, v) to the equation $u^2 - 15Dv^2 = -20$.

3. We have $u = \pm 5 \bmod 15$. Set $x = (-5 \pm u)/15$.

4. Check $q(x)$ is prime.

5. Check $n = q(x) - t(x) + 1$ has a large prime factor r.

6. Construct the corresponding curve E using the CM method.

It can be shown that $n \mid q^{10} - 1$ thus the resulting curve has embedding degree 10.

Type	Supersingular	k	Min Input Size $\lceil \lg_2 q \rceil$	Min Output Size $\lceil \lg_2 q^k \rceil$
A	yes	2	512	1024
B	yes	2	512	1024
C	yes	$6(\approx 3.53)$	≈ 305	≈ 1080
D	no	6	171	1026
E	no	1	1024	1024
F	no	12	160	1920
G	no	10	160	1600

Table 4.2: Pairing comparison.

4.16 Comparing Pairings

Assuming that 160 bits are sufficient to defeat generic discrete log attacks and that 1024 bits are sufficient to defeat index calculus, we can use the embedding degree k to determine a lower bound for q. The results are shown in Table 4.2. In all cases, for the input we would use a subgroup of the elliptic curve group with order r around 160 bits.

The exact efficacy of the Coppersmith attack on characteristic 3 fields of cryptographically-useful sizes has yet to be thoroughly investigated. One conservative estimate by Page, Smart and Vercauteren [52] suggests that the low characteristic effectively changes the embedding degree k from 6 to about $5.6 \log 2/\log 3 \approx 3.53$, suggesting the approximate numbers given in the table. Incidentally, they correspond to an E^- type C curve with $l = 193$ (which does not appear in Table 4.1).

Of course, size is not the only factor. If one must have a symmetric pairing, one is limited to using A, B, C or E pairings.

If speed is the top priority, then A or B pairings should be used. If symmetry is not needed, then B pairings are the better choice because they also allow faster random point generation and point compression.

When asymmetric pairings are required, for example when DDH must be intractable in at least one of the input groups, one must use D, F, or G pairings. Depending on the needed assumptions, one must also select the second input group carefully as in Section 6.4.2.

If a particular group order r is required, one must use A, B or E pairings.

If input size length is important, type D pairings are a good choice. Type F and G pairings allow only slightly shorter inputs, but are slower. On the other hand, they guard against future improvements to finite field discrete log algorithms.

4.17 Pell Equations

For completeness we describe how to solve the Pell-type equations that arise when searching for some of the above pairing types. First we must learn how to solve a standard Pell equation.

A *Pell equation* is an equation of the form

$$x^2 - Dy^2 = 1$$

where D, x, y are integers and D is not a square. We solve such an equation by examining the continued fraction expansion of $\sqrt{D}$. Recall a continued fraction expansion of a real number x is obtained by finding an integer a_0 and positive integers $a_1, a_2, ...$ such that

$$x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + ...}}$$

which we also denote by $x = [a_0, a_1, a_2, ...]$. Consulting any text on basic number theory reveals that this sequence of integers can be found by computing the following.

$$\begin{aligned} P_0 &= 0 \\ Q_0 &= 1 \\ a_0 &= \left\lfloor \sqrt{D} \right\rfloor \\ P_1 &= a_0 \\ Q_1 &= D - a_0^2 \\ a_n &= \left\lfloor \tfrac{a_0 + P_n}{Q_n} \right\rfloor \\ P_n &= a_{n-1} Q_{n-1} - P_{n-1} \\ Q_n &= \tfrac{D - P_n^2}{Q_{n-1}} \end{aligned}$$

One can show for some k we must have $a_{k+1} = 2a_0$, and after this point the a_n sequence begins repeating. That is $\sqrt{D} = [a_0, a_1, ..., a_{k+1}, a_1, ..., a_{k+1}, ...]$.

The convergents are given by

$$p_0 = a_0, p_1 = a_0 a_1 + 1, p_n = a_n p_{n-1} + p_{n-2}$$

and

$$q_0 = 1, q_1 = a_1, q_n = a_n q_{n-1} + q_{n-2}.$$

These satisfy

$$p_n^2 - Dq_n^2 = (-1)^{n+1} Q_{n+1}.$$

It turns out that $(x, y) = (p_k, q_k)$ is the smallest positive integer equation of the Pell equation for odd k, and $(x, y) = (p_{2k+1}, q_{2k+1})$ is smallest for even k. Denote this minimal positive solution by (t, u). Then all positive solutions (x, y) to the Pell equation can be found via

$$x + y\sqrt{D} = (t + u\sqrt{D})^n$$

for all positive integers n. We never need the negative solutions, but these are trivial to find from the positive solutions in any event.

4.18 Generalized Pell Equations

Now suppose we are to solve the *generalized Pell equation*

$$x^2 - Dy^2 = N$$

where D is not a square, x, y are integers.

When $N^2 < D$ we first solve the Pell equation

$$x^2 - Dy^2 = 1$$

using the above method, that is, computing convergents p_n, q_n until the minimal positive solution is found. However, while doing so, we check if $p_n^2 - Dq_n^2 = N/f^2$ for some positive integer f. If so, then we append (fp_n, fq_n) to the list of solutions of the generalized Pell equation.

If no such convergents are found by the time we have reached the minimal positive solution for the Pell equation, then the generalized Pell equation has no solution.

Otherwise let (t, u) be the minimal positive solution of the above Pell equation. Then for each (r, s) on the list of solutions we have a family of solutions (x, y) given by

$$(x + y\sqrt{D}) = (r + s\sqrt{D})(t + u\sqrt{D})^n$$

(for all positive integers n). These families account for all positive integer solutions to the generalized Pell equation.

When $N^2 \geq D$ there are possibly other fundamental solutions to the generalized Pell equation we must add to the list before generating families of solutions. We can use brute force to find them if the numbers are small enough.

For positive N set $L_1 = 0, L_2 = \sqrt{N(t-1)/2D}$. For negative N set $L_1 = \sqrt{-N/D}, L_2 = \sqrt{-N(t+1)/2D)}$. For all integers y satisfying $L_1 \leq y \leq L_2$ check if there exists any integer x such that $x^2 - Dy^2 = N$. Append any solutions (x, y) to our list. Also append $(x, -y)$ if it does not appear in the family of solutions generated by (x, y).

4.19 Hilbert Polynomials

We quote an algorithm to compute Hilbert class polynomials as described by Cohen [23, section 7.6.2]. We need roots of these polynomials over certain fields in order to generate some of the above pairing types, a procedure we shall describe in Section 5.8.

Let $\tau \in \mathbb{C}$ lie in the upper complex plane. Set $q = e^{2i\pi\tau}$. Define

$$\Delta(\tau) = q\left(1 + \sum_{n\geq 1}(-1)^n\left(q^{n(3n-2)/2} + q^{n(3n+1)/2}\right)\right)^{24}$$

and

$$f(\tau) = \frac{\Delta(2\tau)}{\Delta(\tau)}$$

and finally

$$j(\tau) = \frac{(256f(\tau)+1)^3}{f(\tau)}$$

Then the Hilbert class polynomial for the discriminant $-D$ is given by

$$H_D(x) = \prod(X - j(\alpha)$$

where α runs over all complex numbers such that

$$\alpha = \frac{-b+\sqrt{-D}}{2a}$$

where $ax^2 + bxy + cy^2$ is a *primitive reduced positive definite binary quadratic form* of discriminant $-D$, or in other words, $b^2 - 4ac = -D$, $|b| \le a \le \sqrt{|D|/3}$, $\gcd(a, b, c) = 1$ and if $|b| = a$ or $a = c$ then $b \ge 0$.

Hence the following algorithm computes Hilbert class polynomials. Here, P is a polynomial variable, and D is a fundamental discriminant. Recall this means $D > 0$, $D = 0, -1 \pmod 4$ and no odd square divides D.

Algorithm 6 Hilbert class polynomial computation: $P \leftarrow H_D(X)$

```
$P \leftarrow 1$
$b \leftarrow D \pmod 2$
$B \leftarrow \left\lfloor \sqrt{|D|/3} \right\rfloor$
repeat
  $b \leftarrow b + 2$
  $t \leftarrow (b^2 + D)/4$
  $a \leftarrow \max(b, 1)$
  repeat
    if $a^2 \mid t$ then
      $u \leftarrow j((-b + \sqrt{D})/(2a))$
      if $a = b$ or $a^2 = t$ or $b = 0$ then
        $P \leftarrow P(X - u)$
      else
        $P \leftarrow P(X^2 - 2\Re(u)X + |u|^2)$
      end if
    end if
    $a = a + 1$
  until $a^2 > t$
  $b = b + 2$
until $b > B$
round coeffcients of $P$ to nearest integer
return $P$
```

It remains to specify the precision of the floating point operations. Cohen recommends using at least $k + 10$ significant digits where

$$k = \pi \frac{\sqrt{|D|}}{\ln 10} \sum \frac{1}{a}$$

the sum running over all reduced forms (a, b, c) of discriminant D, which can be computed using the following algorithm [23, Algorithm 5.3.5]:

Algorithm 7 Counting reduced forms: $s \leftarrow$ #forms of discriminant D

```
$s \leftarrow 1$, $b \leftarrow D \pmod 2$, $B \leftarrow \left\lfloor \sqrt{|D|/3} \right\rfloor$.
repeat
  repeat
    $q \leftarrow (b^2 - D)/4$, $a \leftarrow b$
    if $a \leq 1$ then
      $a \leftarrow 1$
    else if $a \mid q$ then
      if $a = b$ or $a^2 = q$ or $b = 0$ then
        $s = s + 1/a$
      else
        $s = s + 2/a$
      end if
    end if
    $a \leftarrow a + 1$
  until $a^2 > q$
  $b \leftarrow b + 2$
until $b > B$
return $s$
```

4.20 Arbitrary Embedding Degree

What if other embedding degrees are desired? It turns out that given any positive integer k we can in fact construct pairings with embedding degree k, but the subgroup size r will have far fewer bits than q, the size of the field.

Define $\rho = \lg q / \lg r$. The quantity ρ is in some sense a measure of inefficiency: although we only have r-bit (generic group) discrete log security, we must perform our computations on ρr-bit numbers.

Our main focus is $\rho \approx 1$ pairings, but we now take a small detour and describe an algorithm to find pairings with any given embedding degree k due to Barreto, Lynn and Scott [5]. In fact, this algorithm is relevant to the previous section. A variation of this approach leads to $\rho \approx 1$ type D curves of Scott and Barreto [60].

Lemma. *Let $E(\mathbb{F}_q)$ be an elliptic curve and suppose it has $n = q - t + 1$ points, where t is the trace of Frobenius. Let $n = hr$ where r is prime. Then the embedding degree (of the subgroup of order r) of $E(\mathbb{F}_q)$ is k if and only if $r \mid \Phi_k(t-1)$ and $r \nmid \Phi_i(t-1)$ for all $1 \leq i < k$, where Φ_i denotes the ith cyclotomic polynomial.*

Proof. Then $q = t - 1 \pmod r$. Given any integer i, exponentiating both sides and subtracting 1 gives

$$q^i - 1 = (t-1)^i - 1 \pmod r$$

hence if E has embedding degree k for the subgroup of $E(\mathbb{F}_q)$ of order r then $r \mid (t-1)^k - 1$ and $r \nmid (t-1)^i - 1$ for $1 \le i < k$.

Thus $r \nmid \Phi_i(t-1)$ for $1 \le i < k$ otherwise the embedding degree would be strictly less than k. Since r is prime we must have $r \mid \Phi_k(t-1)$. □

This suggests the following general strategy to finding a curve with a subgroup of order r with given embedding degree k:

Theorem. *Choose an integer x with $|x| > 1$ and any factor r of $\Phi_k(x)$. Choose an integer d. Choose any positive integer h. Let $n = hr$, $q = n + x^d, t = x^d + 1$ (thus $n = q - t + 1$). Suppose $r \nmid \Phi_i(x^d)$ for $1 \le i < k$.*

Then if q is prime and $E(\mathbb{F}_q)$ is a (nonsingular) elliptic curve of order n, then any cyclic subgroup of $E(\mathbb{F}_q)$ of order dividing r has embedding degree k.

Proof. By construction, $t - 1 = x^d$, hence $\Phi_i(t-1) \mid x^{id} - 1$ for any positive integer i. By choice of r we have $r \mid \Phi_k(x)$ thus $r \mid x^k - 1 \mid x^{kd} - 1$ and $r \nmid \Phi_i(t-1)$.

Applying the previous lemma completes the proof. □

Alternative parametrizations are possible. For example, when k is twice an odd number, and d is even, we may use $t = -x^d + 1, q = n - x^d$.

An obvious generalization for the $t = x^d + 1$ case is $q = n + \Phi_k(x)g(x) + x^d$ for any polynomial $g(x)$, though care must be taken to stay within the Hasse bound: we require x^d to cancel out the term of highest degree in $\Phi_k(x)g(x)$ and that the remaining terms have sufficiently low degree. For example, for $k = 9$ we can pick $g(x)$ to obtain $q = n - x^3 - 1$.

In practice we choose d satisfying $1 \le d \le \deg \Phi_k/2$, otherwise the Hasse bound is broken so such a curve $E(\mathbb{F}_q)$ cannot exist and the theorem becomes an empty statement. Also r is usually taken to be the largest prime dividing $\Phi_k(x)$, which almost never divides $\Phi_i(x^d)$ for $1 \le i < k$.

It remains to show how to construct a curve given these parameters. We shall employ the CM method. Recall $r \mid \Phi_k(x)$, so write $mr = \Phi_k(x)$. The CM equation becomes

$$DV^2 = 4h\Phi_k(x)/m - (x^d - 1)^2.$$

If $k = 3, 4, 6$ then $\deg \Phi_k = 2$. Set $m = 1$. Picking some small value for h and fixing D yields a Diophantine equation that can be transformed into a Pell-type equation, which can be solved to find x and V. With luck, $\Phi_k(x)$ will be prime or a small number times a prime, giving a $\rho \approx 1$ pairing. When $h = 1$ we have the equivalent of the MNT method.

Fixing h and D for higher degree cyclotomic polynomials Φ_k result in higher degree Diophantine equations which are not easily solved. Instead, if D is now viewed as a variable and x if fixed, (and h is still fixed) we can find integer solutions for D and V easily. Unfortunately, for cryptographically useful choices of x, the integer D is too large to allow the Hilbert class field polynomial H_D to be computed. However, by using small x we can exhibit toy examples of high embedding degree curves with $\rho \approx 1$.

Now consider fixing D and x. In particular, D is small enough so that H_D can be computed and x is large enough to be cryptographically useful.This gives a Diophantine equation of the form

$$DV^2 = Ah - B$$

where D, A, B are integer constants. We can solve for V and h and obtain a curve containing a cyclic subgroup of order r with embedding degree k. There is a drawback however: h tends to be close in size to r, giving $\rho \approx 2$.

We have set $m = 1$ above. More generally, when $k = 3, 4, 6$, if m is allowed to be some small integer then one can find curves with low h that are not found using the MNT method as originally described [60].

Freeman, Scott and Teske [30] note the approach of Barreto, Lynn and Scott is a special case of a more recent construction by Brezing and Weng [19], which in turn can be thought of as a specific application of a general strategy suggested by Cocks and Pinch [21]. They also outline other methods to construct pairings of arbitrary embedding degree, some of which with ρ substantially less than 2.

Chapter 5

Optimizing Cryptosystems

Improving the efficiency of a pairing-based cryptosystem can be quite involved:

1. Much effort must be devoted to speeding up integer and finite field arithmetic operations, as they are present in large amounts at the lowest level. For best results, platform-dependent hand-coded assembly should be used. Fortunately a lot of research has been conducted in this area [42], and numerous implementations exist, many of them freely available [37, 58].

2. Elliptic curve operations should also be fast. Though a newer subject, much research has been conducted on this area [9].

3. Speeding up the pairing has only been a high priority relatively recently, and is the main topic of the next chapter.

4. Sometimes high-level modifications to a scheme, such as tolerating the loss of a few bits, choosing a different group to contain a key, and so on will garner savings by permitting one to exploit other pairing properties, move exponentiations or multiplications to cheaper groups, precompute more elements, compress or reduce points or pairings, and so on.

5. Needless to say, each of these classes of optimizations cannot be viewed in isolation. An optimization that may not usually help could be beneficial when the behaviour of the whole system is taken into account. Another possibility is that one optimization could adversely affect another.

We have concentrated optimizations from all aspects of a pairing-based cryptosystem into a couple of chapters, rather than introduce each technique along with the algorithm that it affects. Presenting shortcuts in one go as a big bag of tricks is preferable to sprinkling them throughout the text. One motivation is the last item of the above list.

Secondly, an optimization buried in the middle of a section on another topic may be overlooked or forgotten. This may be less likely when it is found in a section dedicated to optimizations.

Also, optimizations from different areas can have much in common. For example, sliding windows and multiexponentiation apply to every repeated-squaring-like algorithm. Seeing all variants at once can provide a deeper understanding of a principle, and may even inspire the application of the same principle to a different situation.

Lastly, the organization of these chapters reflects good software engineering practices. When developing a pairing-based cryptography library from scratch, the initial goal should be to write finite field arithmetic routines. The next step is to implement elliptic curve groups. Once the code for this has been tested satisfactorily, a simple pairing algorithm should be built. Only when the pairing is correctly functioning should one at last consider efficiency improvements. The slowest parts of the system should be attacked first. Educated guessing can meet with some success, but ideally profiling should be employed to identify the bottlenecks.

5.1 Multiprecision Arithmetic

In the author's pairing implementations, to avoid the drawbacks of assembly code (including loss of portability, the need to learn platform-specific low-level tricks, frequent rewrites due to new hardware, fewer potential developers, increase in debugging difficulty, longer development time), pre-existing arbitrary-precision arithmetic libraries were utilized.

One can easily graft finite field routines to any such library, though there are costs. For example, the GMP library has a highly optimized modular exponentiation routine that uses Montgomery reduction, but expects the input not to be in Montgomery form, and also outputs normal integers. Hence if numbers are already internally stored in Montgomery form, they must be converted to a normal integer first, and then converted back to Montgomery form afterwards. Including the unwanted conversions in GMP itself gives a total of four unnecessary conversions.

Another example is that an external library may be able to handle integers of arbitrary length, but a particular deployment of a cryptosystem is concerned only with integers in a certain range. Routines that know in advance how many bytes their input integers take can be faster than their more general counterparts.

Designers of libraries for arbitrary-precision integers should be aware of these issues, and should consider providing access to fairly low-level routines in the library interface.

5.2 All or Nothing

The nature of cryptography in a finite field requiring n bytes per element leads to the value of an integer data structure often containing either zero, or an integer roughly n bytes in length. This is because many cryptographic operations can be viewed as a series of arithmetic operations on n-byte numbers uniformly chosen at random.

One approach to exploit this is to attach a flag to every integer variable that signifies when the variable contains the zero value. Then operations such as addition and multiplication check this flag: if one of the input variables is zero, the output is trivial to determine and no looping is required, otherwise the computation should proceed on all n bytes as with high probability each of the n bytes will be nonzero and thus contribute to the result.

In some cases it may help to test for one as well.

5.3 Montgomery Reduction

For RSA or a discrete-log system in a finite field, it is recommended only to use Montgomery reduction during an exponentiation as the time lost from switching back and forth between representations does not compensate for the savings gained for individual multiplications [47].

Pairing-based cryptosystems on the other hand can benefit from having all coordinates of points stored in Montgomery representation, with conversion only during input and output. This is because during point additions and multiplications, as well as pairing computations, we must perform many finite field operations but never need to convert the coordinates back to its normal representation. In fact, we could take this idea further and avoid conversion completely by storing keys and such in Montgomery representation, though this does requires choosing some fixed machine word size.

Division is slower using Montgomery representation, but this drawback is of no concern if projective coordinates are used (Section 5.9), where each division is replaced by a few multiplications in any case.

The following algorithms can be found in several textbooks in the field [9, 47].

Suppose we are implementing the finite field $\mathbb{F}_p$ for some prime p. Let b be the word size of the machine. Let $R = b^t$ be the smallest power of b greater than p. Let $p' = -p^{-1} \pmod{b}$. Since b is a power of 2, one can use a specialized procedure to find p' [9, Algorithm II.5]. However this is usually unimportant because p' can be precomputed once and stored.

Expressions modulo b should obviously be implemented as operations on machine words. Similarly, multiplications and divisions involving R and b are simply bit shifts.

Then we represent an element $x \in \mathbb{F}_p$ as xR. We only convert back for input/output operations.

The first algorithm allows us to convert quickly from Montgomery representation. It applies to any nonnegative integer $y < pR$. The ith most significant machine word of y is denoted by y_i.

Algorithm 8 Montgomery Reduction: $z \leftarrow yR^{-1}$

```
for $i = 0$ to $t - 1$ do
    $u \leftarrow y_i p' \pmod{b}$
    $y \leftarrow y + upb^i$
end for
$z \leftarrow y/R$
if $z \geq p$ then
    $z \leftarrow z - p$
end if
```

The next algorithm shows why multiplication is faster. Note there is only one step requiring the product of a multiprecision integer and a single word: the other multiplications are single precision.

Inversion is slower: given xR, we use a standard inversion algorithm to find $x^{-1}R^{-1}$. Then performing a Montgomery multiplication with $R^3 \pmod{p}$ (which has been precomputed) yields $x^{-1}R$.

Algorithm 9 Montgomery Multiplication: $Z = XYR^{-1} \pmod{p}$

```
Z ← 0
for i = 0 to t − 1 do
    u ← (z_0 + s_i y_0)p' (mod b)
    Z ← (Z + x_i y + up)/b
end for
if Z ≥ p then
    Z ← Z − p
end if
```

5.4 Cube Roots

When $q = 2 \pmod 3$, for any $x \in \mathbb{F}_q$ we have $(x^{-(q-2)/3})^3 = x^{1-(q-1)} = x/x^{q-1} = x$, thus cube roots can be quickly found via exponentiaton by $-(q-2)/3$. Observe this also means every element of $\mathbb{F}_q$ has a cube root, which implies cube roots are unique.

Thus with curves of the form $Y^2 = X^3 + b$ over such fields, a better way of finding random points is to choose Y randomly and solve for X, which involves taking a cube root. Additionally, point reduction and compression can be achieved by discarding the x-coordinate and only recording the y-coordinate. In particular, for point compression there is no need to store an extra bit and for point reduction no information is lost.

5.5 Random Points

Let $E : Y^2 = X^3 + aX + b$ be an elliptic curve and suppose we are working within a cyclic subgroup G of order r of the points of E.

To find a random point in G, one can randomly choose an x-coordinate and attempt to solve the equation E for the y-coordinate, which involves a finite-field square root algorithm. If no solution exists, more x-coordinates are chosen until a solution for y can be found.

As soon as valid coordinates are obtained, point multiplication by an appropriate factor ensures the resulting point P lies in G.

In certain curves noted above, there are advantages to choosing y and solving for x instead.

In either case, once such a point is found, future random points can be generated by picking a random $k \in \{0, ..., r-1\}$ and returning kP. This can be faster in some curves, though it is an inferior method on others.

To ensure that the point is uniformly chosen from G, the point P must have order r, though in most cases (i.e. when r is a large prime or a product of large primes) this happens with high probability.

When G is not cyclic, but each point has order dividing r, we may of course extend this technique by finding a linearly independent basis of G and form some linear combination using them, where the coefficients are chosen randomly.

However, in this case, many pairing-based cryptosystems still function even if random points are picked from a cyclic subgroup of G only, and we merely need one point of G of order r.

One could consider a hashing to a group by hashing to an integer k in $\{0, ..., r-1\}$ and returning kP for some fixed point P, but for most cryptosystems this cannot be done. In many applications, security is compromised if the discrete log of the outputs of a hash function with respect to some fixed base are always known.

5.6 Dedicated Squaring

A relatively painless way to improve running times is to implement squaring routines for every field, ring or group.

In the next section we describe squaring tricks for $\mathbb{F}_{q^2}$ when q is a prime satisfying $q = 2 \pmod 3$ or $q = 3 \pmod 4$.

For other low degree extensions (e.g. 3, 6), one can write special cases of generalized Karatsuba squaring and multiplication algorithms [66].

5.7 Quadratic Field Extensions

For primes $q = 3 \pmod 4$, a degree two field extension of $\mathbb{F}_q$ can be implemented as $\mathbb{F}_q[i]$ where i is a square root of -1.

We have

$$(a + ib)^2 = (a - b)(a + b) + i(2ab)$$

and

$$(a + ib)(c + id) = (ac - bd) + i[(a + b)(c + d) - ac - bd]$$

As mentioned in the previous chapter, in $\mathbb{F}_q[i]$, exponentiation by q can be performed by simply negating the imaginary part:

$$(a+ib)^q = a - ib$$

which in turn implies computing expressions such as $xz - yz^q$ only costs four multiplications in $\mathbb{F}_q$, where $x, y, z \in \mathbb{F}_q[i]$.

For primes $q = 2 \pmod 3$, the polynomial $X^2 + X + 1$ is irreducible in $\mathbb{F}_q$, thus its roots form an optimal normal basis of $\mathbb{F}_{q^2}$ allowing several shortcuts [44].

In other words, we compute in $\mathbb{F}_q[\alpha]_{\alpha^2+\alpha+1}$ (viewing α as an indeterminate) and represent elements as tuples $(a, b) \in \mathbb{F}_q \times \mathbb{F}_q$ (which means $a\alpha + b\alpha^2$). Note $\alpha^3 = 1$ and $x \in \mathbb{F}_q$ may be written as $-x\alpha - x\alpha^2$.

Then

$$(a\alpha + b\alpha^2)^2 = b(b-2a)\alpha + a(a-2b)\alpha^2$$

and

$$(a\alpha + b\alpha^2)(c\alpha + d\alpha^2) = (bd - ad - bc)\alpha + (ac - ad - bc)\alpha^2$$

We also have

$$(a\alpha + b\alpha^2)^q = b\alpha + a\alpha^2$$

which again implies computing expressions such as $xz - yz^q$ only costs four multiplications in $\mathbb{F}_q$, where $x, y, z \in \mathbb{F}_{q^2}$.

5.8 Finding a Root of a Polynomial

When computing parameters for curves using the CM method, we need to find a root of a Hilbert polynomial modulo a prime. As we only want a single root, we may use the Cantor-Zassenhaus method and skip many steps. For example, we do not care about the multiplicity of the factors, nor the factors of degree higher than 1. Thus we may do the following to find a root of a degree n polynomial $f(x)$ in $\mathbb{F}_q$.

1. Compute $g(x) = \gcd(x^q - x, f(x))$.

2. If $\deg g = 1$ then output the root and stop.

3. Pick a random $r \in \mathbb{F}_q$. If r is a root then stop.

4. Compute $s(x) = (x - r)^{(q-1)/2} \bmod g(x)$.

5. Compute $g'(x) = \gcd(s(x) + 1, g(x))$. This is a proper factor of g with probability $1 - 2^{n-1}$. So if $g'(x) \neq 1$ set $g = g'$ and goto step 2.

6. Goto step 3.

5.9 Projective Coordinates

Instead of using a pair of numbers to represent a point, we can use Jacobian projective coordinates: the triplet (x, y, z) represents the point $(x/z^2, y/z^3)$.

This allows us to replace inversions with several multiplications in a finite field, which is usually worth the trouble though the savings vary from implementation to implementation. Note that Montgomery reduction complements projective coordinates as it speeds up multiplication at the expense of inversion.

We quote algorithms for point addition and doubling using projective coordinates from Blake, Seroussi and Smart [9]:

Algorithm 10 Projective Point Addition, $(x_3, y_3, z_3) = (x_1, y_1, z_1) + (x_2, y_2, z_2)$

```
$\lambda_1 \leftarrow x_1 z_2^2$
$\lambda_2 \leftarrow x_2 z_1^2$
$\lambda_3 \leftarrow \lambda_1 - \lambda_2$
$\lambda_4 \leftarrow y_1 z_2^3$
$\lambda_5 \leftarrow y_2 z_1^3$
$\lambda_6 \leftarrow \lambda_4 - \lambda_5$
$\lambda_7 \leftarrow \lambda_1 + \lambda_2$
$\lambda_8 \leftarrow \lambda_4 + \lambda_5$
$z_3 \leftarrow z_1 z_2 \lambda_3$
$x_3 \leftarrow \lambda_6^2 - \lambda_7 \lambda_3^2$
$\lambda_9 \leftarrow \lambda_7 \lambda_3^2 - 2x_3$
$y_3 \leftarrow (\lambda_9 \lambda_6 - \lambda_8 \lambda_3^3)/2$
```

Note when one input is projective and the other affine fewer multiplications are required. This is called *mixed addition*.

We see the computation of $3x^2 + az^4$ in general requires three squarings and one multiplication. Obeserve we consider multiplication by a small constant to be negligible.

For $a = 0$ the only one squaring is required.

Algorithm 11 Projective Point Doubling, $(x_3, y_3, z_3) = 2(x_1, y_1, z_1)$

```
$\lambda_1 \leftarrow 3x_1^2 + az_1^4$
$z_3 \leftarrow 2y_1z_1$
$\lambda_2 \leftarrow 4x_1y_1^2$
$x_3 \leftarrow \lambda_1^2 - 2\lambda_2$
$\lambda_3 \leftarrow 8y_1^4$
$y_3 \leftarrow \lambda_1(\lambda_2 - x_3) - \lambda_3$
```

For small a or fairly small a with low Hamming weight we can ignore the cost of the multiplication.

For $a = -3$ we may compute $\lambda_1 = 3(x + z^2)(x - z^2)$ which requires one multiplication and one squaring.

For $a = -3d^2$ where d is small or fairly small with low Hamming weight, we may compute $\lambda_1 = 3(x + dz^2)(x - dz^2)$ which has almost the same cost.

In the following chapter, we shall learn how to transform a given curve. Doing so allows us to obtain an a that is faster than the general case.

Cohen, Miyaji and Ono [22] investigate other variants of projective coordinates, where point doubling or multiplication can be even faster.

5.10 Point Multiplication

Point multiplication is the elliptic curve equivalent of modular exponentiation, thus optimizations that apply to one often apply to the other.

One difference is that group inversion is much cheaper in elliptic curves: one simply negates the y-coordinate, an operation that is essentially free, whereas in finite fields one must spend time running the extended Euclidean algorithm.

This allows one work with addition-subtraction chains instead of addition chains, that is, use signed sliding windows [47, Chapter 14] [9, Section IV.2.5].

If it is known in advance that a particular point P will feature in several point multiplications, preprocessing can be employed, namely multiples of P used often in the exponentiation routine are computed and stored.

5.11 Multiexponentiation

When computing expressions of the form $aP + bQ$ (the equivalent of $g^x h^y$ in finite fields), rather than computing aP and bQ separately and adding the results, one can use a multi-exponentiation trick, sometimes referred to as vector addition chains [47, Chapter 14], to roughly halve the number of point doublings (if a and b are about the same length). In its simplest form, $P + Q$ is precomputed and stored, then a double-and-add algorithm is used to compute the final result. Instead of describing the algorithm in full, we present an example:

To compute $12P + 7Q$ one precomputes $P + Q$, then:

1. $R \leftarrow P$
2. $R \leftarrow 2R$
3. $R \leftarrow R + (P + Q)$
4. $R \leftarrow 2R$
5. $R \leftarrow R + Q$
6. $R \leftarrow 2R$
7. $R \leftarrow R + Q$

Of course vector-chain-addition exponentiation generalizes to computing any expression of the form $a_1P_1 + ... + a_tP_t$. Multiexponentiation can also be combined with sliding window techniques. One drawback is the amount of precomputation needed, which worsens as t increases, and also as the size of the sliding window is increased in which case one must store $mP + nQ$ for several m, n.

Note the Tonelli-Shanks algorithm contains a multiexponentiation that benefits from this optimization.

5.12 Floating-Point Complex Numbers

Though not part of any cryptosystem, ideally code that finds suitable curves should also be fast. For type D and G curves, we must find Hilbert polynomials. This requires high precision floating point complex arithmetic.

Packages that provide arbitrary precision floating point numbers may not include routines for complex arithmetic. When implementing complex numbers using such a library, one should be aware of a few basic facts.

Multiplication should use the tricks for quadratic field extensions described earlier.

To minimize underflow, overflow or precision loss [53], one should compute the norm as

$$|a+ib| = \begin{cases} |a|\sqrt{|1+(b/a)^2|} & \text{if } |a| \geq |b| \\ |b|\sqrt{|1+(a/b)^2|} & \text{if } |a| < |b| \end{cases}$$

and divide using

$$\frac{a+ib}{c+id} = \begin{cases} \frac{(a+b(d/c))+i(b-a(d/c))}{d+d(d/c)} & \text{if } |c| \geq |d| \\ \frac{(a(c/d)+b)+i(b(c/d)-a)}{c(c/d)+d} & \text{if } |c| < |d| \end{cases}$$

Chapter 6

Faster Pairings

We now focus on optimizing pairings. Firstly, one can improve the running time of the rational function computation by a factor of 2 by applying a theorem due to Barreto, Kim, Lynn and Scott [4]. This running time can be halved again by using a method described by Barreto, Lynn and Scott [6].

Barreto, Kim, Lynn and Scott [4] also give techniques for speeding up the final powering. The efficiency gain depends on the embedding degree, but for $k = 2$ curves we halve the running time and for $k = 6$ the final powering is three times as fast. (Recall the output of the Tate pairing is a coset representative and we need the final powering to standardize it.)

Because Miller's algorithm has features in common with an exponentation routine, optimizations that improve powering routines can often be tailored to improve pairings.

We mention other optimizations such as preprocessing and pairing compression, though most of these techniques either depend on certain properties of the cryptosystem, or require it to be modified.

6.1 Twist Curves

Let E be an elliptic curve $Y^2 = X^3 + aX + b$ in $\mathbb{F}_q$ where q is a prime power. Let v be a quadratic nonresidue in $\mathbb{F}_q$. Consider the curve E' given by $Y^2 = X^3 + av^2X + v^3b$, which we call the (quadratic) *twist* of the curve E.

Theorem. *Let* $t = q + 1 - \#E(\mathbb{F}_q)$. *Then* $\#E'(\mathbb{F}_q) = q + 1 + t$.

The proof is straightforward. Let $g(X) = X^3 + aX + b$, and $h(X) = X^3 + av^2X + v^3b$.

Note $h(X) = v^3 g(Xv^{-1})$.

There are three cases.

1. If $g(xv^{-1}) = 0$ then $v^3 g(xv^{-1}) = h(x) = 0$ thus $Y = 0$ is the unique solution to both the equations $Y^2 = g(xv^{-1})$ and $Y^2 = h(x)$.

2. If $g(xv^{-1})$ is a quadratic residue then $Y^2 = g(xv^{-1})$ has exactly two solutions, and $Y^2 = v^3 g(xv^{-1}) = h(x)$ has no solutions (since v^3 is a quadratic nonresidue).

3. If $g(xv^{-1})$ is a quadratic nonresidue then the situation is reversed: $Y^2 = g(xv^{-1})$ has no solutions and and $Y^2 = h(x)$ has exactly two solutions.

As x runs through all the elements of $\mathbb{F}_q$, so does xv^{-1}, and we see that total number of solutions to either equation E and E' over $\mathbb{F}_q$ is $2q$. Since O is always a solution of any elliptic curve we have $\#E(\mathbb{F}_q) + \#E(\mathbb{F}_q) = 2q + 2$, proving the theorem.

On the other hand if v is a quadratic residue then $v = c^2$ for some $c \in \mathbb{F}_q$. Then we have a map $\Psi : E'(\mathbb{F}_q) \to E(\mathbb{F}_q)$ given by

$$\Psi(x, y) = (c^2 x, c^3 y)$$

hence in some sense E and E' are the same curve, so the choice of quadratic nonresidue v does not matter as they all lead to the same twist.

Note a quadratic nonresidue v becomes a quadratic residue in $\mathbb{F}_{q^2}$ (that is, we can find a square root c of v) and we have a map between $E'(\mathbb{F}_{q^2})$ and $E(\mathbb{F}_{q^2})$. Roughly speaking, curves that are twists of each other become the same curve when considered in a quadratic extension of the field within which they are defined.

For example, since 2 is a quadratic nonresidue in $\mathbb{F}_{19}$, the curve $E : Y^2 = X^3 + X + 6$ (over $\mathbb{F}_{19}$) has the twist $E' : Y^2 = X^3 + 4X + 10$, and $E'(\mathbb{F}_{19})$ contains 22 points.

In $\mathbb{F}_{19^2}$, both E and E' contain 396 points, and we can map points of E' to points of E via $(x, y) \mapsto (2x, 2\sqrt{2}y)$.

We can use quadratic residues to transform the equation of a given elliptic curve E into a form that allows certain optimizations. We would like $a = 0$ to reduce the amount of multiplications needed for a projective point doubling, but for a general curve this an impossible transformation. Recall from Section 5.9 that a reasonable alternative is $a = -3$.

Then from above, a curve $Y^2 = X^3 + aX + b$ can be transformed into one of the form $Y^2 = X^3 - 3X + b'$ if we can find $v \in \mathbb{F}_q$ satisfying $av^2 = -3$.

When no such c exists, a compromise is to have $a = 1$ or $a = -3d^2$ for some small integer d (or d of low Hamming weight), which can be achieved in a similar manner, again to reduce the number of multiplications required by a projective point doubling.

6.2 Simplified Tate Pairing

A result due to Barreto, Kim, Lynn and Scott states that when the embedding degree is greater than 1 we may simplify the Tate pairing as follows [4, 6].

Theorem. *Let E be an elliptic curve over $\mathbb{F}_q$. Let $P \in E(\mathbb{F}_q)$ be a point of prime order r. Let $G = \langle P \rangle$, and let k be the embedding degree of G.*

If $k > 1$ then

$$e(P, Q) = f_P(Q)^{(q^k-1)/r}$$

is a bilinear nondegenerate map, where f_P is a function with divisor $r\langle P \rangle - r\langle O \rangle$.

Proof. Choose any point $R \in E(\mathbb{F}_q)$ that is not one of $O, -P, Q, -Q, Q - P$, and consider the function f'_P that satisfies $(f'_P) = r\langle P + R\rangle - r\langle R\rangle$. The Tate pairing can be computed by

$$(f'_P(Q)/f'_P(O))^{(q^k-1)/r}$$

We have $f'(O) \in \mathbb{F}_q^*$ since it does not have a zero or pole at O. Hence $f'(O)^{(q^k-1)/r} = 1$ by Fermat's Little Theorem (we know $q - 1$ must divide $(q^k - 1)/r$ since $r \nmid q - 1$) thus

$$e(P, Q) = f'_P(Q)^{(q^k-1)/r}.$$

Let $V_R(X, Y)$ be the equation of a vertical line through R, let $V_P(X, Y)$ be the equation of a vertical line through P, and let $L(X, Y)$ be the equation of a line through $P + R$ and $-R$ (and hence $-P$). Then we have

$$(f'_P V_R^r V_P^r / L^r) = r\langle P\rangle - r\langle O\rangle = (f_P)$$

None of V_R, V_P, L have zeroes or poles at Q by choice of R. Since each of $V_R(Q), V_P(Q), L(Q)$ is ultimately exponentiated by $q^k - 1$ we have $f'_P(Q) = f_P(Q)$. (Alternatively we could appeal to Fermat's Little Theorem again, since the lines can be chosen to have coefficients in $\mathbb{F}_q$.) Hence

$$e(P, Q) = f_P(Q)^{(q^k-1)/r}.$$

□

6.3 Simplified Weil Pairing

We may also simplify the Weil pairing in a similar fashion. Recall that two points R, S are needed in addition to the input points P, Q. We show that we can pick $R = O$ and compute the Weil pairing as

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

where f_P is a rational function satisfying $(f_P) = (P)^r$ (reminiscent of a rational function defined for the Tate pairing), and f_Q is a certain rational function with divisor $(f_Q) = (Q+S)^r/(S)^r$.

In our original definition, the choice of rational function for f_Q did not matter. They are all equivalent up to a constant, which cancels itself out during a division. In the above formula, we never divide f_Q by itself so we build f_Q carefully as follows.

During Miller's algorithm each iteration consists of finding an equation of a line and evaluting it at a certain point before multiplying it to the running product. We compute $f_Q(P)$ in the same way, except we ensure the lines, tangents and verticals have a particular form. For lines and tangents, we pick the equation where the coefficient of Y is unity. For verticals, we pick the equation where the coefficient of X is unity.

For this particular construction of f_Q, we can check that $f_Q(O) = 1$, proving that we can indeed simplify the Weil pairing as claimed.

We can instead choose $S = O$ and compute

$$f(P, Q) = \frac{f_P(Q)}{f_Q(P+R)/f_Q(R)}$$

with similarly defined f_P, f_Q.

Note we cannot simultaneously have $R = S = O$ for this would imply both f_P and f_Q have poles at O hence cannot be evaluated at O.

6.4 Twist Curves and the Trace-Zero Group

Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}_q$, and P be a point of prime order r. Suppose the embedding degree k of $G = \langle P \rangle$ is even.

Let $d = k/2$. Let v be a quadratic nonresidue in $\mathbb{F}_{q^d}$, so that $\mathbb{F}_{q^k} = \mathbb{F}_{q^d}[\sqrt{v}]$. Let E' be the twist of E hence $E' : y^2 = x^3 + v^2ax + v^3b$.

Define the map $\Psi : E' \to E(\mathbb{F}_{q^k})$ by

$$\Psi(x, y) = (v^{-1}x, v^{-3/2}y).$$

Theorem. *Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}_q$, and P be a point of prime order r. Suppose the embedding degree k of $G = \langle P \rangle$ is even. Write $k = 2d$. Let E' be the twist of E in $\mathbb{F}_{q^d}$.*

Then $r \mid \#E'(\mathbb{F}_{q^d})$.

Proof. If $\#E(\mathbb{F}_{q^d}) = q^d + 1 - c$ then

$$\#E'(\mathbb{F}_{q^d}) = q^d + 1 + c$$

and

$$\#E(\mathbb{F}_{q^k}) = (q^d + 1 + c)(q^d + 1 - c).$$

The group $E(\mathbb{F}_{q^d})$ only contains r points of $E[r]$ otherwise the embedding degree would be at most d, and similarly $E(\mathbb{F}_{q^k})$ contains all r^2 points of $E[r]$, hence

$$r \mid q^d + 1 + c = \#E(\mathbb{F}_{q^d}).$$

□

Let P be any point of order r in $E(\mathbb{F}_q)$ and Q' be any point whose order is a multiple of r in $E'(\mathbb{F}_{q^d})$. In practice a randomly chosen point of $E'(\mathbb{F}_{q^d})$ will do.

Let $G = \langle P \rangle, H = \langle Q' + rE'(\mathbb{F}_{q^d}) \rangle$. Note H is a subgroup of $E'(\mathbb{F}_{q^d})/rE'(\mathbb{F}_{q^d})$.

Let f be the Tate pairing. Then e defined by

$$e(P, Q') = f(P, \Psi(Q'))$$

is a bilinear map. We may subsitute the Weil pairing if we ensure $rQ' = O$ by suitable premultiplication.

In practical terms, this means most operations are performed in $\mathbb{F}_{q^d}$ or $\mathbb{F}_q$. The Ψ map and other computations in $\mathbb{F}_{q^k}$ are only performed when a pairing is being evaluated. Also note G_2 is cyclic, and that we can hash into G_2, features useful to some cryptosystems.

We can view this trick as a generalization of type A pairings. Observe we have $k = 2, d = 1$, and that the curve $E : y^2 = x^3 + x$ is equivalent to $E' : y^2 = x^3 + v^2 x$ because either v or $-v$ is a quadratic residue, thus a type A curve is the quadratic twist of itself. In this case, the pairing is symmetric as well.

We can say more about this particular group selection.

Theorem. *Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}_q$, and P be a point of prime order r. Suppose the embedding degree k of $G_1 = \langle P \rangle$ is even. Then write $d = k/2$. Let v be a quadratic nonresidue in $\mathbb{F}_{q^d}$, Let E' be given by $y^2 = x^3 + v^2 ax + v^3 b$. Define the map $\Psi : E' \to E(\mathbb{F}_{q^k})$ by*

$$\Psi(x, y) = (v^{-1}x, v^{-3/2}y).$$

Then $G_2 = \Psi(E'(\mathbb{F}_{q^d})[r])$ is precisely the subgroup of trace zero points in $E(\mathbb{F}_{q^k})[r]$.

Proof. Given $Q' = (X, Y) \in E'(\mathbb{F}_{q^d})$ we have $\Psi(Q') = (a, b)$ where $a \in \mathbb{F}_{q^d}$, and b has the form $b = c\sqrt{v}$ for some $c \in \mathbb{F}_{q^d}$.

Now $\Phi^d(\sqrt{v}) = -\sqrt{v}$, which can be verified using $v^{(q^d-1)/2} = -1$ since v is a quadratic nonresidue, or with Galois theory: $\sqrt{v}$ must be mapped to some other root of its minimal polynomial as Φ^d fixes $\mathbb{F}_{q^d}$ but not $\mathbb{F}_{q^k}$. Thus $\Phi^d(a) = a$ and $\Phi^d(b) = -b$, whence

$$\Phi^d(\Psi(Q')) = \Phi^d(a, b) = (a, -b) = -\Psi(Q')$$

But it can be easily checked that any point $Q \in E(\mathbb{F}_{q^k})$ satisfying $\Phi^d(Q) = -Q$ also satisfies $\operatorname{tr} Q = O$, hence $\Psi(Q')$ is a point of trace zero.

Conversely, suppose Q is a point of trace zero in $E(\mathbb{F}_{q^k})[r]$. From Section 3.3 we have $\Phi(Q) = qQ$, thus $\Phi^d(Q) = q^d Q$.

Since $q^k = 1 \bmod r$, we have $q^d = -1 \bmod r$ (we cannot have $q^d = 1 \bmod r$ since $k > d$ is the embedding degree). Hence $\Phi^d(Q) = -Q$.

If we write $Q = (a, b)$ then this implies $a \in \mathbb{F}_{q^d}$ and $b = c\sqrt{v}$ for some $c \in \mathbb{F}_{q^d}$, thus $\Psi^{-1}(Q)$ lies on $E'(\mathbb{F}_{q^d})$.

Alternatively we may employ a counting argument to show the converse, as there are exactly r points in $E'(\mathbb{F}_{q^d})[r]$ and exactly r points in $E(\mathbb{F}_{q^k})[r]$ of trace zero. □

One consequence is that, aside from type A pairings, this group selection leads to an asymmetric bilinear pairing

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

with no known efficiently computable group isomorphism $\phi : G_2 \to G_1$. Cryptosystems relying on the existence of such a map must be modified accordingly if they are to use this trick.

Recall for a randomly chosen Q in $E(\mathbb{F}_{q^k})[r]$ we have $\operatorname{tr} Q \neq O$ with overwhelming probability, and one can take ϕ to be the trace map as an efficiently computable isomorphism between $\langle Q \rangle$ and $E(\mathbb{F}_q)[r]$. However, as our second input is always a point of trace zero, this choice of ϕ no longer applies, thus we must use the more general definition of a bilinear pairing.

6.4.1 Remarks on Implementation

Observe that all cryptosystem operations except for the bilinear map can be computed in $\mathbb{F}_q$ or $\mathbb{F}_{q^d}$. One only needs to compute in $\mathbb{F}_{q^k}$ for a pairing operation.

In particular, when $k = 2$ all cryptosystem operations except for the pairing can be computed in $\mathbb{F}_q$. More generally, for $k = 2d$ where $d > 1$, if $\mathbb{F}_{q^k}$ is implemented as an extension of $\mathbb{F}_q$ using a minimal polynomial consisting only of terms of even degree (that is as sparse as possible), then the elements of $\mathbb{F}_{q^d}$ are also polynomials containing only terms of even degree.

We have in fact encountered this optimization before. Given a Type A curve

$$E : y^2 = x^3 + x,$$

its distortion map can be thought of as the map ϕ above that takes points on the twist curve $E'(\mathbb{F}_q)$ to $E(\mathbb{F}_{q^2})$, since $E' = E$ in this case.

A dilemma arises with type B curves however:

6.4.2 Twist Curve Trade-offs

Recall a type B curve is given by $E : y^2 = x^3 + 1$ over a field $\mathbb{F}_q$. Its twist E' is given by $E' : y^2 = x^3 + v^3$ for some quadratic nonresidue v and also over $\mathbb{F}_q$. Elements of G_1 are points on $E(\mathbb{F}_q)$. We have two choices for G_2:

1. Take elements of G_2 from $E(\mathbb{F}_q)$, and feed them into the distortion map Ψ (Section 4.8)

to obtain elements of $E(\mathbb{F}_{q^2})$ for the pairing. Recall the distortion map is given by $\Psi(x, y) \mapsto (\zeta x, y)$ where ζ is a primitive cube root of unity.

Then we have symmetry as elements of either input group are chosen from $E(\mathbb{F}_q)$. Unfortunately, we cannot use the important optimization of the following section.

2. Use the twist curve, that is, take elements of G_2 from $E'(\mathbb{F}_q)$, and feed them into the twist map ϕ to obtain elements of $E(\mathbb{F}_{q^2})$ for the pairing.

   This allows us to eliminate denominators as described in the following section, but also means we lose symmetry, since the points from G_2 lie in a different curve.

More generally, for other curves, whether or not to use the twist curve trick depends on if particular properties of the pairing are needed.

Some cryptosystems require DDH to be difficult in both input groups. In such cases we must use the trace zero group for the second input group, otherwise the trace map enables the pairing to break DDH in G_2. Other schemes want DDH hard in G_1 only, but also a surjective map from G_2 to G_1, in which case we cannot use the trace zero group.

Suppose G_2 must be cyclic in an asymmetric pairing. Then if we want the ability to hash to G_2, we must use the trace zero group, and thus cannot have an efficiently computable isomorphism from G_2 to G_1. Note we can hash to a point of trace zero by first hashing to a point $Q \in E(\mathbb{F}_{q^2})$ and returning $\Phi(Q) - Q$. Otherwise, if hashing is not needed for G_2, we may pick any point $Q \in E(\mathbb{F}_{q^2})$ and use $G_2 = \langle Q \rangle$.

6.5 Denominator Elimination

When using twist curves, we can halve the running time of a pairing by applying a technique due to Barreto, Lynn and Scott [6]. We use the same notation and assumptions as the previous section. In other words, again let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}_q$, and P be a point of prime order r. Suppose the embedding degree k of $G = \langle P \rangle$ is even and write $k = 2d$.

The Tate pairing requires us to find $f_P(Q)$. In Miller's algorithm, calculating the denominator of $f_P(Q)$ involves evaluating the equation of various vertical lines at a point. In other words, we compute $x - a$ where a is the x-coordinate of the line and x is the x-coordinate of Q.

Recall we eventually exponentiate the output of Miller's algorithm by $(q^k - 1)/r$ to standardize the coset representative. Observe $q^d - 1$ divides $(q^k - 1)/r$ because if r divides $q^d - 1$ then the embedding degree is at most d, not k.

Thus if both x and a lie in $\mathbb{F}_{q^d}$ then we have $(x - a)^{q^d - 1} = 1$ so they can be omitted during Miller's algorithm. This occurs when twist curves are used, hence we may simplify the computation of $f_P(Q)$ as follows. Recall that T_Z denotes the equation of the tangent at Z and L_Z denotes the equation of the line between Z and P.

Algorithm 12 Miller's Algorithm with Denominator Elimination: $f \leftarrow f_P(Q)$

```
Let the binary representation of $r$ be $b_t ... b_0$.
$f \leftarrow 1$
$Z \leftarrow P$
for $i \leftarrow t - 1$ to 0 do
    $f \leftarrow f^2 \cdot T_Z(Q)$
    $Z \leftarrow 2Z$
    if $b_i = 1$ then
        $f \leftarrow f \cdot L_Z(Q)$
        $Z \leftarrow Z + P$
    end if
end for
```

Assuming r is odd, which holds for all practical applications, the if condition is true during the last iteration and the last multiplication is

$$f \leftarrow f L_{(r-1)P}(Q).$$

Since $(r - 1)P = -P$, this is equivalent to $f \leftarrow f V_P(Q)$ and hence can be skipped since no vertical line computations are needed. In other words, we have $f_r = f_{r-1}$ and the logic can be simplified.

6.6 Input Restriction

We have already covered most of the facts of this section, but we wish to emphasize that, from an efficiency standpoint, choosing the input groups to be certain subgroups is desirable for all pairings.

Consider a supersingular type A, B or C curve E over $\mathbb{F}_q$ containing a cyclic group of order r of embedding degree k. Recall a distortion map ϕ exists for this curve. Let f

be the Weil or Tate pairing for this curve. We can instantiate the symmetric pairings of Section 1.4 (i.e. the input groups are the same, cyclic and lie over smaller fields) by defining $G = E(\mathbb{F}_q)[r]$, G_T to be the group of rth roots of unity in $\mathbb{F}_{q^k}$. and $e : G \times G \to G_T$ by $e(P, Q) = e(P, \phi(Q))$.

More generally let $E(\mathbb{F}_q)$ be an elliptic curve containing a cyclic group of order r of embedding degree k. Distortion maps might not exist for our curve E. However, we can always restrict the first input to $E(\mathbb{F}_q)[r]$, which is cyclic when $k > 1$, and we have just seen that for even $k = 2d$, the other input can be restricted to some curve over $\mathbb{F}_{q^d}$.

This is faster to compute in than $\mathbb{F}_{q^k}$ and allows denominator elimination. Moreover, the second input group is now cyclic and we can hash into it. The only potential drawback is that there is no known efficient method for mapping elements of one input group to the other, which could complicate security proofs or even render the pairing unsuitable for certain cryptosystems.

For certain pairings, the higher degree twists of Section 6.17 can be used for greater savings.

6.7 Miller-Lite Operations

One advantage of restricting the first input P of a pairing to $E(\mathbb{F}_q)[r]$ is that during Miller's algorithm, we perform point additions and doubling involving P and also compute equations of lines that pass through various multiples of P. Thus if $P \in E(\mathbb{F}_q)[r]$ all the arithmetic involved for these operations, which form the bulk of Miller's algorithm, can be performed in $\mathbb{F}_q$ rather than $\mathbb{F}_{q^k}$.

This has been dubbed a *Miller-Lite* operation in the literature. When P does not lie in the base field but rather is some element of $E(\mathbb{F}_{q^k})$ then we refer to an iteration of Miller's algorithm with first input P as a *full Miller*, or *Miller-Full*, operation.

6.8 Last-Second Conversions

One easy optimization is also easy to overlook as it is hidden by notation. In a Miller-Lite operation, at some point we must encounter $\mathbb{F}_{q^k}$ arithmetic. Distortion maps or twist maps must be applied on the second input point Q to obtain a point on $E(\mathbb{F}_{q^k})$, which in turn is fed to line equations.

However this should be done carefully. We wish to avoid computing in $\mathbb{F}_{q^k}$ until absolutely necessary. We describe the procedure in detail for type A pairings. Similar statements can be made for other pairing types.

Let $e : E(\mathbb{F}_q) \times E(\mathbb{F}_q) \to \mathbb{F}_{q^2}$ be a type A pairing and suppose we have chosen to use $\mathbb{F}_q[i]$ to represent $\mathbb{F}_{q^2}$ where $i = \sqrt{-1}$.

During the computation of the pairing, we evaluate $g(Q')$ where $g = aX + bY + c$ is an equation of some line and $Q' = \phi(Q)$, where ϕ is defined by $(X, Y) \mapsto (-X, iY)$ and Q is some point (x, y) in $E(\mathbb{F}_q)$.

If we follow the notation blindly, we would first compute the point Q' which takes twice as much storage as Q (as the field it lies over is twice as big), and perform operations on elements of $\mathbb{F}_q[i]$ to arrive at $g(Q')$.

It is wiser to never explicitly compute Q' and instead do

$$Re(R) \leftarrow c - ax, Im(R) \leftarrow by.$$

Now $R = g(Q')$, and we have only computed in $\mathbb{F}_q$.

6.9 The Final Powering

The last step of a Tate pairing computation is to exponentiate some quantity a by

$$\frac{q^k - 1}{r} = r^{-1} \prod_{d|k} \Phi_d(q)$$

Since k is the embedding degree, we have $r \mid \Phi_k(q)$ (and no cyclotomic polynomial of smaller degree).

The following method for computing $a^{(q^k-1)/r}$ is faster than the obvious approach [4].

1. Compute $b = a^d$ where
$$d = \prod_{d|k, d<k} \Phi_d(q),$$
exploiting the identity $x^q = x$ for all $x \in \mathbb{F}_q$.

2. Since
$$c = \frac{\Phi_k(q)}{r}$$

is an integer, compute the output b^c using a standard exponentiation algorithm.

We describe the steps in detail for $k = 2$. Suppose $\mathbb{F}_{q^2}$ has been implemented as $\mathbb{F}_q[\alpha]$. Typically $\alpha = i = \sqrt{-1}$. Then $q^2 - 1 = \Phi_2(q)(q-1)$ and $r \mid \Phi_2(q) = q+1$. Write $a = u + \alpha v$ where $u, v \in \mathbb{F}_q$. We have

$$b = a^{q-1} = (u + \alpha v)^q a^{-1} = \frac{u + \alpha^q v}{a}.$$

The constant α^q can be precomputed. Usually α is a square root of some quadratic nonresidue in $\mathbb{F}_q$, so $\alpha^q = -\alpha$ and this step is essentially a single division.

Then compute $b^{(q+1)/r}$ using a standard exponentiation algorithm to obtain $a^{(q^2-1)/r}$. We have effectively halved the size of the exponent.

Let us also work through the $k = 6$ case. Suppose we have $\mathbb{F}_{q^6}$ implemented as $\mathbb{F}_q[\alpha]$. Then

$$q^6 - 1 = \Phi_6(q)(q^4 + q^3 - q - 1)$$

(where $\Phi_6(q) = q^2 - q + 1$). If $a = u_0 + u_1\alpha + ... + u_5\alpha^5$ we have

$$b = a^{q^4+q^3-q-1} = \frac{(u_0 + u_1\alpha^{q^4} + ... + u_5\alpha^{5q^4})(u_0 + u_1\alpha^{q^3} + ... + u_5\alpha^{5q^3})}{(u_0 + u_1\alpha^q + ... + u_5\alpha^{5q})a}$$

where each power of α^q can be precomputed. Then exponentiate b by $(q^2 - q + 1)/r$ using a standard algorithm. In this case we have shrunk the exponent to roughly one third its original size.

The $k = 12$ case (which occurs for the type F pairing) is similar. We need to compute $a^{(q^12-1)/r}$ for some $a \in \mathbb{F}_{q^{12}}$ for some prime q and group order r. This is best done by computing $b = a^{q^8+q^6-q^2-1}$ followed by exponentiating b by $\Phi_{12}(q)/r = (q^4 - q^2 + 1)/r$.

Suppose k is even but not divisible by four. Let $k = 2d$ where d is odd. Suppose $\mathbb{F}_{q^d}$ has been implemented as $\mathbb{F}_q[\alpha]$, and $\mathbb{F}_{q^k}$ as $\mathbb{F}_{q^d}[\beta] = \mathbb{F}_q[\alpha, \beta]$. where β is some quadratic nonresidue in $\mathbb{F}_q$.

Then every element $a \in \mathbb{F}_{q^k}$ can be written in the form

$$a = (u_0 + v_0\beta) + (u_1 + v_1\beta)\alpha + ... + (u_{d-1} + v_{d-1}\beta)\alpha^{d-1}$$

Note $\beta^q = -\beta$. Thus for even m we have

$$a^{q^{d+m}} = (u_0 - v_0\beta) + (u_1 - v_1\beta)\alpha^q + ... + (u_{d-1} - v_{d-1}\beta)\alpha^{q^{m-1}}$$

and for odd m we have

$$a^{q^{d+m}} = (u_0 + v_0\beta) + (u_1 + v_1\beta)\alpha^q + ... + (u_{d-1} + v_{d-1}\beta)\alpha^{q^{m-1}}$$

allowing the following simplifications.

When $k = 6$ then any $a \in \mathbb{F}_{q^k}$ can be written in the form

$$a = (u_0 + v_0\beta) + (u_1 + v_1\beta)\alpha + (u_2 + v_2\beta)\alpha^2.$$

Then $b = a^{q^4+q^3-q-1}$ can be computed via

$$\frac{\left((u_0 + v_0\beta) + (u_1 + v_1\beta)\alpha^q + (u_2 + v_2\beta)\alpha^{2q}\right)\left((u_0 - v_0\beta) + (u_1 - v_1\beta)\alpha + (u_2 - v_2\beta)\alpha^2\right)}{\left((u_0 - v_0\beta) + (u_1 - v_1\beta)\alpha^q + (u_2 - v_2\beta)\alpha^{2q}\right)\left((u_0 + v_0\beta) + (u_1 + v_1\beta)\alpha + (u_2 + v_2\beta)\alpha^2\right)}$$

A similar formula applies when $k = 10$ (type G pairings). In this case we compute $b = a^{q^6+q^5-q-1}$, before exponentiating by $\Phi_5(q)/r = (q^4 - q^3 + q^2 - q + 1)/r$.

6.10 Weil Denominator Elimination

Let f be a Weil pairing

$$f : E[r] \times E[r] \to \mathbb{F}_{q^k}$$

for some curve E over $\mathbb{F}_q$ with embedding degree $k > 1$, and for some r.

For pairing-based cryptography, any nondegenerate bilinear map can be used. Thus we may replace f by f^n where n is prime to r. The map f^n is still nondegenerate and bilinear; it merely differs from the Weil pairing by a constant factor.

In particular, if we choose $n = q - 1$ then denominator elimination also applies to the Weil pairing. Additionally, the above powering trick makes the exponentiation cheap.

6.11 Preprocessing

We previously discussed preprocessing for exponentiation in a group. In general opportunities for calculating and storing certain results ahead of time frequently arise.

One trivial application of this principle is generating and storing random points and number needed for certain pairing computations long before they are needed, reusing them when possible. Another is the caching of quadratic nonresidues that are key ingredients in a number of algorithms.

In many pairing-based cryptosystems, a group element such as a system parameter or a key that rarely changes is fed to the pairing over and over again in typical use, behaviour that we can exploit with precomputation.

For example, in the BLS signature scheme, a system parameter and signer's public key are given to the pairing during verification, thus an application that verifies many BLS signatures from the same sender will be a good candidate for this optimization.

6.11.1 Precomputation of Lines

During Miller's algorithm, the coefficients in equations of the form $aX+bY+c$ are calculated. These lines are derived entirely from one of the input points, hence much time can be saved every time the same input point is encountered if we precompute and store a, b, c for every line [4].

6.11.2 Elliptic Net Precomputation

One of the two sequences in the Shipsey-Stange algorithm, which we denoted by c_k is completely determined by the first input. Thus caching c_k^2 and $c_{k-1}c_{k+1}$ for later pairings will improve the running time substantially.

6.12 Compressed Pairings

In the next few sections we quote without proof observations due to Scott and Barreto [59] that speed up pairings and reduce their output size, though in some cases at the cost of losing a few bits.

We first examine the simplest case. Suppose we have constructed a pairing with an elliptic curve E over a field $\mathbb{F}_q$ with embedding degree 2, so that the output of the pairing is an element of order r in $\mathbb{F}_{q^2}$, where r is the order of the cyclic subgroup being used.

Suppose $\mathbb{F}_{q^2}$ has been implemented as $\mathbb{F}_q[\alpha]$. A typical choice is $\alpha = i(= \sqrt{-1})$.

From above, the last step of the optimized Tate exponentiation consists of exponentiating a number of the form $a + \alpha b$ by $(q + 1)/r$.

It can be shown $a + \alpha b$ must be *unitary* (the proof applies to any element of $\mathbb{F}_{q^2}$ whose order divides $q + 1$), which is to say $a^2 - \alpha^2 b^2 = 1$. This in turn implies *Lucas sequences* can be used to compute powers. Let $P = 2a$.

Define

$$V_0 = 2, V_1 = P, V_{n+1} = PV_n - V_{n-1}.$$

It turns out that $(a + \alpha b)^n = V_n/2 + \alpha bU$, where $U = (PV_n - 2V_{n-1})/(P^2 - 4) = (2V_{n+1} - PV_n)/(P^2 - 4)$, and that the following algorithm computes $v_0 = V_n$ and $v_1 = V_{n+1}$ for even n, and $v_0 = V_{n-1}$ and $v_1 = V_n$ otherwise. We can easily compute U in either case.

Algorithm 13 Lucas sequence: $v_0 = V_m$ and $v_1 = V_{m+1}$, where $m = n$ for even n and $m = n - 1$ for odd n

```
Let $n_t ... n_0$ be the binary respresentation of $n$.
$v_0 \leftarrow 2, v_1 \leftarrow P, j \leftarrow t$
while $j > 0$ do
  if $n_j = 1$ then
    $v_0 \leftarrow v_0 v_1 - P, v_1 \leftarrow v_1^2 - 2$
  else
    $v_1 \leftarrow v_0 v_1 - P, v_0 \leftarrow v_0^2 - 2$
  end if
  $j \leftarrow j - 1$
end while
$v_1 \leftarrow v_0 v_1 - P, v_0 \leftarrow v_0^2 - 2$
```

Every pairing-based cryptosystem yet proposed requires r to be an odd prime or a product of two odd primes, thus we assume r is odd. As we avoid characteristic 2 fields, we also assume $q + 1$ is even, so we can assume $n = (q + 1)/r$ is even.

Clearly the above is faster than a standard $\mathbb{F}_{q^2}$ exponentiation procedure as all operations take place in the smaller field $\mathbb{F}_q$.

This method applies to any curve of even embedding degree, but we shall find that for embedding degree $k = 6$ an even better algorithm exists. However, before describing it, we first discuss pairing compression.

6.13 Pairing Compression For Even Embedding Degree

We have seen that the output of a pairing is some unitary element $a + \alpha b \in \mathbb{F}_{q^2}$. Hence for each value of a, there are two possibile values for b since $a^2 - \alpha^2 b^2 = 1$.

Just as point compression works by recording only the x-coordinate and one bit that signifies which y-coordinate to take, we may compress pairing values by recording only the a value and one bit that represents which solutions of $a^2 - \alpha^2 b^2 = 1$ the b takes.

To invert a group element on an elliptic curve is to negate the y-coordinate. Similarly, we have $(a + \alpha b)^{-1} = a - \alpha b$ for pairing values.

Furthermore, in some applications it may be possible to dispose of b entirely and not bother recording which solution to take. This is similar to the point reduction described earlier, and in fact, these two tricks work well in unison.

The BLS signature scheme is a good example of this. Recall a signature is an x-coordinate of some point P and we need to check if $e(P, Q)$ is a certain pairing value v, where Q is a system parameter. Since we have discarded the y-coordinate, when guessing a value for P we may have in fact selected $-P$, in which case we will have computed $e(-P, Q)$ instead, and previously we recommended checking the other possible y-coordinate in event of a mismatch.

But observe that $e(P, Q)$ and $e(-P, Q)$ will have the same a-value, and are the only pairing values that share this a-value. This suggests the following BLS signature verification procedure:

1. Given a signature σ, compute any point P with x-coordinate σ.
2. Compute the a-value of $e(P, Q)$, where Q is the system parameter. Do not bother with the b-value.
3. If this value matches the a-value of v then the signature verifies. Otherwise it is rejected.

6.14 Pairing Compression For Embedding Degree Six

Now consider a curve $E(\mathbb{F}_q)$ with embedding degree $k = 6$. Recall from the discussion of Tate exponentiation that $r \mid \Phi_6(q) = q^2 - q + 1$ (and r does not divide any smaller cyclotomic polynomial).

These are precisely the conditions that occur in the XTR cryptosystem [44], the optimizations and algorithms of which we quote here.

Let tr denote the $\mathbb{F}_{q^2}$-trace in $\mathbb{F}_{q^6}$, that is

$$\operatorname{tr}(x) = x + x^{q^2} + x^{q^4} \in \mathbb{F}_{q^2}.$$

Let $x \in \mathbb{F}_{q^6}$ be an element of order r (such as the output of a pairing). Define $c_k = \operatorname{tr}(x^k)$. It can be shown that

$$c_{u+v} = c_u c_v - c_v^q c_{u-v} + c_{u-2v}$$

for all $u, v \in \mathbb{Z}$, which leads to the identities:

$$c_{n+2} = c_1 c_{n+1} - c_1^q c_n + c_{n-1}$$

$$c_{2n} = c_n^2 - 2c_n^q$$

$$c_{2n-1} = c_n c_{n-1} - c_1 c_n^q + c_{n+1}^q$$

$$c_{2n+1} = c_n c_{n+1} - c_1 c_n^q + c_{n-1}^q$$

Then for any integer n, we can use a repeated-squaring-like algorithm to compute $\operatorname{tr}(x^n)$ from $\operatorname{tr}(x)$.

One can show $\operatorname{tr}(x^n) = (\operatorname{tr}(x^{-n}))^q$ so without loss of generality assume $n \geq 0$. Trivially $c_0 = 3, c_1 = \operatorname{tr}(x)$ and c_2, c_3, c_4 can be computed easily using the above identities, e.g. $c_2 = c_1^2 - 2c_1^q$. Otherwise:

1. If n is odd let $2m + 1 = n$, otherwise let $2m = n$, and let the binary representation of m is $m_t...m_0$.
2. $k \leftarrow 1$.
3. For $j \leftarrow t - 1$ to 0 do
    (a) If $m_j = 0$ then compute $c_{4k}, c_{4k+1}, c_{4k+2}$ from $c_{2k}, c_{2k+1}, c_{2k+2}$, using the above identities.
    (b) Otherwise $m_j = 1$ and compute $c_{4k+2}, c_{4k+3}, c_{4k+4}$ from $c_{2k}, c_{2k+1}, c_{2k+2}$, using the above identities.
    (c) $k \leftarrow 2k + m_j$

4. We have now computed $c_{2m}, c_{2m+1}, c_{2m+2}$. (If n is odd $c_n = c_{2m+1}$ otherwise $c_n = c_{2m}$.)

Thus we can compress the output x of a pairing by a factor of three by using $\operatorname{tr}(x)$ instead of x, and the above shows how to find $\operatorname{tr}(x^n)$ for any integer n, a feature often required by pairing-based cryptosystems. (In general $\operatorname{tr}(x)^n \neq \operatorname{tr}(x^n)$ so we cannot use a standard exponentiation algorithm.) Of course, in doing so we lose some information: x, x^{q^2}, x^{q^4} all have the same trace, but this is tolerable in most cases.

Recall from Section 5.7 that if $q = 2 \pmod 3$ or $q = 3 \pmod 4$, then with a suitable constructed field extension $\mathbb{F}_{q^2}$, for $x, y, z \in \mathbb{F}_{q^2}$:

1. computing x^q is free
2. computing x^2 costs 2 multiplications in $\mathbb{F}_q$
3. computing xy costs 3 multiplications in $\mathbb{F}_q$
4. computing $xz - yz^q$ costs 4 multiplications in $\mathbb{F}_q$

where we assume the time taken by a few additions and subtractions is negligible.

Since these are the operations aside from addition and subtraction involved in the above identities, exponentiating compressed pairings is significantly faster for carefully constructed field extensions.

6.15 Powered Pairings

Michael Scott notes that in many cryptosystems the result of a pairing is raised to some power at some stage. In some cases the output of a pairing is not used until after it has been exponentiated. Thus when designing a pairing library, one should make a powered pairing function available to the user, to take full advantage of the above optimizations that allow faster exponentiation of pairing outputs.

Also, for any m coprime to the group order r, the mth power of a pairing is a nondegenerate bilinear map. Thus in any cryptosystem we can replace the pairing with the mth power of the pairing.

One application of this is to replace a Weil pairing e of even embedding degree $k = 2d$ in any cryptosystem with e^{q^d-1}, which means denominator elimination can be applied during the computation of the Weil pairing. Recall raising to the qth power is a cheap operation, so this will speed up the Weil pairing [43].

In fact, this optimization makes it less clear which pairing is faster. The Tate pairing costs one Miller-Lite operation and a final powering. The powered Weil pairing costs one Miller-Lite and one Miller-Full operation, and needs no final powering. In both cases, denominator elimination applies. Without final powering optimizations, a Miller-Full might outperform it, in which case the Weil pairing would beat the Tate pairing [36].

6.16 Exponentiation Tricks

Miller's algorithm has features in common with exponentation by repeated squaring. Accordingly, tricks that speed up the latter can be adapted for the former.

Let us analyze the analogue of division in Miller's algorithm. Using the notation of Chapter 3, we have

$$(f_{-k}) = (P)^{-k}/(-kP)$$

hence

$$f_{-k}(Q) = \frac{1}{f_k(Q)V_k(Q)}$$

and

$$f_{a-b}(Q) = \frac{f_a(Q)L_{a,-b}(Q)}{f_b(Q)V_b(Q)V_{a-b}(Q)}$$

Since group inversion, i.e. point negation, is practically free, we may neglect the cost of computing $-kP$ from kP. We may treat the cost of computing $(a-b)P$ from aP and bP as the same as the cost of computing $(a+b)P$.

We see a division in Miller's algorithm is slightly more expensive than a multiplication, requiring also an inversion in G_T along with another vertical line.

Denominator elimination allows us to ignore vertical lines. Also, the inversions can be collated. We can defer divisions, and have one inversion at the end instead of one per iteration, by maintaining numerator and denominator variables during the main loop and only dividing at the end. (In other words, compute $(g_1(Q)...g_m(Q))/(h_1(Q)...h_m(Q))$ instead of $(g_1(Q)/h_1(Q))...(g_m(Q)/h_m(Q))$.)

Thus addition-subtraction-chain exponentiation can be better than plain addition-chain exponentiation and as with point multiplication, one can employ signed sliding windows [47, Chapter 14] [9, Section IV.2.5].

As mentioned before the last iteration of Miller's algorithm can be skipped if denomination elimination is applied. This is also true for the signed representation, because

$$f_{r+1} = f_r L_{rP}/V_{r+1}P$$

but since $rP = O$ we have that L_{rP} is a vertical line at P and hence $f_{r+1} = f_r$ (recall we ignore verticals for denominator elimination).

Sometimes it is possible to choose the order r of the input and output groups. In such cases it is desirable to pick r so that the resulting addition-subtraction-chain is as short as possible. For example, a good choice is an r that has the form $2^a \pm 2^b \pm 1$ (Solinas numbers) or has low Hamming weight [4].

Like multiexponentiation, when computing products or quotients of pairings we benefit from using vector addition chains [47, Chapter 14], with much time saved by using a precomputed table.

Although we recommend avoiding characteristic 3 curves, we note some special optimizations for them [4]. Firstly, point tripling is extremely fast. Given a point (x, y) we can quickly compute $(x_3, y_3) = 3(x, y)$ via

$$\begin{aligned} x_3 &= (x^3)^3 - b \\ y_3 &= -(y^3)^3 \end{aligned}$$

since cubing is cheap in characteristic 3 (here b is the constant term of the elliptic curve). Then if we use signed ternary representation point multiplication can be sped up. Furthermore, if r is chosen to a base 3 analogue of a Solinas number $3^a \pm 3^b \pm 1$, then Miller's algorithm is much faster.

6.17 Higher Degree Twists

We quote facts neatly summarized by Hess, Smart and Vercauteren [38].

Let E be an elliptic curve over $\mathbb{F}_q$ where $q \geq 5$ is prime. Let t be the trace of Frobenius, that is, t satisfies $\#E(\mathbb{F}_q) = q - t + 1$. Table 6.1 describes the twists of E for various choices of $v \in \mathbb{F}_q^*$. Some choices of v result in the original curve (or rather a curve easily mapped to the original), while others lead to quadratic, cubic, quartic or sextic twists, which are depicted in the table. We define w, the degree of the twist, as the smallest integer such that

Curve	w	Twist	Twist Map	Point Count
$Y^2 = X^3 + aX + b$	2	$Y^2 = X^3 + a/v^2X + b/v^3$	$vx, v^{3/2}y$	$q + 1 + t$
$Y^2 = X^3 + aX$	2 4	$Y^2 = X^3 + a/vX$	$v^{1/2}x, v^{3/4}y$	$q + 1 + t$ $q + 1 \pm f$ $(t^2 - 4q = -f^2)$
$Y^2 = X^3 + b$	2 3 6	$Y^2 = X^3 + b/v$	$v^{1/3}x, v^{1/2}y$	$q + 1 + t$ $q + 1 - (\pm 3f - t)/2$ $q + 1 - (\pm 3f + t)/2$ $(t^2 - 4q = -3f^2)$

Table 6.1: Twist curves

the twist curve maps to $E(\mathbb{F}_{q^w})$ by taking a point (x, y) to the point shown in the table. We must have $q = 1 \bmod w$ for a twist of degree w to exist.

A simple method to find a desired twist is to

1. Choose $v \in \mathbb{F}_q^*$ and construct the twist E' using v.

2. Generate a random point $P \in E'$.

3. If P does not have the order specified by Table 6.1 then goto step 1.

The twist curve optimization tricks discussed earlier correspond to quadratic twists. For this case, Hess, Smart and Vercauteren note the alternate form:

$$E : vY^2 = X^3 + aX + b$$

with

$$\phi : (x, y) \mapsto (x, v^{1/2}y)$$

may be better from a programmer's point of view.

We can generalize our method to higher-degree twists on certain curves (and embedding degrees), leading to greater time and space savings.

Let G be a cyclic subgroup of $E(\mathbb{F}_q)$ of prime order $r \geq 5$ and embedding degree k. In all cases, it can be shown exactly one of the twist curves has order that is a multiple of r. Let w be the degree of this twist.

We assume that w divides k. When this is not the case, we can use a suitable factor of w instead with reduced savings. In the extreme case, we have $E' = E$ and $d = k$ which implies we are not using the twist curve optimization at all and gain nothing. Let $d = k/w$. Then as before, we can work within the groups $E(\mathbb{F}_q)[r]$ and $E'(\mathbb{F}_{q^d})$ for a suitable twist curve E' at all times except during a pairing computation, where we use the twist map and operate in $\mathbb{F}_{q^k}$.

6.18 The Ate and Twisted Ate Pairing

Barreto et al. discovered a variation on the Tate pairing, dubbed the Eta pairing, that can be more efficient in some cases [2]. Hess, Smart and Vercauteren further refine this new pairing and obtain the Ate pairing [38]. We state their results without proof.

Let E be an elliptic over $\mathbb{F}_q$ containing a cyclic subgroup G_1 of order r and even embedding degree k. Let G_2 be the group of trace zero points in $E'(\mathbb{F}_{q^d})$ where, like the previous section, E' and d represent the twist curve optimization being employed. Let t be the trace of Frobenius, that is t satisfies $\#E(\mathbb{F}_q) = q - t + 1$.

Let $f_{n,P}$ be a rational function with divisor $n(P)/(nP)$ for any integer n and any point P. Then the following are bilinear and nondegenerate up to coset representatives (in practice one needs to execute a final powering on the output of any of these functions).

1. Tate pairing:
$$f_{r,P}(Q)$$
2. Ate pairing:
$$f_{t-1,Q}(P)$$
3. Twisted Ate pairing:
$$f_{(t-1)^d,P}(Q)$$

In some cases, the Ate or twisted Ate pairing may be faster than the Tate pairing. Note that even when $t - 1$ is much smaller than r, the Ate pairing may not necessarily be faster because it requires a Miller-Full operation.

Chapter 7

Summary of Contributions

We briefly reiterate our main original contributions.

7.1 Abstract definitions

In Chapter 1 we gave abstract definitions of pairings that bridge the gap between formal security proofs and bilinear maps used in practice. We began with a symmetric definition, which was the first to appear in the literature, and gradually extended it so that a greater variety of pairings are available.

We presented examples of assumptions that pairing-based cryptosystems rely on, and indicated how to modify them for different pairing definitions.

7.2 The BLS Signature Scheme

We exhibited a practical digital signature scheme with the shortest known signature length at typical security levels. This signature scheme, often referred to as the BLS signature scheme, has many other desirable features [17, 15], but they fall outside the scope of this text. The BLS signature scheme is a pairing-based cryptosystem, and many of these features rely heavily on properties of bilinear maps. It is not known how to construct signature schemes with similar advantages without using pairings.

7.3 Constructing With Prescribed Embedding Degree

We presented the first published method for constructing cryptographically useful pairings with any given embedding degree. Inspired by the work of Miyaji et al. [50], we use cyclotomic polynomials to guarantee certain conditions are met, yielding cryptographically-usedful pairings with any given embedding degree [5].

7.4 Optimizations

In the last chapter we showed how to improve the running time of a pairing by roughly a factor of four over a naive implementation.

Firstly, we proved that the classic definition of the Tate pairing may be replaced by a simpler version that can be evaluated in half the time [4].

Secondly, using twist curves we can ignore denominators in Miller's algorithm, doubling its speed [6]. This technique is now so commonplace that it is considered "standard" [38].

We also described methods for efficiently computing the final powering, a costly operation that is required to standardize coset representatives in the output group [4].

Additionally, we gave more minor optimizations that together substantially speed up a pairing computation, and pairing-based cryptosystems in general [4, 6].

Bibliography

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

[2] P. Barreto, S. Galbraith, C. O'hEigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties, 2004. `http://eprint.iacr.org/2004/375`.

[3] P. S. L. M. Barreto. The pairing-based cryptography lounge. `http://paginas.terra.com.br/informatica/paulobarreto/pblounge.html`.

[4] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 354–368, London, UK, 2002. Springer-Verlag.

[5] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Third Conference on Security in Communication Networks*, 2002.

[6] P. S. L. M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In *Proceedings of Selected Areas in Cryptography – SAC*, 2003.

[7] P. S. L. M. Barreto, B. Lynn, and M. Scott. Efficient implementation of pairing-based cryptosystems. *Journal of Cryptology*, 17(1):321–334, 2004.

[8] D. J. Bernstein. Faster square roots in annoying finite fields. http://cr.yp.to/papers/sqroot.ps.

[9] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, July 1999.

[10] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, 2004.

[11] D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, 2004.

[12] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO 2004*. Springer-Verlag, 2004.

[13] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *Lecture Notes in Computer Science*, 2139:213+, 2001.

[14] D. Boneh, C. Gentry, and M. Hamburg. Space-efficient identity-based encryption without pairings. Cryptology ePrint Archive, Report 2007/177, 2007. `http://eprint.iacr.org/`.

[15] D. Boneh, C. Gentry, H. Shacham, and B. Lynn. Aggregate and verifiably encrypted signatures from bilinear maps. In *In E. Biham, editor, Proceedings of Advances in Cryptology – Eurocrypt'03, LNCS. Springer-Verlag*, 2003.

[16] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography '05*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer-Verlag, 2005.

[17] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Asiacrypt*, volume 2248 of *Lecture Notes in Computer Science*, pages 514+, 2001.

[18] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(1):297–319, 2004.

[19] F. Brezing and A. Weng. Elliptic curves suitable for pairing-based cryptography. `http://eprint.iacr.org/2003/143`.

[20] C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[21] C. Cocks and R. G. E. Pinch. Identity-based cryptosystems based on the Weil pairing. unpublished manuscript, 2001.

[22] Cohen, Miyaji, and Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 1998.

[23] H. Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[24] D. Coppersmith. Fast evaluation of logarithms in fields of characteristics two. In *IEEE Transactions on Information Theory*, volume 30, pages 587–594, 1984.

[25] H. Coxeter and G. Beck. *The Real Projective Plane*. Springer-Verlag, 1992.

[26] C. Diem. The ghs attack in odd characteristic. In *J. Ramanujan Math. Soc.*, volume 18, pages 1–32, 2003.

[27] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[28] D. Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. In *ANTS VII*. LNCS 4076, Springer-Verlag, 2006.

[29] D. Freeman. Constructing pairing-friendly genus 2 curves over prime fields with ordinary jacobians. preprint, 2007. `http://math.berkeley.edu/~dfreeman/papers/pairing-friendly-genus2.pdf`.

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

[31] G. Frey, M. Muller, and H.-G. Ruck. The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *Trans. on Inf. Th.*, 45:1717–1719, 1999.

[32] G. Frey and H. Ruck. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. of Computaions*, 62:865–874, 1994.

[33] S. Galbraith. Supersingular curves in cryptography. In *Asiacrypt*, volume 2248 of *Lecture Notes in Computer Science*, pages 495–513. Springer-Verlag, 2001.

[34] S. Galbraith and N. P. Smart. A cryptographic application of Weil descent. In M. Walker, editor, *Cryptology and Coding*, volume 1746 of *LNCS*, pages 191–200. Springer-Verlag, 1999.

[35] P. Gaudry, F. Hess, and N. P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. Technical Report CSTR-00-016, Department of Computer Science, University of Bristol, 2000.

[36] R. Granger, D. Page, and N. Smart. High security pairing-based cryptography revisited. Cryptology ePrint Archive, Report 2006/059, 2006. `http://eprint.iacr.org/`.

[37] T. Granlund. The GMP library. `http://www.swox.com/gmp/`.

[38] F. Hess, N. Smart, and F. Vercauteren. The eta pairing revisited. In *IEEE Transactions on Information Theory*, volume 52, pages 4595–4602, 2006.

[39] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *Proceeding of Advances in Cryptology – Eurocrypt'02, LNCS. Springer-Verlag*, pages 466–481, 2002.

[40] A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Proc. Fourth Algorithmic Number Theory Symposium*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, 2000.

[41] A. Joux. The Weil and Tate pairings as building blocks for public key cryptosystems. In *Proc. Fifth Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science. Springer-Verlag, 2002.

[42] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming.* Addison-Wesley, Reading, Massachusetts, second edition, 10 Jan. 1981.

[43] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. Cryptology ePrint Archive, Report 2005/076, 2005. `http://eprint.iacr.org/`.

[44] A. K. Lenstra and E. R. Verheul. The XTR public key system. In *CRYPTO*, 2000.

[45] B. Lynn. Authenticated identity-based encryption. Cryptology ePrint Archive, Report 2002/072, 2002. `http://eprint.iacr.org/`.

[46] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 80–89, New York, NY, USA, 1991. ACM Press.

[47] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[48] V. Miller. Short functions for programs on curves. unpublished manuscript.

[49] V. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–262, 2004.

[50] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 2001.

[51] T. Okamoto and D. Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography*, pages 104–118, 2001.

[52] D. Page, N. P. Smart, and F. Vercauteren. A comparison of MNT curves and supersingular curves. Cryptology ePrint Archive, Report 2004/165, 2004. `http://eprint.iacr.org/`.

[53] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1997 (?).

[54] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. Technical Report MIT/LCS/TM-82, 1977.

[55] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. *Lecture Notes in Computer Science*, 2248:552+, 2001.

[56] K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In *Advances in Cryptology – Crypto 2002*, volume 2442 of *Lecture Notes on Computer Science*, pages 336–353. Springer-Verlag, 2002.

[57] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, Okinawa, Japan, 2000.

[58] M. Scott. MIRACL. `http://www.shamus.ie/`.

[59] M. Scott and P. S. L. M. Barreto. Compressed pairings. In *Crypto 2004*, 2004.

[60] M. Scott and P. S. L. M. Barreto. Generating more MNT elliptic curves. Cryptology ePrint Archive, Report 2004/058, 2004. `http://eprint.iacr.org/`.

[61] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - Crypto '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1984.

[62] J. H. Silverman. *The arithmetic of elliptic curves.* Springer-Verlag, Berlin, 1995.

[63] K. E. Stange. The Tate pairing via elliptic nets. Cryptology ePrint Archive, Report 2006/392, 2006. `http://eprint.iacr.org/`.

[64] D. R. Stinson. Some baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. In *Math. Comput 71(237)*, pages 379–391, 2002.

[65] E. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. *Journal of Cryptology*, 17(4):277–296, 2004.

[66] A. Weimerskirch and C. Paar. Generalizations of the Karatsuba algorithm for polynomial multiplication.