Cryptographic Pairings: Efficiency and DLP Security

Naomi Benger

A dissertation submitted to Dublin City University in accordance with the requirements for the award of Doctor of Philosophy (Ph.D.) in the Faculty of Computer Applications, School of Computing

Supervisor: Prof. Michael Scott

May 2010

Claude Shannon Institute
Discrete Mathematics, Coding, Cryptography and Information Security
www.shannoninstitute.ie

sfi
science foundation ireland
fondúireacht eolaíochta éireann

Declaration

I, Naomi Judith Benger, hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed Student ID: 57112207 May 12th 2010

Contents

Abstract

This thesis studies two important aspects of the use of pairings in cryptography, efficient algorithms and security.

Pairings are very useful tools in cryptography, originally used for the cryptanalysis of elliptic curve cryptography, they are now used in key exchange protocols, signature schemes and Identity-based cryptography.

This thesis comprises of two parts: Security and Efficient Algorithms.

In Part I: Security, the security of pairing-based protocols is considered, with a thorough examination of the Discrete Logarithm Problem (DLP) as it occurs in PBC. Results on the relationship between the two instances of the DLP will be presented along with a discussion about the appropriate selection of parameters to ensure particular security level.

In Part II: Efficient Algorithms, some of the computational issues which arise when using pairings in cryptography are addressed. Pairings can be computationally expensive, so the Pairing-Based Cryptography (PBC) research community is constantly striving to find computational improvements for all aspects of protocols using pairings. The improvements given in this section contribute towards more efficient methods for the computation of pairings, and increase the efficiency of operations necessary in some pairing-based protocols.

Acknowledgements

The last 3 years have seen many changes for me, both personally and professionally. Without the friendship and support of many people, writing this thesis would not have been possible.

Firstly, I would like to thank my supervisor, Mike Scott, for giving me the opportunity to learn from him. In my time at DCU I have learnt a lot about researching and writing. I am grateful for his patience as a teacher and infectious enthusiasm. I have enjoyed (nearly) every minute of being a Ph.D. student.

I am thankful to Rob Granger for his invaluable advice, interesting discussions and general support and friendship, especially throughout the last leg of my Ph.D. and thesis writing.

It has been a pleasure working with all the DCU researchers and friends, past and present, Anna Johnston, Paulo Barreto, Neil Costigan, Hyun Sung Kim, Ezekiel Kachisa, Chen Yu and Manuel Charlemagne. In particular, the office would not have been as fun without Luis Julian Dominguez Perez or Denis Butin. Thanks to Luis for the BBQs and cooking lessons and Denis for the language classes (excuses for tea and cake!) and intense thesis editing.

I really appreciated the opportunity to work with David Freeman; he taught me a lot about how to approach some research problems. I also had a lot of support from Fre Vercauteren while developing some ideas and am thankful for his patience and readiness to help. I am grateful to everyone who has worked with me, overlooked my (sometimes) silly questions and encouraged my enthusiasm for research.

Being a member of the Claude Shannon Institute has been a very rewarding experience, I have received a lot of support from everyone, particularly Gary McGuire and Elva O'Sullivan. I have enjoyed the collaborations, workshops, discussions with many other members of the Claude Shannon Institute.

The research I have undertaken was supported financially by the Claude Shannon Insti-

tute and the Science Foundation of Ireland, Grant 06/MI/006.

Working at DCU meant I had to move across the world, leaving my family in Australia. I can't wait to see them again, particularly my nieces who have grown so much and one I am still yet to meet! Thanks to my sister for the constant photo updates!

I am also lucky enough to have wonderful friends in Australia who I can always count on. No matter where we are in the world, we can always rely on each other. It's been great having breakfast/dinner parties with them using Skype (my breakfasts with their dinners), but I can't wait to sit with them all again and have a BBQ in the back yard, without the constant fear of rain!

I would like to particularly thank my friend Eric; without his encouragement and competition I would never have pushed myself to come this far. His friendship, support and humour have helped me to realise my ambition and find a renewed enjoyment of learning and research.

I have had such a great time in Dublin, thanks to all my new friends here who helped me to feel at home. All the Ska parties, hidden pubs, secret gigs, Guinness and cliff walks; I have learnt so much from you all about the world, friendship and myself.

Also thanks to my housemates and 'semi-housemates' who have made living in Dublin fun. Alberto, for driving me to the point of insanity; and Tim, Marie, Gabi, Amira and Mishan, for keeping me sane.

Most of all, I am thankful to my best friend and husband Phil, who makes all places feel like home. I am glad he has been with me for every step of this crazy journey!

List of Tables

List of Algorithms

Publications

Publications resulting from the research conducted are:

- *Constructing Tower Extensions for the implementation of Pairing-Based Cryptography*, N. Benger and M. Scott, to appear in the proceedings of *WAIFI 2010*, Lecture Notes in Computer Science, Springer [13].

  In this article a method for the construction of extensions fields for efficient implementation of Pairing-Based Cryptography is given, presented in §8.

- *On the security of pairing-friendly abelian varieties over non-prime fields*, N. Benger, M. Charlemagne and D. Freeman in H. Shachan and B. Waters, editors, *Pairing-Based Cryptography – Pairings 2009*, volume 5671 of Lecture Notes in Computer Science, pages 52–65, 2009 [12].

  This paper outlines results on the relationship between the two instances of the Discrete Logarithm Problem as they occur in Pairing-Based Cryptography, discussed in §4.

- *On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves*, M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez and E. J. Kachisa in H. Shacham and B. Waters, editors, *Pairing-Based Cryptography – Pairings 2009*, volume 5671 of Lecture Notes in Computer Science, pages 78–88, 2009 [86].

  Faster implementation of the final exponentiation step of the pairing calculation, as given in §9

- *Fast Hashing to G_2 on Pairing-Friendly Curves*, M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez and E. J. Kachisa in H. Shacham and B. Waters, editors, *Pairing-Based Cryptography – Pairings 2009*, volume 5671 of Lecture Notes in Computer Science, pages 102-113, 2009 [85].

An improved method for performing the cofactor multiplication necessary in some Pairing-Based protocols, presented in §10

Chapter 1

Introduction

Begin at the beginning and go on till you come to the end: then stop.

- The King to the White Rabbit, Lewis Carroll's *Alice in Wonderland*,
Chapter XII, Alice's Evidence.

In this chapter, we introduce the goals and results of the research undertaken. We begin with a general introduction to cryptology and the important role it plays in today's society in Section 1.1. In Section 1.1.2 we present some examples of *Public-Key Cryptography* and introduce *Pairing-Based Cryptography* in Section 1.1.4. In Section 1.2, we present the motivation behind the work undertaken for the assembly of this thesis and outline the main results contained herein.

1.1 Cryptology

Cryptology is, in a sense, the study of secrets. More specifically the study of the methods of hiding and trying to retrieve secrets. The word 'cryptology' is derived from the Greek words $\kappa\rho\upsilon\pi\tau\acute{o}\varsigma$ (kryptós), meaning "hidden" and $\lambda\acute{o}\gamma o\varsigma$ (lógos), meaning "word".

There are two main areas of cryptology; *cryptography*, from the Greek kryptós and $\gamma\rho\acute{\alpha}\varphi\omega$ (gráphein) meaning "to write"; and *cryptanalysis*, from the Greek 'kryptós' and $\acute{\alpha}\nu\alpha\lambda\acute{\upsilon}\varepsilon\iota\nu$ (analýein) meaning "to solve a problem" or "to untie". As the name suggests, the aims in cryptography are to write, and devise ways of writing, in such a way as to not reveal the message to unintended recipients; to study and design *cryptosystems*. The goal of cryptanalysis is to extract the message from a cryptographically disguised text, without using the intended system decryption method.

Cryptography has been used for centuries; Julius Caesar used what is known as a "shift cipher" to communicate secretly with officials. Since then, there has been much development in cryptography. Since the nineteenth century, the use of cryptography has become more widespread and diverse; arguably spurred by the first and second World Wars, and the subsequent increasing use of computers and electronic communication and commerce. What we now refer to as *Modern Cryptography* is much more elegant than the cryptography of Caesar and pre-World War II, which was primarily used for secrecy; it has become a science incorporating both mathematics and computer science. Modern cryptography uses organised procedures called *protocols* consisting of a set of steps, *primitives*, such as *encrypt* and *decrypt* used in a secret communication protocol or *sign* used in a digital signature protocol.

Some basic principles observed by modern cryptography were given in 1883 by a Dutch linguist, Auguste Kerckhoffs, with astute foresight [53]:

1. The system must be indecipherable in practice;

2. It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience. The security of the system should rely on the key, not on the secrecy of the system;

3. Its key must be communicable and retainable without the help of written notes, and changeable or modifiable at the will of the correspondents;

4. It must be applicable to telegraphic correspondence;

5. It must be portable, and its usage and function must not require the concourse of several people;

6. It is necessary that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

Both cryptography and cryptanalysis are very important aspects of modern cryptology. This is highlighted by the first principle: a cryptographic protocol must be unbreakable in practice. Devising a cryptographic protocol is only one side of the process, it can only be deemed secure if it has withstood and continues to withstand rigorous cryptanalysis.

The constant scrutiny of new and currently used protocols is necessary, in order to be continually aware of the true level of security offered by a particular protocol and to choose the protocol set up, *system parameters*, to give the appropriate level of security required. This is one of the roles of cryptanalysis: testing the security of new cryptographic protocols before they are deployed. Any weakness in a protocol discovered by cryptanalysis can be removed from the system before it is in use so that once the protocol is implemented it is not as vulnerable to malicious attacks.

Cryptography in some form is constantly being used, often obliviously; for example, simply paying using a bank card and a PIN number. How simply we all use bank cards illustrates the observation of Kerckhoffs' sixth principle; a cryptographic protocol should be accessible enough for people with little understanding of the underlying principles to use, and yet also withstand cryptanalysis from specialists with a complete knowledge of all aspects of the system.

A constant challenge for the cryptographic community is to ensure that both the first and sixth principles are always observed. This entails being one step ahead of cryptanalysis, by constructing cryptographic protocols which can withstand attack, whilst keeping the protocols efficient enough to implement in such a way that they are practical to use. It is important that a cryptosystem is easy to implement and use; retrieving cryptographically disguised information should only be a difficult task for an unauthorised reader, that is,

someone who does not have the key.

The second and arguably most important of Kerckhoffs's Principles has come to be known as *Kerckhoffs's Principle*. It states that the security of a cryptosystem should be dependent only on the security of the key and not on the secrecy of the algorithm. The importance of this principle is highlighted by Schneier in [83]:

> *'If the strength of your new cryptosystem relies on the fact that the attacker does not know the algorithm's inner workings, you're sunk. If you believe that keeping the algorithm's inside secret improves the security of your cryptosystem more than letting the academic community analyze it, you're wrong. And if you think that someone won't disassemble your code and reverse-engineer your algorithm, you're naive.'*

Schneier emphasises that it is only a matter of time until system weaknesses are found, it is better to have the system thoroughly analysed before the system is in use. Thus, it should always be assumed that the encryption and decryption primitives being used are publicly known and that lack of knowledge of the *secret key* is what keeps the information being communicated secure from an adversary. It is therefore reasonable to assume that the following are all public knowledge before communication: the message space, $\mathcal{M}$, of all possible messages constructed from a given alphabet; the cipher text space, $\mathcal{C}$, of all possible cipher texts of messages; the key space of all possible keys, $\mathcal{K}$; and the encryption and decryption primitives, $\mathcal{E}_k : \mathcal{M} \to \mathcal{C}$ and $\mathcal{D}_k : \mathcal{C} \to \mathcal{M}$ with respect to some key $k \in \mathcal{K}$ such that $\mathcal{D}_k(\mathcal{E}_k(m)) = m$ for all $m \in \mathcal{M}$.

Uses of Cryptography

Modern cryptography has an important role in today's society. Aside from its most obvious use, secure communication, cryptography has many other roles; keeping medical records and bank account details not only secret, but also uncorrupted, is essential; signing electronic documents is becoming more important as we increasingly rely on the Internet as a channel for communication and business.

Information Storage and Integrity:

Storing confidential information, such as medical records and bank transactions. These records need to be kept not only confidential, but also tamper proof; the manipulation of such records could have catastrophic effects, for example, the administration of incorrect medication.

Signing:

Digital signatures are used to sign documents electronically. When using an insecure channel, a digital signature gives the receiver reasonable grounds to believe that the message was indeed sent by the claimed author. A digital signature is closely linked to the message.

Authentication:

Communication is presumed to be secure once a cryptosystem and session key have been agreed upon, as only parties using the valid decryption function are able to recover the messages. What happens when the person with whom we have set up our system is not actually who he or she claims to be? Authentication is important to prove one's identity before communication begins, so that no impostor can receive confidential information, simply by impersonating a trusted party.

Non-repudiation of Origin:

Non-repudiation is to ensure that no party can falsely deny having taken part in a communication. This is important for placing orders over the Internet, for example, so that a customer can not purchase goods without paying, by denying placing an order.

Whatever the use of a particular cryptographic protocol, all protocols take on one of two forms: *Symmetric* or *Asymmetric*.

1.1.1 Symmetric Cryptography

Private-Key Cryptography, or *Symmetric* Cryptography, is so called because the *keys* used to encrypt and decrypt a message are either equal or easily derived from one another; the key used for any secure communication must therefore be kept secret.

This is the oldest type of cryptography with a famous example; the afore mentioned *shift cipher*, used by Julius Caesar. Using the shift cipher, the letters of a message were shifted three places in the alphabet to obtain the cipher text so the encryption key is 3 and the decryption key is -3. In this small example, it is obvious why it is imperative to keep the key a secret; anyone who knows the key will be able to decrypt a message, whether they were intended to read it or not.

Modern symmetric algorithms are very efficient, but not without drawbacks. One problem is that each pair of parties involved in any communication needs to securely agree on the secret key before communication or the key must somehow be securely communicated. Needing to communicate a secret key is not the only issue with symmetrical cryptography. *Key management* is also a problem. Each pair of parties wishing to communicate needs a common key; a group of t parties thus requires $t(t-1)/2$ keys. These keys must be kept secure and regularly changed to avoid potential security breaches. The communication of new keys again becomes a problem.

1.1.2 Asymmetric Cryptography

The problems of key distribution and storage associated with symmetric cryptography were the motivation behind the concept of *Asymmetric Cryptography*, also referred to as *Public-Key Cryptography*. The first practical public key protocol was introduced by Whitfield Diffie and Martin Hellman in 1976 [24] in the form of a key exchange protocol. The focus of this thesis is an area of asymmetric cryptography. The idea is to use keys k_E and k_D such that it is infeasible to calculate the one from the other. This way k_E, referred to as the *public* key, can be published and anyone wishing to communicate with Alice, for example, just needs to find Alice's public key from a list and encrypt the message; only Alice will be

able to decrypt the message using her corresponding *private* key. This concept solves the problem of securely distributing keys. What's more, in a network of t people, only t keys are needed, a huge improvement on the situation using symmetric cryptography.

Asymmetric cryptography uses so called *one-way* mathematical problems; one-way problems consist of an operation which is easy to compute, but difficult to invert. Using one-way problems we are able to find *key pairs* k_E and k_D with the desired properties.

One such problem is the *Integer Factoring Problem* (IFP): given two large primes p and q, it is easy to multiply them together to find $l = pq$, but given an integer l, the product of two large primes, it is a much more difficult task to factor l to recover p and q.

1.1.3 The Discrete Logarithm Problem

Some terms used in this section are described in detail in Chapter 2, for other definitions the reader is referred to [4, Chap. 2]. The *Discrete Logarithm Problem* (DLP) is a one-way problem used in asymmetric cryptography, particularly in the area of cryptography of interest in this thesis. It is defined in a finite, abelian group $(G, *)$ as follows: Let $\beta \in G$ be in the group generated by α, that is, there exists an $x \in \mathbb{Z}$ such that

$$\beta = \underbrace{\alpha * \alpha * \ldots * \alpha}_{x \text{ times}} \in G.$$

Definition 1.1.1. The DLP in G is: given α and β in G, compute x; we call x the *discrete logarithm* of β with respect to α and write $\mathrm{Log}_\alpha(\beta) = x$.

The groups in which solving the DLP is hard give a structure for the basis of many cryptographic protocols. The DLP currently occurs in two instances in cryptographyl; in the multiplicative group of a finite field and the group of points on an elliptic curve.

1.1.3.1 Finite Field DLP

The first instance of the DLP used in cryptography is in the multiplicative group of a finite field.

Definition 1.1.2. The *Finite Field Discrete Logarithm Problem* (DLP) is:

Given elements α, β in the multiplicative group of a finite field, $\mathbb{F}_{p^n}^*$, such that β is in the subgroup of $\mathbb{F}_{p^n}^*$ generated by α (so $G = \langle\alpha\rangle$), find x modulo $|G|$ such that $\alpha^x = \beta \in \mathbb{F}_{p^n}^*$.

ElGamal Signatures

An example of the DLP being the basis for the security of a cryptographic protocol is the digital signature scheme suggested in 1985 by ElGamal in [31]. The protocol requires a collision resistant *hash function* H, that is, a function which is non-invertible and for which it is infeasible to find two messages m and m' such that $H(m) = H(m')$.

Key Generation

Using a finite field $\mathbb{F}_p$, p prime, Alice selects an element $g \in \mathbb{F}_p$ and a random number $x \in \{0, \ldots, p-1\}$ and computes $h = g^x$. Alice's public key is the set (p, g, h) and she keeps x secret.

Signing

If Alice wants to sign the message m to send to Bob, she first selects a random integer k which is non-zero mod p and coprime to $p-1$. Alice computes $r = g^k \mod p$ and $s \equiv (H(m) - xr)k^{-1} \mod (p-1)$, where x is Alice's secret key (if $s = 0$ then recompute using a new value for k). Alice sends the pair (r, s) as the signature for m. The signature is unique for each message.

Verification of Signature

When Bob receives the pair (r, s) with $0 < r < p$ and $0 < s < p-1$, he verifies that it is a valid signature by checking that $g^{H(m)} \equiv h^r r^s$, where h is Alice's public key.

It is easy to see that $sk + xr \equiv H(x) \mod (p-1)$ and so by the identity (2.1)

$$
\begin{aligned}
g^{H(m)} &\equiv g^{xr} \cdot g^{ks} \\
&\equiv (g^x)^r (g^k)^s \\
&\equiv h^r \cdot r^s \mod p. \qquad (1.1)
\end{aligned}
$$

To generate false signatures, an adversary would either have to find a message m' such that $H(m') = H(m)$ (which we presume is infeasible) or compute Alice's private key, that is, solve an instance of the DLP.

The ElGamal signature scheme is not used in this exact form, but the Digital Signature Algorithm (DSA) proposed in [18] by National Institute of Standards and Technology (NIST) is a widely used signature algorithm based on the ElGamal signature scheme.

Signing a message does not conceal the content, but using a similar setup we also have the ElGamal encryption protocol.

ElGamal Encryption

When Bob wants to send a message m to Alice, first m is converted to an element $\bar{m}$ of $\mathbb{F}_p$, then Bob chooses a random $y \in \{0, \ldots, p-1\}$ and computes $c_1 = g^y$ and the shared key $k_{AB} = h^y = g^{xy}$ ($h = g^x$ is Alice's public key). The shared key k_{AB} is also referred to as the *ephemeral key* as a new key is calculated each time two parties communicate. Using the ephemeral key, Bob calculates $c_2 = \bar{m} \cdot k_{AB}$ and sends (c_1, c_2) to Alice.

Decryption

On receipt of a cipher text pair (c_1, c_2), Alice computes $k_{AB} = g^{xy} = c_1^x$ and $\bar{m} = c_2 \cdot k_{AB}^{-1}$ and then retrieves m from $\bar{m}$. It is clear that this works as

$$
c_2 \cdot k_{AB}^{-1} = \bar{m} \cdot h^y \cdot (g^{xy})^{-1} = \bar{m} \cdot g^{xy} \cdot g^{-xy} = \bar{m}.
$$

Examining the encryption scheme carefully, we notice that to break the protocol, the

adversary must solve a slight variant of the DLP, not strictly the DLP. An adversary can observe the exchange of $h = g^x$ and $c_1 = g^y$. To recover the message, the adversary has to compute the session key $k_{AB} = g^{xy}$ from h and c_1; this variant of the DLP is known at the Diffie-Hellman Problem and was first suggested in [24].

Diffie-Hellman

The Diffie-Hellman Key Exchange was the first practical public key agreement method suggested [24]. They used a slight variation of the DLP, now known as the Diffie-Hellman Problem.

Definition 1.1.3. The *Finite Field Diffie-Hellman Problem* (DHP) is: Given elements g, g^x and g^y in a finite field $\mathbb{F}_q$, calculate g^{xy}.

The DHP has been shown to be as hard as the DLP in some cases [22] and as yet no groups are known in which the DHP can be solved faster than the DLP. Clearly the DHP can be solved if the DLP can be solved.

The *Decisional Diffie-Hellman Problem* (DDHP) is also a derivative of the DLP and can be used as a basis for security in some settings.

Definition 1.1.4 (Decisional Diffie-Hellman Problem)**.** Given elements g, g^x, g^y and α ,in a finite field $\mathbb{F}_q$, determine if $g^{xy} = \alpha$.

Diffie-Hellman Key Exchange Protocol

As above, we use a prime field $\mathbb{F}_p$ with generator g. Two parties, Alice and Bob, each select a random number x_A and x_B respectively with $x_A, x_B \in \{0, \ldots, q-1\}$ and compute $h_A = g^{x_A}$ and $h_B = g^{x_B}$.

Key Generation

Alice and Bob publish h_A and h_B respectively (they keep x_A and x_B, respectively, secret). Alice computes $k = h_B^{x_A} = g^{x_A \cdot x_B}$ and Bob computes $k = h_A^{x_B} = g^{x_B \cdot x_A} = g^{x_A \cdot x_B}$, now

Alice and Bob have a common secret key.

1.1.3.2 Elliptic Curve DLP

In 1985, Miller [71] and Koblitz [54] independently noticed that the DLP in the group of points on an elliptic curve defined over a finite field is also hard to solve – in most cases harder to solve than in the multiplicative group of a finite field of comparable size.

Definition 1.1.5. The *Elliptic Curve Discrete Logarithm Problem* (ECDLP) is:
Given points Q and P on an elliptic curve such that Q is in the group of points generated by P, find the smallest positive integer l such that $Q = lP$.

Given that the ECDLP is harder than the DLP in the finite field (of equivalent size), there are implementational advantages in adapting to elliptic curves many of the cryptographic protocols originally proposed using the multiplicative group of a finite field.

EC Diffie-Hellman

The elliptic curve analogues to the DHP and DDHP in the finite field are as follows:

Definition 1.1.6. The *Elliptic Curve Diffie-Hellman Problem* (ECDHP) is:
Given the points P, aP and bP on an elliptic curve E, compute abP.

Definition 1.1.7. The *Elliptic Curve Decisional Diffie-Hellman Problem* (ECDDHP) is:
Given points P, aP, bP and Q on an elliptic curve E, determine if $abP = Q$.

Although it may seem, at first glance, that these last two problems would be equivalently hard, that is not always the case. There exist groups in which the DHP is considered computationally infeasible, but the DDHP is easy to solve; these groups are called *Gap groups* [16].

It was believed that all elliptic curves achieve superior efficiency for a given level of security than finite fields (same level of security achievable using smaller system parameters) until the attacks of [69] and [34, 33] showed that some curves admit a bilinear pairing which

maps the ECDLP to a DLP instance in a finite field, which is an extension of the finite field over which the curve is defined. For curves with computable bilinear pairings, the hardness of the ECDLP in those cases is therefore only as hard as the DLP instance in that finite field; if this was "easier" than the ECDLP such curves were considered to be weaker and unsuitable for use in cryptography. With careful selection of the system parameters (discussed in Chapter 7), however, these bilinear pairings can also be used to construct new cryptographic protocols, even giving solutions to some previously unsolved cryptographic problems. The area of public-key cryptography which uses these particular curves and bilinear pairings is known as *Pairing-Based Cryptography*, which is the focus area of this thesis. The groups of points on the curves admitting bilinear pairings are Gap groups. This will be illustrated below.

1.1.4 Pairing-Based Cryptography

As mentioned, bilinear pairings were first introduced to the cryptographic community as a tool for cryptanalysis. They have since been used in many 'constructive' ways; key agreement schemes [49] and short signature schemes [16], for example, have been constructed using bilinear pairings. The problem of developing an *Identity-based* encryption scheme was easily solved using pairings [15]. These developments have spurred the research community into examining more uses of bilinear pairings in cryptography and the development of *Pairing-Based Cryptography* (PBC).

A pairing has two basic properties: it is bilinear and non-degenerate. Non-degeneracy ensures that we do not have a trivial pairing, always evaluating to 1. We denote by $e\langle\cdot,\cdot\rangle$ a bilinear pairing, taking as arguments two points on an elliptic curve, from groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and returning an element of a subgroup, $\mathbb{G}_T$, of a finite field. For points P, P_i, Q, Q_i $(i = 1, 2)$ on an elliptic curve the following properties hold:

$$\begin{aligned} e\langle P, Q_1 + Q_2\rangle &= e\langle P, Q_1\rangle\, e\langle P, Q_2\rangle \\ e\langle P_1 + P_2, Q\rangle &= e\langle P_1, Q\rangle\, e\langle P_2, Q\rangle\,. \end{aligned}$$

From the bilinearity property of a pairing, we see that the DDHP is easy in groups $\mathbb{G}_1$ when there is a known and computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ (such groups are therefore Gap groups). Given an instance of the DDHP (P, aP, bP, Q) we simply compute $e\langle aP, bP\rangle$ and $e\langle P, Q\rangle$, then test for equality. The DHP is not straightforward in these groups. When using a bilinear pairing, the security of a protocol often relies on the hardness of solving the following hard problem [15, 49]:

Definition 1.1.8. The *Bilinear Diffie-Hellman Problem* (BDHP) is:
Given a pairing $e\langle\cdot,\cdot\rangle$ and the points P, a point of order r, $P_1 = aP$, $P_2 = bP$, $P_3 = cP$, where $a, b, c \in \{0, \ldots, r-1\}$, compute $e\langle P, P\rangle^{abc}$.

The *Decisional Bilinear Diffie-Hellman Problem* (DBDHP) is easily deduced from the Bilinear Diffie-Hellman Problem.

Definition 1.1.9. The DBDHP is:
Given a pairing $e\langle\cdot,\cdot\rangle$ and the points P, $P_1 = aP$, $P_2 = bP$, $P_3 = cP$ and Q, where a, b and c are random integers modulo a large prime r, determine if $Q = abcP$.

1.1.4.1 Identity-based Cryptography

A novel way to manage the infrastructure of asymmetric cryptographic protocols is offered by *Identity-based* (ID-based) Cryptography. ID-based cryptography is especially useful in situations where there are many users and hence a need for many public keys. The public key of a user is generated using some specific information, unique to the user's identity, which is freely available to other users. This could be an email or IP address, or even a telephone number. In systems with many users, this has the advantage that public keys need not be stored in a directory; users can determine the necessary public key using the identity of the party with whom they wish to communicate. Another advantage of ID-based cryptography is that a message can be sent to a party who has not yet registered or whose key information is not online. Using their identity information, they obtain their private key and can then decrypt the message after receiving it. The private keys corresponding to the

public keys are generated by a trusted authority, a *Private Key Generator* (PKG).

In 1984, Shamir first proposed the concept of an ID-based protocol [87] which can be used to sign (IBS) and encrypt (IBE), giving also an IBS protocol. There have been proposals for IBE schemes since then; Cock's ID-based encryption scheme, for example, relies on the hardness of distinguishing quadratic residues from quadratic non-residues modulo a product of two large primes for security [19]. This scheme has the drawback that it requires large bandwidth, which results in the scheme being impractical for general use. The most efficient ID-based encryption schemes, proposed to date, are those which use bilinear pairings. The first such ID-based encryption scheme was proposed in 2001 by Boneh and Franklin [15] and uses the *Weil pairing*.

Boneh and Franklin ID-based Encryption

This ID-based encryption scheme is described in [15].

Setup

Choose the system parameters according to the desired security level: E, a pairing-friendly elliptic curve over $\mathbb{F}_q$ with a large subgroup of r points and the pairing $\hat{e}$. The group $\mathbb{G}_1$ $(= \mathbb{G}_2)$ is the group of points of order r over $\mathbb{F}_q$ and $\mathbb{G}_T \subset \mathbb{F}_{q^k}$. Select a generator P of $\mathbb{G}_1$ and a random number $s \in \mathbb{Z}_r^*$, and let $P_{pub} = sP$. Choose 4 hash functions:

$$\begin{aligned} H_1 &: \{0,1\}^* \rightarrow \mathbb{G}_1^* \\ H_2 &: \mathbb{G}_2 \rightarrow \{0,1\}^n \\ H_3 &: \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_r^* \\ H_4 &: \{0,1\}^n \rightarrow \{0,1\}^n. \end{aligned}$$

The system parameters are:

$$\langle r, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$$

The master key, s, is kept secret by the PKG.

Extraction of a Private Key

To obtain her private key, Alice sends her required information ID_A to the PKG who returns $d_A = sQ_A$ to Alice via a secure channel, where $Q_A = H_1(ID_A)$.

Encryption

If Bob wants to send a message m of length n to Alice he first computes $Q_A = H_1(ID_A)$, selects σ, a random binary string of length n and computes $h = H_3(\sigma, m)$. The cipher text is given by

$$c = \left\langle hP, \sigma \oplus H_2(g_A^h), m \oplus H_4(\sigma) \right\rangle$$

where $g_A = \hat{e}\left\langle Q_A, P_{pub} \right\rangle$.

Decryption

When Alice receives the triple $c = \langle u, v, w \rangle$, she first checks that u is a valid element of $\mathbb{G}_1$ and rejects the message if it is not. If the cipher text passes the first test, Alice computes

$$\sigma = v \oplus H_2(\hat{e}\left\langle d_A, u \right\rangle),$$

then

$$m = w \oplus H_4(\sigma).$$

Finally, Alice accepts the message m as being a valid message if $u = H_3(\sigma, m) \cdot P$.

1.2 Motivation and Thesis Outline

As we have briefly seen, there are many benefits of using pairings in cryptographic protocols, in particular for the implementation of ID-based cryptography. The continued interest in, and potential implementation of PBC depend on both the security levels achievable by

protocols using pairings and the efficient implementation of pairings. The intention in this thesis is twofold: to investigate the security offered by pairing-based protocols and to give some improvements for the implementation of pairing-based protocols.

When considering the security of a cryptographic protocol, there are two approaches to be aware of: a cryptographic protocol can be broken if the protocol (or implementation thereof) itself is flawed or if the instances of the hard problem on which the security is based are deemed easy; the second is the focus of this thesis.

The algorithms for solving the ECDLP and DLP as they occur in PBC will be analysed in order to give more precise estimates for the sizes of the system parameters needed for a particular security level to ensure that implementations are as efficient as possible.

The implementation of PBC is not as straightforward as other public-key protocols; there are numerous obstacles to overcome, some of which will be highlighted and addressed.

This thesis is organised as follows: First, the necessary mathematical background will be introduced in Chapter 2. In Chapter 3, the use of pairings in cryptography will be further explored with a more detailed look at the actual pairing functions and how they are computed. In Section 3.2, we present some families of pairing-friendly elliptic curves for use in PBC. The remainder of the thesis is organised into two parts: Security and Algorithms.

In Part I, the security of PBC will be examined by firstly establishing the exact relationship between the two DLP instances occurring in PBC, in Chapter 4. These results then enable us to explore the existing algorithms to solve the DLP in the multiplicative group of the finite field, Chapter 5, and in the group of points on an elliptic curve, Chapter 6. The computational complexity of these Algorithms will be examined in Chapter 7 in order to give detailed security levels achievable by the curves described in Section 3.2.

In Part II, some useful algorithms for the implementation of PBC will be presented; the efficient representation of the finite fields used in PBC, presented in Chapter 8; a faster method for performing the final exponentiation, outlined in 9; and an improved method for performing an expensive cofactor multiplication, given in Chapter 10. The contributions of

this part are summarised in Chapter 11.

Chapter 12 will summarise the contributions of this thesis to PBC and highlight possible areas for further development.

Chapter 2

Mathematics and Notation

'Let me see: four times five is twelve, and four times six is thirteen, and four times seven is –oh dear! I shall never get to twenty at that rate!'

- Alice, Lewis Carroll's *Alice in Wonderland*,
Chapter II, The Pool of Tears.

For the ideas presented in the subsequent sections to be clear, it is necessary to recall some mathematical concepts and set the notation to be used throughout. In this chapter, we present the fundamental areas of mathematics necessary to understand the details of the work presented in this thesis. References used in the compilation of this chapter are [74, 93, 90, 4, 100, 20]; a new property of cyclotomic polynomials is proved, a result of collaboration with Manuel Charlemagne and David Freeman, included in Section 2.2. This property is used in the publication [12], the results of which are detailed in Chapter 4.

2.1 Fields

The theory in the following chapters requires both finite and infinite fields. Let $\mathcal{F}$ be a field and α an element of $\mathcal{F}$. The *characteristic* of $\mathcal{F}$ is the smallest, positive integer n such that for all $\alpha \in \mathcal{F}$,

$$n \cdot \alpha = \underbrace{\alpha + \alpha + \ldots + \alpha}_{n \text{ times}} = 0,$$

we write char$\mathcal{F} = n$. It is true that n is equal to either a prime (in which case $\mathcal{F}$ is a finite field) or 0.

A polynomial is said to be *defined over* a field $\mathcal{F}$ if all its coefficients are contained in $\mathcal{F}$. The *polynomial ring* of $\mathcal{F}$, denoted $\mathcal{F}[x]$, is the collection of all polynomials defined over $\mathcal{F}$.

A field $\mathcal{F}'$ is an *extension field* of $\mathcal{F}$ if it contains $\mathcal{F}$ as a subfield; we write $\mathcal{F}'/\mathcal{F}$. An element δ of any extension field $\mathcal{F}'/\mathcal{F}$ is called algebraic over $\mathcal{F}$ if it is a zero of a polynomial in $\mathcal{F}[x]$. An *algebraic extension* of $\mathcal{F}$ is the field given by adjoining δ to $\mathcal{F}$, denoted $\mathcal{F}(\delta)$, and the *algebraic closure* of $\mathcal{F}$, $\overline{\mathcal{F}}$, is the extension of $\mathcal{F}$ containing all elements algebraic over $\mathcal{F}$. The *extension degree* of $\mathcal{F}(\delta)/\mathcal{F}$ is the degree of the *minimal polynomial* of δ, $f(x) \in \mathcal{F}[x]$ such that f is a monic irreducible polynomial over $\mathcal{F}$ with minimal degree and $f(\delta) = 0$, denoted $[\mathcal{F}(\delta) : \mathcal{F}] = \deg(f)$. The other $\deg(f) - 1$ zeros of $f(x)$ are called the *conjugates* of δ.

It is true that an extension has finite extension degree if and only if it is an algebraic extension.

The multiplicative group of a field $\mathcal{F}$, consisting only of the invertible elements, the *units*, will be denoted $\mathcal{F}^*$.

Finite Fields

A finite field contains p^n elements for some prime p and positive integer n; it is denoted $\mathbb{F}_{p^n}$. Any finite field with p^n elements has a subfield of p elements isomorphic to $\mathbb{F}_p$, often referred to as the *prime subfield*.

For all divisors d of n, $\mathbb{F}_{p^n}$ has a subfield isomorphic to $\mathbb{F}_{p^d}$ and $[\mathbb{F}_{p^n} : \mathbb{F}_{p^d}] = n/d$; all subfields of $\mathbb{F}_{p^n}$ are of this form.

Definition 2.1.1. The *Frobenius automorphism* of the field $\mathbb{F}_{p^n}$ is the map:

$$\begin{aligned} \pi_p : \mathbb{F}_{p^n} &\rightarrow \mathbb{F}_{p^n} \\ \alpha &\mapsto \alpha^p. \end{aligned}$$

The Frobenius automorphism fixes the prime subfield $\mathbb{F}_p$ and generates the *Galois group* of $\mathbb{F}_{p^n}/\mathbb{F}_p$, denoted $G_{\mathbb{F}_{p^n}/\mathbb{F}_p}$, the group of all automorphisms of $\mathbb{F}_{p^n}$ fixing $\mathbb{F}_p$. The Galois group is cyclic and of order n.

Using the Frobenius automorphism we construct two other maps, the *Trace* and *Norm* maps.

Definition 2.1.2. The *Trace* map for an element $\alpha \in \mathbb{F}_{p^n}$ is defined as:

$$\mathrm{Tr}_{\mathbb{F}_{p^n}}(\alpha) = \sum_{i=1}^{n-1} \pi_p^i(\alpha).$$

The Trace of α, $\mathrm{Tr}_{\mathbb{F}_{p^n}/\mathbb{F}_p}(\alpha)$, is the sum of all its conjugates.

Definition 2.1.3. The *Norm* map for an element $\alpha \in \mathbb{F}_{p^n}$ over $\mathbb{F}_p$ is defined as:

$$N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(\alpha) = \prod_{i=0}^{n-1} \pi_p^i(\alpha).$$

The Norm of α, $N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(\alpha)$, is the product of all its conjugates. The norm is multiplicative, that is, for $\alpha_1, \alpha_2 \in \mathbb{F}_{p^n}$,

$$N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(\alpha_1 \cdot \alpha_2) = N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(\alpha_1) \cdot N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(\alpha_2)$$

and so for any $\ell \in \mathbb{Z}^+$ we have $N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(\alpha^\ell) = N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(\alpha)^\ell$.

Number Fields

A *Number field* is an algebraic extension of the rational numbers $\mathbb{Q}$. Suppose $\mathcal{K}$ is a number field of the form $\mathbb{Q}(\theta)$ where θ is a zero in $\mathbb{C}$ of the irreducible polynomial $f(x) \in \mathbb{Q}[x]$ of degree d. All number fields are subfields of $\mathbb{C}$. Labeling the conjugates, $\theta = \theta_1, \ldots, \theta_d \in \mathbb{C}$, the mappings φ_j for $j \in [0, \ldots, d-1]$ which permute the roots of $f(x)$ ($\varphi_j(\theta_i) = \theta_{i+j \mod d}$) give us d embeddings of $\mathcal{K}$ in $\mathbb{C}$. The number of embeddings of $\mathcal{K}$ which are contained in $\mathbb{R}$ is denoted r_1 and the number of unique embeddings of $\mathcal{K}$ which are contained in $\mathbb{C}\backslash\mathbb{R}$ is denoted r_2 so $d = r_1 + 2r_2$. We call the pair (r_1, r_2) the *signature* of $\mathcal{K}$.

We can write the elements of $\mathcal{K} = \mathbb{Q}(\theta)$ as degree $d-1$ expressions in θ and for an element $a \in \mathcal{K}$, $a = a_0 + a_1\theta + \ldots + a_{d-1}\theta^{d-1}$ the *Norm* of a is given by the *resultant* of the polynomials $a(x) = \sum_{j=0}^{d-1} a_j x^j$ and $f(x)$; that is, $N_{\mathcal{K}/\mathbb{Q}}(\sum_{j=0}^{d-1} a_j\theta^j) = \mathrm{Res}(a(x), f(x))$. The resultant is computed by finding the determinant of the Sylvester matrix, a $(d+\ell)$ square matrix, where ℓ is the largest positive integer $\ell \leq d$ such that $a_\ell \neq 0$.

An *algebraic integer* of $\mathcal{K}$ is an element of $\mathcal{K}$ which is algebraic over $\mathbb{Q}$ and whose minimal polynomial is monic with integer coefficients. The set of all algebraic integers of $\mathcal{K}$ forms a ring, called the *Ring of integers*, denoted $\mathcal{O}_\mathcal{K}$. It is true that $N_\mathcal{K}(\alpha)$ is an integer for any $\alpha \in \mathcal{O}_\mathcal{K}$ (this is easily shown by considering the coefficients of the minimal polynomial of α). The ring of integers is not necessarily a unique factorisation domain, which can lead to problems in the algorithms presented in later sections of this thesis. There is a way of "measuring" how close the ring of integers is to being a unique factorisation domain, called the *class number* of $\mathcal{K}$.

Definition 2.1.4. Let $\mathcal{K}$ be a number field with ring of integers $\mathcal{O}_\mathcal{K}$. An ideal I of $\mathcal{K}$ is a *fractional ideal* of $\mathcal{O}_\mathcal{K}$ if αI is an ideal of $\mathcal{O}_\mathcal{K}$ for some $\alpha \in \mathcal{O}_\mathcal{K}$.

Definition 2.1.5. A fractional ideal I of $\mathcal{O}_\mathcal{K}$ belongs to the equivalence class $[J]$ if $\alpha I = J$ for some $\alpha \in \mathcal{O}_\mathcal{K}$. The number of equivalence classes is finite and is called the *class number* of $\mathcal{K}$, often denoted by h. Defining multiplication of fractional ideals to be $[I][J] = [IJ]$,

the set of equivalence classes forms an abelian group under multiplication called the *class group* of $\mathcal{K}$.

A ring of integers with class number $h = 1$ has unique factorisation.

2.2 Cyclotomic Polynomials

Euler's phi function, $\phi(n)$, which denotes the number of integers $\leq n$ which are coprime to n, has the following properties:

$$\begin{aligned} \phi(mn) &= \phi(m)\phi(n) \text{ for } m, n \text{ coprime integers.} \\ \phi(p^n) &= (p-1)p^{n-1} \text{ for } p \text{ prime, } n \text{ a positive integer..} \end{aligned}$$

Anothr usefull property is given by the following theorem:

Theorem 2.2.1 (Euler-Fermat Theorem)**.** *For positive, coprime integers x and m,*

$$x^{\phi(m)} \equiv 1 \mod m. \tag{2.1}$$

Definition 2.2.2. A zero of the polynomial $x^n - 1$ is called an *nth root of unity*.

From the above definition, it is easy to deduce that for any divisor d of n, a dth root of unity will also be an nth root of unity. An nth root of unity ζ is called a *primitive nth root of unity* if $\zeta^n - 1 = 0$, but $\zeta^d - 1 \neq 0$ for any divisor d of n. The number of primitive nth roots of unity is $\phi(n)$. We denote by μ_n the group of nth roots of unity and by $\hat{\mu}_n$ the set of primitive nth roots of unity. It is the primitive roots of unity which define the cyclotomic polynomials.

Definition 2.2.3. The nth cyclotomic polynomial is given by:

$$\Phi_n(x) = \prod_{\zeta \in \hat{\mu}_n} (x - \zeta).$$

Cyclotomic polynomials have many uses in algebra and some well known properties.

Fact 2.2.4. [60, §VI.3]

1. $x^k - 1 = \prod_{d|k} \Phi_d(x)$.

2. From the definition, we see that $\deg(\Phi_n(x)) = \phi(n)$.

3. If ℓ is a prime not dividing k, then $\Phi_k(x^\ell) = \Phi_{k\ell}(x)\Phi_k(x)$.

4. If ℓ is a prime dividing k, then $\Phi_k(x^\ell) = \Phi_{k\ell}(x)$.

The following property has recently been proved by Benger et al. in [12, Lemma 2.8], an alternative proof for a similar property can be found in [80, Lemma 5.2].

Lemma 2.2.5. *If k and m are coprime, then*

$$\Phi_k(x^m) = \prod_{d|m} \Phi_{kd}(x). \tag{2.2}$$

Proof. Since the polynomials on both sides of the equation are monic, to show the polynomials are equal, it suffices to show that they are of the same degree and have the same roots.

- Degrees of the polynomials:

  Clearly, the left hand side of (2.2) has degree $m\phi(k)$. For any coprime numbers x and y it is true that $\phi(xy) = \phi(x)\phi(y)$. Since $(k, m) = 1$ by assumption, it is also true that $(k, d) = 1$ for all $d \mid m$. It follows that the degree of the right hand side of (2.2) is $\phi(k) \sum_{d|m} \phi(d)$, which by Fact 2.2.4 (1) and (2) is equal to $m\phi(k)$.

- Roots of the polynomials:

  Suppose ζ is a root of $\Phi_{kd}(x)$ for some $d \mid m$. Since ζ is a primitive kdth root of unity, ζ^d is a primitive kth root of unity. Write $m = de$. Since $\gcd(k, e) = 1$, it follows that $(\zeta^d)^e = \zeta^m$ is also a primitive kth root of unity, so ζ is also a root of $\Phi_k(x^m)$.

Since the two monic polynomials in (2.2) have the same degree, and all roots of the right hand side are also roots of the left hand side, we conclude that the two polynomials are equal. $\square$

2.3 Elliptic Curves

Elliptic curves have been of interest for hundreds of years and are used as very efficient structures upon which to base cryptographic protocols. In the foreword of [59], Lang writes that *"It is possible to write endlessly on Elliptic Curves. (This is not a threat.)"* In this section we present only the definitions, properties and results relevant to this thesis, compiled from numerous sources [74, 93, 90, 4].

We retain the finite field notation from above.

Definition 2.3.1. An *Elliptic Curve* over a finite field $\mathbb{F}_q$, $q = p^n$ for some prime p, is the set of all solutions over $\overline{\mathbb{F}}_q$ (called *points*) $[x : y : z]$ in $\mathbb{P}^2(\overline{\mathbb{F}}_q)$ satisfying the (*Weierstraß*) equation

$$\mathbb{E} : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3,\ a_i \in \mathbb{F}_q.$$

The set of all solutions, denoted $\mathbb{E}(\overline{\mathbb{F}}_q)$, with elliptic curve addition (to be explained momentarily) forms a group with identity element $[0 : 1 : 0]$, denoted $\mathcal{O}$, called the *point at infinity*.

Often, for clarity, we use the affine version of the curve:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,\ a_i \in \mathbb{F}_q.$$

For any field F, $\mathbb{F}_q \subseteq F \subseteq \overline{\mathbb{F}}_q$ a solution (x, y) is *defined over* F if $x, y \in F$. The set of points defined over F with $\mathcal{O}$, denoted $E(F)$,

$$E(F) = \{(x, y) \in \mathbb{A}^2(F) \mid (x, y) \text{ satisfies the equation}\} \cup \mathcal{O},$$

forms a group under the elliptic curve addition operation, called the F-*rational* points. We consider $\mathcal{O}$ to be the point of infinity of the pencil of projective lines associated (by homogenisation with respect to Z) with the lines in $\mathbb{A}^2(\overline{\mathbb{F}}_q)$ parallel to the y-axis.

Without loss of generality we will use the affine Weierstraß equation throughout this thesis. (It is easy to map between the two representations using *(de)homogenisation* with respect to Z.)

If the elliptic curve is defined over a field with characteristic $p \neq 2, 3$ then by finding the correct isomorphism over $\mathbb{F}_q$ we are able to write the elliptic curve in the *short Weierstraß form*

$$E : y^2 = x^3 + ax + b.$$

We use the notation $E_{a,b}$ for this curve. The curves we will be examining in more detail in this thesis will all be of this form.

Two invariants used to classify elliptic curves are the *discriminant*,

$$\Delta = -16(a^3 + 27b^2),$$

and the *j-invariant*,

$$j = 1728a^3/4\Delta.$$

The group operation elliptic curve addition (henceforth referred to simply as *addition*) is performed as follows:

Let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be two points in $E(\overline{\mathbb{F}}_q)$ (not $\mathcal{O}$, $P \neq -Q$); let ℓ denote the straight line passing through P and Q and let S be the third point of intersection of ℓ with E (this exists as the defining equation of E has a cubic monomial). Let ν be the vertical line through S (the line connecting S and $\mathcal{O}$), then the point $P + Q$ is given by the second point of intersection of ν with E (which is equivalent to $-S$). This is illustrated more clearly in Figure 2.1. To complete the addition law, we also set

$$P + \mathcal{O} = P,$$

for any point $P \in E(\overline{\mathbb{F}}_q)$ and

$$P + -P = \mathcal{O}.$$

The inverse of a point $P = (x_P, y_P)$ is given by $-P = (x_P, -y_P - a_1 x_P - a_3)$ or simply $-P = (x_P, -y_P)$ when the curve is in short Weierstraß form.

We have concrete formulæ for point addition: For two points P, Q on E we define $S = P + Q$ where

$$x_S = \lambda^2 + a_1\lambda - a_2 - x_P - x_Q,$$

and

$$y_S = \lambda(x_P - x_S) - y_P - a_1 x_S - a_3,$$

and λ is the slope of the line through P and Q, which is

$$\frac{y_P - y_Q}{x_P - x_Q} \text{ if } P \neq \pm Q,$$

and

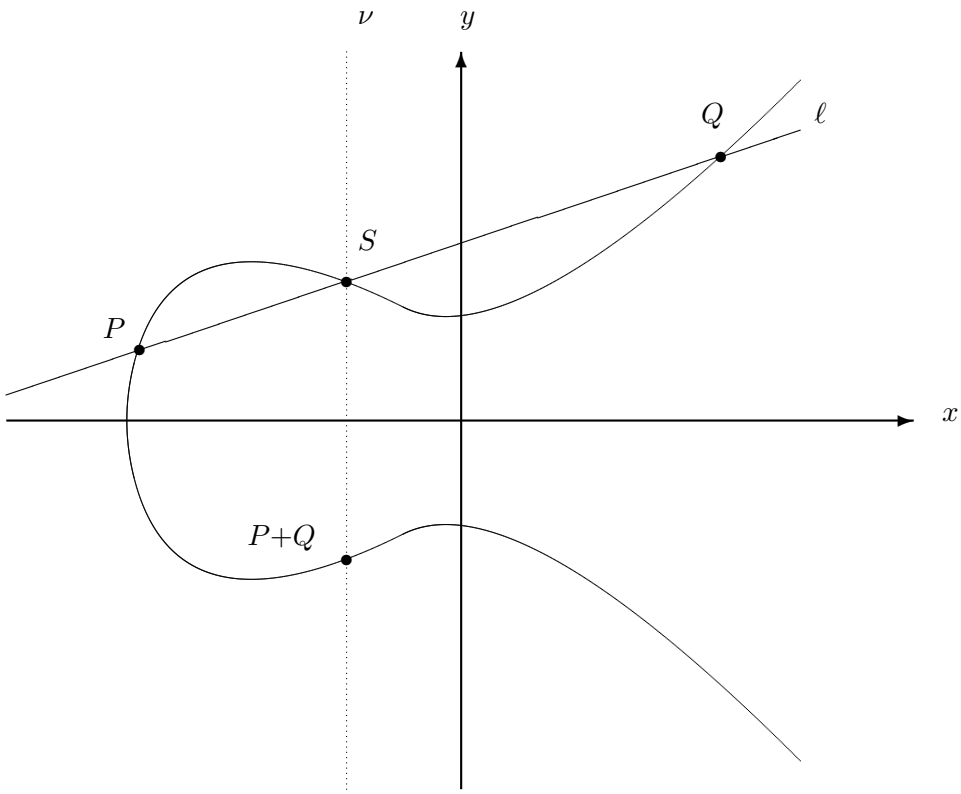
$$\frac{3x_P^2 + 2a_2 x_P + a_4 - a_1 y_P}{2y_P + a_1 x_P + a_3} \text{ if } P = Q.$$

From Figure 2.1 it is clear that the group of points on an elliptic curve is abelian; it is also cyclic, or isomorphic to a product two of cyclic groups. We write the number of points, $\#E(\overline{\mathbb{F}}_q)$, as a product $c \cdot r$, $c \geq 1$, and for cryptographic purposes we like to choose curves with a large, prime r and small c (called the *cofactor*).

Definition 2.3.2. The group of points with order dividing r is called the *r-torsion group* of $E(\overline{\mathbb{F}}_q)$, denoted $E[r]$.

The r-torsion group is the *kernel* of the map $P \mapsto rP$ (where iP for some $i \in \mathbb{Z}$ denotes $\underbrace{P + P + \cdots + P}_{i \text{ times}}$), that is, $E[r]$ is the group of points such that $rP = \mathcal{O}$.

Figure 2.1: Adding two points on an elliptic curve



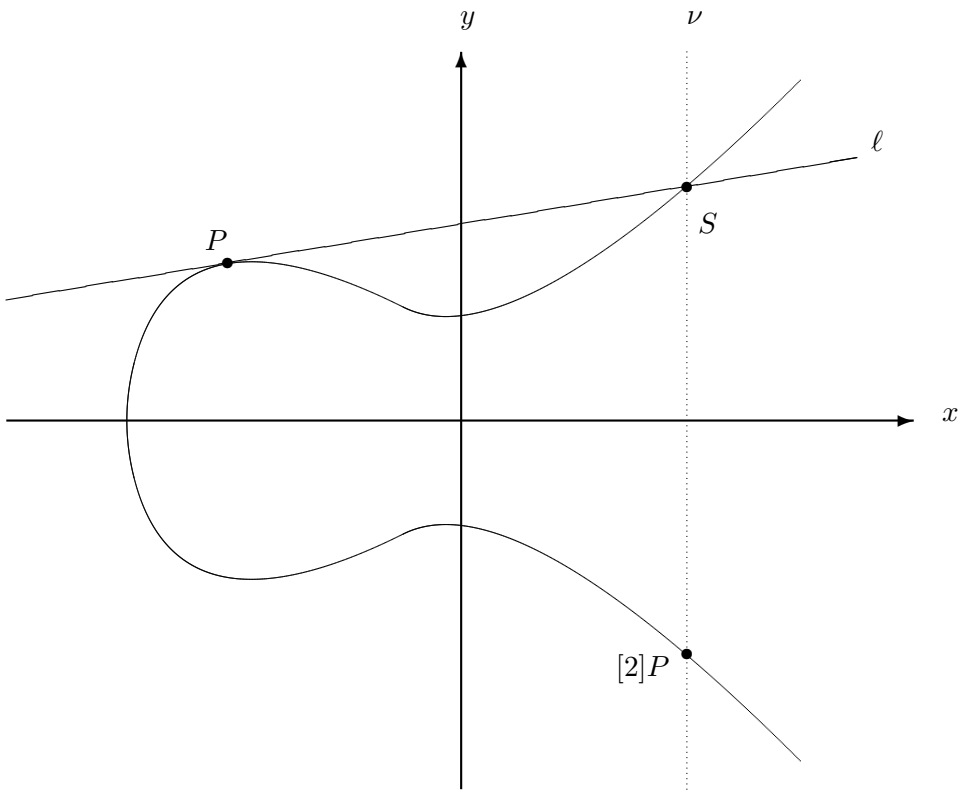
Definition 2.3.3. Let E be an elliptic curve defined over $\mathbb{F}_q$, $q = p^n$. Let $r \neq p$ be a prime dividing $\#E(\mathbb{F}_q)$. The *embedding degree of E with respect to r* is the smallest integer k such that r divides $q^k - 1$ and r does not divide $q^i - 1$ for all $1 \leq i < k$.

With r and k defined as above, if $k > 1$ then all of the r-torsion points of E will be defined over $\mathbb{F}_{q^k}$.

Definition 2.3.4. Let E, q, and r be as above. The *minimal embedding field of E with respect to r*, denoted F, is the smallest extension of $\mathbb{F}_p$ containing the rth roots of unity $\mu_r \subset \overline{\mathbb{F}}_q$.

Clearly the minimal embedding field is a subfield of $\mathbb{F}_{q^k}$; if $n = 1$, then the minimal embedding field is $\mathbb{F}_{q^k}$. We will see, however, that this is not always the case (Chapter 4)

Figure 2.2: Doubling a point on an elliptic curve



when $n > 1$. Knowing the minimal embedding field is important for selecting parameters for elliptic curve cryptographic protocols correctly.

As for finite fields, we are able to define the Frobenius endomorphism on the group of points on an elliptic curve:

$$\begin{aligned} \pi_q : E &\rightarrow E \\ \pi_q((x, y)) &= (x^q, y^q). \end{aligned}$$

The trace of the Frobenius endomorphism is denoted by t.

There are two types of elliptic curves which will be mentioned throughout this thesis, *supersingular* and *ordinary* elliptic curves.

2.3.1 Supersingular Elliptic Curves

Supersingular elliptic curves have many equivalent defining properties:

An elliptic curve E defined over $\mathbb{F}_q$, where $q = p^n$ is a supersingular elliptic curve if and only if:

- the greatest common divisor of q and t (trace of the Frobenius map), (q, t), is > 1;
- E has no p^i-torsion points for some $i \in \mathbb{Z}^+$;

One other useful property of supersingular elliptic curves is that their endomorphism ring is non-commutative and so there exist endomorphisms which map a point to a different, disjoint group of points (of the same order). These maps have been termed *distortion maps* [99]; their use in pairing-based cryptography will become apparent in Section 3.1. The details of these maps are not necessary to understand their use.

Menezes, Okamoto and Vanstone [69] gave a complete classification of supersingular elliptic curves over finite fields $\mathbb{F}_q$, with $q = p^n$. They showed that five possible embedding degrees k can occur, $k \in \{1, 2, 3, 4, 6\}$, corresponding to five possible absolute values of the trace of Frobenius t:

k	t	$\#E(\mathbb{F}_q)$	p, n
1	$\pm 2\sqrt{q}$	$q \mp 2\sqrt{q} + 1$	any p, n even
2	0	$q + 1$	any p, any n
3	$\pm\sqrt{q}$	$q \mp \sqrt{q} + 1$	$p \equiv 2 \bmod 3$, n even
4	$\pm\sqrt{2q}$	$q \mp \sqrt{2q} + 1$	$p = 2$, n odd
6	$\pm\sqrt{3q}$	$q \mp \sqrt{3q} + 1$	$p = 3$, n odd

The supersingular elliptic curves defined over binary and ternary fields have a slightly different short Weierstraß form from the one given above but this will not be used in this thesis.

2.3.2 Non-supersingular Elliptic Curves

An elliptic curve which is not supersingular is called *non-supersingular* or *ordinary*. In this thesis we shall adopt the following convention: ordinary elliptic curves will simply be referred to as 'elliptic curves' and if a discussion refers to supersingular elliptic curves this will be stated explicitly. Let $E(\overline{\mathbb{F}}_p)$ (p a prime) be an elliptic curve. As for supersingular elliptic curves, we know how many points will be in the group $E(\mathbb{F}_p)$:

Theorem 2.3.5. *(Hasse, Weil) An elliptic curve over a prime field $\mathbb{F}_p$ has $p+1-t$ points where $|t| \leq 2\sqrt{p}$.*

More details about properties of certain elliptic curves will be given in Chapter 3, when the particular curves of interest to this work will be presented.

The set of isomorphisms of an elliptic curve E to itself preserving the point at infinity forms a group, called the *Automorphism* group of E, denoted $\mathrm{Aut}(E)$. The order of $\mathrm{Aut}(E)$ is given by [90, Theorem III.10.1]:

$$\begin{aligned} j \neq 0, 1728 \quad &: \quad \mathrm{Aut}(E) \text{ has order } 2; \\ j = 0 \quad &: \quad \mathrm{Aut}(E) \text{ has order } 6; \\ j = 1728 \quad &: \quad \mathrm{Aut}(E) \text{ has order } 4. \end{aligned}$$

We also know that if m is the order of $\mathrm{Aut}(E)$, then by [90, Corollary III.10.2]

$$\mathrm{Aut}(E) \cong \mu_m,$$

so we are able to fully determine the Automorphisms of E; we identify $\mathrm{Aut}(E)$ with μ_m by the following map: if ζ is a primitive mth root of unity

$$\begin{aligned} [\cdot] : \mu_m &\mapsto \mathrm{Aut}(E) \\ [\zeta](x,y) &= (\zeta^2 x, \zeta^3 y). \end{aligned}$$

If E has a non-trivial automorphism group, these can be used to the advantage of the implementer (Chapter 3) and the cryptanalyst (Section 6.2.1).

The elliptic curves over $\mathbb{F}_q$ with j-invariant 0 or 1728 are of the forms

$$E_b : y^2 = x^3 + b,$$

and

$$E_a : y^2 = x^3 + ax,$$

respectively. When $p \equiv 1 \mod 4$ and $p \equiv 1 \mod 3$ respectively, then the curves E_b and E_a are non-supersingular. These curves both have non-trivial automorphism groups and also higher degree *twists*.

2.3.2.1 Twists

An elliptic curve E' over $\mathbb{F}_q$ is said to be a *twist of E of degree d*, where E is an elliptic curve also defined over $\mathbb{F}_q$, if there exists an isomorphism defined over $\mathbb{F}_{q^d}$

$$\phi_d : E' \mapsto E,$$

where d is minimal. The curves we are interested in for PBC are those with higher order twists because such curves will be used to speed up the calculations of pairings as will be described in Section 3.1. The highest degree twist possible for a curve can be categorised by the j-invariant of the curve [46]:

$$\begin{aligned} j \neq 0, 1728 \quad &: \quad d = 2; \\ j = 0 \quad &: \quad d = 6; \\ j = 1728 \quad &: \quad d = 4. \end{aligned}$$

For each elliptic curve E with twists of degree d, there are $\phi(d)$ unique twists, that is 2

possible twists of each of E_b and E_a. Suppose now that $d = \gcd(d, k)$ and let $e = k/d$.

Both the twists and the isomorphisms from the twists to the curve E can be defined using an element $i \in \mathbb{F}_{q^e}$, which is not a power of any prime dividing e.

For $E_a : y^2 = x^3 + ax$, the quartic twists are given by $E_1' : y^2 = x^3 + ax/i$ and $E_2' : y^2 = x^3 + ax/i^3$. The respective isomorphisms are given as [4]:

$$E_1' \to E : (x, y) \to (i^{1/2}x, i^{3/4}y),$$

and

$$E_2' \to E : (x, y) \to (i^{3/2}x, i^{5/4}y).$$

Similarly, for $E_b : y^2 = x^3 + b$, the sextic twists are given by $E_1' : y^2 = x^3 + b/i$ and $E_2' : y^2 = x^3 + b/i^5$; the respective isomorphisms are given as:

$$E_1' \to E : (x, y) \to (i^{1/3}x, i^{1/2}y),$$

and

$$E_2' \to E : (x, y) \to (i^{5/3}x, i^{5/2}y).$$

For both cases, we see that the twist equations are defined over $\mathbb{F}_{q^e}$ though the isomorphisms are defined over $\mathbb{F}_{q^k}$.

2.3.3 Divisors

Divisors are used in the calculation of pairings. They form a new group structure, constructed from the points of an elliptic curve, the *divisor group*, denoted $\mathrm{Div}(E)$.

Definition 2.3.6. A *Divisor* $D \in \mathrm{Div}(E)$ is a formal sum

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

where $n_P \in \mathbb{Z}$ and only finitely many n_P are non zero. The *degree* of a divisor D is defined

to be $\sum_{P \in E} n_P$.

From the definition it is obvious that the set of divisors forms a group under addition and that the set of *degree* 0 divisors forms a subgroup, denoted $\mathrm{Div}^0(E)$.

Definition 2.3.7. For a divisor $D = \sum_{P \in A} n_P(P)$, the set of points P for which $n_P \neq 0$ is called the *support* of D.

Another subgroup of $\mathrm{Div}(E)$ is the group of *principal divisors*, defined using functions on the points of E.

Definition 2.3.8. For a given function f, the *order* of f at a point P, denoted $\mathrm{ord}_P(f)$, is given by

$$ord_P(f) = \begin{cases} n > 0 \text{ if } P \text{ is a zero of } f \text{ of order } n \\ n < 0 \text{ if } P \text{ is a pole of } f \text{ of order } -n \\ 0 \text{ if } P \text{ is neither a pole nor a zero of } f \end{cases}$$

As functions have a finite number of zeros and poles, it follows that we can define divisors of functions using the order function:

Definition 2.3.9. The *divisor of a function* f is given as $(f) = \sum_P \mathrm{ord}_P(f)[P]$. Divisors which are divisors of a function are called *principal* divisors. The principal divisors form a group under addition (as detailed below) denoted $\mathrm{Div}^q(A)$.

From the definition of a divisor of a function, we can easily deduce the following properties: for two functions f and g, $g \neq 0$,

$$\begin{aligned} (fg) &= (f) + (g); \\ (f/g) &= (f) - (g). \end{aligned}$$

Using the divisors of functions we are able to define equivalence classes in $\mathrm{Div}(E)$:

Definition 2.3.10. Two divisors D_1 and D_2 are said to be *linearly equivalent*, denoted $D_1 \sim D_2$ if there exists a function f such that $D_1 = D_2 - (f)$.

Given any degree 0 divisor $D = \sum_{P \in A} n_P(P)$, it is simple to check whether D is a principal divisor. The sum $\sum_{P \in A} n_P P$ evaluates to the point of infinity if and only if D is a principal divisor.

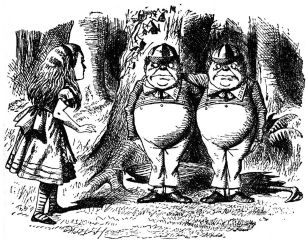
Definition 2.3.11. The *divisor class group*, often referred to as the *Picard group*, is the group of equivalence classes of degree 0 divisors under linear equivalence, that is, $\mathrm{Div}(A)^0/\mathrm{Div}^q(A)$, denoted $\mathrm{Pic}^0(A)$.

2.3.4 Weil Restriction and Abelian Varieties

Suppose E is an ordinary or supersingular elliptic curve over the extension field $\mathbb{F}_{p^n}$. We can choose an explicit polynomial representation of $\mathbb{F}_{p^n}$ by taking a monic, irreducible, degree n polynomial $f \in \mathbb{F}_p[t]$, then $\mathbb{F}_{p^n} \equiv \mathbb{F}_p[t]/f(t)$. The *Weil restriction* A of E is given by the set of $2m$-tuples $(x_0, \ldots, x_{m-1}, y_0, \ldots, y_{m-1})$ where $(x, y) \in E(\mathbb{F}_{p^n})$ for $x = x_0 + x_1 t + \ldots + x_{m-1} t^{m-1}$ and $y = y_0 + y_1 t + \ldots + y_{m-1} t^{m-1}$. The group law on A is derived directly from the group law on E and as such, A is an *abelian variety* of dimension n over $\mathbb{F}_p$. An abelian variety can be considered as a generalisation of an elliptic curve (an elliptic curve is an abelian variety of *dimension* 1); this idea is sufficient for this thesis. Abelian varieties will be mentioned in a general sense but no details or specific properties will be used, excepting those pertaining also to elliptic curves and mentioned in this section. The *genus* of a curve, denoted by g, is an invariant which describes the surface and is equal to the degree of the associated abelian variety. This will be referred to in the following sections but no particular details are used, nor are they necessary to understand the results.

Chapter 3

Pairings



- Tweedledee and Tweedledum, Lewis Carroll's *Through the Looking-Glass and What Alice Found There*,

Chapter IV, Tweedledee and Tweedledum, illustration by Sir John Tenniel.

In this chapter pairings, as used in Pairing-Based Cryptography (PBC), will be examined in more detail. After a basic introduction to pairings, we consider their first use in cryptology, for cryptanalysis. We introduce the algorithm for computing pairings for PBC in §3.1, with some optimisations and modifications giving the pairings currently used in cryptography.

Definition 3.0.12. A *pairing* is a map, e, from additive groups $\mathbb{G}_1$ and $\mathbb{G}_2$ into a multiplicative group $\mathbb{G}_T$, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. For a given P_1, P_2 and $P \in \mathbb{G}_1$ and Q_1, Q_2 and

$Q \in \mathbb{G}_2$ a pairing has the following properties:

- *Bilinearity*:

$$\begin{aligned} e(P, Q_1 + Q_2) &= e(P, Q_1)e(P, Q_2), \\ e(P_1 + P_2, Q) &= e(P_1, Q)e(P_2, Q); \end{aligned}$$

- *Non-degeneracy*:

$$\begin{aligned} &\forall P \in \mathbb{G}_1, P \neq \mathcal{O} \quad \exists Q \in \mathbb{G}_2 \text{ such that } e(P, Q) \neq 1, \\ &\forall Q \in \mathbb{G}_2, Q \neq \mathcal{O} \quad \exists P \in \mathbb{G}_1 \text{ such that } e(P, Q) \neq 1; \end{aligned}$$

- *Computability:* e is efficiently computable.

We introduce here some notation for this section. A *Miller function*, denoted $f_{s,P}$, for a point P on an elliptic curve E and some integer s, is a function with divisor of the form $(f_{s,P}) = s(P) - ([s]P) - (s-1)(\mathcal{O})$; if P has order dividing s, this becomes $(f_{s,P}) = s(P) - s(\mathcal{O})$. For any point P, we denote by D_P any divisor $\sim (P) - (\mathcal{O})$.

Weil Pairing

Let E be an elliptic curve defined over over a finite field $\mathbb{F}_q$. The *Weil* pairing ([90, §III.8] and [74, §16]) maps from groups of r-torsion points of E to a subgroup of the multiplicative group of $\mathbb{F}_{q^k}$, where k is the embedding degree of E with respect to r. The Weil pairing is defined as:

$$\begin{aligned} \omega_r : E[r] \times E[r] &\rightarrow \mu_r \in \mathbb{F}_{q^k} \\ \omega_r(P, Q) &= f_{r,P}(D_Q)/f_{r,Q}(D_P). \end{aligned}$$

If P and Q are linearly independent and the divisors D_P and D_Q have supports disjoint from those of $f_{r,Q}$ and $f_{r,P}$ respectively, then this pairing is non-degenerate.

Tate Pairing

The *Tate* pairing [28] is defined as the map:

$$\begin{aligned} t_r : E[r](\mathbb{F}_p) \times E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k}) &\rightarrow \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r \\ t_r \langle P, Q \rangle &= f_{r,P}(Q). \end{aligned}$$

This will be degenerate if Q is disjoint from the support of $f_{r,P}$.

As the second argument, Q, is chosen as a representative of an equivalence class of $E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k})$ it follows that the result of the pairing will also be a representative of an equivalence class of $\mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r$, which is only unique up to a multiple of a power of r. As a unique value is required for cryptographic applications, after computing $f_P(D_Q)$ the value is raised to the power $(p^k - 1)/r$. The pairing $t_r \langle P, Q \rangle^{(p^k-1)/r}$, often called the *reduced Tate pairing*, will be the pairing refered to throughout the rest of this thesis.

3.0.5 Cryptanalysis

The first application of pairing maps to cryptography was to attack cryptosystems whose security is based on the hardness of solving the ECDLP on particular types of curves. From the bilinearity property of a pairing, we can deduce that $e([a]P, Q) = e(P, Q)^a$ and so solving the DLP in the group group $\mathbb{G}_1$ (similarly, $\mathbb{G}_2$) is at most as hard as solving the DLP in $\mathbb{G}_T$, whenever such a pairing is computable. This was noticed by the authors of [69] and [34, 33], who used the Weil and Tate pairings respectively to map the DLP in groups of points on supersingular elliptic curves, $\mathbb{G}_1$ and $\mathbb{G}_2$, to a subgroup of the multiplicative group of a relatively small field, $\mathbb{G}_T$. The most efficient algorithms for solving the ECDLP and DLP in the finite field differ substantially (as will be explained in detail in Chapters 5 and 6). For most supersingular elliptic curves the ECDLP is more efficiently solved in $\mathbb{G}_T$ than in $\mathbb{G}_1$, thus the attacks in [69, 34, 33] were successful.

3.1 Computing Pairings

The pairings above are only useful for cryptography if we have some way of evaluating them efficiently.

The Weil pairing can be considered as a ratio of two instances of the Tate pairing without the requirement for the final exponentiation, thus we continue this section with discussions for evaluating the Tate pairing; the reader may assume that the same methods apply for computing the Weil pairing.

This evaluation requires a function $f_{r,P}$, for a point P of order r, with a given divisor $(f_{r,P}) = r(P) - r(\mathcal{O})$ to be evaluated at a divisor $D_Q \sim (Q) - (\mathcal{O})$ with support disjoint from that of $(f_{r,P})$.

The divisor can be easily constructed by taking a random point $S \notin \{P, \mathcal{O}\}$ and defining $D_Q = (Q + S) - (S)$, checking that the support is disjoint from sup$(f_{r,P})$.

Miller's algorithm is a polynomial time algorithm for evaluating the Weil [73] and Tate pairings.

3.1.1 Miller's Algorithm

Instead of first trying to find the function $f_{r,P}$ and then evaluating it at D_Q, the ingenious idea behind Miller's algorithm [72] is to use the linear functions used in the geometric representation of the addition operation to iteratively construct $f_{r,P}(D_Q)$, evaluating at each step. We therefore require that the divisor D_Q satisfies the slightly stricter condition that it is disjoint from the support of the divisors of all intermediate functions. This idea relies on the important observation [72] that

$$f_{i+j,P} = f_{i,P} \cdot f_{j,P} \cdot \frac{\ell_{[i]P,[j]P}}{\nu_{[i+j]P}},$$

where $\ell_{[i]P,[j]P}$ is the line connecting $[i]P$ and $[j]P$ and $\nu_{[i+j]P}$ is the vertical line through $[i+j]P$ (as used for the point addition computation outlined in Chapter 2).

For Miller's algorithm we first write r in binary form, $r = \sum b_i 2^i$, $b_i \in \{0, 1\}$ then we

calculate $f_{2^i,P}$ for all $b_i \neq 0$ and use Miller's observation to put them together to obtain $f_{r,P}$.

Algorithm 1 Miller's Algorithm

[72]

INPUT: P and Q, points on an elliptic curve E, P a point of order r.

OUTPUT: $\langle P, Q \rangle_r$.

1: Chose a suitable point S on E
2: $Q' \leftarrow Q + S$
3: $T \leftarrow P$
4: $m \leftarrow \lfloor \log_2(r) \rfloor - 1$, $f \leftarrow 1$
5: **while** $m \geq 0$ **do**
6: Calculate l and ν for doubling T
7: $T \leftarrow 2T$
8: $f \leftarrow f^2 \frac{l(Q')\nu(S)}{\nu(Q')l(S)}$
9: **if** mth bit if r is 1 **then**
10: calculate l and ν for addition of T and P
11: $T \leftarrow T + P$
12: $f \leftarrow f \frac{l(Q')\nu(S)}{\nu(Q')l(S)}$
13: **end if**
14: $m \leftarrow m - 1$
15: **end while**
16: return f

To compute a pairing, the algorithm must evaluate the *Miller loop* (lines 5 to 15 in Algorithm 1) $\log(r)$ times. This is relatively expensive as it requires a lot of extension field arithmetic and r is a large prime (the size of r will be discussed in Section 7).

If there exists a line defined over $\mathbb{F}_{q^k}$, denoted $u_{\mathcal{O}}$, which passes through $\mathcal{O}$ and is not tangent to E such that $u_{\mathcal{O}^r} f_{r,P}(\mathcal{O}) = 1$ then we say that $f_{r,P}$ is normalised ($u_{\mathcal{O}}$ is called a *rational uniformiser* at $\mathcal{O}$). In this case we can work with Q instead of D_Q. This is not restrictive so all further pairings will use $f_{r,P}(Q)$ instead of $f_{r,P}(D_Q)$.

In an effort to speed up the computation of the pairing, new pairings have been developed. These pairings have been made more efficient by incorporating observations about the structure of particular types of curves which can lead to shorter Miller loops. The idea of shortening the loop originally came from Duursma and Lee [30] and has been adapted to yield the following pairings:

3.1.2 Eta Pairing

In [8], a generalisation of the work done in [30] resulted in the *eta pairing* on points of supersingular curves. The eta pairing of points P and Q ($Q \in \langle P \rangle$, P of order r) is given by

$$e_T(P, Q) = f_{T,P}(\psi(Q)),$$

where $T = t - 1$ and ψ is a distortion map (as mentioned in §2.3.1).

3.1.3 Ate Pairing

To generalise the Eta pairing to a pairing defined over non-supersingular elliptic curves, the *Ate pairing* was developed by Hess, Smart and Vercauteren [46].

All versions of the Ate pairing can be considered as optimised versions of the Tate pairing taking as arguments points from the groups of points formed by the eigenspaces of the Frobenius endomorphism. As explained in §2, all points of order r are defined over $\mathbb{F}_{q^k}$. If r is a prime divisor of $\#E(\mathbb{F}_q)$, then there is a group of points of order r defined over $\mathbb{F}_q$. The groups formed by the eigenspaces of the Frobenius endomorphism are: $\mathbb{G}_1 = E[r] \cap \text{Ker}(\pi_q - [1]) = E(\mathbb{F}_q)[r]$, and $\mathbb{G}_2 = E[r] \cap \text{Ker}(\pi_q - [q])$, the q-eigenspace of the Frobenius endomorphism on $E[r]$.

The Ate pairing is given by:

$$e_T(Q, P) = f_{T,Q}(P)^{c(q^k-1)/N}$$

where $T = t - 1$ (as for the Eta pairing), $N = \gcd(T^k - 1, q^k - 1)$ and $c \equiv kq^{k-1} \mod r$. This gives a non-degenerate, bilinear pairing whenever $r \nmid c$ as it is clear that the Ate pairing evaluates to be a power (by c) of the Tate pairing.

Notice the reversal of parameters (in the Tate pairing the point from $\mathbb{G}_2$ is the second argument, in the Ate pairing is is the first). This makes the Miller loop more computationally complex as Q is defined over an extension field, but the shortening of the loop means the resulting pairing might be computed faster than the Tate pairing.

3.1.3.1 Twisted Ate pairing

Suppose the elliptic curve E admits a twist of order d, where $d \mid k$. From the discussion of [46], there is a unique twist E' of E or order d defined over $\mathbb{F}_{q^e}$ ($e = k/d$) with a group of points of order r. From [46, §IV], there exists a primitive dth root of unity ζ_d such that $\mathrm{Ker}([\zeta]\pi_q^e - 1)$ is isomorphic to $E'(\mathbb{F}_{q^e})$. We know that $E'(\mathbb{F}_{q^e})$ is stable under π_q (as is $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi_q - [q])$) so we may associate $\mathbb{G}_2$, a group defined over $\mathbb{F}_{q^k}$, with $E[r] \cap \mathrm{Ker}([\zeta]\pi_q^e - [1]) \cong E'[r] = \mathbb{G}_2'$, a group defined over $\mathbb{F}_{q^e}$.

The (reduced) *twisted Ate* pairing of $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2'$ is given by:

$$e_r(P, Q) = f_{T^e,P}(\varphi_d(Q))^{(q^k-1)/r},$$

where φ_d is the monomorphism $\varphi_d : E'(\mathbb{F}_{q^e}) \to E(\mathbb{F}_{q^k})$. This shortens the Miller loop when $|T^e| < r$, resulting in another speedup.

For the supersingular case, taking $P, Q \in \mathbb{G}_1$ and using the distortion map $\psi : \mathbb{G}_1 \to \mathbb{G}_2$ recovers the Eta pairing.

The Ate pairing can achieve a Miller loop length as short as $\log(T) \sim \log(r^{1/\phi(k)})$, a huge improvement over the original Tate pairing. Results of [67] showed that the Ate pairing is always at least twice as fast as the Tate pairing (in the optimal setting).

3.1.3.2 R-ate pairing

Generalising further, the R-ate pairing [61] uses ratios (hence the name) of pairings to achieve shorter Miller loops for even more sets of curves and is currently the most efficient pairing. For D_1 and D_2 divisors of E, over $\mathbb{F}_q$ with large prime order r, $r \mid \#E(\mathbb{F}_q)$ and A, B, a, $b \in \mathbb{Z}$ with $A = aB + b$, the R-ate pairing is given as:

$$R_{A,B}(D_2, D_1) = f_{a,BD_2}(D_1) \cdot f_{b,D_2}(D_1) \cdot G_{aBD_2,bD_2}(D_1),$$

where $G_{iD,jD}$ is a function with divisor $(G_{iD,jD}) = iD + jD - (iD + jD)$ (for some divisor D).

For the R-ate pairing to be non-degenerate and bilinear, the pair (A, B) must be selected carefully. Let T_i denote $q^i \mod r$, the pairs (A, B) which may render the R-ate pairing non-degenerate and bilinear, as given in [61], are:

$A = q^i$ and $B = r$,

$A = q$ and $B = T_1$, (where $T_1 < q$)

$A = T_i$ and $B = T_j$,

$A = r$ and $B = T_j$.

3.1.3.3 Optimal Pairings

The Ate and R-ate pairings reduce the length of the Miller loop of the pairing computation, for some families of curves, to $\log(r^{1/\phi(k)})$ from the length $\log(r)$ necessary to compute the Tate pairing by computing what is essentially a power of the Tate pairing. In [98], Vercauteren conjectures that $\log(r^{1/\phi(k)})$ is actually a lower bound for the number of Miller operations required to compute a pairing on an elliptic curve with no efficiently computable endomorphisms (apart from the Frobenius endomorphism).

The main idea used by the Ate and R-ate pairings to reduce the length of the Miller loop is to write a multiple of r, $c \cdot r$, to the base π_q^i, where π_q^i $(i = 0, \ldots, k-1)$ are the Frobenius endomorphisms. The Ate pairing then computes the cth power of the Tate pairing. Presuming that no other efficiently computable endomorphisms exist for a curve, Vercauteren showed that computation of the Miller loop will require at least $(1 - \varepsilon)\log(r^{1/\phi(k)})$ iterations.

Definition 3.1.1. (Optimal Pairing) A bilinear, non-degenerate pairing over a finite field $\mathbb{F}_{q^k}$, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ for the groups $\mathbb{G}_1$, G_2 and G_T as above, is called an *optimal pairing* if it can be computed in $\log(r^{1/\phi(k)} + \varepsilon(k))$ basic Miller operations (where $\varepsilon(k) \leq \log(k)$).

In [98], a method for deriving an optimal pairing for most families of elliptic curves is given.

3.2 Pairing-Friendly Elliptic Curves

We cannot use just any elliptic curves for the implementation of pairing-based protocols. We need elliptic curves with a subgroup of points of prime order r and embedding degree k with respect to r such that r is 'large enough' for the ECDLP to be hard and k is 'small enough' for the pairing to be computable and at the same time, large enough so that the DLP in $\mathbb{F}_{q^k}$ is hard (as discussed in Chapter 4).

For r to be 'large enough', we need to examine the most efficient algorithm to date for solving the ECDLP; this will be done in Chapter 6.

A 'small enough' k is the defining property of a *pairing-friendly elliptic curve* as given in [32]:

Definition 3.2.1. An elliptic curve E defined over $\mathbb{F}_q$ is said to be *pairing-friendly*, if there exists a large prime $r \geq q^{1/2}$, such that $r|\#E(\mathbb{F}_q)$, and the embedding degree k of E is less than $\log_2(r)/8$.

The bound $\log_2(r)/8$ imposed on k is to ensure that the pairing is computable.

In [7], Balasubramanian and Koblitz investigate the probability that the methods, described in Section 3.0.5, for reducing the ECDLP to the DLP in a finite field are feasible for a random elliptic curve with a large prime-order group of points. They showed that the attacks are successful when $k < \log p^2$, and that this inequality is satisfied with a probability less than $\log p^{9+\epsilon}/p$. We see now that such curves are quite specialised and rare, and though they were originally known as 'weak' curves, to be avoided for implementation, these are exactly the curves we need to implement PBC.

For PBC to be viable, we need to find rare, pairing-friendly elliptic curves $E/\mathbb{F}_q$ for which the DLP in the group of points of order r on E is as hard as the DLP in the minimal embedding field (a subfield of $\mathbb{F}_{q^k}$). Obviously, given the results of Balasubramanian and Koblitz [7], simply selecting random curves and testing for these properties is not likely to be a successful method for finding a suitable curve. Finding constructions of pairing-friendly elliptic curves (and in general, pairing-friendly abelian varieties) is an active area

of research, with many constructions already available for the implementation of PBC.

To construct pairing-friendly elliptic curves we choose the suitable parameters, p, r, k and t, (satisfying the restraints as above) then obtain the curve $E : y^2 = x^3 + ax + b$ using the *Complex Multiplication (CM) method* [3]. For this construction to be computationally feasible, the parameters must satisfy the *CM equation*, $Dv^2 = 4p - t^2$, for some small ($\lesssim 10^{16}$ [96]) square-free integer D and some integer v. The integer D is known as the *CM discriminant* (or just *discriminant*) of the endomorphism ring of E.

Supersingular Elliptic Curves

Supersingular curves were the first curves to be recognised as pairing-friendly by Menezes, Okamoto and Vanstone in [69].

Supersingular curves can have embedding degrees $k \in \{1, 2, 3, 4, 6\}$ [69], obtaining maximum embedding degrees of $k = 4$ over $\mathbb{F}_{2^n}$, $k = 6$ over $\mathbb{F}_{3^n}$ and $k = 2$ over prime fields $\mathbb{F}_p$ for $p \geq 5$. This is very restrictive, and does not allow us to implement higher levels of security efficiently; so, we will focus on non-supersingular (*ordinary*) pairing-friendly elliptic curves defined over prime fields ($n = 1$), as they are more flexible for implementation.

Ordinary Pairing-Friendly Elliptic Curves

There are two types of ordinary pairing-friendly elliptic curves, those in *families* and those not. For curves not in families, the parameters p, r, k and t are calculated separately each time. The pairing-friendly elliptic curves in families have p, r and t parametrised by univariate polynomials $p(x)$, $r(x)$ and $t(x)$ for a fixed value of k. The polynomials satisfy $Dv(x)^2 = 4p(x) - t(x)^2$, for some fixed D and $v(x)$, and $r(x) \mid (p(x) - t(x) + 1)$. A value x_0 is sought, such that $p(x_0)$ and $r(x_0)$ are primes of the correct size to give the desired security level. By construction, the values $p(x_0)$, $r(x_0)$ and $t(x_0)$ will satisfy the CM equation and the resulting elliptic curve will have CM discriminant D. Of course, for such an x_0 to be found easily enough, we place the restraint on $p(x)$ and $r(x)$ that they should

be polynomials that *represent primes*. A function $f(x)$ is said to represent primes if:

- it is non-constant and irreducible;
- the leading coefficient is positive;
- for some $x_1, x_2 \in \mathbb{Z}$, $\gcd(f(x_1), f(x_2)) = 1$ and
- for some $x' \in \mathbb{Z}$, $f(x') \in \mathbb{Z}$.

When considering which curves we would like to use in practice, we take into account the ρ-value, given by $\frac{\log(p)}{\log(r)}$, which is a measure of the efficiency of the curve. The aim is to have a maximal r for a minimal p, as the complexity of the arithmetic is determined by $\log(p)$ and we would like this to be balanced with the security given, measured by $\log(r)$; so for elliptic curves we would like $1 \leq \rho \leq 2$, with ρ closer to 1 at lower levels of security. For families of pairing-friendly curves, the ρ-value is taken to be $\lim_{x\to\infty} \frac{p(x)}{r(x)} = \frac{\deg(p)}{\deg(r)}$. Using families of pairing-friendly elliptic curves has some advantages; most notably, the families often have a smaller ρ-value than the generic curve parameter generation methods.

There are a variety of curves available for implementation, each having different advantages. Here we will consider the families of pairing-friendly elliptic curves satisfying the following criteria: We prefer to use curves with large degree twists, as these curves have a small discriminant and are therefore easy to construct and also give significant speed-up using the twisted Ate pairing [46] (as described in Section 3.1). We restrict our attention to the case of even embedding degrees, which are more useful and practical, as they support the important *denominator elimination* optimization [9], which eliminates inversions generally required to evaluate a pairing. We also follow the recommendations of Koblitz and Menezes [57, §8.3] to use curves for which the embedding degree k is of the form $k = 2^i \cdot 3^j$, for integers i and j with $i, j > 0$. Taking all these constraints into consideration and also notes in [32] on efficiency and security, Table 3.1 summarises the curves recommended for efficient implementations of pairing based protocols. These criteria are concerned with *efficiency* alone. There are some who believe that curves with too many fixed, specific parameters

Table 3.1: Suggested Curves for use in PBC

k	ρ	D	Twist d	Construction
4	2	1	4	FST [32]
6	2	3	6	FST/BLS [32, 10]
8	1.5	1	4	KSS [52]
12	1	3	6	BN [11]
16	1.25	1	4	KSS [52]
18	1.333	3	6	KSS [52]
24	1.25	3	6	BLS [10]
32	1.125	1	4	KSS [52]
36	1.167	3	6	KSS [52]
48	1.125	3	6	BLS [10]

also have vulnerabilities, if for no other reason than the fixed parameters giving a more defined target for a cryptanalyst to attack. Ideally, all system parameters should be chosen with the *"maximal possible degree of randomness"* (the "Hardline position" [55]). This is, indeed, a justified prejudice, as [29] illustrates a slightly improved Pollard's Rho algorithm for computing discrete logarithms in the groups of points on an elliptic curve with CM discriminant $D = 1$ or 3 (this is discussed in Chapter 6). This attack, however, only gives a slight advantage and though it should be taken into account, it currently does not pose a significant threat. When efficiency is a priority, we may prefer to be more lenient in our idea of which parameters should be used; taking specific parameters for which there is currently no known feasible attack but which allow more efficient implementations than truly random parameters (the "Kinder, gentler viewpoint" [55]). In practice, one may wish to take more than the speed of the calculations into account. For example, one may prefer to minimise the bandwidth requirements of a curve used in a protocol for which much interaction is necessary; in some situations, minimising the storage space or complexity of the elliptic curve arithmetic may be a priority. It is not the goal of this section to be a comprehensive guide to implementers, but to present some ideas for speeding up implementations, illustrated using the chosen curve families as examples.

3.2.1 Parameter Generation of Pairing-Friendly Elliptic Curves

First we present here a general construction method for a pairing-friendly elliptic curve of any desired embedding degree k, then we present the families of pairing-friendly elliptic curves which can be used for more efficient implementations, introduced in Table 3.1.

3.2.1.1 Cocks-Pinch curves

The Cocks-Pinch curves are described in detail in [32] and [14, Chapter IX]. Suppose the group in which the discrete exponentiation is being performed is of prime size r, then $p + 1 - t = \alpha r$ for some $\alpha \in \mathbb{Z}$.

Set the desired embedding degree k and then choose r, such that $k|r - 1$ and using a random $\omega \in \mathbb{Z}$, set $t = \omega^{\frac{r-1}{k}} \mod r$. Then from [9] we have the following:

$$(t-1)^k \equiv 1 \mod r \Rightarrow r|(t-1)^k - 1 \Rightarrow r|\Phi_k(t-1).$$

Let $v = (t-2)/\sqrt{-D} \mod r$ for some small fixed D. (This arises from solving the CM equation $4p - t^2 = Dv^2$.) As t, v and D are known, the appropriate prime can be found by testing numbers of the form $p = (t^2 d(mr + v)^2)/4$ for primality; incrementing m until a suitable prime is found.

Cocks-Pinch curves usually have a ρ-value of 2 which is by no means optimal, but they give a kind of 'safety net' to PBC. A ρ-value of 1 is achieved by some families of pairing-friendly curves, which will be presented shortly, and so a much more efficient implementation is possible. Families of pairing-friendly curves could have a potential disadvantage, should a more efficient algorithm to solve the ECDLP on a particular family of curves be developed, then the security of the entire family of curves would suffer. Cocks-Pinch curves do not seem to belong to a family of curves and so a more general technique is required to attack the DLP on each curve individually. It is an open problem to show whether or not a particular Cocks-Pinch curves belong to a family of curves or not.

3.2.1.2 Families of Pairing-Friendly Elliptic Curves

All non-supersingular pairing-friendly elliptic curves are *CM curves*; that is, the curves are constructed using the CM method from the chosen parameters. The Taxonomy of Pairing-Friendly Elliptic Curves [32] presents all currently known methods for selecting the parameters for the construction of pairing-friendly elliptic curves using the CM method. Some new families of constructions are also given, these are known as FST curves.

FST $k = 4$

$$\begin{aligned} p(x) &= \frac{1}{4}(x^4 - 2x^3 + 2x^2 + 2x + 1); \\ r(x) &= \Phi_4(x); \\ t(x) &= x + 1. \end{aligned}$$

This family of curves has a ρ-value of 2 and a discriminant $D = 1$ which means that this curve had a quartic twist defined over $\mathbb{F}_p$.

FST $k = 6$

$$\begin{aligned} p(x) &= \frac{1}{3}(x-1)^2(x^2 - x + 1) + x; \\ r(x) &= \Phi_6(x); \\ t(x) &= x + 1. \end{aligned}$$

The FST construction given for curves with embedding degree $k \equiv 0 \mod 6$ is actually the same as the BLS construction for all k of the form $k = 2^i 3^j$ when $i, j > 0$. This family has a ρ-value of 2 and discriminant $D = 3$, which means that the curve constructed using these parameters has a sextic twist, also defined over $\mathbb{F}_p$, which can be used to implement the Ate pairing efficiently.

BN Curves

The construction of BN pairing-friendly elliptic curves is given by Barreto and Naehrig in [11]. BN curves are a family of curves with embedding degree $k = 12$ and $D = 3$, with the parameters p, r and t given by

$$\begin{aligned} p(x) &= 36x^4 + 36x^3 + 24x^2 + 6x + 1; \\ r(x) &= 36x^4 + 36x^3 + 18x^2 + 6x + 1; \\ t(x) &= 6x^2 + 1. \end{aligned}$$

The Elliptic curve resulting from such parameters will take the form $E : y^2 = x^3 + b$, for some non-zero b and, since $D = 3$ and 6 divides 12, we know that it has a sextic twist, defined over $\mathbb{F}_{p^2}$. BN curves are plentiful and easy to find.

KSS Curves

The construction of KSS curves, as described in [52], finds pairing friendly elliptic curves with embedding degrees $k = 8, 16, 18, 32, 36$ and 40 with better ρ-value than has previously been achieved for these embedding degrees. The most noted, however, is the construction of the family of curves with embedding degree $k = 18$ with ρ-value, $4/3$. Although this ρ-value does not seem to be sufficiently low, the fact that these curves have sextic twists defined over $\mathbb{F}_{p^3}$ makes the implementation of the KSS curves with embedding degree 18 very efficient. We present here the KSS families of focus, those with embedding degrees $k = 8, 16, 18, 32$ and 36. All KSS curves with embedding degree divisible by 6 have a CM discriminant $D = 3$, and so have sextic twists. KSS curves with embedding degree a power of 2 have CM discriminant $D = 1$, and therefore have quartic twists.

KSS $k = 8$

For embedding degree $k = 8$, the system parameters are given by the following polynomials:

$$\begin{aligned} p(x) &= (x^6 + 2x^5 - 3x^4 + 8x^3 - 15x^2 - 82x + 125)/180; \\ r(x) &= (x^4 - 8x^2 + 25)/450; \\ t(x) &= (2x^3 - 11x + 15)/15. \end{aligned}$$

For these curves $\rho = 3/2$. In this case, if we take $x \equiv \pm 5 \mod 30$ then $t(x)$ has integer values and $r(x)$ and $p(x)$ represent primes.

KSS $k = 16$

The family of KSS curves with embedding degree $k = 16$ is parameterised by the following polynomials:

$$\begin{aligned} p(x) &= (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 + 625x^2 + 2398x + 3125)/980; \\ r(x) &= (x^8 + 48x^4 + 625)/61250; \\ t(x) &= (2x^5 + 41x + 35)/35. \end{aligned}$$

This family of curves has ρ-value $5/4$. For values of x satisfying $x \equiv \pm 25 \mod 70$, $t(x)$ has integer values and $r(x)$ and $p(x)$ represent primes.

KSS $k = 18$

For embedding degree $k = 18$, we use the following parameterisation:

$$\begin{aligned} p(x) &= (x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401)/21, \\ r(x) &= (x^6 + 37x^3 + 343)/343, \\ t(x) &= (x^4 + 16x + 7)/7. \end{aligned}$$

These curves have $\rho = 4/3$. We need to find a value for x which is $14 \mod 42$ to find appropriate values for p, r and t.

KSS $k = 32$

The family of KSS curves with embedding degree $k = 32$ has parameters given by:

$$\begin{aligned} p(x) &= (x^{18} - 6x^{17} + 57120x^{10} - 344632x^9 + 742560x^8 + 815730721x^2 \\ &\quad -4948305594x + 10604499373)/2970292; \\ r(x) &= (x^{16} + 57120x^8 + 815730721)/93190709028482; \\ t(x) &= (-2x^9 - 56403x + 3107)/3107. \end{aligned}$$

These curves have a ρ-value of $9/8$. Taking $x \equiv \pm 325 \mod 6214$, we find that $t(x)$ has integer solutions, and $r(x)$ and $p(x)$ represent primes.

KSS $k = 36$

The KSS $k = 36$ curves have a ρ-value $7/6$ and the system parameters are given by:

$$\begin{aligned} p(x) &= (x^{14} - 4x^{13} + 7x^{12} + 683x^8 - 2510x^7 + 4781x^6 + 117649x^2 - 386569x \\ &\quad +823543)/28749; \\ r(x) &= (x^{12} + 683x^6 + 117649)/161061481; \\ t(x) &= (2x^7 + 757x + 259)/259. \end{aligned}$$

If we take $x \equiv 287 \mod 777$, then $t(x)$ has integer solutions, and $r(x)$ and $p(x)$ represent primes.

BLS Curves

In [10] Barreto, Lynn and Scott give a method for constructing pairing-friendly elliptic curves with embedding degree $k = 2^i 3^j q^s$, known as the BLS curves. These curves are

favourable for implementation for the values $k = 24$ and $k = 48$.

BLS $k = 24$

For the case $k = 24$ the parameters are given by:

$$\begin{aligned} r(x) &= \Phi_{24}(x); \\ p(x) &= \frac{1}{3}(x-1)^2 r(x) + x; \\ t(x) &= x + 1. \end{aligned}$$

BLS $k = 48$

The parameters for $k = 48$ are given by:

$$\begin{aligned} r(x) &= \Phi_{48}(x); \\ p(x) &= \frac{1}{3}(x-1)^2 r(x) + x; \\ t(x) &= x + 1; \end{aligned}$$

for both the $k = 24$ and the $k = 48$ cases, the chosen value of x needs to satisfy $x \equiv 1 \mod 3$ for $r(x)$ and $p(x)$ to be integers (and potentially primes).

In the following sections $r(x)$, $p(x)$ and $t(x)$ denote the parameterisation of the primes and the trace of the Frobenius of families of pairing friendly-elliptic curves. The particular value of x chosen to generate the appropriate parameters will be denoted x_0, and the values given by $r(x_0)$, $p(x_0)$ and $t(x_0)$ will be denoted simply as r, p and t respectively.

From the discussion in [65] we see that in the near future, a security level of 128-bits is expected to become the standard minimum. We shall use the BN curves and the KSS $k = 18$ curves as the main example families throughout this thesis, to illustrate the methods given; these curve families are appropriate for use at ~ 128 and ~ 192-bit security levels respectively, as will be made clear in Chapter 7. The methods are applied to other selected families in the appendix.

Part I

Security

Security

- Fighting Red and White Knights, Lewis Carroll's *Through the Looking-Glass and What Alice Found There*,

Chapter VIII, It's my own Invention, illustration by Sir John Tenniel.

All public-key cryptosystems are breakable in theory, using brute force (or blind luck); given the public key, one could try every possible private key and eventually find the correct one. Simply because an attack exists does not mean a cryptosystem is insecure. The attack may not be practical; if one chooses the system parameters correctly, the attacks are rendered computationally infeasible. For example, if the only known attack on a particular protocol is an exhaustive private-key search, we can choose the system parameters such that there is a very low probability that an adversary will find the correct key by simply 'testing' until his computational resources are exhausted. Of course testing all keys, an adversary will eventually find the correct key but this could take far too long, possibly longer than the

lifetime of the attacker. The attacker may also be lucky and guess the correct key the first try, but the probability of this happening is very small. When assessing the level of security a protocol offers, one needs to consider all possible attacks which can be used and what kind of adversary one could be up against. One needs to take into account the following: the knowledge and capabilities of an adversary and the resources they have available; to what extent we consider an attack successful (when the private key is obtained or just a few bits of a single message); the computational complexity of the attacks and how long the computation will take given the adversary's available resources; and, if a weakened version of the protocol is broken or its most general setting.

We see, therefore, that it would be impossible to give the exact security of all pairing-based protocols in one thesis; there are too many variables to take into account and a continually growing number of protocols. We must content ourselves with an examination of one aspect of the security. Here we choose to focus on the hardness of the problem underlying the security of most pairing-based protocols.

One of the concerns about the security of pairing-based protocols is that it relies on the hardness of solving the BDHP, which is not a well understood problem. It is easily seen that if we can solve the ECDLP or the DLP in a finite field, then we are able to solve the BDHP. What is not known is if the BDHP is as hard as these two problems or if it is solvable using other methods. In [56] Koblitz and Menezes discuss results of [99], which show that we are only able to prove that the ECDHP on a pairing-friendly elliptic curve is as hard as the DHP in a finite field if both problems are easy. Koblitz and Menezes speculate that, in a similar fashion, the BDHP is not likely to be shown equivalent to any 'standard' problem which is considered hard, unless both problems are easy. They argue that this need not be an issue. Currently, the best method for solving the BDHP is by solving the ECDHP or the DHP, which is solved by solving one of the DLP instances (as summarised in [56], there is significant evidence that (EC)DLP and (EC)DHP are equivalent). To give minimal requirements for system parameters, we examine the most efficient algorithms, to date, to solve the ECDLP and DLP. This approach does simplify the setting significantly and is

by no means exhaustive, but can be considered to give minimal requirements; a pairing-based protocol will not achieve high levels of security unless the DLP and ECDLP are hard. Choosing the parameters to ensure this does not necessarily imply that a protocol is secure, but is a necessary first step. The same approach is taken in [63].

We continue with the following notation: E is a pairing-friendly elliptic curve defined over a finite field $\mathbb{F}_q$ with a group of points of large, prime order r and embedding degree k with respect to r.

We begin in Chapter 4 by looking at the relationship between the ECDLP and the DLP instances occurring in PBC. We know that the DLP instance arises in a subgroup of a finite field contained in $\mathbb{F}_{q^k}$, but in order to give accurate parameter size recommendations for implementations we need to establish exactly the size of the minimal embedding field. The work of this section resulted in the publication [12] and is joint work with Manuel Charlemagne and David Freeman.

In Chapters 5 and 6 the algorithms used to solve the DLP in the finite field and the ECDLP, respectively, will be presented.

In Chapter 7 we use the details of these algorithms to give some approximations for the sizes of the parameters necessary to achieve certain security levels.

Chapter 4

ECDLP and DLP Relationship in PBC

'Sometimes I've believed as many as six impossible things before breakfast.'

- The Queen, Lewis Carroll's *Through the Looking-Glass and What Alice Found There*, Chapter, Wool and Water.

From Chapter 3 we have seen that an instance of the ECDLP maps to an instance of the DLP in a finite field. To be able to set up an efficient pairing-based protocol, we need to choose the system parameters such that the two instances of the DLP are equally hard; in order to do this, we need to know into which finite field the DLP maps. In this section the relationship between the instances of the DLP occurring in PBC will be presented. The results of this section do not take into account any specific properties of elliptic curves and will therefore be presented in their most general sense, for abelian varieties. The definitions of embedding degree and minimal embedding field of an elliptic curve given in Chapter 2 can be generalised to definitions for abelian varieties. First we set some notation for this section: Let A be an abelian variety of genus g defined over $\mathbb{F}_q$, $q = p^m$ for some prime p

and integer m, and denote by r, $r \neq p$, a large prime dividing $\#A(\mathbb{F}_q)$.

Embedding Degree: The *embedding degree of A with respect to r* is the smallest integer k such that r divides $q^k - 1$ and r does not divide $q^i - 1$ for all $1 \leq i < k$.

Minimal Embedding Field: The *minimal embedding field of A with respect to r*, denoted F, is the smallest extension of $\mathbb{F}_p$ containing the rth roots of unity $\mu_r \subset \bar{\mathbb{F}}_p$.

From the definitions it is clear that the minimal embedding field of A with respect to r will be contained in the field $\mathbb{F}_{q^k}$. When $m = 1$ the abelian variety is defined over a prime field and it follows that the minimal embedding field of A with respect to r is exactly $\mathbb{F}_{q^k}$.

The situation becomes more complex when $m > 1$; the minimal embedding field can be a subfield of $\mathbb{F}_{q^k}$, much smaller than $\mathbb{F}_{q^k}$ itself.

4.1 Determining the Minimal Embedding Field

If an abelian variety $A/\mathbb{F}_q$ has embedding degree k with respect to r, then $\mathbb{F}_{q^k}$ is the smallest extension of $\mathbb{F}_q$ containing the rth roots of unity. The essential point to note is that the minimal embedding field is defined as an extension of the base field $\mathbb{F}_p$ and not of $\mathbb{F}_q$. It was observed by Rubin and Silverberg [79] and Hitt [47] that bilinear pairings return values in the minimal embedding field, not necessarily $\mathbb{F}_{q^k}$ as was previously believed. In particular, the Weil pairing and the Tate pairing (and variations thereof) take values in a subgroup and a quotient group of $\mathbb{F}_{q^k}^*$, respectively. In both papers it was shown that it is likely that the minimal embedding field is a proper subfield of $\mathbb{F}_{q^k}$.

This observation, found in different forms in each paper, is expressed by Hitt as follows:

Lemma 4.1.1. *[47] Let $q = p^m$, for p a prime and m a positive integer. For some prime*

$r \neq p$ and k the smallest positive integer such that $q^k \equiv 1 \mod r$, k is given by:

$$k = \frac{\operatorname{ord}_r p}{\gcd(\operatorname{ord}_r p, m)},$$

where $\operatorname{ord}_r p = x > 0 \in \mathbb{Z}$ such that $p^x \equiv 1 \mod r$ and x minimal. It suffices to have $k' \in \mathbb{Q}$, that is, have k' such that $r | q^{k'} - 1$ and $k' = \frac{\operatorname{ord}_r p}{m}$. The minimal embedding field is given by $\mathbb{F}_{q^{k'}}$.

The result of this lemma is that the minimal embedding field of an abelian variety $A/\mathbb{F}_q$ is $\mathbb{F}_{q^{k'}}$, where $k' = \operatorname{ord}_r(p)/m \in \mathbb{Q}$, which is not necessarily the same as $\mathbb{F}_{q^k}$. In both papers [79] and [47], Rubin and Silverberg and Hitt pointed out this implies that previous assumptions of the security of pairing-based protocols being based on the hardness of the DLP in the group of points of size r on A and in the finite field $\mathbb{F}_{q^k}$ give a misleading overestimation of the security of the protocol on the finite field side.

It is important to note that when the abelian variety is defined over a prime field (that is, when $m = 1$) these observations have no effect as the minimal embedding field is always $\mathbb{F}_{p^k}$.

Hitt gives examples of abelian varieties where $k/k' = m$, which is the largest possible ratio for these parameters [47, §4]. In this case, the DLP occurs in $\mathbb{F}_{q^k}$, not $\mathbb{F}_{q^{mk}}$ which would have been used for the security estimate. This is clearly a problem for implementers of PBC. In order to give a realistic estimate of the hardness of the DLP instance which occurs in the protocol, it is imperative that one knows the finite field the pairing maps to.

For supersingular abelian varieties, which are currently the only known constructable pairing-friendly abelian varieties over extension fields, Rubin and Silverberg propose a solution. In [79] they give a theorem which distinguishes the minimal embedding field of a supersingular abelian variety.

Theorem 4.1.2 ([79, Theorem 7])**.** *Suppose A is an elementary supersingular abelian variety of dimension g over $\mathbb{F}_q$, $q = p^m$, $r \neq p$ is a prime divisor of $\#A(\mathbb{F}_q)$, and s is the multiplicative order of p mod r. Let $F_A(x) \in \mathbb{Z}[x]$ be the characteristic polynomial of*

Frobenius for A, and let f be the unique integer such that $F_A(x)^{1/f}$ is irreducible in $\mathbb{Z}[x]$. If q is a square, assume $r > (1+p)^{mg/2f}$. If q is not a square, assume $r > (1+\sqrt{p})^{2mg/3f}$ and $r > 7$. Then $p^s = q^{c_{A,q}}$, so $\mathbb{F}_{q^{c_{A,q}}}$ is the smallest extension of $\mathbb{F}_p$ whose multiplicative group has a subgroup of order r.

In other words, if $A/\mathbb{F}_q$ is an elementary supersingular abelian variety, where $q = p^m$, with embedding degree k with respect to r then we can expect the minimal embedding field to take on one of two forms:

- If m is odd and $r > (1+\sqrt{p})^{2mg/3}$, then the minimal embedding field of A with respect to r is $\mathbb{F}_{q^k}$.
- If m is even and $r > (1+p)^{mg/2}$ then the minimal embedding field of A with respect to r is either $\mathbb{F}_{q^k}$ or $\mathbb{F}_{q^{k/2}}$.

In the case of supersingular curves, the minimal embedding field is therefore either $\mathbb{F}_{q^k}$ or $\mathbb{F}_{q^{k/2}}$. The actual field is determined using the *cryptographic exponent*, the $c_{A,q}$ defined as in Theorem (4.1.2). This theorem is slightly cumbersome and has two disadvantages:

- The genus g of the abelian variety affects the lower bound on r. This can become quite restrictive.
- It is only defined for supersingular abelian varieties.

Only supersingular abelian varieties are considered in [79], with the restult that the minimal embedding field is either $\mathbb{F}_{q^k}$ or $\mathbb{F}_{q^{k/2}}$. As illustrated by the examples given by Hitt [47], however, the minimal embedding field of ordinary abelian varieties could be up to m times smaller than $\mathbb{F}_{q^k}$. It was clear that there was room for more investigation into the sizes of the minimal embedding fields of ordinary abelian varieties.

In [12] we give a new theorem:

Theorem 4.1.3. *Let k be a positive integer, p^m a prime power, and r a prime. Write $m = \alpha\beta$, where every prime dividing α also divides k and $\gcd(k, \beta) = 1$. (This factorization is*

unique.) Denote by e the smallest prime factor of β. Suppose $r \mid \Phi_k(p^m)$ and that one of the following holds:

1. $m = \alpha$ *(and* $\beta = 1$*);*
2. β *is prime and* $r > \Phi_{k\alpha}(p)$*;*
3. $r > p^{km/e}$*; or*
4. $4 \mid m$*; or* $2 \mid k$ *and* $r > p^{km/2e} + 1$.

Then $r \mid \Phi_{km}(p)$.

Proof. The proof uses results and properties presented in Section 2.2. We first note that Fact 2.2.4 (4) implies

$$\Phi_k(p^m) = \Phi_{k\alpha}(p^\beta). \tag{4.1}$$

Since $k\alpha$ and β are coprime, Lemma 2.2.5 implies that $\Phi_k(p^m)$ has $\Phi_{km}(p)$ as a factor. Our strategy in each case is to show that the remaining factors of $\Phi_k(p^m)$ are all smaller than r. Since r is prime, it then follows that if r divides $\Phi_k(p^m)$ then r divides $\Phi_{km}(p)$.

We now consider each case separately:

1. Since $m = \alpha$ it follows immediately that $\Phi_k(p^m) = \Phi_{km}(p)$.
2. Since β is a prime not dividing $k\alpha$, equation (4.1) and Fact 2.2.4 (3) imply that

$$\Phi_k(p^m) = \Phi_{k\alpha\beta}(p)\Phi_{k\alpha}(p) = \Phi_{km}(p)\Phi_{k\alpha}(p).$$

   Since $r > \Phi_{k\alpha}(p)$, it follows that $r \mid \Phi_{km}(p)$.

3. By equation (4.1) and Lemma 2.2.5 we have

$$\Phi_k(p^m) = \prod_{d|\beta} \Phi_{kd\alpha}(p) = \prod_{d|\beta} \Phi_{km/d}(p). \tag{4.2}$$

   By assumption we have $r > p^{km/d}$ for all $d \mid \beta$ except for $d = 1$, and by Fact 2.2.4 (1) we have $p^{km/d} > \Phi_{km/d}(p)$ for all such d. It follows that $r \mid \Phi_{km}(p)$.

4. Given the factorization of $\Phi_k(p^m)$ as in (4.2), the same analysis as in Case 3 shows that $r > \Phi_{km/d}(p)$ for all $d \mid \beta$ with $d \geq 2e$. Since e is the smallest prime dividing β, if $d \mid \beta$ and $1 < d < 2e$ then d is prime, so it suffices to show that $r > \Phi_{km/d}(p)$ for all primes d dividing β. Let d be such a prime. The assumption $4 \mid m$ or $2 \mid k$ then implies that km/d is even. In this case we have $x^{km/d} - 1 = (x^{km/2d} + 1)(x^{km/2d} - 1)$, and $\Phi_{km/d}(x)$ must divide the first factor by Fact 2.2.4 (1). Since $d \geq e$, if $r > p^{km/2e} + 1$ then $r > \Phi_{km/d}(p)$.

□

The theorem is stated in terms of cyclotomic polynomials, with no reference to abelian varieties, particularly no reference to properties of specific types of abelian varieties. We need another Lemma to clearly show how this theorem determines the minimal embedding field of an abelian variety.

Lemma 4.1.4. *Let $q = p^m$ be a prime power, and $A/\mathbb{F}_q$ be an abelian variety. Let $r \neq p$ be a prime dividing $\#A(\mathbb{F}_q)$, and let k, s be integers not divisible by r. Then*

1. *A has embedding degree k with respect to r if and only if $r \mid \Phi_k(q)$.*

2. *The minimal embedding field of A with respect to r is $\mathbb{F}_{p^s}$ if and only if $r \mid \Phi_s(p)$.*

Proof. The first statement appears e.g. as [32, Proposition 2.4]; we observe that the same proof applies to the second statement. □

Using Lemma 4.1.4 to interpret Theorem 4.1.3 in the context of abelian varieties, we obtain the following corollary:

Corollary 4.1.5. *Let A, $q = p^m$, r and k be as above. Assume that $r \nmid km$. If q, k, and r satisfy any of the conditions* (1)–(4) *of Theorem 4.1.3, then the minimal embedding field of A with respect to r is $\mathbb{F}_{p^{km}}$.*

We note that in the case where m is prime, which is usually recommended for cryptographic applications in order to prevent Weil descent attacks (e.g., [37, 38]) we usually have

$r \approx p^{mg}$ (where $g = \dim A$) and $m \gg k$, so we are in case (2) of Theorem 4.1.3. If p is small ($p = 2$ and $p = 3$ are common choices) then in this situation the bound on r given by the theorem is very weak; that is, A will have minimal embedding field $\mathbb{F}_{p^{km}}$ with respect to any r of cryptographic size.

Ideally we would also like to apply Theorem 4.1.3 to abelian varieties over finite fields that are *not* pairing-friendly. Specifically, if $A/\mathbb{F}_q$ is an abelian variety chosen for a *non*-pairing-based cryptographic protocol, one wants to make sure that the discrete logarithm problem in $A(\mathbb{F}_q)[r]$ cannot be reduced to a more tractable discrete logarithm problem in a finite field. Thus one must ensure not only that the embedding degree k is sufficiently large, but also that the minimal embedding field is sufficiently large. However, if $k \gg m$ and the genus g is small then none of the conditions of Theorem 4.1.3 can be expected to hold: Condition (1) is very unlikely and conditions (2)–(4) would require $r \gg q^g$, which is impossible (as the number of points is $\approx q^g$ and r is a divisor of the number of points).

Remark 4.1.6. If k is odd and m is even then $\Phi_k(x^m) = \Phi_k(x^{m/2})\Phi_{2k}(x^{m/2})$. Since $\varphi(k) = \varphi(2k)$ for odd k, these two factors have the same degree and we cannot use the above techniques to show that r divides $\Phi_{km}(p)$ and does not divide $\Phi_{km/2}(p)$. Applying Theorem 4.1.3 recursively to each factor allows us to determine conditions on q, k, and r guaranteeing that r divides one of the two expressions $\Phi_{km}(p)$ and $\Phi_{km/2}(p)$, but additional information is needed to determine which one. In the context of pairing-friendly curves, this situation rarely occurs as even embedding degrees are preferred, as are prime values for m.

To summarise the contribution of Theorem 4.1.3, we now have conditions on the parameters p, k, m, and r for use in PBC which, when satisfied, ensure that the minimal embedding field of an abelian variety A over $\mathbb{F}_{p^m}$ with group of points of size r has minimal embedding field $\mathbb{F}_{p^{km}}$ with respect to r. These parameters already satisfy some basic conditions simply because of the role they have in cryptography and for implementation's sake: the ECDLP must be computationally infeasible in the group of size r and m is prime or near prime. These conditions are almost enough to ensure that the minimal embedding field is $\mathbb{F}_{p^{km}}$. Two advantages of Theorem 4.1.3 over Theorem 4.1.2 are:

- The bound on r given in Theorem 4.1.3 is not augmented by the genus, this means that the conditions do not become more restrictive for higher genus abelian varieties.

- Theorem 4.1.3 is applicable to all abelian varieties, not just supersingular ones. Currently, the only constructable abelian varieties over non prime fields are the supersingular varieties over fields of characteristic 2 and 3. Future discoveries of pairing-friendly abelian varieties over non prime fields will be immediately assessable using Theorem 4.1.3.

We are now easily able to verify whether the minimal embedding field is $\mathbb{F}_{p^{km}} = \mathbb{F}_{q^k}$ using Theorem 4.1.3, so without loss of generality we may now safely presume that the finite field DLP instance occurring in PBC is in $\mathbb{F}_{q^k}$.

Chapter 5

Solving the DLP in Finite Fields

'You may call it 'nonsense' if you like ... but I've heard nonsense, compared with which that would be as sensible as a dictionary!'

- The Queen, Lewis Carroll's *Through the Looking-Glass and What Alice Found There*, Chapter II, The Garden of Live Flowers.

From the results presented in Chapter 4 we may presume that the finite field DLP instance occurring in PBC is in a field of the form $\mathbb{F}_{q^k}$, where k is the embedding degree of the elliptic curve with respect to the group size. The methods for solving the DLP in prime, binary and ternary fields are well understood and a lot of interest has been focused on these cases. Until now, the DLP in medium prime extension fields has received relatively little attention. Many of the limitations on the 'exploration' of the use of the method for these cases are computational restrictions; the asymptotic complexity can not be thoroughly tested in practice as this is computationally infeasible.

Before presenting the algorithms of this and the subsequent chapter we need to introduce some notation. To describe the performance of an algorithm we use the following notation:

Definition 5.0.7. *Big O notation* is used to describe an algorithm's behaviour as input parameters of the algorithm tend to a specific value (such as infinity). It gives an estimate for the necessary resources to run a particular algorithm, for example computing power or storage space.

In the context here, the big O notation will be used to describe the growth rate of running time and storage requirements as the size of the the finite field in which the DLP instance occurs tends towards infinity.

Sub-exponential complexity is expressed using L-notation, which is slightly more concise than the big O notation; it is used to describe the behaviour of algorithms with run time which is between polynomial and exponential.

Definition 5.0.8. The L-notation is given by

$$L_n(\alpha, c) = O(\exp\{(c + o(1)) \ln(n)^{\alpha} \ln(\ln(n))^{1-\alpha}\})$$

where c is a positive constant and $0 < \alpha < 1$.

When $\alpha = 0$ we see that the run time of the algorithm is polynomial in $\log(n)$, and when $\alpha = 1$ the algorithm is fully exponential in n.

5.1 Early Attacks

There are a few obvious ways to solve the DLP in $\mathbb{F}_q$. A *brute force* attack computes all a^x for all $x \in 1, \ldots, \phi(q)$. This method needs only $O(1)$ memory, but takes $O(q)$ time. This attack does not use available computing resources to their full potential. Similarly, precomputing all possible values and storing the pairs $(a^x \in \mathbb{F}_q, x)$ takes $O(q)$ memory, $O(q)$ precomputation time and constant time to solve the discrete logarithm. To have an effective algorithm, there should be a balance between memory used and run time.

BSGS

The 'Baby-Step-Giant-Step' algorithm (developed by Shanks) [20] is also often referred to as Shanks' Algorithm. It has running time $O(\sqrt{q})$, where q is the size of the field. The basic idea used in this algorithm is to write $x = \mathrm{Log}_a(b)$ as $x = mj+i$ for integers $m = \sqrt{p-1}$, j and i, so as to find x. The algorithm generates lists ordered with respect to the second coordinate $L_1 = (j, a^{mj} \mod p)$ and $L_2 = (i, ba^{-i} \mod p)$ by systematically 'stepping' through the positive integers i and j (L_1 is the giant step list, stepping m at a time, L_2 is the baby step list). If there are elements $(j, y) \in L_1$ and $(i, y) \in L_2$ then $a^{mj} = y = ba^{-i}$, which gives $a^{mj+i} = b = a^x$. Thus $x = \mathrm{Log}_a(b)$ has been found.

Pollard's Rho Method

This algorithm, based on the birthday paradox, can be used to solve the DLP in any group in run time $O(\sqrt{q})$, where q is the size of the group. It is the most efficient known algorithm for solving the DLP in a generic group and will be presented in detail in Section 6.2.

Pohlig-Hellman

Suppose $n = \prod_{i=1}^{r} p_i^{e_i}$ is the prime factorisation of $n = \phi(q)$. Since $\mathrm{Log}_b(a) = x$ is unique modulo n, the idea is to find a system of equations $x_i \equiv x \mod p_i^{e_i}$ for $1 \leq i \leq r$, and then find $x \mod n$ using the Chinese remainder theorem. The x_i are computed by computing the coefficients of the x_i in the p_i-ary representation: $x_i = l_{i,0} + l_{i,1}p_i + \ldots + l_{i,e_i-1}p_i^{e_i-1}$. For the computation of each x_i, it is necessary to solve the DLP in a group of size p_i.

5.2 Index Calculus

The methods mentioned above are known as *generic* algorithms, the run time (and, in some cases, memory required) for the algorithm is $O(\sqrt{q})$, where q is the cardinality of the group in which the DLP is to be solved.

Currently, the most efficient algorithms for solving the DLP in finite fields are index

calculus methods which have run time and memory requirements both $L_q(1/3, c)$, where q is the cardinality of the field. These particular index calculus methods were developed to factor large numbers, but have been modified to computing discrete logarithms with similar success.

To find the discrete logarithm x of $b \in \mathbb{F}_{p^k}$ with respect to a ($b \in \langle a \rangle$), the index calculus method has 2 preliminary stages, a sieving stage and a linear algebra stage.

5.2.1 Sieving

The goal of the sieving stage is to find a series of linear relations in the logarithms of a fixed set of elements. This is done by selecting two isomorphic representations of $\mathbb{F}_{p^k}$, say $\mathcal{K}_1$ and $\mathcal{K}_2$, and fixing a subset of elements in each of the representations, called the *Factor Base*, denoted by $\mathcal{F}_1$ and $\mathcal{F}_2$. The Factor Bases usually contain the elements of small norm or degree (bounded by a *smoothness bound* $\mathcal{B}_i$, $i = [1, 2]$), depending on the particular representations of the field.

The elements of $\mathbb{F}_{p^k}$ are then 'sieved' to find pairs (α, β), $\alpha \in \mathcal{K}_1$ and $\beta \in \mathcal{K}_2$, with $\alpha \cong \beta$, and α and β *smooth* over their respective factor bases; that is, α and β can be written as a product of elements of their respective factor bases. Such pairs are called *doubly smooth*. The sieving continues until enough relations $\alpha = \prod_{\gamma_i \in \mathcal{F}_1} \gamma_i^{a_i} \cong \prod_{\gamma_j \in \mathcal{F}_2} \gamma_j^{b_j} = \beta$ are found (more than the number of elements in the factor base).

These relations are converted to linear equations in the logarithms of the factor base elements by taking the discrete logarithms of each side with respect to fixed (isomorphic) elements in each representation.

5.2.2 Linear Algebra

The equations obtained in the sieving stage are solved using linear algebra modulo the group order to find the logarithms of the elements in the factor bases.

Solving the linear equations is no trivial task. Though the matrix of equations is originally sparse, after only a few operations it becomes dense. There are a few popular meth-

ods for reducing this matrix and solving for the unknown logarithms: Lanczos's algorithm, Wiederman's algorithm and structured Gaussian elimination. Some algorithms use a combination of these methods [58].

5.2.3 Finding the Individual Logarithm

Once the preliminary stages have been completed and the logarithms of the factor base elements have been found, the next step is to calculate the discrete logarithms of individual elements, not included in the factor base. The methods used vary between algorithms, usually using a variation of the *special-*$\mathfrak{q}$ *descent*.

Suppose we wish to calculate the discrete logarithm of b with respect to a. If b is in the factor base (or $a = b$), the discrete logarithm has already been found. For some element b not in the factor base, denote the ideal generated by b by $\mathfrak{q} = (b)$. To find the discrete logarithm of b: search for an element δ such that $\mathfrak{q}$ divides $\Delta = (\delta)$ and the norm of Δ factors into primes smaller than some value D. Factor Δ into a product of ideals and repeat this process on the factors, continually lowering the bound D, until $D < \mathcal{B}$ and all factors are in the factor base.

Replace the large factors in the previous step with the smooth elements found until $\mathfrak{q}$ is written as a product of factor base elements and the logarithm of b is easily determined.

Currently, the most efficient index calculus methods have complexity $L_q(1/3, c)$, where $(32/9)^{1/3} \leq c \leq (64/9)^{1/3}$ using the Function Field Sieve [50] and the Number Field Sieve [51]. The best sieve to use for a particular instance of the DLP depends on the size of the field $\mathbb{F}_{p^k}$ and also on the relative sizes of p and k (as specified in [50] and [51]). The optimal run time will not always be achievable for all combinations of p and k.

Implementation

In some cases, using a combination of an index calculus method and a square root method can be more efficient. For example, since $\log_b(a) = x$ is calculated in a group of size $p^k - 1$, if $p^k - 1$ factorises into a product of relatively small primes, the problem of finding discrete

logarithms can be reduced.

- For each small prime (power) factor m of $p^k - 1$, the discrete logarithm modulo m can be computed using Pollard's Rho method.
- For the larger prime factors l of $p^k - 1$, the index calculus method can be used to compute the discrete logarithm modulo l.

These results can then be combined using the Chinese Remainder Theorem [20] to compute the logarithm modulo $p^k - 1$.

The factorisation of $p^k - 1$ can be found efficiently by considering the factorisation of $x^k - 1 = \prod_{l|k} \Phi_l(x)$, where $\Phi_l(x)$ is the lth cyclotomic polynomial, and substituting p for x, then factorising the smaller factors further using, if necessary, a powerful factoring algorithm. Factoring algorithms using the number field sieve have complexity $L_q(1/3)$. As mentioned, the algorithms for finding discrete logarithms in this case also have complexity $L_q(1/3)$, and so the factoring of $p^k - 1$ adds nothing to the overall complexity of the algorithm and in all practical settings $p^k - 1$ contains a large prime factor.

In the PBC setting we already know a large prime factor of $\Phi_k(p)$, r, the divisor of $\#E(\mathbb{F}_{p^k})$; in fact, we only need to compute the discrete logarithm modulo r in this case.

5.2.4 DLP Index Calculus Algorithms

There are two main index calculus methods for solving the DLP, the Function Field Sieve (FFS) and the Number Field Sieve (NFS). The most obvious difference between these two methods is the choice of representations used for $\mathbb{F}_{p^k}$: the FFS uses $\mathbb{F}_{p^k} \cong \mathbb{F}_p[x]/f(x)\mathbb{F}_p[x]$ where $f(x)$ is an irreducible polynomial of degree k; the NFS uses $\mathbb{F}_{p^k} \cong \mathcal{O}_{\mathcal{K}}/(p)$ where $\mathcal{K} = \mathbb{Q}(\theta)$, (p) is the ideal generated by p and the minimal polynomial of θ is of degree k and is irreducible over $\mathbb{F}_p$. The factor bases in the FFS consist of ideals of $\mathbb{F}_p[x]/f(x)\mathbb{F}_p[x]$ with norm a small degree polynomial, and the factor bases of the NFS are the ideals of $\mathcal{O}_{\mathcal{K}}$ with norms less than a given bound.

The FFS and the NFS have similar complexities so we need to examine the details of the asymptotic analysis to decide which of the two is the most appropriate for use in the context of PBC. Once we know which algorithm to use we will be able to make more accurate estimates of the sizes of the fields needed to balance the hardness of the ECDLP with the DLP.

The most efficient version of the FFS is given in [50]. The complexity achieved is $L_{p^m}(1/3)$ when $\log(p) < \sqrt{m}\log(m)$, making it the most appropriate for binary and ternary fields (which arise when supersingular elliptic curves are used).

To complement the curves suggested in Table 3.1 in Section 3.2, we focus on the finite fields of the form $\mathbb{F}_{p^k}$ where p is a medium prime. For such cases we know that k will be in the range $[4 \ldots 48]$ implying that $\log(p) < 26$ (if $k = 48$) if the FFS is to perform optimally; clearly this is too small, using a brute force attack one could solve the ECDLP quickly. The NFS in the medium prime case is the appropriate algorithm to use when $p \geq L_{p^k}(1/3)$, which is exactly the setting in PBC using the curves of Table 3.1.

5.3 NFS

The NFS algorithm developed by Joux et al. in [51] is an improvement of the originally proposed algorithm by Schirokauer [81].

Examining the parameters of the curves given in Chapter 3 it is easily verified that the NFS is indeed the most appropriate algorithm to use to solve the DLP in the context of PBC.

The NFS follows the basic index calculus algorithm as outlined above. The isomorphic representations of $\mathbb{F}_{p^k}$ used are given by the quotient by (p) of the rings of integers of two number fields. The number fields are constructed using two monic polynomials $f_1(x)$ and $f_2(x)$ with a common root in $\mathbb{F}_{p^k}$. In the context of PBC, this is done by taking $f_1(x)$ to be of degree k and irreducible modulo p with small coefficients and then taking $f_2(x) = f_1(x) + p$. The number fields are $\mathcal{K}_1 \cong \mathbb{Q}[\theta_1]$ and $\mathcal{K}_2 \cong \mathbb{Q}[\theta_2]$ where θ_1 and θ_2 zeros

of $f_1(x)$ and $f_2(x)$ respectively in $\mathbb{C}$. If we denote the ring of integers of $\mathcal{K}_i$ by $\mathcal{O}_i$ then $\mathcal{O}_i/(p) \cong \mathbb{F}_{p^k}$.

The following lemma from [51] (a generalisation of [20, Lemma 10.5.1]) establishes the construction for the factor bases.

Lemma 5.3.1. *Let $\mathcal{K} = \mathbb{Q}[\theta]$ and $(a_0, \ldots, a_l)$ be an $(l+1)$-tuple, $a_i \in \mathbb{Q}$ for $i \in [0, \ldots, l]$, with $\gcd(a_0, \ldots, a_l) = 1$, then a prime ideal $\mathfrak{p}$ dividing the principal ideal generated by $\sum_{i=0}^{l} a_i\theta^i$ either divides $f_\theta = [\mathcal{O}_\mathcal{K} : \mathbb{Z}[\theta]]$, or has degree $\leq l$.*

The factor bases are therefore:

$\mathcal{F}_i = \{$degree $\leq l$ ideals of $\mathcal{O}_i$ with norm $\leq \mathcal{B}_i$, prime ideals with norms $n | f_{\theta_i}\}$,

for $l \in \mathbb{Z}^+$ determined by the relative sizes of p and k, and $f_{\theta_i} = [\mathcal{O}_{\mathcal{K}_i} : \mathbb{Z}[\theta_i]]$, $i = 1, 2$.

The smoothness and sieving bounds $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{S}$ are set, depending on p, k and l. We sieve through the $l+1$-tuples, $(a_0, \ldots, a_l)$ where a_j are integers, $|a_j| \leq \mathcal{S}$, with $\gcd(a_0, \ldots, a_l) = 1$. We want to collect pairs $\sum_{j=0}^{l} a_j\theta_i^j$ for $i = 1, 2$ which have $\mathcal{B}_i$-smooth norms in $\mathcal{K}_i$. Once a sufficient number of smooth elements have been found, the ideal generated by $\sum_{j=0}^{l} a_j\theta_i^j$ is to be factored into a product of ideals from the respective factor bases $\mathcal{F}_i$. The factorisation of the ideal $\sum_{j=0}^{l} a_j\theta_i^j$ consists of principal ideals over the primes in the factorisation of the norm. These ideals are in the factor bases by Lemma 5.3.1.

Let $N_{\mathcal{K}_i/\mathbb{Q}}(\sum_{j=0}^{l} a_j\theta_i^j) = \prod_v p_v^{e_v}$ for p_v primes, $p_v \leq \mathcal{B}_i$, be the factorisation of the norm. For all primes p_v in the factorisation of the norm which do not divide f_{θ_i}, the irreducible factors of $\gcd(A(x), f_i(x))$ over $\mathbb{F}_{p_v}$ correspond to ideals $\mathfrak{p}_{\mathfrak{v}_s}$ lying above p_v which occur in the decomposition of the ideal. If $\gcd(A(x), f_i(x))$ is irreducible then $\mathfrak{p}_v$ is the only prime ideal lying above p_v and the $\mathfrak{p}_\mathfrak{v}$-adic valuation of $(\sum_{j=0}^{l} a_j\theta_i^j)$ is given by $m/\deg \mathfrak{p}_v$; otherwise, the valuation of the ideal corresponding to each irreducible factor of $\gcd(A(x), f_i(x))$ is calculated using [20, Algorithm 4.8.17], as for the primes p_v which divide f_{θ_i}.

The relations of products of powers of factor base elements are converted to linear equations of logarithms of factor base elements by taking the logarithms with respect to a

fixed (isomorphic) element on both sides. Once this conversion has been performed, the system of resulting linear equations is solved for the logarithms of the ideals in the factor bases in the linear algebra step. The conversion is by no means trivial.

5.3.1 Relation Conversion

For clarity and ease of notation we now drop the index i, referring to the number fields as $\mathcal{K}$ and sieve elements as $\sum_{j=0}^{l} a_j\theta^j$, keeping in mind that the process is performed in both $\mathcal{K}_1$ and $\mathcal{K}_2$ simultaneously. Similarly, we take $\mathcal{O}$ to be the ring of integers of $\mathcal{K}$, $f(x)$ to be the polynomial defining $\mathcal{K}$ and $\mathcal{B}$ the smoothness bound in $\mathcal{K}$. Let (s_1, s_2) be the signature of $\mathcal{K}$ and define $s = s_1 + s_2 - 1$, then the unit group of $\mathcal{K}$, is $\mathcal{U} \cong \mu(\mathcal{K}) \times \mathbb{Z}^s$, where $\mu(\mathcal{K}) = \langle u_0 \rangle$ is a finite cyclic group of order. We denote by h the class number of $\mathcal{K}$.

As explained in Section 5.2, we use the NFS to compute discrete logarithms modulo a large prime factor r of $p^k - 1$. We assume that r does not divide h and that r does not ramify in $\mathcal{K}$; this is only a minor restriction [51]. Once the ideal decomposition of the smooth elements has been obtained:

$$(\sum_{j=0}^{l} a_j\theta^j) = \prod_v \mathfrak{p}_v^{e_v},$$

each equation is raised to the power h to obtain

$$(\sum_{j=0}^{l} a_j\theta^j)^h = u \prod_v \delta_v^{e_v}, \tag{5.1}$$

for elements $\delta_v^{e_v} \in \mathcal{O}$ such that $(\delta_v) = \mathfrak{p}_v^h$ and for some $u \in \mathcal{U}$ (note that the left hand side is the hth power of the element $\sum_{j=0}^{l} a_j\theta^j$ and not the ideal).

Each equation is likely to have a different u so taking the logarithms at this stage would introduce too many new unknowns and thus drastically increase the number of equations needed and the run time of the linear algebra. This is not an issue when $\mathcal{K}$ has a computable unit group. Instead of logarithms, Joux et al. used *Schirokauer maps* as in [81].

5.3.2 Schirokauer maps

We highlight again that the logarithms are being computed modulo r, so instead of working with the entire unit group of $\mathcal{K}$, denoted $\mathcal{U}$, it is sufficient to work with the unit group modulo the rth powers of units since if $u \in (\mathcal{U})^r$ then $\mathrm{Log}_\alpha(u) \equiv 0 \mod r$; that is, we work with $\mathcal{U}/(\mathcal{U})^r$ instead of $\mathcal{U}$.

Denote by Γ_1 the multiplicative set $\{\gamma \in \mathcal{U} : r \nmid N_{\mathcal{K}/\mathbb{Q}}(\gamma)\}$. For each prime ideal $\mathfrak{p}_i$ lying over r in $\mathcal{O}$ if we take $\epsilon_{\mathfrak{p}_i}$ to be $|\mathcal{O}/\mathfrak{p}_i|^*$ then the lowest common multiple of the $\epsilon_{\mathfrak{p}_i}$, denoted ε, will give:

$$\forall \gamma \in \Gamma_1, \gamma^\varepsilon \equiv 1 \mod r.$$

It is easy to compute ε as each prime ideal, $\mathfrak{p}_i$, lying over r in $\mathcal{O}$ corresponds to an irreducible factor $f_i(x)$ of $f(x) \mod r$. Then, $|\mathcal{O}/\mathfrak{p}_i|$ is given by $r^{\deg(f_i(x))}$ and $\varepsilon = r^D - 1$, where D is the lowest common multiple of the degrees of the irreducible factors of $f(x) \mod r$.

Using the relation $\gamma^\varepsilon \equiv 1 \mod r$, Schirokauer defines the map λ_1:

$$\begin{aligned} \lambda_1 : \Gamma_1 &\rightarrow r\mathcal{O}/r^2\mathcal{O}, \\ \gamma &\rightarrow (\gamma^\varepsilon - 1) + r^2\mathcal{O}. \end{aligned}$$

Given that $r\mathcal{O}/r^2\mathcal{O}$ is a $\mathbb{Z}/r^2\mathbb{Z}$-module of rank k [81], there exists a basis for $r\mathcal{O}/r^2\mathcal{O}$ given by $\{rb_\ell + r^2\mathcal{O} : \ell = 1, \ldots, k\}$ such that $\{rb_\ell + r^2\mathcal{O} : \ell = 1, \ldots, s\}$ is a basis for $\lambda_1(\mathcal{U})$.

In a similar way, let Γ_i denote the set $\{\gamma \in \Gamma_{i-1} : \lambda_{i-1}(\gamma) = 0\}$ where the maps λ_i are given by:

$$\begin{aligned} \lambda_i : \Gamma_i &\rightarrow r^{2^{i-1}}\mathcal{O}/r^{2^i}\mathcal{O}, \\ \gamma &\rightarrow (\gamma^\varepsilon - 1) + r^{2^i}\mathcal{O}. \end{aligned}$$

As above, $r^{2^{i-1}}\mathcal{O}/r^{2^i}\mathcal{O}$ is a $\mathbb{Z}/r^2\mathbb{Z}$-module of rank k so there is also a basis $\{r^{2^{i-1}}b_\ell +$

$r^{2^i}\mathcal{O} : \ell = 1, \ldots, k\}$, then the maps λ_i can be given by the maps $\lambda_{i,w}$ such that the following congruence is satisfied:

$$(\gamma^{\varepsilon} - 1) \equiv r^{2^{i-1}} \sum_{w=1}^{k} \lambda_{i,w}(\gamma) b_w \mod r^{2^i}.$$

The maps λ_i and $\lambda_{i,w}$ are homomorphisms on $\mathcal{U}$ satisfying $\lambda_i(\gamma_1\gamma_2) = \lambda_i(\gamma_1) + \lambda_i(\gamma_2)$ and $\lambda_{i,w}(\gamma_1\gamma_2) = \lambda_{i,w}(\gamma_1) + \lambda_{i,w}(\gamma_2)$ for $i = 1, \ldots, k$ and are therefore logarithmic maps on Γ_i.

From the definition of the group Γ_1 and the fact that units have norm 1 and the norms of the elements $\sum_{j=0}^{l} a_j\theta^j$ are smooth, we know that all units and elements $\sum_{j=0}^{l} a_j\theta^j$ will be in Γ_1 (as r is a large prime – larger than the smoothness bound). Joux et al. used λ_1 in their algorithm, constructing the homomorphism

$$\begin{aligned} \bar{\lambda} : \mathcal{U}/(\mathcal{U})^r &\rightarrow (\mathbb{Z}/r\mathbb{Z})^s \\ u &\mapsto (\lambda_{1,1}(u), \ldots, \lambda_{1,s}(u)), \end{aligned}$$

for the $\lambda_{1,w}$ and $\{rb_w + r^2\mathcal{O} : w = 1, \ldots, s\}$ is a basis for $\lambda(\mathcal{U})$ as defined above.

Assuming that $\bar{\lambda}$ is injective ($\bar{\lambda}(a) = \bar{\lambda}(b) \Rightarrow a = b$) and following the results of Schirokauer [81], an element in $\mathcal{U}$ will be an rth power if and only if $\lambda(u)$ is the zero vector [51]. It is reasonable to assume that $\mathcal{U}$ does not contain a primitive rth root of unity so it can be concluded that $\mathcal{U}/(\mathcal{U})^r$ has r^s elements and therefore $\bar{\lambda}$ is an isomorphism. It is therefore possible to find a (non-unique) set of units $u_1, \ldots, u_s$ such that $\lambda_{1,w}(u_w) = 1$ and $\lambda_{1,w}(u_z) = 0$ for $z \neq w$ and write all units in the form

$$u = \bar{u}^r \prod_{i=1}^{s} u_i^{\lambda_i(u)}, \tag{5.2}$$

for some unit $\bar{u}$.

Returning now to the expression (5.1), multiplying each δ_v by the unit

$$u_{\delta_v} = \prod_{i=1}^{s} u_i^{-\lambda_i(\delta_v)},$$

we obtain elements $\delta_v^{'}$ such that $\lambda_i(\delta_v^{'}) = 0$ for $i \in [1 \ldots s]$ such that $\delta_v^{'}$ is a generator for $\mathfrak{p}_v^h$. Rewriting equation (5.1), denote $A = \sum_{j=0}^{l} a_j \theta^j$, using the form given in equation (5.2) we have:

$$(A)^h = \bar{u}_A^r \prod_{i=1}^{s} u_i^{h\lambda_i(A)} \prod_v \delta_v^{e_v}, \tag{5.3}$$

for some unit $\bar{u}_{a,b}$. Taking logarithms modulo r of both sides and dividing by h, we obtain

$$\mathrm{Log}_\alpha(A) \equiv \sum_{i=0}^{s} \lambda_i(A) \,\mathrm{Log}_\alpha(u_i) + \sum_v e_v h^{-1} \,\mathrm{Log}_\alpha(\delta_v^{'}) \mod r.$$

The unknowns, the logarithms modulo r, can be solved for, using linear algebra.

5.3.3 Exploiting Automorphisms

In [51, Section 4.3] Joux et al. discuss an improvement to the algorithm as described above. They propose using automorphisms to reduce the size of the factor bases in order to shorten the linear algebra stage – the bottle-neck of the algorithm. They show that if the automorphism group of a number field $\mathcal{K}$, $\mathrm{Aut}(\mathcal{K})$, is non-trivial, then the size of the factor base can be reduced by a factor of $\#\,\mathrm{Aut}(\mathcal{K})$, which is a divisor of k.

5.3.4 Large Prime Variations

One method to find more smooth relations is to allow relations with a limited number of primes larger than the smoothness bound $\mathcal{B}$ but lower than a secondary bound $\mathcal{B}'$. We call such relations *partial* and this improvement is based on the birthday paradox (explained in Section 6.2). If the number of primes between $\mathcal{B}$ and $\mathcal{B}'$ is approximately v and we allow norm factorisations with one large prime, then after collecting around $\sqrt{\pi\mathcal{B}'/2}$ such equations we expect to find partial relations with a common large prime. This large prime

is then removed from one relation using the other, giving us a new smooth relation from two relations which would otherwise have been discarded. In [101] and [97] this method is tested with positive results. The double large prime variation (allowing up to two large primes) is used in [97]. Using more than two large primes has been successful when using the NFS for factorisation. For example, in [25], a variation of the NFS for factoring using four large primes was successful; but it is still unknown if it would be useful to adapt this to the discrete logarithm NFS algorithm setting as it is slightly more complicated and may introduce unnecessary computational complexity [97]. In [101] the experimental results support the suggestion of [64] that $\mathcal{B}^{1.2} < \mathcal{B}' < \mathcal{B}^{1.4}$ using the single prime variant (the 'unsmooth' part bounded by $\mathcal{B}'^2$ when using the double prime variant).

5.3.5 The Multiple Polynomial NFS

We see that the polynomial selection has a significant effect so it may be beneficial to use multiple number fields instead of just two. This approach is called the *Multiple Polynomial NFS*. It is easy to generalise the NFS above to a multiple polynomial number field sieve with n polynomials. As above, we select a degree k, irreducible polynomial $f_0(x)$ in $\mathbb{F}_p[x]$. Let α be a root of $f_0(x)$ in $\mathbb{C}$ and $\mathcal{K}_0 = \mathbb{Q}(\alpha)$. For the other number fields, we let $g_i(x) = f_0(x)+p\cdot h_i(x)$, for $i \in [1 \ldots n]$, where $h_i(x)$ has degree less than k and $g_i(x)$ is irreducible in $\mathbb{F}_p[x]$. We denote by β_i a zero of $g_i(x)$ in $\mathbb{C}$ and $\mathcal{K}_i = \mathbb{Q}(\beta_i)$.

The sieving is performed as above, only now we collect elements in the sieving region which are smooth in $\mathcal{K}_0$ and (at least) one of the $\mathcal{K}_i$. After sieving, for all i such that the number of smooth points of $\mathcal{K}_i$ is less than $\mathcal{B}_i$ we discard $\mathcal{K}_i$ and the set of relations. We then set up a system of equations from $\mathcal{K}_0$ and a remaining $\mathcal{K}_i$ and solve for the discrete logarithms of the factor bases as usual.

5.3.6 Computational issues with the NFS

For each step of the NFS there are parameter choices to be made and implementation options which affect the performance; the best parameter selection for one step of the NFS

can result in the suboptimal performance of another step and in turn of the NFS as a whole procedure. The most straightforward example of this is the choice of the smoothness bound $\mathcal{B}$; taking a higher bound decreases the sieving time, as it is easier to find smooth values, but increases the time and space required in the linear algebra stage. The straightforward approach for an implementation would be to simply choose $\mathcal{B}$ so that the running time and space requirement of each step is equal; this, however is not as straightforward as it seems: the sieving step is parallelisable, whereas the linear algebra stage is not as flexible [39]. The optimal choice for $\mathcal{B}$ not only depends on the particular field $\mathbb{F}_{p^k}$ and the relative sizes of p and k but also on the particular implementation and the platform on which it is being run.

The choice of $f_1(x)$ also has an unpredictable effect on the runtime of the NFS. The choice of $f_1(x)$ not only determines $\mathcal{K}_1$, but also $\mathcal{K}_2$. It therefore makes sense to not only assess the structure of $\mathcal{K}_1$, but also to assess the structure of $\mathcal{K}_2$, when making this choice. In [51] it is suggested that $f_1(x)$ should be chosen so that $\mathcal{K}_1$ has a large cyclic automorphism group (with order k), thus decreasing the number of relations required by a factor of k (and decreasing the size of the matrix accordingly). In [101] the author investigates the probability of finding relations (doubly smooth elements) for different choices of $f_1(x)$. It was shown that different choices of polynomial have a significant effect on the runtime of the sieving stage: changing the polynomial which defines the number fields can increase the probability of finding relations at a rate comparable with a four-fold increase in the size of the factor base [101]. As the degree of elements sieved increases, finding enough relations becomes a computational issue with the NFS so the selection of $f_1(x)$ is clearly important, but the best choice for $f_1(x)$ can only be made on a case by case basis.

The smoothness bounds and polynomial selections are not the full extent of the variability of the NFS. Other choices affecting the runtime of the NFS include the choice of the sieving region, the dimensions of the sieving space and whether to use large primes and partially smooth relations, or not. Though the NFS has an asymptotic complexity of $L_{p^k}(1/3, (64/9)^{1/3} + o(1))$ as p and k both tend to infinity, given the many implementation variables it is clear that we are not able to say much about the expected run time of the NFS

in practice, particularly for a fixed k.

In the PBC setting, as outlined in Chapter 3, the fixed extension degrees of the finite fields and the relative sizes of the prime fields used for the particular embedding degrees mean that the NFS is the most appropriate algorithm to use for solving the finite field DLP instance. Given the approximate field size, we are still not able to run experiments to test the runtime of the NFS as the parameters in such a simulation would be much too large; it is currently computationally infeasible. Taking a smaller instance of the prime for a particular embedding degree would not give us a realistic estimate of the runtime, as the relative sizes of k and p have a significant effect on the runtime. In fact, it is possible that for such an example the FFS would be the most appropriate algorithm to use, not the NFS. This leaves us with the problem of not being able to thoroughly test the NFS in the context of PBC and thus we are unable to give a concrete estimate of the hardness of the DLP in the context of PBC; we will have to rely on the heuristic analysis of [51]. In fact, this can be said for the use of the NFS in general: the NFS would obtain optimal performance (asymptotically) only for fields for which it is currently computationally infeasible to compute examples; this means that for all example cases, the parameters will have to be chosen such that the NFS can not possibly achieve the optimal runtime; this is an advantage for PBC.

Complexity of the NFS in the context of PBC

There are three cases for the complexity of the NFS in the medium prime case. In the case relevant to PBC, the complexity of the NFS depends on the relative sizes of p and k, this will determine the degree of elements sieved. The parameters are such that p can be written as $p = L_{p^k}(2/3, c)$ for some constant c. The complexity of the NFS is given by

$$L_{p^k}(1/3, c')$$

where $c' = \frac{8}{3}\left(\frac{3l}{4(l+1)}\right)^{1/3}$ and l is the degree of the elements sieved, given by the closest integer solution to the real solution of $3tl^3(l+1)^2 - 32 = 0$ [51].

Chapter 6

Solving the ECDLP

'Sentence first – verdict afterwards.'

- The Queen, Lewis Carroll's *Alice in Wonderland*,
Chapter XII, Alices Evidence.

In this chapter, the hardness of solving the DLP in the groups of points on elliptic curves will be examined. In contrast to the finite field case, the methods for solving the ECDLP are well understood and have been rigorously tested.

6.1 Index Calculus for the ECDLP

One of the main arguments made by Miller [71] for the use of the group of points on an elliptic curve in cryptography as an alternative to the multiplicative group of a finite field is the improbability of the existence of a computationally feasible index calculus method for solving the ECDLP. The analogue of the index calculus methods to solve the DLP in a group of points on $E(\mathbb{F}_p)$, $G = \langle P \rangle$, is to lift E to a curve defined over $\mathbb{Q}$, $\mathcal{E}$, and then lift the points in G to points on $\mathcal{E}(\mathbb{Q})$. Once this has been done, the DLP can be solved

efficiently [71]. There are a couple of issues with this method, outlined in [71]. The first being that $E(\mathbb{F}_p)$ needs to be lifted to a curve $\mathcal{E}(\mathbb{Q})$ with large rank and points with a "small" enough representation (bounded by a value polynomial in $\log(p)$), which is not achievable in practice [71, 92]. In the lucky case that an appropriate $\mathcal{E}(\mathbb{Q})$ is found another problem arises: there are many ways to lift points to $\mathcal{E}(\mathbb{Q})$, finding the correct points for the index calculus algorithm to work is arguably a more complicated problem than the original ECDLP.

These issues are investigated in depth in [92] with the authors giving more evidence to support the claim of Miller in [71] that *"... it is extremely unlikely that an 'index calculus' type attack on the elliptic curve method will ever be able to work."*

In [92] the authors show that computing a factor base for an index calculus method for the ECDLP is exponentially more difficult than for the finite field case. The main explanation given by the authors in [71, 92] that the index calculus method is not a practical method in G, despite the success in the finite field, is the 'size' of the elements of the factor base. Using a finite field index calculus method, the elements of a factor base with v elements can be represented using $\leq \log(v)$ bits, whereas in the group G the factor base elements are significantly 'larger', around $v \log(v)$ bits in size [92].

In an effort to get around these problems, Silverman developed the Xedni calculus method [91]. The Xedni calculus method begins by lifting v points $P_1, \ldots, P_v$ from $E(\mathbb{F}_p)$ to points $Q_1, \ldots, Q_v$ with integer coefficients. Next an elliptic curve $\mathcal{E}/\mathbb{Q}$ which passes through all v points must be found such that $\mathcal{E}$ has minimal rank. The Xedni calculus method is thus the reverse process of the index calculus method, hence the name. If the points Q_i happen to be dependent in the group $\mathcal{E}(\mathbb{Q})$ then the relationships between the points can be used to solve the ECDLP. The problem of finding points of small height in the index calculus method has been replaced by the very low probability of the points Q_i being dependent and the experimental results [91] show that the Xedni calculus is also impractical and the analysis of [48] shows that the algorithm will most certainly fail.

As there is currently no algorithm which can take into account the structure of the

group G, the best known algorithms for computing discrete logarithms in groups of points on an elliptic curve are the generic methods, algorithms for computing discrete logarithms in any general group. Such algorithms have expected run time $O(\sqrt{|G|})$. There are a few well known square root methods: Baby-step Giant-step (also known as Shank's method), Pollard's Kangaroo method (or Pollard's Lambda method) and Pollard's Rho method [4, 95].

6.2 Pollard's Rho Method

We present here Pollard's Rho method as it requires less storage space than other algorithms and also has a speed-up, relevant in the security discussion of the subsequent chapter.

Pollard's Rho method is based on a random walk crossing its own path; finding a 'collision' at different stages in the walk results in the discrete logarithm being determined very easily. This idea is based on the *Birthday Paradox*: if we select v random values (with repetition) from a set of r elements we expect to have two values the same if v is around $\sqrt{\pi r/2}$. The effectiveness of this method relies on us wanting to find a match for any element; not on us wanting to find a match to one particular element.

Let G be a group of points of order r on an elliptic curve, and suppose we wish to compute a such that $[a]P = Q$ for two points $P, Q \in G$. In practice we do not use a truly *random* walk; to make use of a collision in the path we need to retain information about each step. The best way to do this is to step through the path in a deterministic manner, which resembles a random path in G. We use a random mapping $f : G \to G$ and define the random walk to be $\{s_i\}_{i=0}^{\infty}$ where $s_{i+1} = f(s_i)$.

The name, Pollard's Rho method, reflects the shape of the path resulting from this random walk. The walk begins in the 'tail' and, by the birthday paradox, will meet itself at some point leaving a path which resembles the Greek letter rho: ρ.

An example of such a random mapping f is given by the following [78, 4]: The group G is randomly partitioned into 3 groups of approximately the same size: G_0, G_1 and G_2.

We then define a random walk $\{P_i = [a_i]P + [b_i]Q\}_{i=0}^{\infty}$ by:

$$P_{i+1} = \begin{cases} P_i + P, & P_i \in G_0, \\ [2]\, P_i, & P_i \in G_1, \\ P_i + Q, & P_i \in G_2, \end{cases} \qquad (a_i, b_i) = \begin{cases} (a_i + 1, b_i) & \omega_i \in G_0, \\ (2a_i, 2b_i) & \omega_i \in G_1, \\ (a_i, b_i + 1) & \omega_i \in G_2, \end{cases}$$

starting with $a_0 = 1$ and $b_0 = 0$. When a collision occurs we have $[a_i]P + [b_i]Q = [a_j]P + [b_j]Q$ for some i and j so if $b_j - b_i \neq 0$ we are able to recover $\mathrm{Log}_P(Q) = \frac{a_i - a_j}{b_j - b_i}$.

This example is a mixed additive and multiplicative random walk (it has two additive steps and a multiplicative step) usually taken to be the most efficient type of random walk [29]. Pollard's Rho method is flexible; it can be parallelised [4, 29], the random walks can use more than 3 partitions [4], and different methods of determining a collision are available, *cycle finding algorithms* [4], which trade off space requirements against running time. One particular optimisation of Pollard's Rho method is given below for curves $E_a : y^2 = x^3 + ax$ and $E_b : y^2 = x^3 + b$ defined over $\mathbb{F}_p$ and $p \equiv 1 \mod 4$ or $p \equiv 1 \mod 3$ respectively [29].

6.2.1 Speed-up for curves with non-trivial automorphism group

If an elliptic curve $E/\mathbb{F}_p$ has a point of order r, a large prime, then we know that $r^2 \nmid \#E(\mathbb{F}_p)$ so every point in $G = E(\mathbb{F}_p)$ or order r must be in $\langle P \rangle$, the group generated by P. For any automorphism $\sigma \in \mathrm{Aut}(E)$ the point $\sigma(P)$ also has order r and so must be contained in $\langle P \rangle$. We say that $\langle P \rangle$ is *stable* under $\mathrm{Aut}(E)$ and consider the automorphisms σ as a restriction of σ to $\langle P \rangle$. If σ is a non-trivial automorphism, then using Pollard's Rho method and an additive walk we are able to reduce the expected running time of the algorithm by a factor of $\sqrt{\mathrm{ord}(\sigma)}$, where $\mathrm{ord}(\sigma)$ is the smallest, positive integer i such that σ^i is the identity map [29].

The elliptic curves of the particular form given above have non-trivial automorphisms of order 4 and 6 respectively, so using an additive walk the expected run time of Pollard's

Rho method is $\sqrt{\pi|G|/8}$ and $\sqrt{\pi|G|/12}$ respectively [29]. Elliptic curves defined over $\mathbb{F}_p$ and considered over fields of the form $\mathbb{F}_{p^n}$ for $n > 1$ have a non-trivial Frobenius endomorphism in which case a speed-up of $\sqrt{n}$ can be achieved.

Chapter 7

Part I Summary and Security Levels of Suggested Curves

She generally gave herself very good advice, (though she very seldom followed it).

- Lewis Carroll's *Alice in Wonderland*,

Chapter I, Down the Rabbit-Hole.

A chain is only as strong as its weakest link. In the same way, a cryptographic protocol is only as secure as its most vulnerable point. Choosing the parameters correctly is no guarantee that a particular implementation of a protocol is secure, but it is the starting point. When assessing the security offered by a particular set of system parameters, we need to first be clear on what we consider a 'break' of the system; here we take it to be a solution to the ECDLP or DLP being computed. Following still the reasoning of [63], we consider the computational equivalence of solving the ECDLP and DLP instances to give us the estimated security level for particular embedding degrees and families of curves. We say a pairing-based protocol is *efficient* if both the ECDLP and DLP require an approximately equal amount of computational power. We do not take into account the cost of the resources

Table 7.1: Security Level of Suggested Curves

Security Level (in bits)	k	ρ	D	Twist d	Construction
$\sim 107 - 112$	4	2	1	4	FST [32]
$\sim 145 - 150$	6	2	3	6	FST/BLS [32, 10]
$\sim 144 - 149$	8	1.5	1	4	KSS [52]
$\sim 145 - 150$	12	1	3	6	BN [11]
$\sim 208 - 214$	16	1.25	1	4	KSS [52]
$\sim 237 - 242$	18	1.333	3	6	KSS [52]
$\sim 277 - 281$	24	1.25	3	6	BLS [10]
$\sim 313 - 318$	32	1.125	1	4	KSS [52]
$\sim 347 - 353$	36	1.167	3	6	KSS [52]
$\sim 411 - 416$	48	1.125	3	6	BLS [10]

necessary, or the cost associated with the running of the algorithms in the assessment of the security.

We focus now on the families of curves suggested in Table 3.1. All of these curves are vulnerable to the speed-up to the ECDLP algorithm as described in Section 6.2.1 as they all have CM discriminant $D = 1$ or 3. We take into account the ρ-value of each curve and the degree of the sieving elements for the NFS algorithm; this is where our method varies slightly from the method in [63]. As the analysis of [63] precedes the algorithm of [51] it is appropriate that the security levels be reassessed, taking the NFS into account. The goal of [63] was to generate guidelines for public-key sizes used in different types of public-key protocols. By focusing specifically on the PBC setting, we are able to take more variables into account, such as the degree of elements sieved and the relative sizes of p and k, which have a significant effect on the heuristic complexity of the algorithm, as described in Section 5.3 (and an even stronger effect on the actual run-time, predicted by [101]).

We use the convention that a cryptographic protocol offers the security level s if the complexity of solving the underlying hard problem using the best known algorithm is equal to s. For an efficient implementation of a pairing-based protocol, we aim to have $s_{EC} \approx s_F$, where $2^{s_{EC}}$ is the complexity of solving the ECDLP and 2^{s_F} the complexity of solving the DLP.

The sizes of the parameters needed for a particular security level are given easily by $\log(r) = 2 \cdot (\text{security level})$ and $\log(p) = 2\rho \cdot (\text{security level})$. These values correspond with the families of pairing-friendly elliptic curves in column 6, with embedding degree k with respect to a group of points of size r as computed above, the ρ-value and the CM discriminant of the curve, D, in columns 2, 3 and 4 respectively.

Interpreting Table 7.1

The values in column 1 of Table 7.1 are the approximate intervals for which the complexity of Pollard's Rho algorithm is within a small constant factor of the complexity of the NFS (factor between 1 and 3). We emphasize again that Pollard's Rho algorithm has been tested in numerous situations and the run-time in practice is very well understood. There have been no algorithms developed to solve the ECDLP which exploit the group structure of the points on an elliptic curve, and for many reasons, outlined in Chapter 6 given in [71, 92], it is not likely that such an algorithm will be developed.

The actual run-time of the NFS, on the other hand, can not be so rigorously tested in practice. There have also been suggestions that the DLP in the context of PBC is actually more vulnerable than the general DLP in extension fields with medium prime characteristic. In [82], Schirokauer discusses potential improvements to the NFS which exploit the low hamming weight of the primes and the special form of primes used in PBC (the analogue of the special number field sieve (SNFS) for factoring integers of a special form). There are also other properties which are not yet exploited by the NFS which could lead to run-time improvements. The composite extension degree (k), for example, gives the finite field a much richer structure which could be used by the algorithm. The extension degree is fixed for a particular family of pairing-friendly elliptic curves, this family could be vulnerable to a specific variation of the NFS, adapted to the particular k (the complexity of the general algorithm assumes that k is tending towards infinity). Another point to take into account is that we can not use the L-notation as an exact count of computational steps, especially as this is the limiting complexity as p and k tend to infinity, but we can use this complexity

to give us a range of security levels [65]. Taking the above into account, the security levels given in column 1 are slightly overestimated and there is some flexibility. Also, the general convention in cryptography is to err on the side of caution. To illustrate, it appears that from column 1 of Table 7.1 that there is no appropriate curve admitting a higher order twist for the implementation of a pairing-based protocol with a 128-bit level of security. We emphasize that these values should be interpreted more as an upper bound for efficiency, rather than a lower bound. It is generally accepted that the BN curves would be appropriate to use for the 128-bit level of security, and we support this assumption. We also support the opinion that the KSS $k = 18$ curves are appropriate for the 192-bit security level. From the analysis of [65], the 128-bit and 192-bit security levels will become increasingly important, so to illustrate the algorithms presented in the subsequent chapters, the BN and KSS $k = 18$ curves will be used as the primary examples.

Part II

Efficient Algorithms

Efficient Algorithms

'Now, here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!'

- The Queen, Lewis Carroll's *Through the Looking Glass and What Alice Found There*, Chapter II, The Garden of live Flowers.

In this section, some aspects of the implementation of pairings for cryptography will be discussed. There is an important issue facing implementers of PBC, which does not have to be considered for other public-key cryptographic schemes, such as RSA, or other schemes based on the DLP. The issue is as follows: It is possible to write an implementation of RSA or a discrete logarithm based protocol, which performs reasonably efficiently for any level of security. For example, an RSA implementation with a 1024-bit modulus can easily be modified to use a 4096-bit modulus, maybe by just changing a single parameter within the program. The same applies to elliptic curve cryptography, where a generic implementation will perform reasonably well for a curve with a subgroup of points of size 160-bits, 192-bits or 256-bits. Of course, an implementation specially tailored for, and hard-wired to, a particular level of security will perform somewhat better, but not spectacularly so.

The situation for PBC is fundamentally different. An efficient implementation at the 80-bit level of security, using the Tate pairing on a Cocks-Pinch pairing-friendly curve, will

be completely different from an implementation at the 128-bit level, using the R-ate pairing on a BN curve, and very little code will be reusable between the two implementations. In this situation the development and maintenance of good quality pairing code becomes difficult and there is a compelling case for the development of some kind of automatic tool – a *cryptographic compiler* – which can generate good quality code for each case [26]. Much of the work presented in this part was motivated by the intention to contribute to such a compiler.

For efficient implementations we see that using an optimal pairing and a curve admitting a high degree twist is favourable. We have placed some restrictions on the embedding degrees used, for practical reasons, and seen that choosing a security level restricts the choice of embedding degree. One implementation issue remaining is how to represent the finite field $\mathbb{F}_{p^k}$ given a particular set of parameters; this choice is important as it will affect the general efficiency of the protocol and also the ability to use several computational improvements available for computing pairings and for use in pairing-based protocols. In Chapter 8 we present results of work undertaken with Michael Scott, published in [13], addressing the issue of how to best implement these fields for use in PBC.

In Chapter 8.5, work of Shirase [89] which determines the equation of some BN curves is extended to KSS $k = 18$ curves. For some KSS $k = 18$ curves the elliptic curve equation is given if x_0 satisfies an easily checked equivalence relation. In these cases the necessity of performing the point scalar multiplication test is removed.

The 'final exponentiation' step of the Tate pairing, and variations thereof, including optimal pairings, is expensive but necessary. In Chapter 9 we present a fast algorithm for performing the final exponentiation. This method comes from a collaboration with Michael Scott, Manuel Charlemagne, Luis Dominguez Perez and Ezekiel Kachisa and resulted in the publication [86]. It uses the polynomial description of the system parameters and addition chains to reduce the cost of performing the final exponentiation.

This part concludes with a faster algorithm to perform an expensive cofactor multiplication of a point on a twisted curve in Chapter 10. This algorithm extends ideas of [36] and

also exploits the polynomial parameterisation of the primes p and r, and the trace t, and uses some curve automorphisms to reduce the cost of the cofactor multiplication. The development of this algorithm is a result of collaboration with Michael Scott, Manuel Charlemagne, Luis Dominguez Perez and Ezekiel Kachisa, published in [85].

Notation

Following on from Chapter 3, some notation will be constant throughout this section is: We take E to be an elliptic curve defined over a finite field $\mathbb{F}_q$, where $q = p^n$ for p a prime and $n \in \mathbb{Z}^+$. The number of points on $E(\mathbb{F}_q)$ is divisible by a large prime number r and the embedding degree k of E with respect to r, is the smallest positive integer such that $r \mid (q^k - 1)$. The number of points $E(\mathbb{F}_q)$ is $q + 1 - t$, where t denotes the trace of the Frobenius endomorphism and t satisfies $\mid t \mid \leq 2\sqrt{q}$.

Chapter 8

Representation of Finite Fields in PBC

'Curtsey while you're thinking what to say, it saves time.'

- The Queen, Lewis Carroll's *Through the Looking Glass and What Alice Found There*, Chapter II, The Garden of live Flowers.

Let us assume now that we have selected a pairing-friendly elliptic curve for our implementation. We now have to implement, as efficiently as possible, all structures associated with the particular curve. Not only the finite field $\mathbb{F}_p$ is used, but it will also be necessary to perform arithmetic in $\mathbb{F}_{p^k}$ and some intermediate fields $\mathbb{F}_p \subset \mathbb{F}_{p^e} \subset \mathbb{F}_{p^k}$.

8.1 Extension Fields Represented Using Towers

First consider the implementation of a general extension field $\mathbb{F}_{p^k}$. The natural representation of elements of this field is as polynomials of degree $k-1$, $\mathbb{F}_{p^k} \cong \mathbb{F}_p[x]/f(x)\mathbb{F}_p[x]$

where $f(x)$ is an irreducible polynomial in $\mathbb{F}_p[x]$ of degree k. For efficiency reasons some effort might be made to choose $f(x)$ to have a minimal number of terms and small coefficients. For example, for the field $\mathbb{F}_{p^2}$, where p is a prime and $p \equiv 3 \mod 4$, a good choice for $f(x)$ would be $x^2 + 1$, and elements can be represented as $ax + b$, with $a, b \in \mathbb{F}_p$. For the case $p \equiv 5 \mod 8$, a good choice for $f(x)$ would be $x^2 - 2$. For the final case $p \equiv 1 \mod 8$ there is no immediately obvious way to choose a suitable irreducible binomial, but for some small value i which is a quadratic non-residue in $\mathbb{F}_p$, $x^2 - i$ would be appropriate.

In some settings, the value of the extension degree k can be much greater than 2, as in PBC, in which case the direct polynomial representation becomes more arithmetically complex. For elliptic curve cryptography implemented over "Optimal Extension Fields" (OEFs), as suggested by Bailey and Paar [5], extensions as high as $\mathbb{F}_{p^{30}}$ are considered; in pairing-based cryptosystems, an extension degree of up to 50 is reasonable, as discussed in the previous section. OEFs are usually defined as extensions with respect to a small single-word pseudo-Mersenne prime. The extension fields that arise in the context of efficient implementations of PBC, however, are rather different.

If the extension degree is a parameter of the implementation, then the potentially uncomfortable situation arises where, if the extension degree changes, an optimal implementation must be re-written, largely "from scratch". The alternative seems to be to use generic polynomial code to construct the extension field, making the implementation slow and bulky. A nice compromise that applies when the extension k is smooth (that is, has only small factors) is to use a "tower" of extensions, where one layer builds on top of the last, and ideally where each sub-extension is quite small. For example, $\mathbb{F}_{p^{12}}$ could be implemented as a quadratic extension, of a cubic extension, of a very efficiently implemented (and reusable) quadratic extension field $\mathbb{F}_{p^2}$, as implemented by Devegili et al. [23].

This idea of using a tower of extensions was suggested by Baktir and Sunar [6] as a better way of implementing OEFs, and in the process of doing this they discovered that the resulting simpler implementation resulted in an asymptotically improved method for performing field inversion. It is relatively easy to implement quadratic and cubic extensions

efficiently, whereas the complexity of implementing generic methods over large extensions might result in the inadvertent use of sub-optimal methods.

It is also proposed in the IEEE draft standard "P1363.3: Standard for Identity-Based Cryptographic Techniques using Pairings" that extensions of odd primes are constructed using a tower of extensions created using irreducible binomials at each stage [1], adjoining roots of elements to give the extension.

There are other supporting factors for the use of tower extensions in PBC. Many of the operations necessary in PBC can be computationally expensive so it is important that every possible optimisation is exploited. Optimisations can be made by speeding up and shortening the Miller loop (as described in Section 3.1.1) by carefully selecting curves and twists of curves used in the pairing, or by speeding up operations required in pairing-based protocols. Many of these optimisations have one detail in common: they assume that the finite fields used in PBC can be (and are) represented in the most efficient way. That is, it is presumed that $\mathbb{F}_{p^k}$ can be given as a sequence of tower extensions of $\mathbb{F}_p$, using only one element of $\mathbb{F}_p$ to automatically construct the towers.

For example, in [42], a method for speeding up squaring of elements in the cyclotomic subgroup of fields of the form $\mathbb{F}_{p^k}$ where k is divisible by 6 is presented. This results in faster manipulation of elements used in pairing-based protocols. The results of [42] rely on a tower representation of the extension field in order to 'unravel' operations in the extension field down to a subfield, using also the Frobenius automorphism, to compute operations in a subfield, instead of in the full extension field $\mathbb{F}_{p^k}$.

In [21], a method for speeding up the computation of the Miller loop in the pairing computation is given. This is achieved by replacing the double-and-add method by an 2^n-tuple-and-add. The speed-up is a result of a reduction of the number of operations in the full extension field replaced by an increase in the number of operations in a subfield. This relies on both the extension field $\mathbb{F}_{p^k}$ and the subextension fields, $\mathbb{F}_{p^e} \subset \mathbb{F}_{p^k}$ over $\mathbb{F}_p$ (for $e \mid k$) having an efficient representation, that is, it requires the extension field to be implemented as a tower of extensions. In [75] a method to speed up the computation of the Tate pairing

is given, which relies on a tower extension ending with a cubic extension.

Clearly it is advantageous to use this towering method when implementing a pairing-based protocol. One issue remains: finding the best tower for a particular value of k. Obviously, for different values of k, we will need to use different towers. A very reasonable approach in the context of PBC would be to fix the tower for a particular k.

The construction does not only depend on k however, but also on p, the characteristic of the base field. There is an existing method for constructing such towers given by Koblitz and Menezes in [57], which can only be used for some p with specific properties, so relying on this method alone places unnecessary restrictions on the parameters of a pairing-friendly curve. Given the relative rarity of pairing-friendly elliptic curves it is clear that we should aim to reduce the number of constraints on the parameters that may compromise the efficiency of the implementation.

8.2 Existing Ideas for Constructing General Towers

Let p be an odd prime, and let $n, m > 0$ be integers. The most obvious way to construct the tower of sub-extensions of the field $\mathbb{F}_{p^{nm}}$ over $\mathbb{F}_{p^n}$ would be to use a binomial $x^m - \alpha$ which is irreducible over $\mathbb{F}_{p^n}$ and successively adjoin roots of the previously adjoined root until the tower has been constructed (we refer to this as the 'general method'). We are able to test $x^m - \alpha$ for irreducibility using the following theorem:

Theorem 8.2.1. *[66, Theorem 3.75] Let $m \geq 2$ be an integer and $\alpha \in \mathbb{F}_{p^n}^*$. Then the binomial $x^m - \alpha$ is irreducible in $\mathbb{F}_{p^n}[x]$ if and only if the following two conditions are satisfied:*

1. *Each prime factor of m divides the order, e, of $\alpha \in \mathbb{F}_{p^n}^*$, but not $(p^n - 1)/e$;*
2. *If $m \equiv 0 \mod 4$ then $p^n \equiv 1 \mod 4$.*

The *order* of γ is the smallest positive integer e such that $\gamma^e = 1$ in $\mathbb{F}_{p^n}$ and the order is a divisor of $p^n - 1$.

By Theorem 8.2.1 we see that we can use the the general method to construct towers when $m \not\equiv 0 \mod 4$ or if $p^n \equiv 1 \mod 4$ when $m \equiv 0 \mod 4$.

Given the constraints outlined in Section 3.2, with $m = 2^i 3^j$, it is clear that the tower of extensions used in PBC can be built using a sequence of cubic and quadratic sub-extensions. This was recognised by Koblitz and Menezes in [57]. They called a field $\mathbb{F}_{p^k}$ *pairing-friendly* (not to be confused with a pairing-friendly elliptic curve) if $p \equiv 1 \mod 12$ and k is of the form $k = 2^i 3^j$, in which case by [57, Theorem 2] (which is derived from Theorem 8.2.1 above), the polynomial $x^k - \alpha$ is irreducible if α is neither a square nor a cube in $\mathbb{F}_p$. The extension can be constructed using the general method by simply adjoining a cube or square root of some small α and then successively adjoining a cube or square root of the previously adjoined root until the tower has been constructed. If $j = 0$ then it is sufficient that $p \equiv 1 \mod 4$ and that α be a quadratic non-residue in $\mathbb{F}_p$. This result gives us an easy method for building towers over pairing-friendly fields; simply find an element $\alpha \in \mathbb{F}_p$ which is a quadratic (and, when necessary, a cubic) non-residue and adjoin successive cube and square roots of α to $\mathbb{F}_p$.

There is one major issue remaining, the strict condition that $p \equiv 1 \mod 12$ to give a pairing-friendly field. When searching for pairing-friendly curves of a suitable size, there are typically other criteria that we wish to meet (for example, it is preferred that the Hamming weight of the variable that controls the Miller loop in the pairing calculation should be as small as possible [23]). Having to skip a nice set of parameters because p doesn't satisfy this equivalence seems unnecessarily restrictive. Since the publication of [57], new families of pairing-friendly elliptic curves have been discovered, which the results of [57] could not have taken into account. In particular, the KSS $k = 18$ curves are good for implementation given the many optimisations possible using these curves. The condition that $p \equiv 1 \mod 12$ here is completely unnecessary as this restriction arises from condition 2 of Theorem 8.2.1, which is not applicable when $k = 18$.

Given that the parameters of a pairing-based protocol are already subject to quite strict constraints, it is clear that there is a necessity for a method to construct towers for fields,

which would not be considered pairing-friendly (in the sense of Koblitz and Menezes), but would otherwise be favourable for implementation of a pairing-based protocol. The term 'pairing-friendly field' is slightly misleading, as there are families of pairing-friendly elliptic curves attractive for implementation, which are defined over fields which do not necessarily satisfy $p \equiv 1 \bmod 12$. In a sense, the pairing-friendly fields of [57] are the fields, in the context of pairings, over which it is easy to build the towers. We instead refer to these fields as *towering-friendly* as this gives a more accurate description of these fields – the towers over such fields are easily constructed. This definition is not specific to pairings, but in this setting we would like to use towering-friendly fields for the most efficient implementation possible.

Definition 8.2.2. A *towering-friendly* field is a field of the form $\mathbb{F}_{q^m}$, where q is a prime power, for which all prime divisors of m also divide $q-1$.

In essence, towering-friendly fields are fields for which the tower of sub-extensions can be easily (and most efficiently) constructed; that is, using binomials. The OEFs of Bailey and Paar [5] are, by definition, towering-friendly fields with characteristic a prime of a special form. The fields said to be pairing-friendly by Koblitz and Menezes are indeed towering-friendly fields, but these are not the only ones which occur in the context of PBC.

8.3 General Tower Construction Method

Consider first the general case where $q = p^n$ is an odd prime power, $m > 1$ is an integer and we want to construct the tower of sub-extensions of the towering-friendly finite field $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. The most obvious method (which we shall refer to as the *general method*) would be to use a binomial $x^m - \alpha$ which is irreducible in $\mathbb{F}_q[x]$ and successively adjoin roots of the previously adjoined root until the tower has been constructed. This way the only restriction on α would be that α should not be a sth power in $\mathbb{F}_q$ for any prime divisor s of m. This method works for all m, $m \not\equiv 0 \mod 4$. When $m \equiv 0 \mod 4$, this method will work if $q \equiv 1 \mod 4$ (which is always true for even n).

The two issues to address now are:

- finding a method to build a tower when $m \equiv 0 \mod 4$ and $q \equiv 3 \mod 4$;
- finding a suitable irreducible binomial $x^m - \alpha \in \mathbb{F}_q[x]$ to construct the tower.

The first problem has a relatively simple solution. We first construct a quadratic extension of $\mathbb{F}_{q^2}$, which we will refer to as a *base tower*, using a binomial. We now have $q^2 \equiv 1 \mod 4$, so we can use the general method to build the rest of the tower above $\mathbb{F}_{q^2}$, using a binomial $x^{m/2} - \alpha$, where $\alpha \in \mathbb{F}_{q^2}/\mathbb{F}_q$. In the particular case of $n = 1$ this can be done by simply adjoining a square root of -1. This idea is a generalisation of the approach taken by Barreto and Naehrig in [11] to construct the field $\mathbb{F}_{p^{12}}$ over $\mathbb{F}_p$. They first implement an efficient quadratic extension over the base field, and then look for irreducible polynomials of the form $x^6 - \alpha$, where $\alpha \in \mathbb{F}_{p^2}/\mathbb{F}_p$ is neither a square nor a cube.

Remark 8.3.1. The idea of a base tower can be generalised: Suppose $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ is not a towering-friendly field. Write $m = m_1 m_2$ such that $\gcd(q-1, m_2) = 1$ and all the primes dividing m_1 divide $q - 1$. If all the primes dividing m_2 divide $q^{m_1} - 1$ then the tower of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ can be constructed in two parts using the general method. First $\mathbb{F}_{q^{m_1}}$ over $\mathbb{F}_q$ is constructed using a binomial, this is the base-tower. Then $\mathbb{F}_{q^m} = \mathbb{F}_{q^{m_1 m_2}}$ over $\mathbb{F}_{q^{m_1}}$ is constructed using a binomial defined over $\mathbb{F}_{q^{m_1}}$ (not over any subfield of $\mathbb{F}_{q^{m_1}}$). This method can be implemented recursively to achieve an efficient tower for a non-towering-friendly extension.

As to the problem of finding a suitable α for constructing the tower (and also the base tower when necessary), Theorem 8.2.1 provides a means for determining whether a given binomial is irreducible, but it does not give an efficient method for constructing the towers; taking random small elements, then computing their order in the extension field and verifying that the conditions hold is quite cumbersome, the order could be large and this could require a lot of extension field computation for a single element. Using Theorem 8.2.1, however, we are able to prove a theorem which results in a simpler method for checking the irreducibility of a polynomial $x^m - \alpha$ in certain cases and hence a more practical method

for finding irreducible polynomials to construct the towering-friendly field extensions, particularly in the context of PBC.

Recall the definition and properties of the *Norm* of $\mathbb{F}_q$ over $\mathbb{F}_p$ from Chapter 2.

Theorem 8.3.2. *Let $m > 1$, $n_1 n_2 = n > 0$ be integers, p an odd prime, $q = p^n$ and $\alpha \in \mathbb{F}_q^*$. The binomial $x^m - \alpha$ is irreducible in $\mathbb{F}_q[x]$ if the following two conditions are satisfied:*

1. *Each prime factor s of m divides $p^{n_1} - 1$ and $N_{\mathbb{F}_q/\mathbb{F}_{p^{n_1}}}(\alpha) \in \mathbb{F}_{p^{n_1}}$ is not a sth residue in $\mathbb{F}_{p^{n_1}}$;*

2. *If $m \equiv 0 \mod 4$ then $q \equiv 1 \mod 4$.*

Proof. To prove this theorem, we show that condition 1 of Theorem 8.3.2 implies condition 1 of Theorem 8.2.1. We assume that condition 1 of Theorem 8.3.2 is true. Let e denote the order of α in $\mathbb{F}_q$ and s denote a prime divisor of m.

Suppose that $s \mid (q-1)/e$. This implies that $e \mid (q-1)/s$ and so α is a sth power in $\mathbb{F}_q$. Let $\delta \in \mathbb{F}_q$ be such that $\delta^s = \alpha$. Taking the norm of α we see that $N_{\mathbb{F}_q/\mathbb{F}_p}(\alpha) = N_{\mathbb{F}_q/\mathbb{F}_{p^{n_1}}}(\delta^s) = N_{\mathbb{F}_q/\mathbb{F}_{p^{n_1}}}(\delta)^s$ where $N_{\mathbb{F}_q/\mathbb{F}_{p^{n_1}}}(\delta) \in \mathbb{F}_{p^{n_1}}$ and thus $N_{\mathbb{F}_q/\mathbb{F}_{p^{n_1}}}(\alpha)$ is a sth residue in $\mathbb{F}_{p^{n_1}}$, a contradiction, so $s \nmid (q-1)/e$.

We have also assumed that $s \mid (q-1)$ and since $s \nmid (q-1)/e$ it is clear that $s \mid e$ and so condition 1 of Theorem 8.3.2 is satisfied. □

Using Theorem 8.3.2 we are able to verify the irreducibility of a binomial $x^m - \alpha$ over an extension field $\mathbb{F}_q$, where α is an element of $\mathbb{F}_q$, by checking the properties of a single element of the base field, namely the norm of $\mathbb{F}_q$ over $\mathbb{F}_{p^{n_1}}$ of α – a simpler task than computing the order of an element in $\mathbb{F}_q$. Theorem 8.3.2 can be used in all cases for which the prime divisors of m also divide $p^{n_1} - 1$ to automatically generate towers of extensions over all towering-friendly fields and to build an efficient tower of extensions for the extension field $\mathbb{F}_{q^m}$. As already mentioned, if condition 2 of Theorem 8.2.1 is not satisfied, the towers can still be easily constructed by first constructing a base tower, a quadratic extension, then using the theorem to construct the tower over the base tower.

We now illustrate the usefulness of Theorem 8.3.2 in the context of PBC.

8.4 Towers in Pairing-Based Cryptography

Given the constraints outlined in Section 3.2, it is clear that the tower of extensions for the recommended curves can be built as a sequence of quadratic and cubic sub-extensions. There is some freedom as to the best way to order the extensions. The choice here may be influenced by whether or not it is intended to compress the value of the pairing [84, 41]. This compressed value can then be further efficiently exponentiated in its compressed form, by using Lucas or XTR based methods for a compression factor of 2 or 3 respectively. This is facilitated by terminating with a quadratic or a cubic extension respectively.

Consider for example the BN curves which have embedding degree $k = 12$ and which support the sextic twist $d = 6$. In this case $E(\mathbb{F}_{p^2})$ arithmetic must be supported, and so it makes sense that the tower should start with a quadratic extension over the base field. This can be followed by a cubic extension and then a quadratic, or indeed the other way around. Assuming that the highest possible compression should be supported, the tower of choice in this case is $1 - 2 - 4 - 12$. This particular tower construction is given as an example by the IEEE draft standard [1, §5.3.2]. To utilise a base tower when necessary, starting with a quadratic extension where possible is preferred. Taking all these constraints into account, we extend Table 3.1 in Section 3.2 to include the recommended tower progressions given in Table 8.1.

The recommended towers given in column 6 of Table 8.1 should be interpreted as illustrated here for BN curves: we construct $\mathbb{F}_{p^{12}}$ starting with a quadratic extension, given by adjoining a square root of some non-square in $\mathbb{F}_p$ to $\mathbb{F}_p$, we then construct a quadratic extension of $\mathbb{F}_{p^2}$ to get $\mathbb{F}_{p^4}$ then with a cubic extension of $\mathbb{F}_{p^4}$ we have $\mathbb{F}_{p^{12}}$.

$$\mathbb{F}_p \rightarrow \mathbb{F}_{p^2} \rightarrow \mathbb{F}_{p^4} \rightarrow \mathbb{F}_{p^{12}}$$

Table 8.1: Suggested Towers for Curves with Efficient Arithmetic

k	ρ	D	Twist d	Construction	Recommended Tower
4	2	1	4	FST [32]	1-2-4
6	2	3	6	FST/BLS [10, 32]	1-2-6
8	1.5	1	4	KSS [52]	1-2-4-8
12	1	3	6	BN [11]	1-2-4-12
16	1.25	1	4	KSS [52]	1-2-4-8-16
18	1.333	3	6	KSS [52]	1-3-6-18
24	1.25	3	6	BLS [10]	1-2-4-8-24
32	1.125	1	4	KSS [52]	1-2-4-8-16-32
36	1.167	3	6	KSS [52]	1-2-6-12-36
48	1.125	3	6	BLS [10]	1-2-4-8-16-48

8.4.1 Tower Construction for PBC

From the definition of towering-friendly fields we are only able to distinguish on a specific case-to-case basis if a general extension field is a towering-friendly field.

In the PBC setting we have a little more information. We are able to determine information about some of the parameters for particular curves in advance by making some observations. We see from the following discussion that all the fields $\mathbb{F}_{p^k}$ arising when using the families of pairing-friendly curves in Table 3.1 are towering-friendly.

Elliptic Curves with CM discriminant $D = 1$

Elliptic curves from Table 3.1 with CM discriminant $D = 1$ have equations of the form $E : y^2 = x^3 + Ax$. We know that these curves are not supersingular (which is the case for curves with such equations defined over a prime field with characteristic $p \equiv 3 \mod 4$ [90]), and so $p \equiv 1 \mod 4$. This means that the field is towering-friendly (also pairing-friendly), as all $D = 1$ cases in Table 3.1 have $k = 2^n$, so the Koblitz-Menezes strategy appears to be optimal. Indeed, in the case of $p = 5 \bmod 8$, we can always choose $\alpha = 2$, which leads to fast reduction. An implementation can simply tower up quadratically, by adjoining the square root of the last adjoined element to build the next extension at each step.

Elliptic Curves with CM discriminant $D = 3$

For elliptic curves with CM discriminant $D = 3$, p will not always be a pairing-friendly prime in the sense of the Koblitz and Menezes definition, but we do have some information which will aid us in the construction of the towers over $\mathbb{F}_p$. Given that the CM discriminant is $D = 3$, we know that the elliptic curve must have an equation of the form $E : y^2 = x^3 + B$. If $p \equiv 2 \mod 3$, then such a curve would be supersingular [90] and so $p \equiv 1 \mod 3$. We see then that all the fields resulting from this construction are towering-friendly.

For the KSS $k = 18$ and FST $k = 6$ curves, we are able to use the general method in every case, without a base tower (as $k \not\equiv 0 \mod 4$ and both 2 and 3 divide $p - 1$). We simply adjoin successive cubic and quadratic roots of some cubic and quadratic non-residue $\alpha \in \mathbb{F}_p$, in the recommended order.

For all other families of curves, if the prime $p \not\equiv 1 \mod 4$ then we need to use a base tower to construct the tower. One advantage in this case is that we know $p \equiv 3 \mod 4$, and so the base tower $\mathbb{F}_{p^2}$ over $\mathbb{F}_p$ can be efficiently constructed by adjoining a square root of -1. This may actually be more efficient than an implementation using a pairing-friendly field, as the arithmetic in $\mathbb{F}_p(\sqrt{-1})$ can be performed faster than in $\mathbb{F}_p(\sqrt{\tau})$, for some other quadratic non-residue $\tau \in \mathbb{F}_p$ [35]. The following Corollary (drawing on ideas from Barreto and Naehrig in [11]) gives a method for finding an appropriate value α, such that the polynomial $x^m - \alpha$ is irreducible over a finite field of the form $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{-1})$.

Corollary 8.4.1. *The polynomial $x^m - (a \pm b\sqrt{-1})$ is irreducible over $\mathbb{F}_{p^2}$, for $m = 2^i 3^j$, $i, j > 0$, if $a^2 + b^2$ is neither a square nor a cube in $\mathbb{F}_p$.*

Proof. For any element $a \pm b\sqrt{-1}$, $N_{\mathbb{F}_{p^2}/\mathbb{F}_p}(a \pm b\sqrt{-1}) = (a + b\sqrt{-1})(a - b\sqrt{-1}) = a^2 + b^2$. The integer m is of the form $2^i 3^j$ and so, by Theorem 8.3.2, if $a^2 + b^2$ is neither a quadratic nor a cubic residue modulo p, then $x^m - (a \pm b\sqrt{-1})$ is irreducible over $\mathbb{F}_{p^2}$. $\square$

This Corollary is basically Theorem 8.3.2 in the case $p \equiv 3 \mod 4$, $n = 2$ and $m = k/2$; this is the case of most concern in PBC. Using this corollary, in order to construct the tower, small values of a and b can be tested, until a combination is found such that $a^2 + b^2$

is neither a square nor a cube in $\mathbb{F}_p$. This process only requires a few cubic and quadratic non-residue tests to be performed on elements of the base field. Small values for a and b can be found to help improve efficiency.

As $\frac{1}{2}$ of the non-zero elements of $\mathbb{F}_p$ are non-squares and $\frac{2}{3}$ of the non-zero elements are non-cubes, such an element must exist; in fact, on heuristic grounds, it is expected that $\frac{1}{3}$ of the elements will be neither squares nor cubes, which the experimental evidence supports [11].

Given a little more information about p, which is easily found, we are able to give some more specific constructions.

Construction 8.4.2. For approximately $2/3$ of the primes $p \equiv 3$ modulo 8 the polynomial $x^m - (1 + \sqrt{-1})$ is irreducible in $\mathbb{F}_{p^2}[x]$ for $m = 2^i 3^j$, $i, j > 0$.

Proof. In this case $a^2 + b^2 = 2$. The polynomial will be irreducible if 2 is neither a square nor a cube modulo p. We know that 2 is a quadratic non-residue modulo p when $p \equiv 3 \mod 8$. The only remaining condition is that 2 is not a cube modulo p.

All primes $p \equiv 1 \mod 3$ can be written in the form $p = 3u^2 + v^2$. As Euler conjectured (proved by Gauss, proof given in [62]), 2 is a cubic residue modulo p if, and only if, $3 \mid u$. Instinctively we would presume that this occurs $1/3$ of the time. There is currently no proof concerning the number of primes in a quadratic sequence, but this is supported by experimental results. So 2 is a cubic non-residue modulo for approximately $2/3$ of the values of p. □

When $p \equiv 7 \mod 8$ the following corollary may be useful:

Construction 8.4.3. For approximately $2/3$ of the primes $p \equiv 2$ or 3 modulo 5 the polynomial $x^m - (2 + \sqrt{-1})$ is irreducible in $\mathbb{F}_{p^2}[x]$ for $m = 2^i 3^j$, $i, j > 0$.[1]

Proof. The values of a and b in Corollary 8.4.1 in this case are 2 and 1 respectively, so $a^2 + b^2 = 5$. The polynomial will be irreducible if 5 is neither a square nor a cube modulo

[1] In this case, the polynomial $x^m - (1 + 2\sqrt{-1})$ is also irreducible.

p. When $p \equiv 2$ or 3 modulo 5 we know that 5 is a quadratic non-residue modulo p, and so the only condition left is that 5 should not be a cube in $\mathbb{F}_p$. With p written in the form $p = 3u^2 + v^2$, we know that 5 is a cube if $15 \mid a$, or $3 \mid a$ and $5 \mid b$, or $15 \mid (a \pm b)$, or $15 \mid (a \pm 2b)$ [62]. Again, there is currently no proof concerning the number of primes in a quadratic sequence, but, as supported by experimental results, we expect that this occurs $1/3$ of the time. So 5 is a cubic non-residue modulo for approximately $2/3$ of the values of p. □

The result of Constructions 8.4.2 and 8.4.3 is that for around $2/3$ of the fields not considered pairing-friendly we have a more automatic and often more efficient implementation than is possible for pairing-friendly fields.

8.4.2 Euler's Conjectures

Primes p for which $p \equiv 2 \mod 3$, it is easily shown that every element is a cubic residue modulo p. For primes which are $1 \mod 3$, Fermat showed that p can be written as the sum $p = a^2 + 3b^2$ for some integers a and b. Euler conjectured (and Gauss proved) that using this form we can easily determine if some small elements are cubic residues:

1. 2 is a cubic residue $\Leftrightarrow 3 \mid b$.
2. 3 is a cubic residue $\Leftrightarrow 9 \mid b$; or $9 \mid (a \pm b)$.
3. 5 is a cubic residue $\Leftrightarrow 15 \mid b$; or $3 \mid b$ and $5 \mid a$; or $15 \mid (a \pm b)$; or $3 \mid (2a \pm b)$.
4. 6 is a cubic residue $\Leftrightarrow 9 \mid b$; or $9 \mid (a \pm 2b)$.
5. 7 is a cubic residue $\Leftrightarrow 21 \mid b$; or $3 \mid b$ and $7 \mid a$; or $21 \mid (a \pm b)$; or $7 \mid (a \pm 4b)$; or $7 \mid (2a \pm b)$.

These conjectures can be used once p has been constructed to decide if constructions 8.4.2 or 8.4.3 can be used. For some cases we have this information already.

BN Towers

Is was noticed by Shirase [88], that the polynomial parameterising the BN primes, $p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$, can be written in the form $p(x) = a(x)^2 + 3b(x)^2$, thus giving us more information about the towers we can construct for certain values of x_0, without having to perform the quadratic and cubic residue tests modulo p. We have $a(x) = 6x^2+3x+1$ and $b(x) = x$. We let x_0 denote the x value chosen to construct the parameters. With the additional information about the structure of $p(x)$ we are able to use Theorem 8.3.2 to put conditions on the values of x_0, which, when satisfied, give an immediate construction for the tower of fields of degree 12 over BN primes.

Considering first BN primes $p \equiv 3 \mod 4$ we know that $x_0 \equiv \pm 1 \mod 4$ and that we have a towering-friendly field, which requires a base tower $\mathbb{F}_{p^2}$, which can be constructed by adjoining $\sqrt{-1}$ to $\mathbb{F}_p$. We now need to find an element $a + b\sqrt{-1} \in \mathbb{F}_{p^2}$, such that $x^6 - (a + b\sqrt{-1})$ is irreducible, to construct the remaining extensions. From Corollary 8.4.1 we know that $x^6 - (a + b\sqrt{-1})$ is irreducible if $a^2 + b^2$ is neither a square nor a cube in $\mathbb{F}_p$. We know from Conjecture 1 that if $x_0 \equiv \pm 1 \mod 3$ then 2 is a cubic non-residue modulo p. For 2 to be a non-quadratic residue also we need $p \equiv 3 \mod 8$; this implies that $x_0 \equiv 3 \mod 4$. Together, these two constraints imply:

- If $x_0 \equiv 7$ or $11 \mod 12$ then $x^6 - (1 + \sqrt{-1})$ is irreducible over $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{-1})$.

In [88] the same conclusion is drawn, but using a much more elaborate method. We see that this result supports Construction 8.4.2 as $2/3$ of the possible values of x_0 give a p for which 2 is a quadratic non-residue.

Using Theorem 8.3.2 we are also able to classify more constructions than given in [88]. Using a similar method as above:

- If x_0 is odd and $x_0 \equiv 1, 3, 7, 11, 12$ or $13 \mod 15$ then $x^6 - (1 + 2\sqrt{-1})$ is irreducible over $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{-1})$.

Using Euler's conjectures it is also straight forward to set constructions for BN primes $p \equiv 1 \mod 4$, not needing a base tower.

- If $x_0 \not\equiv 0 \mod 3$ and $x_0 \equiv 2, 6 \mod 8$, then $x^{12} - 2$ is irreducible;
- If $x_0 \equiv 1, 3, 7, 11, 12$ or $13 \mod 15$, then $x^{12} - 5$ is irreducible;
- If $x_0 \not\equiv 0, 2$ or $4 \mod 9$ and $x_0/2$ is odd, then $x^{12} - 6$ is irreducible.

BN curves are quite plentiful and easy to find. Using BN curves in pairing-based protocols requires an efficient implementation of $\mathbb{F}_{p^{12}}$ and also of $\mathbb{F}_{p^2}$, as we would use a degree 6 twist. It may be favourable to choose $x_0 \equiv 1 \mod 2$ and x_0 satisfying one of the equivalences above, so that $\mathbb{F}_{p^2}$ can be constructed as $\mathbb{F}_p(\sqrt{-1})$ and the tower for $\mathbb{F}_{p^{12}}$ can be constructed, using one of Constructions 8.4.2 or 8.4.3, though these fields would not have originally been considered towering-friendly. Given that BN curves are so plentiful, this restriction would not impede finding curves appropriate for use.

KSS $k = 18$ Towers

When $k = 18$ the parameterisation of $p(x)$ can also be written in the form $a(x)^2 + 3b(x)^2 = p(x)$ where $a(x)$ and $b(x)$ have integer coefficients. In these cases we are also able to give the tower construction if the value x_0 satisfies some easily checked conditions.

The polynomial parameterisation of p for a KSS $k = 18$ curve is given by

$$p(x) = (x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401)/21.$$

We also know that $x \equiv 14 \mod 42$, so substituting $x = 42x' + 14$ we obtain the equation

$$p(x') = 461078666496x'^8 + 1284433428096x'^7 + 1564374047040x'^6 + 1088278335648x'^5 + 473078255328x'^4 + 131624074008x'^3 + 22896702948x'^2 + 2277529014x' + 99213811.$$

Using four values x'_0 for x' giving different primes, we are able to use Euclid's algorithm to find a and b such that $p = a^2 + 3b^2$. With these a, b values we then proceeded with

polynomial interpolation to find

$$a(x') = 444528x'^4 + 629748x'^3 + 333396x'^2 + 78321x' + 6908,$$

and

$$b(x') = 296352x'^4 + 407484x'^3 + 209916x'^2 + 48091x' + 4143,$$

such that $a(x')^2 + 3b(x')^2 = p(x')$. (We needed four x' values as it is clear that $a(x')$ and $b(x')$ would be at most degree 4 polynomials.) Using Euler's Conjectures we see that:

- If $x'_0 \equiv 1, 4, 5, 8 \mod 12$ then $x^{18} - 2$ is irreducible over $\mathbb{F}_p$;
- If $x'_0 \not\equiv 2, 3, 4 \mod 9$ then $x^{18} - 3$ is irreducible over $\mathbb{F}_p$;
- If $x'_0 \equiv 7, 9, 12, 14 \mod 15$ then $x^{18} - 5$ is irreducible over $\mathbb{F}_p$;
- If $x'_0 \equiv a \mod 42$ then $x^{18} - 6$ is irreducible over $\mathbb{F}_p$,

where $a = \{2, 3, 4, 9, 10, 11, 12, 13, 18, 20, 21, 22, 27, 28, 30, 31, 35, 36, 37, 38,$
$38, 40, 44, 45, 46, 48, 49, 53, 54, 55, 56, 57, 58, 62, 63, 64, 65, 66\}$;

- If $x'_0 \equiv 2 \mod 7$ then $x^{18} - 7$ is irreducible over $\mathbb{F}_p$.

The many possible constructions listed here mean that a tower can often be automatically constructed for an implementation using KSS $k = 18$ curves without further testing necessary.

8.4.3 Twists and Choosing α

When choosing a particular value of α to construct the tower, we may find that there are numerous potential values we could use. This is illustrated in the following example.

Example 1

The value $x_0 = 4008804000000009_{16}$ generates suitable parameters for a BN curve. Using this x_0 we see that $p \equiv 3 \mod 4$ and we first need a base tower $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{-1})$, before

we use the general construction method. We see also that $x_0 \equiv 3 \mod 15$ and so as, shown in Section 8.4.2, we know immediately that 5 is a cubic and quadratic non-residue in $\mathbb{F}_p$, and so $x^6 - (1 + 2\sqrt{-1})$ is irreducible over $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{-1})$. Using the same reasoning, however, we also know that $x^6 - (2 + 1\sqrt{-1})$, $x^6 - (2 - 1\sqrt{-1})$, $x^6 - (-2 - 1\sqrt{-1})$ and $x^6 - (-2 + 1\sqrt{-1})$ are all irreducible over $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{-1})$. Using this particular value of x_0, we also see that $a^2 + b^2$ is neither a square nor a cube for the (unordered and unsigned) pairs $(a, b) = (1, 3)$, $(1, 5)$, $(2, 3)$, as well as for $(1, 2)$. This example raises an important question:

How do we decide which value will be the best for implementation?

A simple analysis indicates that the optimal choice is the one which minimises $\omega(a)+\omega(b)$, where $\omega(n)$ is the number of additions required to perform a multiplication by n. There is another important point to take into account when choosing α and that is the construction of the twists of the elliptic curve used when computing the pairing.

In Section 3.1, we mentioned how twists are used to improve the efficiency of the pairing computation. To construct a twist of degree d and the isomorphism from the twist to the curve, we need an element from $\mathbb{F}_{p^{k/d}}$, which is a sth non-residue for all divisors s of k/d. We see now that Theorem 8.3.2 also gives us this element. In fact, it would make sense to use the same element to define the twist and to construct the tower. We will have to be slightly more careful, however, in our selection of the element α.

From Section 2.3.2.1, we see that there are two possible twists. The element i we choose to define our mappings to the twist with the correct number of points should be of the simplest form for the isomorphism to be as inexpensive as possible. If we select $i = \alpha^{(1/e)}$, where $e = k/d$, then the isomorphism is basically a free computation. If the curve defined choosing $i = \alpha^{(1/e)}$ does not give the correct number of points, then we must take $i = \alpha^{(3/e)}$, if E' is a quartic twist, or $i = \alpha^{(5/e)}$, if E' is a sextic twist. In these cases the isomorphism will be slightly more expensive. This is also discussed in [46].

To summarise, when selecting the element α to define the tower, both $\omega(\alpha)$ and the structure of the twist should be taken into account.

8.5 Curve Construction for BN and KSS $k = 18$

As mentioned above, once the system parameters have been selected, the curve itself is constructed using the CM method. This is a non-trivial process and it seems unnecessary when $D = 3$. The curve $E_b : y^2 = x^3 - b$ is usually constructed by chosing b at random and performing a trial scalar multiplication of a point on the curve, to check it has the correct order. In practice we find many of the same curves recurring for particular families (as first noted in [23]), and in these cases the trial scalar multiplication also becomes unnecessary. For example, for the BN curves $1/6$ choices for b will give an elliptic curve with the correct number of points [11].

Recent work by Shirase [89] has shown that even the scalar multiplication step can be omitted as some BN curves will always have the correct number of points when the x_0 value defining the parameters satisfies some easily verified equivalences. Shirase used:

Theorem 8.5.1 (Gauss)**.** *Let N_p be the number of solutions to the equation*

$$x^3 + y^3 = z^3$$

over $\mathbb{F}_p$. When $p \equiv 1 \mod 3$ there exists integers A and B such that

$$4p = A^2 + 27B^2.$$

Fixing the sign of A so that $A \equiv 1 \mod 3$ we have that $N_p = p + 1 + A$.

This is a shortened version of a Theorem in [93, Chapter IV, §2]. It is known that the solutions to $C : x^3 + y^3 = 1$ over $\mathbb{Q}$ correspond with $\mathbb{Q}$-rational points on the elliptic curve $E : y^2 = x^3 - 432$ and it has been shown that this also holds for C and E defined over $\mathbb{F}_p$ when $p \equiv 1 \mod 3$.

The results of [89] are:

- if $x_0 \equiv 2, 5 \mod 12$ then $E : y^2 = x^3 - 2$ is a BN curve,

- if $x_0 \equiv 2, 11 \mod 12$ then $E : y^2 = x^3 + 2$ is a BN curve.

KSS $k = 18$ curves also have the form $E : y^2 = x^3 + b$. Using similar methods, we have been able to show some results for KSS $k = 18$ curves. Specifically, if $x_0' \equiv 1 \mod 3$ we can always take $b = 16$ and also:

- if $x_0' \equiv 7, 10 \mod 12$ then $E : y^2 = x^3 + 2$ is an appropriate curve,
- if $x_0' \equiv 7 \mod 12$ then $E : y^2 = x^3 - 2$ is an appropriate curve.

For KSS $k = 18$ curves with defining parameter x_0' satisfying one of the above equivalences, we no longer need to use trial scalar multiplication to obtain the curve, it is given immediately.

Chapter 9

Performing the Final Exponentiation

- White Rabbit checking watch, Lewis Carroll's *Alice in Wonderland*, Chapter I, Down the Rabbit-Hole, illustration by Sir John Tenniel.

As mentioned in Chapter 3, the Tate pairing and all variations thereof actually evaluate as representatives of a cosets in $\mathbb{F}_{p^k}^*/(\mathbb{F}_{p^k}^*)^k$. This representative is not unique and is affected by the choice of divisor $D_Q \sim (Q) - (\mathcal{O})$ used in the evaluation of $e_r(P,Q)_r = f_{r,P}(D)$. In order to obtain a unique element, we raise the output of the Miller loop to the

power $(p^k - 1)/r$; this "kills off" any factors of order dividing $(p^k - 1)/r$, and results in a unique rth root of unity.

From Section 3.2.1.2, we notice that the final exponentiation is defined by the fixed system parameters and is constant for every pairing computation, so methods of exponentiation optimised for fixed exponents are applicable here.

Firstly, the final exponent can be broken down into three components. We have chosen the embedding degree k so that it is even, so let $u = k/2$. Then

$$(p^k - 1)/r = (p^u - 1) \cdot [(p^u + 1)/\Phi_k(p)] \cdot [\Phi_k(p)/r].$$

The first two parts of the exponentiation are "easy" since raising to the power of p is an almost free application of the Frobenius operator, as p is the field characteristic. The first part of the exponentiation is not only cheap (although it does require an extension field division) but also simplifies the rest of the final exponentiation. After raising to the power of $(p^u - 1)$, the field element becomes "unitary" [84], that is, an element α with norm $N_{\mathbb{F}_{p^k}/\mathbb{F}_{p^u}}(\alpha) = 1$. This has important implications, as squaring of unitary elements is significantly cheaper than squaring of non-unitary elements, and any future inversions can be implemented by simple conjugation [94, 84, 43, 75].

9.1 'Hard part' of the Final Exponentiation

We now consider the "hard part" of the final exponentiation, raising to the power of $\Phi_k(p)/r$. The usual approach is to express this exponent in base p, as $\lambda_{n-1} \cdot p^{n-1} + ... + \lambda_1 \cdot p + \lambda_0$, where $n = \phi(k)$, and $\phi(\cdot)$ is the Euler Totient function. Let m be the value given by result of Miller's algorithm, raised to the power $(p^u - 1) \cdot [(p^u + 1)/\Phi_k(p)]$. To complete the final exponentiation, we need to calculate

$$m^{\lambda_{n-1} \cdot p^{n-1}} \cdots m^{\lambda_1 \cdot p} \cdot m^{\lambda_0},$$

which is the same as

$$(m^{p^{n-1}})^{\lambda_n - 1} \cdots (m^p)^{\lambda_1} \cdot m^{\lambda_0}.$$

The m^{p^i} can be calculated using the Frobenius, and the hard part of the final exponentiation can be calculated using a fast multi-exponentiation algorithm [45, 40, 70].

These methods, however, do not exploit the polynomial description of p and r. It is our intention to do so, and hence obtain a faster hard-part of the final exponentiation. Each curve family is different in detail, so we will proceed on a case-by-case basis.

The BN curves

For the BN family of pairing-friendly curves, the hard part of the final exponentiation is computing m to the power of $(p^4 - p^2 + 1)/r$. After substituting the polynomials $p(x)$ and $r(x)$, this can be expressed in base $p(x)$ as:

$$\lambda_3.p(x)^3 + \lambda_2.p(x)^2 + \lambda_1.p(x) + \lambda_0,$$

where

$$\begin{aligned}
\lambda_3(x) &= 1; \\
\lambda_2(x) &= 6x^2 + 1; \\
\lambda_1(x) &= -36x^3 - 18x^2 - 12x + 1; \\
\lambda_0(x) &= -36x^3 - 30x^2 - 18x - 2.
\end{aligned}$$

Now we take a new approach. BN curves are plentiful, and choosing x_0 to have a low Hamming weight reduces the number of additions in the Miller loop, resulting in a faster pairing computation; this will have benefits in the final exponentiation evaluation also. Nogami et al. [76] have suggested the nice value $x_0 = -4080000000000001_{16}$ for a curve appropriate for the 128-bit level of security. We compute m^{x_0}, $m^{x_0^2} = (m^{x_0})^{x_0}$

and $m^{x_0^3} = (m^{x_0^2})^{x_0}$. These are simple exponentiations, and the low Hamming weight of x_0 ensures that each requires a minimum number of multiplications when using a simple square-and-multiply algorithm. We next calculate m^p, m^{p^2}, m^{p^3}, $(m^{x_0})^p$, $(m^{{x_0}^2})^p$, $(m^{{x_0}^3})^p$ and $(m^{{x_0}^2})^{p^2}$ using the Frobenius. Now group the elements of the exponentiation together, and the expression becomes:

$$[m^p \cdot m^{p^2} \cdot m^{p^3}] \cdot [1/m]^2 \cdot [(m^{{x_0}^2})^{p^2}]^6 \cdot [(m^{x_0})^p]^{12} \cdot [m^{x_0}/((m^{{x_0}^2})^p)]^{18} \cdot [1/m^{{x_0}^2}]^{30} \cdot [m^{{x_0}^3} \cdot (m^{{x_0}^3})^p]^{36}.$$

The individual components between the square brackets are then calculated with just 4 multiplications (recalling that division costs the same as a multiplication, as inversion is just a conjugation), and we end up with a calculation of the form:

$$y_0 \cdot {y_1}^2 \cdot {y_2}^6 \cdot {y_3}^{12} \cdot {y_4}^{18} \cdot {y_5}^{30} \cdot {y_6}^{36}.$$

Note that the exponents here are simply the coefficients that arise in the λ_i equations above. Now, how do we evaluate this expression most efficiently?

There is a well known algorithm to evaluate expressions of this form, which minimizes the number of required multiplications. Examples are given in [77] and also [4, Section 9.2]. The starting point is to find an *addition sequence*: an addition chain, which includes within it the elements of the set of integers which occur as exponents. In this case it is not hard to see that an optimal addition sequence (the shortest sequence containing all values) is given by:

$$\{1, 2, \underline{3}, 6, 12, 18, 30, 36\}.$$

Note that 3 is the only member of this sequence which is not a member of the set of exponents. This is certainly serendipitous, as it means less work to do the evaluation. Observe here that an addition-subtraction chain is also a possibility (as divisions are as cheap as mul-

tiplications as a consequence of the unitary property); but in this case it gives no advantage over the above addition chain. Application of the Olivos algorithm results in the following vectorial addition chain:

$$
\begin{array}{rrrrrrr}
(1 & 0 & 0 & 0 & 0 & 0 & 0) \\
(0 & 1 & 0 & 0 & 0 & 0 & 0) \\
(0 & 0 & 1 & 0 & 0 & 0 & 0) \\
(0 & 0 & 0 & 1 & 0 & 0 & 0) \\
(0 & 0 & 0 & 0 & 1 & 0 & 0) \\
(0 & 0 & 0 & 0 & 0 & 1 & 0) \\
(0 & 0 & 0 & 0 & 0 & 0 & 1) \\
(2 & 0 & 0 & 0 & 0 & 0 & 0) \\
(2 & 0 & 1 & 0 & 0 & 0 & 0) \\
(2 & 1 & 1 & 0 & 0 & 0 & 0) \\
(0 & 1 & 0 & 1 & 0 & 0 & 0) \\
(2 & 2 & 1 & 1 & 0 & 0 & 0) \\
(2 & 1 & 1 & 0 & 1 & 0 & 0) \\
(4 & 4 & 2 & 2 & 0 & 0 & 0) \\
(6 & 5 & 3 & 2 & 1 & 0 & 0) \\
(12 & 10 & 6 & 4 & 2 & 0 & 0) \\
(12 & 10 & 6 & 4 & 2 & 1 & 0) \\
(12 & 10 & 6 & 4 & 2 & 0 & 1) \\
(24 & 20 & 12 & 8 & 4 & 2 & 0) \\
(36 & 30 & 18 & 12 & 6 & 2 & 1)
\end{array}
$$

which in turn allows us to evaluate the expression as follows, using just two temporary variables:

$$
\begin{aligned}
T_0 &\leftarrow (y_6)^2 \\
T_0 &\leftarrow T_0 \cdot y_4 \\
T_0 &\leftarrow T_0 \cdot y_5 \\
T_1 &\leftarrow y_3 \cdot y_5 \\
T_1 &\leftarrow T_1 \cdot T_0 \\
T_0 &\leftarrow T_0 \cdot y_2 \\
T_1 &\leftarrow (T_1)^2 \\
T_1 &\leftarrow T_1 \cdot T_0 \\
T_1 &\leftarrow (T_1)^2 \\
T_0 &\leftarrow T_1 \cdot y_1 \\
T_1 &\leftarrow T_1 \cdot y_0 \\
T_0 &\leftarrow (T_0)^2 \\
T_0 &\leftarrow T_0 \cdot T_1
\end{aligned}
$$

The final result is in T_0. This part of the calculation requires only 9 multiplications and 4 squarings. We find this approach to the hard part of the final exponentiation for the BN curves to be about 4% faster than the rather ad hoc method proposed by Devegili et al. [23] (7156 modular multiplications/squarings over $\mathbb{F}_p$, compared to 7426 for the choice of x_0 suggested above). Moreover, our more general method is applicable to all families of pairing-friendly curves.

KSS Curves $k = 18$

The KSS $k = 18$ curves introduce another aspect of this method: removing denominators from the coefficients of the base $p(x)$ representation, which occur in some families.

The base $p(x)$ representation of $\Phi_k(p(x))/r(x)$ is given by:

$$\begin{aligned}
\lambda_0(x) &= (-3x^7 - 15x^6 - 21x^5 - 62x^4 - 319x^3 - 434x^2 + 3)/3; \\
\lambda_1(x) &= (14x^6 + 70x^5 + 98x^4 + 273x^3 + 1407x^2 + 1911x)/3; \\
\lambda_2(x) &= (-49x^5 - 245x^4 - 343x^3 - 931x^2 - 4802x - 6517)/3; \\
\lambda_3(x) &= (-5x^7 - 25x^6 - 35x^5 - 87x^4 - 450x^3 - 609x^2 + 54)/3; \\
\lambda_4(x) &= (7x^6 + 35x^5 + 49x^4 + 112x^3 + 581x^2 + 784x)/3; \\
\lambda_5(x) &= (49x^2 + 245x + 343)/3.
\end{aligned}$$

A minor difficulty arises due to the common denominator of 3 which occurs here. We suggest a simple solution: evaluate instead the cube of the pairing. We use the fact that for any $\ell \in \mathbb{Z}$ such that $\gcd(\ell, r) = 1$, computing $m^{\ell(p^k-1)/r}$ will give us an element of order r and thus will retain the desired properties of the pairing. When r is of cryptographic size, 3 is co-prime to r, so we can simply ignore the denominator. The denominator in this case is not unusual – it appears in the base $p(x)$ representation of $\Phi_k(p(x))/r(x)$ for the other KSS curves, FST curves and BLS curves mentioned in Table 3.1. We therefore use:

$$\begin{aligned}
\lambda_0(x) &= -3x^7 - 15x^6 - 21x^5 - 62x^4 - 319x^3 - 434x^2 + 3; \\
\lambda_1(x) &= 14x^6 + 70x^5 + 98x^4 + 273x^3 + 1407x^2 + 1911x; \\
\lambda_2(x) &= -49x^5 - 245x^4 - 343x^3 - 931x^2 - 4802x - 6517; \\
\lambda_3(x) &= -5x^7 - 25x^6 - 35x^5 - 87x^4 - 450x^3 - 609x^2 + 54; \\
\lambda_4(x) &= 7x^6 + 35x^5 + 49x^4 + 112x^3 + 581x^2 + 784x; \\
\lambda_5(x) &= 49x^2 + 245x + 343.
\end{aligned}$$

The coefficients again "nearly" form a natural addition chain. Our best attempt to find an addition sequence containing all of the exponents in the above is:

$\{\underline{1},\underline{2},3,\underline{4},5,7,\underline{8},14,15,\underline{16},21,25,\underline{28},35,\underline{42},49,54,62,70,87,98,112,\underline{147},245,273,\underline{294},$
$319,343,\underline{392},434,450,581,609,784,931,\underline{1162},1407,\underline{1862},1911,\underline{3724},\underline{4655},4802,6517\}.$

The underlined numbers are the ones added in order to complete the sequence.

Proceeding as in the BN case, we find that the vectorial chain derived from this addition sequence requires just 56 multiplications and 14 squarings to complete the calculation of the hard part of the final exponentiation. We did eventually find (by partial computer search) an addition sequence one element shorter than the above, but it required 61 multiplications and only 7 squarings. We prefer to use the solution above as the computations are performed in an extension field, which is also a *squaring-friendly* field as defined in [42], and squarings are notably cheaper than multiplications.

Final Exponentiation: Discussion

Here we make a few general observations. First, it seems that the proposed method results in surprisingly compact addition sequences. We note also that the coefficients in the $\lambda_i(x)$ tend to be "smooth" numbers, having only relatively small factors. More evidence for both of these claims is given by the $\lambda_i(x)$ computations in the Appendix. This may facilitate the construction of addition sequences. Other intriguing patterns emerge – observe for example that, for the KSS $k = 18$ curves, the three most significant coefficients of the $\lambda_i(x)$ are all in the same ratio $1 : 5 : 7$. Coefficients also appear to follow the same kind of distribution as numbers in a typical addition chain.

We have also used the proposed method for other families of pairing-friendly curves, and have observed that, for example, for the FST and BLS curves, the resulting addition sequence is often as easy as:

$$\{1, 2, 3\}.$$

Since squarings are significantly faster than multiplications (as our computations are over extension fields) it may, as we have seen, be sometimes preferable to select a slightly longer addition sequence which trades additions for doublings. Addition-subtraction se-

quences may also be an attractive alternative in other cases.

Finding the shortest addition sequence is an NP-complete problem [27]. The values we obtained in each set, however, are relatively small, and so the sets themselves already contained some addition 'subchains;' it was not too difficult to generate, either with a computer or manually, addition sequences containing the specific entries, with length close to the lower bound given for the length of addition chains [17]. Should a particular curve result in larger or more numerous coefficients to be constructed into a sequence, Bos and Coster suggest an algorithm for that scenario in [17].

Chapter 10

Cofactor Multiplication to Obtain a Point in $\mathbb{G}_2$

'Curiouser and curiouser!'

- Alice, Lewis Carroll's *Alice in Wonderland*, Chapter II, The Pool of Tears.

When using ordinary elliptic curves to implement identity-based protocols, there is often a need to hash identities to points on one or both of the two elliptic curve groups involved in the pairing. The first group, denoted $\mathbb{G}_1$, consists of points on a pairing-friendly elliptic curve E of prime order r that are defined over the base field $\mathbb{F}_p$. The second group, denoted $\mathbb{G}_2'$, is instantiated as a group of points on a twisted curve E', which have coordinates in some extension field $\mathbb{F}_{p^e}$, where e divides the embedding degree k, which is isomorphic to a group $\mathbb{G}_2$ of order r in $E(\mathbb{F}_{p^k})$ (as detailed in Chapter 3).

The Tate pairing and its variants only require one of the input points to be of prime order, as it is sufficient for the other argument to be a coset representative. For the Weil pairing, both input points must have prime order. The most efficient pairings, to date, are

the Ate and R-ate pairings as described in Section 3.1 and both specifically require $\mathbb{G}_2$ to have prime order. Whereas hashing to a point in $\mathbb{G}_1$ is relatively easy, hashing to a point in $E(\mathbb{F}_{p^e})$ of specific order requires a multiplication by a large cofactor of a point in $E(\mathbb{F}_{q^e})$. In this section we consider the problem of reducing the cost of hashing to a point in $\mathbb{G}_2'$, by reducing the cost of performing the multiplication by the large cofactor. This step may be necessary to ensure efficient implementations of protocols using Weil, ate or R-ate pairings.

Some reluctance to use PBC stems from the necessary implementation of extension field arithmetic and handling of points in the, potentially cumbersome, group $\mathbb{G}_2$, defined over an extension field. In [36], however, Galbraith and Scott observe that arithmetic in $\mathbb{G}_2$ is not as difficult as might be thought, as an efficient homomorphism can be exploited. In this section we extend the ideas of [36] to the related problem of cofactor multiplication in $E'(\mathbb{F}_{p^e})$, which is required to hash an identity to a point of prime order in $\mathbb{G}_2$.

Let E be an elliptic curve defined over $\mathbb{F}_p$ with a large prime order subgroup of order r and embedding degree k with respect to r. Let E' be a twist of degree $d \mid k$ of E, with a group of points of order r defined over $\mathbb{F}_{p^e}$, where $e = k/d$ (Section 2.3.2.1).

If $d = k$ (in which case $E \cong E'$), we define $\mathbb{G}_2$ to be the cyclic subgroup of $E[r]$, on which the p-power Frobenius of E acts as multiplication by p.

From Chapter 2 we know that $\#E(\mathbb{F}_p) = p+1-t$, where t is the trace of the Frobenius, which obeys the Hasse bound $|t| \leq 2\sqrt{p}$. Consider now points whose coordinates are defined over an extension field $\mathbb{F}_{p^m}$, and the number of such points on the same elliptic curve [68]. It is well known for example, that

$$\begin{aligned} \#E(\mathbb{F}_{p^2}) &= p^2 + 1 - (t^2 - 2p), \\ \#E(\mathbb{F}_{p^3}) &= p^3 + 1 - (t^3 - 3tp). \end{aligned}$$

In the general case the number of points can be calculated by the following Algorithm 2 [68].

Algorithm 2 Calculate $\#E(\mathbb{F}_{p^m})$

INPUT: m, p, t: m a positive integer, p a prime, t the trace of Frobenius of an elliptic curve E defined over $\mathbb{F}_p$.
OUTPUT: $\#E(\mathbb{F}_{p^m})$.

```
τ_0 ← 2
τ_1 ← t
for i ← 1 to m − 1 do
    τ_{i+1} ← t · τ_i − p · τ_{i−1}
end for
q ← p^m
τ ← τ_m
Return q + 1 − τ
```

To represent the group $\mathbb{G}_2$, we like to use an isomorphic group on a twisted curve, over the smallest possible extension field. The number of points on the twisted curve can also easily be determined from the output of Algorithm 2.

The following formulæ are for quadratic, quartic and sextic twists, as given in [46]:

$$\begin{aligned}
\text{quadratic: } \#E'(\mathbb{F}_q) &= q + 1 + \tau; \\
\text{quartic: } \#E'(\mathbb{F}_q) &= q + 1 \pm f_1 \text{ where } f_1 = \sqrt{4q - \tau^2}; \\
\text{sextic: } \#E'(\mathbb{F}_q) &= q + 1 \pm (3f_2 + \tau)/2 \text{ where } f_2 = \sqrt{(4q - \tau^2)/3},
\end{aligned}$$

where $q = p^m$ and τ is the trace of the q-power Frobenius on E, as calculated in Algorithm 2.

To hash to a point in $\mathbb{G}_2$, the standard approach would be to first hash to a general point on $E'(\mathbb{F}_{p^e})$ and then multiply by the cofactor $c = \#E'(\mathbb{F}_{p^e})/r$. Consider now a pairing-friendly curve with $k = 10$, $e = 5$ and $r \approx p$. In this case, using the quadratic twist, this cofactor c would be of a size, in bits, approximately the same as p^4. This would be prohibitively slow. Here we will show that the same outcome can be achieved in all cases with the equivalent work of a multiplication by a value less than p, and in some cases much less than p. Clearly, this is a very expensive operation, supporting the decision to use ordinary pairing-friendly elliptic curves, which support the highest possible twist, such as those presented in Section 3.2.

10.1 A Fast Cofactor Multiplication Algorithm

The issue of fast cofactor multiplication of points in $E'(\mathbb{F}_{p^e})$ was briefly considered for the BN curves [11] by Galbraith and Scott [36, Section 8]. We have generalised and extended their idea. In that paper the authors introduce the homomorphism $\psi = \varphi^{-1}\pi_p\varphi$, where $\varphi : E' \to E$ is the isomorphism which maps points from the twisted curve $E'(\mathbb{F}_{p^e})$ to the isomorphic group in $E(\mathbb{F}_{p^k})$, as actually required by the pairing algorithm, and π_p is the p-power Frobenius map on E. Note that $\psi(P)$ can be calculated very quickly.

General points on $E'(\mathbb{F}_{p^e})$ obey the identity [35, Theorem 1]:

$$\psi^2(P) - [t]\psi(P) + [p]P = 0.$$

Our main idea is to first express the cofactor c to the base p:

$$c = c_0 + c_1 \cdot p + c_2 \cdot p^2 ...$$

and then use the identity

$$[p]P = [t]\psi(P) - \psi^2(P) \tag{10.1}$$

repeatedly, if necessary, to reduce the cofactor multiplication to a form

$$[c_0 + p(c_1 + p(c_2 + ...))]P = [g_0]P + [g_1]\psi(P) + [g_2]\psi^2(P) + ..., \tag{10.2}$$

where all of the g_i are less than p. Observe that $[c_1 \cdot p]P = [c_1 \cdot t]\psi(P) - [c_1]\psi^2(P)$, and that $c_1 \cdot t$ may be of a size in bits 50% larger than p (recall that t can be up to half the size of p as a consequence of the Hasse condition). Further applications of the homomorphism may therefore be necessary to effect a complete reduction. The end result is a recoding of c from a base p representation to a base $\psi(\cdot)$ representation, with all coefficients less than p. The number of terms in the representation increases with each application of the identity

(10.1), so in some circumstances, we will also find the following identity to be useful:

$$\Phi_k(\psi(P)) = 0, \tag{10.3}$$

where Φ_k is the kth cyclotomic polynomial. This identity allows terms of degree greater than or equal to $\phi(k)$ (the Euler totient function) to be replaced with terms of lower degree.

In the case that $k = de$ and $(d, e) = 1$, we observe that the twisting isomorphism φ defining a twist of degree d can be chosen, so that the twisted curve E' is actually defined over $\mathbb{F}_p$ (in which case φ is defined over $\mathbb{F}_{p^d}$). In this case, the cofactor c can be factored into $h \cdot c_1$, where $c_1 = \#E'(\mathbb{F}_p)$. The endomorphism $\pi'_p - 1$ (where π'_p is the p-power Frobenius map on E') projects into the subgroup of $\#E'(\mathbb{F}_{p^e})$ of order $h \cdot r$, thus we only need to perform a multiplication by h to obtain a point of order r. In this case, our algorithm only needs to be applied to the smaller factor h.

10.2 Application to Ordinary Pairing-Friendly Elliptic Curves

It is our aim to exploit the polynomial parameterisation of the prime modulus p, the group r and the trace t in a systematic way, to further speed up the cofactor multiplication required for hashing to $\mathbb{G}_2$.

First, we formally describe the method of the previous section: an algorithm for reducing the cofactor multiplication to the evaluation of a polynomial of the powers $\psi^i(P)$, with coefficients less than p. When p is itself expressed as a polynomial $p(x)$, these coefficients can, in turn, be calculated as polynomials in x, and this we choose to do, as it leads to further optimizations. In these cases the cofactor c itself can also be calculated and presented as a polynomial in x. We emphasise, however, that the basic idea (with minor modifications) applies equally to non-parameterised Cocks-Pinch curves. See Algorithm 3. For a step-by-step walk-through of the algorithm, see the section on BN curves below.

Algorithm 3 Reduction of the cofactor $c(x)$ to base $\psi(\cdot)$

INPUT: $k, p(x), t(x)$, and $c(x)$: embedding degree k and polynomials $p(x)$, $t(x)$, $c(x)$ parametrising the field size, trace, and $\mathbb{G}_2$ cofactor of a pairing-friendly elliptic curve, respectively.

OUTPUT: $g_0(x), g_1(x), \ldots, g_{\varphi(k)-1}(x)$, $\deg g_i(x) < \deg p(x)$, $g_i(x)$ are the coefficients of a base $\psi(\cdot)$ representation of the cofactor $c(x)$.

```
f ← ⌊deg(c(x))/ deg(p(x))⌋
◊ First express c(x) to the base p
for i ← 0 to f do
    c_i(x) ← c(x) mod p(x)
    c(x) ← c(x) div p(x)
end for
◊ Make a first pass to determine the coefficients g_i of c(x) in base ψ(·), using equation (10.1).
for j ← 0 to 2f − 1 do
    g_j ← 0
    for i ← 0 to ⌊j/2⌋ do
        g_j ← g_j + C(j−i, i) t(x)^{j−2i} (−1)^i c_{j−i}(x)
    end for
end for
◊ Make a second pass to finally force all coefficients to have degree < deg p
g_{2f+1} ← 0, g_{2f+2} ← 0
for j ← 1 to 2f do
    w(x) ← g_j(x) div p(x)
    g_j(x) ← g_j(x) mod p(x)
    g_{j+1}(x) ← g_{j+1}(x) + t(x)w(x)
    g_{j+2}(x) ← g_{j+2}(x) − w(x)
end for
◊ Finally exploit equation (10.3); a_i is the coefficient of x^i in Φ_k(x)
for j ← 2f + 2 downto φ(k) do
    for i ← 1 to φ(k) do
        g_{j−i}(x) ← g_{j−i}(x) − a_{φ(k)−i} · g_j(x)
    end for
    g_j(x) ← 0
end for
```

Algorithm 3 Summary

Algorithm 3 takes as input the embedding degree k, and the polynomials $p(x)$, $t(x)$ and $c(x)$, where the polynomial $c(x)$ parametrises the hard part of the multiplication to be performed to obtain a point of order r on the twist of the elliptic curve. (The polynomials $p(x)$ and $t(x)$ are the parametrisations of p and t as given for a particular family with embedding degree k.) The first step is to recode $c(x)$ to the base $p(x)$ (lines 3–6), then, using this representation of $c(x)$, recode $c(x)$ to the base $\psi(\cdot)$ (lines 8–13). The coefficients of the base $\psi(\cdot)$ representation are computed using the coefficients of the base $p(x)$ representation and the appropriate coefficients of the equation,

$$[p^l]P = \sum_{i=0}^{l} \binom{l}{i} t(x)^{l-i}(-1)^i \psi^{l+i}(P),$$

obtained by applying induction on equation (10.1). Once $c(x)$ has been written to base $\psi(\cdot)$, the coefficients $g_i(x)$ are checked. If $\deg g_i(x) \geq \deg p(x)$ then the identity $[p]P = [t]\psi(P) - \psi^2(P)$ is reapplied (lines 15–21). Finally the relation (10.3) is exploited to obtain a base $\psi(\cdot)$ representation of $c(x)$ of degree $< \phi(k)$ (lines 23–28).

We now proceed to use this algorithm to find a faster way to perform the cofactor multiplication required to hash to a point of order r in $\mathbb{G}_2$. We proceed on a case-by-case basis for certain selected popular families of pairing-friendly elliptic curves.

BN curves

We now step through Algorithm 3, using the BN family of curves to illustrate each step. The BN curves have embedding degree 12 and CM discriminant 3, so they have sextic twists defined over $\mathbb{F}_{p^2}$. As stated in [11], the twist with the correct number of points has

$$N = p^2(x) + t^2(x) - 1,$$

which we rearrange to obtain

$$(p(x) + 1 - t(x))(p(x) - 1 + t(x)) = r(x)(p(x) - 1 + t(x))$$

so the cofactor in this case is

$$c(x) = p(x) - 1 + t(x).$$

Writing $c(x)$ to the base $p(x)$ (lines 3–6) is straightforward in this case with coefficient polynomials

$$\begin{aligned} g_0 &= t(x) - 1, \\ g_1 &= 1. \end{aligned}$$

Now apply equation (10.1) to each term involving a power of $p(x)$, and use it to express $[c(x)]P$ in base $\psi(\cdot)$ form (lines 8–13 of the algorithm). We find that, taking into account the parameterisation of t,

$$[c(x)]P = \psi(6x^2P) + 6x^2P + \psi(P) - \psi^2(P).$$

This supports the results of [36, section 8] where the cofactor multiplication for BN curves was examined.

The major work remaining here is the point multiplication by $6x^2$. Since BN curves are plentiful, it is not hard to find a value of x with a very low Hamming weight (as is already commonly done to optimize the main Miller loop of the pairing algorithm), and this will further speed up the calculation, as the point multiplication will consist largely of point doublings, which are significantly faster than point additions in most curve and point representations.

KSS $k = 18$

We now apply Algorithm 3 to the KSS $k = 18$ family of pairing-friendly elliptic curves. We start by computing $c(x) = \#E'(\mathbb{F}_{p^3})/r(x)$ which is:

$$1/27x^{18} + 5/9x^{17} + 32/9x^{16} + 409/27x^{15} + 199/3x^{14} + 881/3x^{13} + 27539/27x^{12} + 27220/9x^{11} + 85636/9x^{10} + 757927/27x^{9} + 601228/9x^{8} + 1351828/9x^{7} + 9658007/27x^{6} + 2162818/3x^{5} + 1142985x^{4} + 85636/9x^{10} + 757927/27x^{9} + 601228/9x^{8} + 1351828/9x^{7} + 9658007/27x^{6} + 2162818/3x^{5} + 1142985x^{4} + 50075833/27x^{3} + 27518078/9x^{2} + 29615306/9x + 40301641/27.$$

After applying Algorithm 3 to write $c(x)$ in base $\psi(\cdot)$ we obtain:

$$\begin{aligned}
g_0(x) &= (-5x^7 - 26x^6 - 98x^5 - 381x^4 - 867x^3 - 1911x^2 - 5145x - 5774)/3; \\
g_1(x) &= (-5x^7 - 18x^6 - 38x^4 - 323x^3 - 28x^2 + 784x)/3; \\
g_2(x) &= (-5x^7 - 18x^6 - 38x^4 - 323x^3 + 1029x + 343)/3; \\
g_3(x) &= (-11x^6 - 70x^5 - 98x^4 - 176x^3 - 1218x^2 - 2058x - 686)/3; \\
g_4(x) &= (28x^2 + 245x + 343)/3;
\end{aligned}$$

A minor difficulty arises due to the common denominator of 3 which occurs here. As in Chapter 9, the solution is quite simple – for any $\ell \in \mathbb{Z}$ coprime to r, the point $[\ell \cdot c(x_0)]P$ will also be a point of order r. We can take $[\ell \cdot c(x_0)]P$, instead of $[c(x_0)]P$, to be our point of order r, simplifying the calculations. We use the g_i with the denominator removed to evaluate $[3 \cdot c(x)]P$.

In this case, the best addition sequence we could find that includes all of the coefficients was:

{$\underline{1}$, $\underline{2}$, $\underline{3}$, 5, $\underline{7}$, $\underline{8}$, 11, 18, 26, 28, $\underline{31}$, 38, $\underline{45}$, $\underline{69}$, 70, $\underline{78}$, 98, 176, 245, $\underline{253}$, 323, 343, 381, $\underline{389}$, 686, 784, $\underline{829}$, 867, 1029, 1218, $\underline{1658}$, 1911, 2058, $\underline{4116}$, 5145, 5774},

where the underlined numbers are the extra numbers included to complete the sequence. This sequence can be used to complete the calculation in 51 point additions and 5 point

doublings.

Cofactor Multiplication: Discussion

For similar reasons as in the previous chapter, it may sometimes be preferable to select a slightly longer addition sequence which trades additions for doublings: in most cases (dependent on the curve representation and the projective coordinate method used) point doublings are significantly faster than point additions. The situation is complex, however, and requires further study. For example, if doubling or adding a point on $E'(\mathbb{F}_{p^5})$ it is likely that affine coordinates will in fact be faster than any kind of projective coordinates, in which case, using the standard short Weierstraß representation, additions may actually be faster than doublings [44]. Addition-subtraction sequences may also be an attractive alternative in other cases.

Chapter 11

Part II Summary

'It's a poor sort of memory that only works backwards.'

- The Queen, Lewis Carroll's *Through the Looking Glass and What Alice Found There*, Chapter V, Wool and Water.

Through the work done in this section, the contributions to PBC are:

- A method for constructing tower extensions for the fields used in PBC. These tower extensions are necessary, as it is well recognised that the implementation of finite extension fields as a sequence of Kummer extensions (when possible) is the most efficient. Tower extensions are assumed in many of the improvements, such as those suggested in [42, 21, 2]. The results of this section show that this tower construction is available for more fields occurring in the context of PBC than were previously considered for use and gives explicit constructions for some fields accompanying BN curves and KSS $k = 18$ curves.

- A fixed elliptic curve equation for some KSS $k = 18$ curves when x_0, the parameter defining the system parameters, satisfies some easily checked equivalence relations.

- A general method for the implementation of the hard part of the final exponentiation in the calculation of the Tate pairing and its variants, which is faster, generally applicable, and requires less memory than previously described methods.

- A method for deriving a point on $E'(\mathbb{F}_{p^e})$ of order r given an initial hashing to a general point on $E'(\mathbb{F}_{p^e})$, the degree d twist of an ordinary pairing-friendly elliptic curve with embedding degree k. The proposed method is significantly faster than the naive approach, which would require multiplication by a very large cofactor.

These methods can contribute to a cryptographic compiler, specialised for pairing-based cryptography, as described in [26]. Given only the polynomial equations defining a pairing-friendly family of elliptic curves, it should now be possible, and indeed appropriate, to write a computer program which would automatically generate very efficient R-ate pairing code (for the curves discussed in this section), using the efficient representation of the finite extension field, a faster cofactor multiplication formula and a final exponentiation method.

Chapter 12

Closing Remarks

'Everything has got a moral if you can only find it.'

- The Duchess, Lewis Carroll's *Alice in Wonderland*,

Chapter IX, The Mock Turtle's Story.

In this chapter, we summarise the contributions of this thesis and introduce some points worthy of continued research.

12.1 Summary

PBC is by nature fundamentally different from other areas of cryptography. There are implementation issues which do not have to be considered for other public-key cryptographic schemes such as RSA, or other schemes based on the DLP: It is not possible to write a general implementation of a pairing-based protocol which performs reasonably efficiently for any level of security. Different security levels require different pairing-friendly elliptic curves and different pairing variations (specifically an Ate pairing variation). The development and maintenance of good quality pairing code is therefore difficult. Much of the work

presented in this thesis was motivated by the intention to contribute to a cryptographic compiler: a tool for automatically generating good quality code for PBC at all security levels, as first introduced by Dominguez and Scott [26].

In Chapter 8, a method for constructing the extension fields used in PBC is given. The optimal tower extensions can be automatically generated using the results of Chapter 8 and allow various optimisations to also be incorporated into a cryptographic compiler such as those presented in [42, 21, 2]. Such constructions are proposed in the IEEE draft standard [1].

In Section 8.5, we have built on work by Shirase [89], which automatically gives BN curve equations. We have given elliptic curve equations for KSS $k = 18$ curves for which the defining parameter x_0' satisfies some easily verified equivalence equations. This result removes the necessity of performing expensive trial point scalar multiplications to find the curve equation.

The final exponentiation step of the pairing computation is expensive, but necessary; in Chapter 9 we present a faster method of performing the final exponentiation using the Frobenius automorphism and addition chains.

For many protocols it is necessary to hash an identity to a random point on an elliptic curve, then perform a scalar multiplication by a large cofactor to obtain a point of a particular order. This cofactor is quite large and so the scalar multiplication is quite expensive. In Chapter 10, drawing on ideas of [36], we used a group automorphism and addition chains to speed up the cofactor multiplication process.

After a detailed examination of the algorithms used to solve the DLP and ECDLP as they occur in PBC (Chapters 5 and 6 respectively), we were able to compile Table 7.1 to give a more precise approximation of the levels of security achievable for efficient implementations using the curves recommended in Table 3.1. From Table 7.1 we are easily able to deduce the size of the parameters necessary to obtain these security levels; this can be used as a guideline for implementers of pairing-based protocols.

12.2 Future Work

PBC is a relatively new area of research; there are many aspects which can be developed.

It is an open problem to construct non-supersingular abelian varieties — including elliptic curves — over non-prime fields with small embedding degree and $\rho < 16$. Such a construction would not only expand our library of pairing-friendly abelian varieties but could potentially lead to improvements in the performance of pairing-based protocols, in the same way that elliptic curves over non-prime fields can lead to performance improvements for standard elliptic curve cryptography [35]. In [80] Rubin and Silverberg show that using supersingular abelian varieties (with genus $2 \leq g \leq 4$) can lead to more efficient implementations than those using supersingular elliptic curves.

A pattern of constant coefficients of pairing-friendly elliptic curves has emerged, first noticed by Devegili et al. in [23]. Some of these patterns have been explained in [89] and Part II, but there are many curves for which this is yet to be explained. The observation in [23] that when p satisfies $p \equiv 7 \mod 8$, $p \equiv 4 \mod 9$ and $p \equiv 1 \mod 6$ then we can take $E : y^2 = x^3 + 3$, is an interesting case worth further exploration.

The NFS could be adapted to the context of PBC. As described in [51], the NFS presumes general parameters. In PBC the parameters are quite specific, in particular, the extension degree k which is a product of powers of 2 and 3. Such extension fields have a rich structure, which could still be exploited by the attacker to reduce the hardness of the DLP instance occurring in pairing-based protocols. The extension degrees are also fixed (not tending to infinity) and the prime characteristics are of a particular form. These could potentially introduce vulnerabilities to the systems (as also hinted by Schirokauer [82], as the system parameters are often chosen to have low weight). This is yet to be thoroughly investigated.

Appendix A

Computing the Security Levels of Suggested Curves

We describe here in more detail the method used to obtain the security levels given in Table 7.1. There is little known about the actual runtime of the NFS as it is still computationally infeasible test the NFS, even when the parameters are selected to be 'optimal' for the run-time (selecting the p and k such that degree of the elements sieved gives the lowest possible heuristic run-time). Even the worked example in [51] does not have 'optimal' parameters (from the perspective of the NFS). This is to the advantage of PBC, the implementational issues associated with using the NFS mean that the complexity analysis slightly over-estimates the capabilities of the NFS in practice and therefore underestimates the level of security offered; in cryptography it is usual to be conservative when estimating security levels.

The analysis method used to compile the table in Chapter 7 takes the specific properties of the curves used throughout this thesis, but could easily be adapted to all pairing friendly elliptic curves. The analysis takes into account the following:

1. The size of the autmorphism group of the curve;

2. the ρ-value of each curve;

3. the degree of elements sieved in the sieving stage;

4. the relative sizes of p and k.

In Chapter 6 we saw that elliptic curves with non-trivial automorphism group allow modifications to Pollard's Rho method which give a speed up of a factor of $\sqrt{|\mathrm{Aut}(E)|}$, this is the case for all curves in Table 3.1 an is therefore taken into account in the analysis.

The ρ-value of each curve determines the size relationship between the group of points on the elliptic curve and the field over which it is defined, this will in turn affect the relative sizes of p and k, for fixed k and group size r.

Both points 3 and 4 have an effect on the heuristic run-time of the NFS and, as illustrated by [101], an even stronger effect on the actual run-time. The choice of the polynomials used in the NFS to construct the number fields also has a significant effect on the run-time of the NFS; this can only be determined on a case by case basis and so could not be taken into account for this analysis.

The basic idea is, for a fixed embedding degree k and corresponding ρ-value, find the security interval for which the complexity of the NFS is approximately equal to the complexity of Pollard's Rho method. This was done using a small Magma program. We step through an example here to illustrate.

KSS $k = 18$ curves

Listing A.1: Code in Magma for a Generic Assignment

```
// Input  <- LH, RH, notation, operator(s), library definitions
// Output -> Corresponding output code to assign Left <- Right
// Example Input: GenericAssign("fm1","c",lang[tgtRATE]["powerPOS"],"
    powerMAP","powerMAP2","RATEmiracl",lang)
// Example Output: pow(fm1,c)
GenericAssign:=function(Left,Right,operatorPOS,operator1,operator2,
    target,language)
  Code:="";
  case operatorPOS:
    when 0:

function L(q,a,c)
```

```
  l := Exp(c*(Log(q)^(a))*(Log(Log(q))^(1-a)));

  return l;

end function;


curves[4] := "BN";
complexity_multiple[4] := Sqrt(Pi(RealField())/12);
k[4] := 12;
rho[4] := 1/1;
m[4] := 250;  //expected value, used as starting point for testing

bound := 3;
PrintFile("Pollard_NFS_complexity.txt",Sprintf("bound:␣%o\n\n", bound
    ));


for j in [4 .. 4] do


  PrintFile("NFS_complexity_trials_method_2.txt",Sprintf("Family:␣%o\
    n\n", curves[j]));

  p := NextPrime(2^m[j]);

  count := m[j]+1000;

  repeat
    q := p^k[j];
    c := (Log(p)/(k[j]^(1/2)*(Log(Log(q)))))^(2/3); //find the c
    value such that p = L(q,a,c)

    C<x> := PolynomialRing(RealField());
    t_val := Roots(3*c*x*(x+1)^2-32);

    t1 := Floor(t_val[1][1]);
    t2 := Ceiling(t_val[1][1]); //the value of t used in the paper is
     sometimes rounded down or up, and may vary, this takes both
    variations into account
    c_dash1 := (8/3)*(3*t1/(4*(t1+1)))^(1/3);
    c_dash2 := (8/3)*(3*t2/(4*(t2+1)))^(1/3);

    NFS_complexity1 := L(q,1/3,c_dash1);
    NFS_complexity2 := L(q,1/3,c_dash2);  //the NFS complexities for
    the two possible values of t

    r := p^(1/rho[j]);
    Pollard_complexity := complexity_multiple[j]*r^(1/(2)); //the
    complexity of Pollard's rho method.

    ratio := [];
```

```
    ratio[1] := Pollard_complexity/NFS_complexity1;
    ratio[2] := Pollard_complexity/NFS_complexity2;
    ratio[3] := NFS_complexity1/Pollard_complexity;
    ratio[4] := NFS_complexity2/Pollard_complexity;

    for i in [1 .. 4] do  //check if the ratio of the complexities is
     small enough (here we set the ratio to be between 1 and 3)
      if Abs(ratio[i]) ge 1 then
        if Abs(ratio[i]) le bound then
          PrintFile("Pollard_NFS_complexity.txt",Sprintf("Log_2␣p␣:=␣
   %o\nSecurity␣Level␣:=␣%o\n\n", Log(p)/Log(2), Log(
   Pollard_complexity)/Log(2)));  \\record the size of p in bits and
    the complexity in a file
          break;
        end if;
      end if;
    end for;

    m[j] := m[j] + 1;
    p := NextPrime(2^m[j]);

  until m[j] eq count;

end for;
```

Appendix B

Final Exponentiation

We apply the method presented in Chapter 9 to some other selected families of pairing-friendly elliptic curves. In the following examples ℓ denotes the smallest common multiple of the denominators of the coefficients of the λ_i in the base $p(x)$ representation of $\Phi_k(p(x))/r(x)$. Some addition chains and polynomial parametrisations of $\ell\Phi_k(p)/r$ are not included for space and time reasons.

The $k = 8$ family of curves

We take $\ell = 6$ in this case and find the parameterisation of $\ell\Phi_8(p)/r = (p^4 + 1)/r$ to be:

$$\begin{aligned}
\lambda_3(x) &= 15x^2 + 30x + 75; \\
\lambda_2(x) &= 2x^5 + 4x^4 - x^3 + 26x^2 - 55x - 144; \\
\lambda_1(x) &= -5x^4 - 10x^3 - 5x^2 - 80x + 100; \\
\lambda_0(x) &= x^5 + 2x^4 + 7x^3 + 28x^2 + 10x + 108.
\end{aligned}$$

We find by brute-force computer search that we can construct the following optimal addition sequence which contains all the exponents in the above equations:

$$\{1, 2, 4, 5, 7, 10, 15, \underline{25}, 26, 28, 30, \underline{36}, \underline{50}, 55, 75, 80, 100, 108, 144\}.$$

As in Chapter 9, the underlined numbers are the extra numbers added in order to complete the sequence. We find that the vectorial addition chain derived from this addition sequence requires just 27 multiplications and 6 squarings to complete the calculation of the hard part of the final exponentiation.

The $k = 16$ family of curves

Taking $\ell = 14$ we find the parameterisation of $\ell\Phi_8(p)/r = (p^4 + 1)/r$ to be:

$$\begin{aligned}
\lambda_0(x) &= -11x^9 - 22x^8 - 55x^7 - 278x^5 - 1172x^4 - 1390x^3 + 1372; \\
\lambda_1(x) &= 15x^8 + 30x^7 + 75x^6 + 220x^4 + 1280x^3 + 1100x^2; \\
\lambda_2(x) &= 25x^7 + 50x^6 + 125x^5 + 950x^3 + 3300x^2 + 4750x; \\
\lambda_3(x) &= -125x^6 - 250x^5 - 625x^4 - 3000x^2 - 13000x - 15000; \\
\lambda_4(x) &= -2x^9 - 4x^8 - 10x^7 + 29x^5 - 54x^4 + 145x^3 + 4704; \\
\lambda_5(x) &= -20x^8 - 40x^7 - 100x^6 - 585x^4 - 2290x^3 - 2925x^2; \\
\lambda_6(x) &= 50x^7 + 100x^6 + 250x^5 + 1025x^3 + 4850x^2 + 5125x; \\
\lambda_7(x) &= 875x^2 + 1750x + 4375.
\end{aligned}$$

FST $k = 4$

For this family of curves, $\ell = 4$ and the base $p(x)$ representation of $\ell\Phi_k(p(x))/r(x)$ is given by:

$$\begin{aligned}
\lambda_0(x) &= x^3 - 2x^2 + x + 4; \\
\lambda_1(x) &= x^2 - 2x + 1.
\end{aligned} \tag{B.1}$$

Clearly, the coefficients already form a complete addition chain

$$\{1, 2, 4\}.$$

BLS $k = 6$ **/ FST** $k = 6$

Taking $\ell = 3$, the base $p(x)$ representation of $\ell\Phi_k(p(x))/r(x)$ is given by:

$$\begin{aligned} \lambda_0(x) &= x + 2; \\ \lambda_1(x) &= 1. \end{aligned} \tag{B.2}$$

Again, we immediately have a complete addition chain

$$\{1, 2\}.$$

KSS $k = 36$

For this family, the hard part of the final exponentiation is $111(p^k - 1)/r$ (that is, $\ell = 111$). We obtain:

$$\begin{aligned} \lambda_0(x) &= -87x^{13} + 348x^{12} - 609x^{11} - 25807x^7 + 83914x^6 - 180649x^5 + 151959; \\ \lambda_1(x) &= -385x^{12} + 1540x^{11} - 2695x^{10} - 128499x^6 + 428526x^5 - 899493x^4; \\ \lambda_2(x) &= -931x^{11} + 3724x^{10} - 6517x^9 - 333347x^5 + 1126706x^4 - 2333429x^3; \\ \lambda_3(x) &= -1029x^{10} + 4116x^9 - 7203x^8 - 433895x^4 + 1507142x^3 - 3037265x^2; \\ \lambda_4(x) &= 2401x^9 - 9604x^8 + 16807x^7 + 597849x^3 - 1858374x^2 + 4184943x; \\ \lambda_5(x) &= 16807x^8 - 67228x^7 + 117649x^6 + 5428661x^2 - 17983490x + 38000627; \\ \lambda_6(x) &= -62x^{13} + 248x^{12} - 434x^{11} - 25539x^7 + 88392x^6 - 178773x^5 - 1478520; \\ \lambda_7(x) &= 112x^{12} - 448x^{11} + 784x^{10} + 26075x^6 - 79436x^5 + 182525x^4; \\ \lambda_8(x) &= 882x^{11} - 3528x^{10} + 6174x^9 + 283073x^5 - 936488x^4 + 1981511x^3; \\ \lambda_9(x) &= 2744x^{10} - 10976x^9 + 19208x^8 + 949767x^4 - 3189900x^3 + 6648369x^2; \\ \lambda_{10}(x) &= 4802x^9 - 19208x^8 + 33614x^7 + 1817557x^3 - 6204184x^2 + 12722899x; \\ \lambda_{11}(x) &= 621859x^2 - 2487436x + 4353013. \end{aligned} \tag{B.3}$$

BLS $k = 24$ **& BLS** $k = 48$

For both the BLS $k = 24$ and the BLS $k = 48$ cases we have $\ell = 3$ and complete addition chains in the coefficients

$$\{1, 2, 3\}.$$

The base $p(x)$ representation of $\ell\Phi_k(p(x))/r(x)$ is given by:

BLS $k = 24$

$$\begin{aligned}
\lambda_0(x) &= x^9 - 2x^8 + x^7 - x^5 + 2x^4 - x^3 + 3; \\
\lambda_1(x) &= x^8 - 2x^7 + x^6 - x^4 + 2x^3 - x^2; \\
\lambda_2(x) &= x^7 - 2x^6 + x^5 - x^3 + 2x^2 - x; \\
\lambda_3(x) &= x^6 - 2x^5 + x^4 - x^2 + 2x - 1; \\
\lambda_4(x) &= x^5 - 2x^4 + x^3; \\
\lambda_5(x) &= x^4 - 2x^3 + x^2; \\
\lambda_6(x) &= x^3 - 2x^2 + x; \\
\lambda_7(x) &= x^2 - 2x + 1.
\end{aligned} \tag{B.4}$$

BLS $k = 48$

$$\begin{aligned}
\lambda_0(x) &= x^{17} - 2x^{16} + x^{15} - x^9 + 2x^8 - x^7 + 3; \\
\lambda_1(x) &= x^{16} - 2x^{15} + x^{14} - x^8 + 2x^7 - x^6; \\
\lambda_2(x) &= x^{15} - 2x^{14} + x^{13} - x^7 + 2x^6 - x^5; \\
\lambda_3(x) &= x^{14} - 2x^{13} + x^{12} - x^6 + 2x^5 - x^4; \\
\lambda_4(x) &= x^{13} - 2x^{12} + x^{11} - x^5 + 2x^4 - x^3; \\
\lambda_5(x) &= x^{12} - 2x^{11} + x^{10} - x^4 + 2x^3 - x^2; \\
\lambda_6(x) &= x^{11} - 2x^{10} + x^9 - x^3 + 2x^2 - x; \\
\lambda_7(x) &= x^{10} - 2x^9 + x^8 - x^2 + 2x - 1;
\end{aligned}$$

$$
\begin{aligned}
\lambda_8(x) &= x^9 - 2x^8 + x^7; \\
\lambda_9(x) &= x^8 - 2x^7 + x^6; \\
\lambda_{10}(x) &= x^7 - 2x^6 + x^5; \\
\lambda_{11}(x) &= x^6 - 2x^5 + x^4; \\
\lambda_{12}(x) &= x^5 - 2x^4 + x^3; \\
\lambda_{13}(x) &= x^4 - 2x^3 + x^2; \\
\lambda_{14}(x) &= x^3 - 2x^2 + x; \\
\lambda_{15}(x) &= x^2 - 2x + 1. \qquad \text{(B.5)}
\end{aligned}
$$

Appendix C

Cofactor Multiplication to obtain a point in $\mathbb{G}_2$

We apply the algorithm given in Chapter 10 to more families of pairing-friendly elliptic curves. Adapting the notation from above, ℓ denotes the smallest common multiple of the denominators of the coefficients of the g_i. We calculate $[\ell \cdot c(x)]\, P$ instead of $[c(x)]\, P$, for some (ℓ is coprime to r). Some addition chains are not included for space and time reasons.

KSS $k = 8$

Proceeding as above we obtain:

$$\begin{aligned} g_0(x) &= 2x^5 + 4x^4 - x^3 + 50x^2 + 65x - 36; \\ g_1(x) &= 2x^5 + 4x^4 - x^3 - 7x^2 - 25x + 75; \\ g_2(x) &= -15x^2 - 30x - 75. \end{aligned}$$

We take $\ell = 6$ and compute $[6 \cdot c(x)]\, P$. To complete the calculation we need an

addition sequence which includes all of the integer coefficients that arise here:

$$\{1, 2, 4, \underline{5}, \underline{6}, 7, \underline{10}, 15, 25, 30, 36, 50, 65, 75\},$$

where the underlined numbers are the extra numbers included to complete the sequence. The computation using this addition sequence can be completed with 18 point additions and 5 point doublings.

KSS $k = 16$

We compute $[98 \cdot c(x_0)]\, P$ using:

$$\begin{aligned}
g_0 &= 210x^9 + 1634x^8 - 2282x^7 + 224x^6 + 27167x^5 + 68296x^4 - 51681x^3 + 4592x^2 \\
&\quad +509096x + 932532; \\
g_1 &= -14x^9 - 168x^8 + 322x^7 - 924x^6 - 19509x^5 - 3689x^4 - 1239x^3 - 13307x^2 \\
&\quad -408016x - 344960; \\
g_2 &= -14x^9 - 168x^8 + 1953x^5 - 3689x^4 + 1715x^2 + 39795x + 344960; \\
g_3 &= -14x^9 - 168x^8 + 1953x^5 - 3689x^4 + 48125x + 30625; \\
g_4 &= 768x^9 - 156x^8 - 14x^7 + 5768x^6 + 31698x^5 + 5357x^4 + 7063x^3 + 118244x^2 \\
&\quad +337347x + 108125; \\
g_5 &= 154x^7 - 5068x^6 + 1666x^5 + 10997x^3 - 101199x^2 - 54537x + 29155; \\
g_6 &= -10045x^2 + 113190x + 32095.
\end{aligned}$$

BLS curves

In both BLS $k = 24$ and BLS $k = 48$ cases, we have $\ell = 3$ and so compute $[3 \cdot c(x)]\, P$ instead of $[c(x)]\, P$ to remove the inconvenient denominator. We use the following polynomials:

BLS $k = 24$

$$
\begin{aligned}
g_0 &= -2x^8 + 4x^7 - 3x^5 + 3x^4 - 2x^3 - 2x^2 + x + 4; \\
g_1 &= x^5 - x^4 - 2x^3 + 2x^2 + x - 1; \\
g_2 &= x^5 - x^4 - x + 1; \\
g_3 &= x^5 - x^4 - x + 1; \\
g_4 &= -3x^4 + x^3 + 4x^2 + x - 3; \\
g_5 &= 3x^3 - 3x^2 - 3x + 3; \\
g_6 &= -x^2 + 2x - 1.
\end{aligned} \tag{C.1}
$$

We use the complete addition chain:

$$\{1, 2, 3, 4\}.$$

BLS $k = 48$

$$
\begin{aligned}
g_0 &= -6x^{16} - 2x^{15} + 8x^{14} + 14x^{13} - 14x^{11} - 8x^{10} + 3x^9 + 11x^8 + 8x^7 \\
&\quad -14x^5 - 14x^4 + 8x^2 + 5x + 4; \\
g_1 &= 10x^{15} + 6x^{14} - 26x^{13} - 22x^{12} + 22x^{11} + 26x^{10} - 5x^9 - 11x^8 - 16x^7 \\
&\quad -24x^6 + 10x^5 + 46x^4 + 24x^3 - 16x^2 - 19x - 5; \\
g_2 &= -14x^{14} + 4x^{13} + 34x^{12} - 34x^{10} - 3x^9 + 13x^8 + 24x^6 + 26x^5 - 34x^4 \\
&\quad -56x^3 + 29x + 11; \\
g_3 &= 8x^{13} - 8x^{12} - 16x^{11} + 16x^{10} + 9x^9 - 9x^8 - 22x^5 - 10x^4 + 40x^3 \\
&\quad +24x^2 - 19x - 13; \\
g_4 &= -4x^{12} + 8x^{11} - 7x^9 + 3x^8 + 12x^4 - 4x^3 - 20x^2 + 3x + 9; \\
g_5 &= x^9 - x^8 - 4x^3 + 4x^2 + 3x - 3; \\
g_6 &= x^9 - x^8 - x + 1; \\
g_7 &= x^9 - x^8 - x + 1;
\end{aligned}
$$

$$
\begin{aligned}
g_8 &= -7x^8 - 13x^7 - 8x^6 + 14x^5 + 28x^4 + 14x^3 - 8x^2 - 13x - 7; \\
g_9 &= 21x^7 + 43x^6 + 6x^5 - 70x^4 - 70x^3 + 6x^2 + 43x + 21; \\
g_{10} &= -35x^6 - 55x^5 + 34x^4 + 112x^3 + 34x^2 - 55x - 35; \\
g_{11} &= 35x^5 + 29x^4 - 64x^3 - 64x^2 + 29x + 35; \\
g_{12} &= -21x^4 + x^3 + 40x^2 + x - 21; \\
g_{13} &= 7x^3 - 7x^2 - 7x + 7; \\
g_{14} &= -x^2 + 2x - 1.
\end{aligned}
$$

We construct the following addition chain from the coefficients of the g_i:

$$
\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 19, 20, 21, 22, 24, 26, 28, 29, 34, 35, 40, \\
43, 46, 55, 56, 64, 70, 112\}.
$$

Remarkably, the coefficients form a complete addition chain already.

Bibliography

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

[2] C. Arène, T. Lange, M. Naehrig, and C. Ritzenthaler. Faster Computation of the Tate Pairing. To appear in Journal of Number Theory. Cryptology ePrint Archive, Report 2009/155, 2009. `http://eprint.iacr.org/`.

[3] A. Atkin and F. Morain. Elliptic Curves and Primality Proving. *Mathematics of Computation*, 61(203):29 – 68, 1993.

[4] R. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman and Hall/CRC, 2006.

[5] D. Bailey and C. Paar. Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms. In H. Krawczyk, editor, *Advances in Cryptology – Crypto '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 472–485. Springer-Verlag, 1998.

[6] S. Baktir and B. Sunar. Optimal Tower Fields. *IEEE Transactions on Computers*, 53(10):1231–1243, October 2004.

[7] 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, 1998.

[8] P. S. L. M. Barreto, S. Galbraith, C. OhEigeartaigh, and M. Scott. Efficient Pairing Computation on Supersingular Abelian Varieties. *Designs, Codes and Cryptography*, 42:239–271, 2007. `http://eprint.iacr.org/2004/375`.

[9] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In M. Yung, editor, *Advances in Cryptology – Crypto 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag, 2002.

[10] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing Elliptic Curves with Prescribed Embedding Degrees. In C. Galdi and G. Persiano, editors, *Security in Communication Networks – SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 263–273. Springer-Verlag, 2002.

[11] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer-Verlag, 2006.

[12] N. Benger, M. Charlemagne, and D. Freeman. On the security of pairing-friendly abelian varieties over non-prime fields. In H. Shacham and B. Waters, editors, *Pairing-Based Cryptography – Pairings 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 52–65. Springer-Verlag, 2009.

[13] N. Benger and M. Scott. Constructing Tower Extensions for the implementation of Pairing-Based Cryptography, 2010.

[14] I. F. Blake, G. Seroussi, and N. P. Smart, editors. *Advances in Elliptic Curve Cryptography*. Number 317 in London Mathematical Society Lecture Note Series. Cambridge University Press, 2005.

[15] D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. In J. Kilian, editor, *Advances in Cryptology – Crypto 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 213–229, London, UK, 2001. Springer-Verlag.

[16] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In C. Boyd, editor, *Advances in Cryptology – Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001.

[17] J. Bos and M. Coster. Addition chain heuristics. In G. Brassard, editor, *Advances in Cryptology – Crypto '89*, volume 434 of *Lecture Notes in Computer Science*, pages 400–407, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[18] R. H. Brown and A. Prabhakar. Digital Signature Standard (DSS). `http://www.itl.nist.gov/fipspubs/fip186.htm`.

[19] C. Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In B. Honary, editor, *Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.

[20] H. Cohen. *A Course in Computational Algebraic Number Theory*. Number 138 in Graduate Texts in Mathematics. Springer-Verlag, Berlin, 1993.

[21] C. Costello, C. Boyd, J. M. González-Nieto, and K. K.-H. Wong. Avoiding full extension field arithmetic in pairing computations. In D. J. Bernstein and T. Lange, editors, *Progress in Cryptology – Africacrypt 2010*, volume 6055 of *Lecture Notes in Computer Science*, pages 203–224. Springer-Verlag, 2010.

[22] B. den Boer. Diffie-Hellman is as strong as Discrete Log for certain primes. In S. Goldwasser, editor, *Advances in Cryptology – Crypto '88*, volume 403 of *Lecture Notes in Computer Science*, pages 530–539, 1988.

[23] A. J. Devegili, M. Scott, and R. Dahab. Implementing cryptographic pairings over Barreto-Naehrig curves. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto,

editors, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 197–207. Springer-Verlag, 2007.

[24] W. Diffie and M. E. Hellman. New Directions in Cryptography. In *Information Theory, IEEE Transactions on Information Theory*, volume 22, pages 644–654, 1976.

[25] B. Dodson and A. K. Lenstra. NFS with four large primes: An Explosive Experiment. In D. Coppersmith, editor, *Advances in Cryptology – Crypto '95*, pages 372–385, London, UK, 1995. Springer-Verlag.

[26] L. J. Dominguez Perez and M. Scott. Automatic Generation of Optimised Cryptographic Pairing Functions. pages 55–71, 2009.

[27] P. Downey, B. Leony, and R. Sethi. Computing sequences with addition chains. *Siam Journal of Computing*, 3:638–696, 1981.

[28] S. Duquesne and G. Frey. Background on pairings. In *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, pages 115–124. Chapman & Hall/CRC, Boca Raton, FL, 2006.

[29] I. Duursma, P. Gaudry, and F. Morain. Speeding up the discrete log computation on curves with automorphisms. In K.-Y. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology – Asiacrypt '99*, pages 103–121. Springer-Verlag, 1999.

[30] I. Duursma and H. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In C.-S. Laih, editor, *Advances in Cryptology – ASIACRYPT '03*, Lecture Notes in Computer Science, pages 111–123. Springer-Verlag, 2003.

[31] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – Crypto '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[32] D. Freeman, M. Scott, and E. Teske. A Taxonomy of Pairing Friendly Elliptic Curves. *Journal of Cryptology*, 23(2):224–280, 2010. Also available from `http://eprint.iacr.org/2006/372`.

[33] G. Frey, M. Müller, and H. Rück. The Tate pairing and the Discrete Logarithm applied to Elliptic Curve Cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717 – 1718, 1999.

[34] G. Frey and H. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62(206):865–874, 1994.

[35] S. Galbraith, X. Lin, and M. Scott. Endomorphisms for Faster Elliptic Curve Cryptography on a Large Class of Curves. In A. Joux, editor, *Advances in Cryptology – Eurocrypt 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 518–535. Springer-Verlag, 2009.

[36] S. Galbraith and M. Scott. Exponentiation in Pairing-Friendly Groups using Homomorphisms. In S. Galbraith and K. Patterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 211–224. Springer-Verlag, 2008.

[37] P. Gaudry. Index Calculus for Abelian Varieties and the Elliptic Curve Discrete Logarithm Problem. *Journal of Symbolic Computation*, (44):1690–1702, 2009. Also available at `http://www.loria.fr/~gaudry/publis/indexcalc.pdf`.

[38] P. Gaudry, F. Hess, and N. P. Smart. Constructive and Destructive facets of Weil descent on Elliptic Eurves. *Journal of Cryptology*, 15(1):19–46, 2002.

[39] D. M. Gordon and K. S. McCurley. Massively Parallel Computation of Discrete Logarithms. In E. F. Brickell, editor, *Advances in Cryptology – Crypto '92*, pages 312–323, London, UK, 1993. Springer-Verlag.

[40] R. Granger, D. Page, and N. Smart. High Security Pairing-Based Cryptography Revisited. In F. Hess, S. Pauli, and M. Pohst, editors, *Algorithmic Number Theory Symposium – ANTS VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 480–494. Springer-Verlag, 2006.

[41] R. Granger, D. Page, and M. Stam. On Small Characteristic Algebraic Tori in Pairing Based Cryptography. *LMS Journal of Computation and Mathematics*, 9:64–85, 2006.

[42] R. Granger and M. Scott. Faster Squaring in the Cyclotomic Subgroup of Sixth Degree Extensions. In *Public Key Cryptography – PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 209–223, 2010.

[43] D. Hankerson, A. Menezes, and D. Scott. Chapter XII: Software Implementation of Pairings. In M. Joye and G. Neven, editors, *Identity-Based Cryptography*, volume 2 of *Cryptology and Information Security Series*. IOS Press, 2009. Available at `http://www.cacr.math.uwaterloo.ca/`.

[44] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[45] L. Hei, J. Dong, and D. Pei. Implementation of Cryptosystems based on Tate Pairing. *J. Comput. Sci & Technolgy*, 20(2):264–269, 2005.

[46] F. Hess, N. Smart, and F. Vercauteren. The Eta Pairing Revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

[47] L. Hitt. On the minimal embedding field. In *Pairing-Based Cryptography — Pairing 2007*, volume 4575 of *Springer LNCS*, pages 294–301, 2007.

[48] M. J. Jacobson, N. Koblitz, J. H. Silverman, A. Stein, and E. Teske. Analysis of the xedni calculus attack. *Des. Codes Cryptography*, 20(1):41–64, 2000.

[49] A. Joux. A One Round Protocol for Tripartite Diffie – Hellman. In *Journal of Cryptology*, volume 17, pages 263–276, 2004.

[50] A. Joux and R. Lercier. The Function Field Sieve in the Medium Prime Case. In S. Vaudenay, editor, *Advances in Cryptology – Eurocrypt 2006*, number 4004 in Lecture Notes in Computer Science, pages 254–270, 2006.

[51] A. Joux, R. Lercier, N. Smart, and F. Vercauteren. The Number Field Sieve in the Medium Prime Case. In C. Dwork, editor, *Advances in Cryptology – Crypto 2006*, number 4117 in Lecture Notes in Computer Science, pages 323–341, 2006.

[52] E. Kachisa, E. Schaefer, and M. Scott. Constructing Brezing-Weng Pairing-Friendly Elliptic Curves using elements in the Cyclotomic Field. In S. Galbraith and K. Patterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 126–135. Springer-Verlag, 2008.

[53] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, 9:5–38, 1883.

[54] N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203 – 209, 1987.

[55] N. Koblitz. Good and bad uses of Elliptic Curves in Cryptography. *Mosc. Math. J.*, 2:693–715, 2002.

[56] N. Koblitz and A. Menezes. Pairing-Based Cryptography at high security levels. *IMA Int. Conf.*, 3796:13 – 36, 2005.

[57] N. Koblitz and A. Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. *Journal of Mathematical Cryptology*, pages 311 – 326, 2008.

[58] B. A. LaMacchia and A. M. Odlyzko. Solving large sparse linear systems over finite fields. In A. Menezes and S. A. Vanstone, editors, *Advances in Cryptology - Crypto 1990*, number 537 in Lecture Notes in Computer Science, pages 109–133. Springer-Verlag, 1990.

[59] S. Lang. *Elliptic Curves: Diophantine Analysis*, volume 231 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1978.

[60] S. Lang. *Algebra*, volume 211 of *Graduate Texts in Mathematics*. Springer, New York, revised third edition, 2002.

[61] E. Lee, H-S. Lee, and C-M. Park. Efficient and Generalized Pairing Computation on Abelian Varieties. Cryptology ePrint Archive, Report 2008/040, 2008. `http://eprint.iacr.org/2008/040`.

[62] F. Lemmermeyer. *Reciprocity Laws: From Euler to Eisenstein*. Springer Monographs in Mathematics. Springer-Verlag, 2000.

[63] A. K. Lenstra. Unbelievable Security: Matching AES Security Using Public Key Systems. In C. Boyd, editor, *Advances in Cryptology – Asiacrypt '01*, volume 2248 of *Lecture Notes in Computer Science*, pages 67–86, London, UK, 2001. Springer-Verlag.

[64] A. K. Lenstra and H. W. Lenstra Jr., editors. *The development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993.

[65] A. K. Lenstra and E. R. Verheul. Selecting Cryptographic Key Sizes. In H. Imai and Y. Zheng, editors, *Public-Key Cryptography 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 446–465, London, UK, 2000. Springer-Verlag.

[66] R. Lidl and H. Niederreiter. *Finite Fields*. Number 20 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, UK, 2nd edition, 1997.

[67] S. Matsuda, N. Kanayama, F. Hess, and E. Okamoto. Optimised versions of the Ate and Twisted Ate Pairings. In S. Galbraith, editor, *Cryptography and Coding*, volume 4887 of *Lecture Notes in Computer Science*, pages 302–312. Springer, 2007.

[68] A. Menezes. *Elliptic Curve Public Key Cryptosystems*, volume 234 of *series in engineering and computer science, SECS*. Kluwer Academic Publishers, 1993.

[69] A. Menezes, T. Okamoto, and S. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. In *Symposium on Theory of Computing – STOC '91*, pages 80–89, New York, NY, USA, 1991. ACM. `http://doi.acm.org/10.1145/103418.103434`.

[70] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, Florida, 1996. URL: `http://cacr.math.uwaterloo.ca/hac`.

[71] V. S. Miller. Use of Elliptic Curves in Cryptography. In H. C. Williams, editor, *Advances in Cryptology - Crypto '85*, number 218 in Lecture Notes in Computer Science, page 417. Springer, 1985.

[72] V. S. Miller. Short Programs for Functions on Curves. unpublished manuscript, 1986.

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

[74] J. S. Milne. Abelian Varieties. In G. Gornell and J. Silverman, editors, *Arithmetic Geometry*, pages 103–150, New York, 1986. Springer.

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

[76] Y. Nogami, M. Akane, Y. Sakemi, H. Kato, and Y. Morikawa. Integer variable X-based Ate Pairing. In S. Galbraith and K. Patterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 178–191. Springer-Verlag, 2008.

[77] J. Olivos. On vectorial addition chains. *Journal of Algorithms*, 2:13–21, 1981.

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

[79] K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In M. Yung, editor, *Advances in Cryptology – Crypto 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 336–353, 2002.

[80] K. Rubin and A. Silverberg. Using abelian varieties to improve pairing-based cryptography. *Journal of Cryptology*, 22(3):330–364, 2009.

[81] O. Schirokauer. Using Number Fields to Compute Logarithms in Finite Fields. *Mathematics of Computation*, 231(69):1267–1283, 2000. http://www.jstor.org/stable/2585027.

[82] O. Schirokauer. The Number Field Sieve for integers of low weight. Cryptology ePrint Archive, Report 2006/107, 2006. `http://eprint.iacr.org/`.

[83] B. Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[84] M. Scott and P. Barreto. Compressed Pairings. In M. K. Franklin, editor, *Advances in Cryptology – Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156. Springer-Verlag, 2004. Also available from `http://eprint.iacr.org/2004/032/`.

[85] M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez, and E. J. Kachisa. Fast Hashing to G_2 on Pairing-Friendly Curves. In H. Shacham and B. Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 102–113. Springer, 2009.

[86] M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez, and E. J. Kachisa. On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves.

In H. Shacham and B. Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 78–88. Springer, 2009.

[87] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – Crypto '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, 1984.

[88] M. Shirase. Universally Constructing 12-th Degree Extension Field for Ate Pairing. Cryptology ePrint Archive, Report 2009/623, 2009. `http://eprint.iacr.org/`.

[89] M. Shirase. Barreto-Naehrig Curve With Fixed Coefficient - efficiently constructing pairing-friendly curves -. Cryptology ePrint Archive, Report 2010/134, 2010. `http://eprint.iacr.org/`.

[90] J. H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, New York, 1986.

[91] J. H. Silverman. The Xedni Calculus and the Elliptic Curve Discrete Logarithm Problem. *Designs, Codes and Cryptography*, 20(1):5–40, 2000.

[92] J. H. Silverman and J. Suzuki. Elliptic Curve Discrete Logarithms and the Index Calculus. In K. Ohta and D. Pei, editors, *Advances in Cryptology – Asiacrypt '98*, pages 110–125, London, UK, 1998. Springer-Verlag.

[93] J. H. Silverman and J. Tate. *Rational Points on Elliptic Curves*, volume 106 of *Undergraduate Texts in Mathematics*. Springer, New York, 1992.

[94] M. Stam and A. K. Lenstra. Efficient Subgroup Exponentiation in Quadratic and Sixth Degree Extensions. In *CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 318–332. Springer-Verlag, 2002.

[95] D. R. Stinson. *Cryptography: Theory and Practice, Second Edition*. Chapman & Hall/CRC, 2002.

[96] A. Sutherland. Record CM constructions of elliptic curves, 2010. http://www-math.mit.edu/~drew/CMRecords.html.

[97] E. Thomé. Computation of Discrete Logorithms in GF(2^{607}). NMBRTHRY mailing list, Feb 2002.

[98] F. Vercauteren. Optimal Pairings. *IEEE Transactions on Information Theory*, (56):455–461, 2010. http://eprint.iacr.org/2008/096.

[99] E. R. Verheul. Evidence that XTR is more secure than Supersingular Elliptic Curve Cryptosystems. *Journal of Cryptology*, 17(4):277–296, 2004.

[100] L. C. Washington. *Introduction to Cyclotomic Fields*, volume 83 of *Graduate Texts in Mathematics*. Springer-Verlag, 1997.

[101] P. Zajac. Basic remarks on the NFS complexity. Cryptology ePrint Archive, Report 2008/064, 2008. http://eprint.iacr.org/.