Implementing the Schoof-Elkies-Atkin Algorithm with NTL

by

Yik Siong Kok

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

Waterloo, Ontario, Canada, 2013

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

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

Abstract

In elliptic curve cryptography, cryptosystems are based on an additive subgroup of an elliptic curve defined over a finite field, and the hardness of the Elliptic Curve Discrete Logarithm Problem is dependent on the order of this subgroup. In particular, we often want to find a subgroup with large prime order. Hence when finding a suitable curve for cryptography, counting the number of points on the curve is an essential step in determining its security.

In 1985, René Schoof proposed the first deterministic polynomial-time algorithm for point counting on elliptic curves over finite fields. The algorithm was improved by Noam Elkies and Oliver Atkin, resulting in an algorithm which is sufficiently fast for practical purposes. The enhancements leveraged the arithmetic properties of the lth classical modular polynomial, where l is either an Elkies or Atkin prime. As the Match-Sort algorithm relating to Atkin primes runs in exponential time, it is eschewed in common practice.

In this thesis, I will discuss my implementation of the Schoof-Elkies-Atkin algorithm in C++, which makes use of the NTL package. The implementation also supports the computation of classical modular polynomials via isogeny volcanoes, based on the methods proposed recently by Bröker, Lauter and Sutherland.

Existing complexity analysis of the Schoof-Elkies-Atkin algorithm focuses on its asymptotic performance. As such, there is no estimate of the actual impact of the Match-Sort algorithm on the running time of the Schoof-Elkies-Atkin algorithm for elliptic curves defined over prime fields of cryptographic sizes. I will provide rudimentary estimates for the largest Elkies or Atkin prime used, and discuss the variants of the Schoof-Elkies-Atkin algorithm using their run-time performances.

The running times of the SEA variants supports the use Atkin primes for prime fields of sizes up to 256 bits. At this size, the selective use of Atkin primes runs in half the time of the Elkies-only variant on average. This suggests that Atkin primes should be used in point counting on elliptic curves of cryptographic sizes.

Acknowledgements

I would like to thank my supervisor David Jao for his guidance. I would also like to thank Alfred Menezes and Eric Katz for their comments.

Dedication

This is dedicated to my wife Ling Wen Wan.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In elliptic curve cryptography (ECC), cryptosystems are based on an additive subgroup of an elliptic curve defined over a finite field, and the hardness of the Elliptic Curve Discrete Logarithm Problem is dependent on the order of this subgroup. In particular, we often want to find a subgroup with large prime order. Hence, when finding a suitable curve for cryptography, counting the number of points on the curve is an essential step in determining its security.

In 1985 [17], René Schoof proposed the first deterministic polynomial-time algorithm for point counting on elliptic curves over finite fields, using the Chinese Remainder Theorem (CRT) to break the problem down into subproblems for a set of (much smaller) primes, which can be solved independently. Though the algorithm runs in polynomial-time, it was on the order of $O(\log^8 p)$ bit operations, which for elliptic curves defined over field of cryptographic sizes was still too slow for practical purposes at the time.

Improvements to Schoof's algorithm were independently introduced by Noam Elkies and Oliver Atkin resulting in an algorithm which, with a running time in the order of $O(\log^6 p)$, is now fast enough for curves in use for ECC. Using a different approach, Satoh [15] developed a polynomial-time algorithm for point counting on curves over finite fields of small characteristic.

1.2 Motivation

1.2.1 Existing Point Counting Implementations

There are various implementations of the SEA algorithm for which source code is freely available online, such as that in GP/PARI (used in SAGE), and Mike Scott's implementation in C/C++ as part of the MIRACL package. The algorithm has also been implemented in MAGMA, although its source code is not readily available.

The existing C/C++ MIRACL implementation of SEA makes use of lower-level optimizations provided by the MIRACL library to provide very good running times. It also outputs the intermediate values during its execution, which allows for comparisons of the choice of primes. One observation is that the implementation eschews the consideration of a subset of small primes that Atkin uses, instead choosing larger primes. While the larger primes translate to more expensive field arithmetic, it may be more than offset by computation time savings elsewhere, which reasonably explains the choice of approach. This approach is also reflected in Sutherland's point counting record for elliptic curves over finite fields (`http://math.mit.edu/~drew/SEArecords.html`).

Nicole Pitcher [14] proposed the use of Schönhage-Strassen's algorithm [16], the fastest integer multiplication method for integers beyond $2^{2^{15}}$, to speed up multiplication of polynomials over finite fields in Schoof's algorithm. This is done by lifting polynomials from $\mathbb{F}_p[x]$ to $\mathbb{Z}[x]$, and mapping them into the set of integers by substituting x with $256^{\lceil \frac{\lg p}{4} + \frac{\lg N}{8} \rceil}$. The resulting integers will be sufficiently large for Schönhage-Strassen's algorithm to provide speedups, thus improving the performance of Schoof's algorithm.

1.2.2 Project Aims

As with algorithms in general, there are often gaps in understanding that can only be filled by the actual process of implementing the algorithm. Our implementation serves as a tool to investigate why the MIRACL implementation diverges from Atkin's original approach for Atkin primes, and if there are possible improvements to support the use of Atkin's approach.

The aim is to provide an implementation of SEA, with the use of some Atkin primes, that can perform better than an Elkies prime-only approach for curves of cryptographic sizes.

1.3 Author's Contributions

Computing Classical Modular Polynomials. I have implemented the complex analytical approach to compute Hilbert class polynomials, as a precursor to computing classical modular polynomials via isogeny volcanoes, as proposed in [2]. Using MAGMA for verification, I wrote the routines for finding suitable parameters for prime l, which then used the parameters to construct isogeny volcanoes, as well as compute the l-th classical modular polynomial in $\mathbb{Z}[X]$.

Schoof-Elkies-Atkin. The modular polynomials are subsequently used in the implementation of the Schoof-Elkies-Atkin algorithm for elliptic curves over prime fields with characteristic larger than 3. In the implementation of the Match-Sort algorithm, I proposed sorting by absolute values and performing scalar multiplication iteratively to reduce the size of the scalar multiplications in the routine.

Complexity of Match-Sort. In the existing literature, the complexity analysis of the SEA algorithm focuses on the asymptotic performance. As such, there is no estimate of the size of the Match-Sort problem and how much it impacts the overall running time of the SEA algorithm. I have provided some rudimentary estimates of the size of the largest prime used in SEA and the number of cases to be checked via Match-Sort. The run-time performance of the variants of SEA is included as a basis for understanding the size of the Match-Sort problem.

The running times of the SEA variants supports the use Atkin primes for prime fields of sizes up to 256 bits. At this size, the selective use of Atkin primes runs in half the time of the Elkies-only variant on average. This suggests that Atkin primes should be used in point counting on elliptic curves of cryptographic sizes.

1.4 Organization of Thesis

In Chapter 2, I will explain the ideas behind in Schoof's algorithm, and give an overview of the Schoof-Elkies-Atkin algorithm. In Chapter 3, I will discuss isogeny volcanoes and their role in the computation of classical modular polynomials. In Chapter 4, I will discuss the implementation in the cases of Elkies and Atkin primes. I will also provide a detailed description of the Match Sort algorithm. This allows us to describe the SEA algorithm in full. In Chapter 5, I will compare the run-time performance of the implementation, using the different variants of SEA.

Chapter 2

Schoof and SEA

In this chapter we provide a few preliminary definitions about elliptic curves over finite fields, leading up to the description of Schoof's algorithm. Though the material in this section can be found in most textbooks on elliptic curve cryptography, our treatment is based on Blake et al. [1] as it contains a more detailed discussion of the Schoof-Elkies-Atkin algorithm.

2.1 Preliminaries

2.1.1 Elliptic Curves

Definition 2.1.1. An **elliptic curve** E, defined over a finite field $\mathbb{F}_q$ with prime characteristic p, is the set of points (i.e. solutions) $(x, y) \in \mathbb{F}_q^2$ satisfying the equation

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

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_q$. As the equation defines E, we will equate the definition of E with that of the equation. Two elliptic curves E (with variables x, y) and E' (with variables x', y') defined over $\mathbb{F}_q$ are **isomorphic** over $\mathbb{F}_q$ if and only if there exists constants $r, s, t, u \in \mathbb{F}_q$ with $u \neq 0$, such that the change of variables

$$x = u^2x' + r, \qquad y = u^3y' + su^2y' + t \tag{2.2}$$

transforms E into E'.

For $p > 3$, the equation above can be replaced by

$$E : y^2 = x^3 + ax + b, \tag{2.3}$$

for some $a, b \in \mathbb{F}_q$. We call this the Weierstrass form of E. We say that E is **singular** if $4a^3 + 27b^2 = 0$. For the rest of this report, we assume $p > 3$, and E is assumed to be non-singular, and hence can be described by its Weierstrass form.

We write $E/\mathbb{F}_q$ to denote that the coefficients of E are in $\mathbb{F}_q$ and write $E(\mathbb{F}_q)$ to denote the points of E in $\mathbb{F}_q$. Although the coefficients of E may be in a field extension and in practice for $p > 3$ we usually have $q = p$, we nevertheless distinguish between the notations here.

2.1.2 Group Law

Definition 2.1.2. For two points $P := (x_1, y_1), Q := (x_2, y_2)$ on $E(\mathbb{F}_q)$, we define the binary operation **point addition** $\oplus$ so that $P \oplus Q := (x_3, y_3)$ is given by

$$m : = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if} \quad x_1 \neq x_2 \\ \frac{3x_1^2 + b}{2y_1} & \text{if} \quad x_1 = x_2, y_1 = y_2 \end{cases}, \tag{2.4}$$

$$x_3 = m^2 - a - x_2 - x_1, \tag{2.5}$$

$$y_3 = m(x_1 - x_3) - y_1, \tag{2.6}$$

Note that when $x_1 = x_2, y_1 \neq y_2$, we have $y_2 = -y_1$. In this case we write $-P := Q$, and define

$$\infty_E := P \oplus (-P),$$

which is referred to as the point at infinity. When it is unambiguous, we will use ∞ instead. We also define

$$P \oplus \infty = \infty \oplus P = P$$

and

$$P \ominus Q := P \oplus (-Q).$$

For $n \in \mathbb{N}$, we define $nP := P \oplus \cdots \oplus P$ (n copies of P). If $n = -n'$ is negative, then $nP = -(n'P)$. Thus for $n \in \mathbb{Z}$,

$$[n] : E \longrightarrow E, \qquad [n](P) = nP \tag{2.7}$$

is the multiply-by-n function.

Theorem 2.1.3. *$(E, \oplus)$ is an abelian group, with ∞ as the identity element and $-P$ as the inverse element for each point $P \in E$.*

2.1.3 Frobenius Endomorphism

Definition 2.1.4. For the finite field $\mathbb{F}_q$ with algebraic closure $\overline{\mathbb{F}}_q$, the Frobenius map ϕ_q is defined as

$$\phi_q : \overline{\mathbb{F}}_q \longrightarrow \overline{\mathbb{F}}_q, \qquad \phi_q(x) = x^q \tag{2.8}$$

Proposition 2.1.5. *The Frobenius map ϕ_q is an automorphism on $\overline{\mathbb{F}}_q$, and $\phi_q(x) = x$ if and only if $x \in \mathbb{F}_q$.*

We can extend the definition of the Frobenius map to elliptic curves over $\mathbb{F}_q$:

$$\phi_q : E(\overline{\mathbb{F}}_q) \longrightarrow E(\overline{\mathbb{F}}_q), \qquad \phi_q(x, y) = (\phi_q(x), \phi_q(y)) = (x^q, y^q), \quad \phi_q(\infty) = \infty$$

Proposition 2.1.6. *ϕ_q is a group endomorphism on $E(\mathbb{F}_q)$.*

Theorem 2.1.7. *The characteristic equation of ϕ_q on E is:*

$$Z^2 - t \cdot Z + q = 0 \tag{2.9}$$

for some integer t. In other words, for all $(x, y) \in E(\overline{\mathbb{F}}_q)$,

$$\left(x^{q^2}, y^{q^2}\right) - t(x^q, y^q) + q(x, y) = \infty, \tag{2.10}$$

*where addition here is point addition on E. t is called the **trace** of the Frobenius endomorphism ϕ_q.*

2.1.4 Number of Points

Theorem 2.1.8. *The number of points (x, y) on E, denoted as $\#E(\mathbb{F}_q)$, is given by*

$$\#E(\mathbb{F}_q) = q + 1 - t, \tag{2.11}$$

where t is the trace of the Frobenius endomorphism.

Definition 2.1.9. Let $E : y^2 = x^3 + ax + b$ denote an elliptic curve over $\mathbb{F}_q$. The **quadratic twist** of E, is the elliptic curve over $\mathbb{F}_q$ given by $E' : dy^2 = x^3 + ax + b$, for some non-square $d \in \mathbb{F}_q$. If $\#E(\mathbb{F}_q) = q + 1 - t$, then $\#E'(\mathbb{F}_q) = q + 1 + t$.

Theorem 2.1.10. (Hasse's Theorem) *The value of t is bounded as follows[20]:*

$$|t| = |q + 1 - E(\mathbb{F}_q)| \leqslant 2\sqrt{q}. \tag{2.12}$$

The interval $[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$ is known as the Hasse interval.

Definition 2.1.11. An integer m is called a **quadratic residue** modulo n if

$$x^2 \equiv m \pmod{n}$$

for some integer x. Otherwise it is called a **quadratic nonresidue** modulo n. We use QR_n and $\overline{\mathrm{QR}}_n$ to denote the set of quadratic residues and nonresidues modulo n respectively.

We can use the quadratic character to count the number of points, but this yields an exponential time algorithm, thus only feasible for small fields.

Definition 2.1.12. For each $x \in \mathbb{F}_q$, the **quadratic character** $\chi_q(x)$ is

$$\chi_q(x) = \begin{cases} 1 & \text{if} \quad x \text{ is a square in } \mathbb{F}_q \\ 0 & \text{if} \quad x = 0 \text{ in } \mathbb{F}_q \\ -1 & \text{if} \quad x \text{ is not a square in } \mathbb{F}_q \end{cases} \tag{2.13}$$

From the equation of the Weierstrass form of an elliptic curve, we observe that, besides the point at infinity, a point $(x, y) \in \mathbb{F}_q^2$ is a point on E if and only if $x^3 + ax + b$ is a square in $\mathbb{F}_q$ (y is a corresponding square root). Note that when q is a prime, then

$$\chi_q(x) = \left(\frac{x}{q}\right). \tag{2.14}$$

If $x^3 + ax + b = 0$ in $\mathbb{F}_q$, then y can only take one value, 0. If $\chi_q(x^3 + ax + b) = 1$, then $y = \pm\sqrt{x^3 + ax + b}$, which are two distinct values since the characteristic p is odd. If $\chi_q(x^3 + ax + b) = -1$, then there are no possible values of y for which $(x, y) \in E$. Hence

$$\#E(\mathbb{F}_q) = 1 + \sum_{x \in \mathbb{F}_q} \big(1 + \chi_q(x^3 + ax + b)\big) \tag{2.15}$$

For small values of q, this formula is very efficient. Unfortunately as this approach requires a computation over all the elements in $\mathbb{F}_q$, it is exponential in the length of the size of $\mathbb{F}_q$, and thus becomes impractical very quickly as the size of the field increases.

2.1.5 Baby-Step Giant-Step

In an elliptic curve discrete logarithm problem, we are given an elliptic E with $N := \#E(F_q)$, two points $P, Q \in E$, and that $Q = [m]P$, for some integer $0 \leqslant m < N$. Shanks' Baby-Step Giant-Step (BSGS) algorithm first computes and stores a table of 'baby steps'; the values

$$R_b := Q - [b]P,$$

with b ranging from 0 to $\lceil\sqrt{N}\rceil - 1$. Next, the 'giant steps'

$$S_a := [a]\left([\lceil\sqrt{N}\rceil]P\right)$$

are computed, incrementally from $a = 0$. Whenever we find that S_a matches with some R_b in the table, we will have found

$$Q - [b]P = [a]\left([\lceil\sqrt{N}\rceil]P\right)$$

which allows us to compute in $O(\sqrt{N})$ computations, the value of

$$m = a\lceil\sqrt{N}\rceil + b.$$

Shanks-Mestre. To find N, the order of the elliptic curve group, we begin with a random point $P \in E(\mathbb{F}_q)$, and compute its order in the group o_P. It is known that an elliptic curve group is isomorphic to the direct product of either one or two cyclic groups, and that we can expect to find a point P with its order in the elliptic curve group $o_P > 4\sqrt{q}$ [17]. We define the points

$$Q = [q+1]P \qquad \text{and} \qquad Q_1 = Q + [\lfloor 2\sqrt{q}\rfloor]P,$$

and let $t' = t + \lfloor 2\sqrt{q}\rfloor$. Note that $t' \in [0, 4\sqrt{q}]$ by the Hasse bound 2.12. Since

$$Q_1 = [q+1]P + [\lfloor 2\sqrt{q}\rfloor]P = [q+1-t]P + [t + \lfloor 2\sqrt{q}\rfloor]P = [N]P + [t']P = [t']P,$$

and $o_P > 4\sqrt{q}$, solving the discrete log problem $Q_1 = [t']P$ by BSGS gives us the exact value of t'. From this, we can compute the value of t and thus the order of the elliptic curve group. This algorithm, referred to the Shanks-Mestre algorithm, requires $O(q^{1/4+\epsilon})$ computations and storage, for some arbitrarily small $\epsilon > 0$.

2.2 Schoof's Algorithm

In 1985, Schoof proposed the first polynomial-time algorithm for point counting on elliptic curves, by computing $t = \#E(\mathbb{F}_p) - p - 1$ via a series of modular computations. Schoof's approach is to find the value of $t_l = t \bmod l$, for $l \in S$, where $S = \{2, 3, 5, \cdots, L\}$ is a set of primes. When S is chosen so that $m_S := \prod_{l \in S} l > 4\sqrt{p}$, by the Chinese Remainder Theorem (CRT) we can solve for $t' = t \bmod m_S$ uniquely.

Since $m_r > 4\sqrt{p}$, we can determine t by finding the only value in the Hasse interval which is congruent to $t' \bmod m_S$. It can be shown using the prime number theorem that the number of primes required is roughly $O(\log p)$ (see Section 4.4.3).

In the case $l = 2$, and computing t_l means checking if N is even or odd, which is equivalent to asking if $E(\mathbb{F}_p)$ has a point of order 2. From Definition 2.4, a point of order 2 exists if and only if $x^3 + ax + b$ has roots in $\mathbb{F}_p$, which is true if and only if $\gcd(x^p - x, x^3 + ax + b) \neq 1$ in the polynomial ring $\mathbb{F}_p[x]$.

Let ModularExponentiation$(a(X), e, f(X))$ denote the function which returns $a(X)^e \bmod f(X)$. To compute the gcd above, we observe that

$$\gcd(x^p - x, x^3 + ax + b) = \gcd(x^p - x \bmod x^3 + ax + b, x^3 + ax + b). \tag{2.16}$$

We compute $x^p \bmod x^3 + ax + b =$ ModularExponentiation$(X, p, x^3 + ax + b)$, a polynomial of degree at most two, and subtract x from the result. This reduces the gcd computation (by the Euclidean algorithm) to one involving two polynomials of low degree, which can be performed very efficiently.

Modular exponentiation can be performed in time $O(\log p)$ by square-and-multiply. However, as p is fixed regardless of prime l, we can incur some computational overhead to find an addition chain for p, and possibly use the chain to compute the modular exponentiation more efficiently than square-and-multiply. Chapter 9 of [5] provides an overview of how addition chains can be computed and used.

2.2.1 Division Polynomials

Definition 2.2.1. For each non-negative integer l, the **l-th division polynomial** ψ_l over $\mathbb{F}_p[x]$ is defined recursively as follows:

$$\begin{aligned}
\psi_0 &= 0 \\
\psi_1 &= 1 \\
\psi_2 &= 2y \\
\psi_3 &= 3x^4 + 6ax^2 + 12bx - a^2 \\
\psi_4 &= 4y(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - 8b^3 - a^3) \\
\psi_{2m+1} &= \psi_{m+2}\psi_m^3 - \psi_{m-1}\psi_{m+1}^3 \qquad \text{for } m \geqslant 2 \\
\psi_{2m} &= (2y)^{-1}(\psi_m)(\psi_{m+2}\psi_{m-1}^2 - \psi_{m-2}\psi_{m+1}^2) \qquad \text{for } m \geqslant 3
\end{aligned}$$

Lemma 2.2.2. *ψ_n is a polynomial in $\mathbb{Z}[x, y^2, a, b]$ when n is odd and ψ_n is a polynomial in $2y\mathbb{Z}[x, y^2, a, b]$ when n is even.*

Proof. We prove the lemma by induction on n. For $n \leqslant 4$, the lemma is true by definition. Suppose $n = 2m$ for some integer m.

If m is even, then by the induction hypothesis $\psi_m, \psi_{m+2}, \psi_{m-2}$ are in $2y\mathbb{Z}[x, y^2, a, b]$ and ψ_{m+1}, ψ_{m-1} are in $\mathbb{Z}[x, y^2, a, b]$, which implies that ψ_n is in $2y\mathbb{Z}[x, y^2, a, b]$.

If m is odd, then by the induction hypothesis $\psi_m, \psi_{m+2}, \psi_{m-2}$ are in $\mathbb{Z}[x, y^2, a, b]$ and ψ_{m+1}, ψ_{m-1} are in $2y\mathbb{Z}[x, y^2, a, b]$, which again implies that ψ_n is in $2y\mathbb{Z}[x, y^2, a, b]$.

Similarly, we can check that if $n = 2m+1$ for some integer m, then ψ_n is in $\mathbb{Z}[x, y^2, a, b]$. □

Lemma 2.2.3. *Define the following polynomials:*

$$\begin{aligned}
\varphi_n &= x\psi_n^2 - \psi_{n+1}\psi_{n-1} \\
\omega_n &= (4y)^{-1}(\psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2)
\end{aligned}$$

For all n, $\varphi_n \in \mathbb{Z}[x, y^2, a, b]$. Also, if n is odd, then $\omega_n \in y\mathbb{Z}[x, y^2, a, b]$. If n is even, then $\omega_n \in \mathbb{Z}[x, y^2, a, b]$.

Lemma 2.2.4.

$$\begin{aligned}
\varphi_n &= x^{n^2} + \textit{lower degree terms} \\
\omega_n &= n^2x^{n^2-1} + \textit{lower degree terms}
\end{aligned} \tag{2.17}$$

Theorem 2.2.5. *Let $P = (x, y)$ be a point on the elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$, and let $n \in \mathbb{N}$. Then*

$$[n]P = \left(\frac{\varphi_n(x)}{\psi_n^2(x)}, \frac{\omega_n(x,y)}{\psi_n^3(x,y)} \right). \tag{2.18}$$

Corollary 2.2.6. *$\varphi_n(x)$ and $\psi_n^2(x)$ have no common roots. Therefore given an elliptic curve E, multiplication-by-n is an endomorphism on E with degree n^2.*

From Lemma 2.2.2, we know that if n is odd, then ψ_n is a polynomial in $\mathbb{Z}[x, y^2, a, b]$, which implies that given a fixed elliptic curve $E : y^2 = x^3 + ax + b$, ψ_n is a polynomial in x. Therefore by Corollary 2.2.6 and Theorem 2.2.5, we have that $nP = \infty$ if and only if $\psi_n(x) = 0$.

2.2.2 Torsion Points

Definition 2.2.7. For all primes l, we define the l-torsion subgroup of $E(\overline{\mathbb{F}}_p)$ as

$$E[l] := \{P \in E(\overline{\mathbb{F}}_p) : l \cdot P = \infty\}. \tag{2.19}$$

Thus for all $P = (x, y) \in E(\mathbb{F}_p)$, $P \in E[l]$ iff $\psi_l(x) = 0$.

Theorem 2.2.8. *For $l \neq p$, $E[l] \cong \mathbb{Z}_l \times \mathbb{Z}_l$.*

Recall that to compute the order of $E(\mathbb{F}_p)$ is equivalent to finding t, the trace of the Frobenius endomorphism, which satisfies $Z^2 - t \cdot Z + p = 0$ or equivalently,

$$(x^{p^2}, y^{p^2}) - t(x^p, y^p) + p(x, y) = \infty. \tag{2.20}$$

for all $(x, y) \in E(\overline{\mathbb{F}}_p)$.

One approach is to compute, for a given point $P = (x, y)$, (x^{p^2}, y^{p^2}), (x^p, y^p), $p(x, y)$, and check which value of $t = 0, 1, ..., 2\sqrt{p}$ satisfies the equation above. This would require $O(p^{1/2})$ computations, which is prohibitively large.

By consider the restriction of the characteristic polynomial of ϕ_p modulo l, we obtain the reduced characteristic equation $Z^2 - t_l \cdot Z + p_l = 0$ where $t_l = t \bmod l$ and $p_l = p \bmod l$ and $0 \leqslant t_l < l$, $|p_l| < l/2$. We also have

$$\left(x^{p^2}, y^{p^2}\right) + p_l(x, y) = t_l(x^p, y^p) \tag{2.21}$$

for all the points $(x, y) \in E[l]$. The idea now is, given a point $(x, y) \in E[l]$, to compute the expressions for $\left(x^{p^2}, y^{p^2}\right)$, (x^p, y^p), $p_l(x, y)$, and check which value of $t_l = 0, 1, ..., l-1$ satisfies the equation above.

We find the expression for x^{p^2}, x^p in the ring $\mathbb{F}_p[x]/(\psi_l(x))$ and y^{p^2}, y^p in the ring $\mathbb{F}_p[x, y]/(\psi_l(x), y^2 - x^3 - ax - b)$. These require $O(\log p(l^2 \log p)^2)$ arithmetic operations in $\mathbb{F}_p$. Finding t_l requires l additions of (x^p, y^p), which requires $O(l(l^2 \log p)^2)$ computations. Since l is of size $O(\log p)$, the total work required is $O(\log^8 p)$ bit operations.

Suppose $\left(x^{p^2}, y^{p^2}\right) \neq \pm p_l(x, y)$ for some $(x, y) \in E[l]$. Then $(x', y') := \left(x^{p^2}, y^{p^2}\right) + p_l(x, y) \neq \infty$, which implies that $t_l \not\equiv 0 \pmod{l}$

Let $(x_m, y_m) := m(x, y)$ for any integer m. Then $x^{p^2} \neq x_{p_l}$. From Theorem 2.2.5, we note that

$$x_m = \frac{\varphi_m(x)}{\psi_m^2(x)}, \qquad y_m = \frac{\omega_m(x, y)}{\psi_m^3(x, y)} \tag{2.22}$$

i.e. $x_m = f_{1,m}(x)$ and $y_m = y f_{2,m}(x)$ for some rational functions $f_{1,m}, f_{2,m} \in \mathbb{F}_p[x]$. We have

$$x' = \left(\frac{y^{p^2} - y_{p_l}}{x^{p^2} - x_{p_l}}\right)^2 - x^{p^2} - x_{p_l} \tag{2.23}$$

Since

$$\begin{aligned} \left(y^{p^2} - y_{p_l}\right)^2 &= y^2 \left(y^{p^2-1} - f_{2,p_l}(x)\right)^2 \\ &= (x^3 + ax + b)\left((x^3 + ax + b)^{(p^2-1)/2} - f_{2,p_l}(x)\right)^2, \end{aligned}$$

x' can be written as a rational function in x. Our aim is to find m such that $(x', y') = (x_m^p, y_m^p)$. Since the roots of $\psi_l(x)$ are the x-coordinates of points in $E[l]$, $x' - x_m^p \equiv 0 \pmod{\psi_l}$. Note that the roots of $\psi_l(x)$ have multiplicity 1, as $E[l]$ has $l^2 - 1$ distinct points of order l and ψ_l has degree $(l^2 - 1)/2$.

After m is found, where $x' - x_m^p \equiv 0 \pmod{\psi_l}$, we have

$$(x', y') = \pm(x_m^p, y_m^p) = (x_m^p, \pm y_m^p). \tag{2.24}$$

If $(y' - y_m^p)/y \equiv 0 \pmod{\psi_l}$, then $t_l \equiv m \pmod{l}$, otherwise $t_l \equiv -m \pmod{l}$.

In the remaining case $\left(x^{p^2}, y^{p^2}\right) = \pm p_l(x, y)$. Suppose

$$\phi_p^2(x, y) = \left(x^{p^2}, y^{p^2}\right) = p_l(x, y). \tag{2.25}$$

We have

$$t_l \phi_p(x, y) = \phi_p^2(x, y) + p_l(x, y) = 2p_l(x, y). \tag{2.26}$$

Therefore

$$t_l^2 p_l(x, y) = t_l^2 \phi_p^2(x, y) = 4p_l^2(x, y). \tag{2.27}$$

Thus $t_l^2 p_l \equiv 4p_l^2 \pmod{l}$, which implies that p_l is a square modulo l. Let $p_l = w^2 \pmod{l}$. Hence

$$(\phi_p + w)(\phi_p - w)(x, y) = (\phi_p^2 - p_l)(x, y) = \infty. \tag{2.28}$$

for all $(x, y) \in E[l]$.

If $\phi_p(x, y) = w(x, y)$, then $t_l \equiv 2w \pmod{l}$. If $\phi_p(x, y) = -w(x, y)$, then $t_l \equiv -2w \pmod{l}$. If neither condition is met for some point $(x, y) \in E[l]$, then $t_l \equiv 0 \pmod{l}$.

If p_l is not a square modulo l, then the initial supposition in Equation 2.25 is false. Therefore $\left(x^{p^2}, y^{p^2}\right) = -p_l(x, y)$, which implies $t_l \equiv 0 \pmod{l}$. Note that since l is odd, $p_l(x, y) \neq -p_l(x, y)$ for all values of p_l.

2.2.3 Algorithm Details

Description of Schoof's Algorithm We now state Schoof's algorithm in full:

1. Choose a set of primes $S = \{2, 3, 5, \cdots, L\}$ (with $p \notin S$) such that $\prod_{l \in S} l > 4\sqrt{p}$.
2. If $l = 2$, then $t_2 \equiv 0 \pmod 2$ iff $\gcd(x^p - x, x^3 + ax + b) \neq 1$.
3. For each odd prime $l \in S$ do:
   (a) Let $p_l \equiv p \pmod l$ such that $|p_l| < l/2$.
   (b) Compute x', the x-cordinate of $(x', y') = \left(x^{p^2}, y^{p^2}\right) + p_l(x, y) \pmod{\psi_l}$.
   (c) For $m = 1, 2, \cdots, (l-1)/2$ do:
      i. Compute the x-coordinate x_m of $(x_m, y_m) = m(x, y)$.
      ii. If $x' - x_m^p \not\equiv 0 \pmod{\psi_l}$, try the next value of m in step (c).
      iii. If $x' - x_m^p \equiv 0 \pmod{\psi_l}$, check if $(y' - y_m^p)/y \equiv 0 \pmod{\psi_l}$). If so, then set $t_l = m$, else set $t_l = l - m$; proceed to step 4.
   (d) If $p \in \overline{\mathrm{QR}}_l$, then set $t_l = 0$ and proceed to step 4. Otherwise define w so that $w^2 = p \pmod l$.
   (e) Compute $(x_w, y_w) := w(x, y)$.
   (f) If $x^p \neq x_w$, then set $t_l = 0$ and proceed to step 4.
   (g) If $y^p = y_w$, then set $t_l = 2w \pmod l$, else set $t_l = -2w \pmod l$.
4. Given $t \equiv t_l \pmod l$ for all $l \in S$, use CRT to compute t as the unique value satisfying the congruences and the condition $|t| \leqslant 2\sqrt{p}$. Return $\#E(\mathbb{F}_p) = p + 1 - t$.

Although Schoof's algorithm runs in polynomial-time, it is not sufficiently fast for practical use with curves of cryptographic sizes. For example, for curves defined over 256-bit finite fields, the algorithm's running time is $(\log 2^{256})^8 = 2^{64}$, which is barely feasible.

The computations for Schoof's algorithm are performed in the ring

$$\mathbb{F}_p[x, y] / \gcd(\psi_l(x), y^2 - x^3 - ax - b)$$

and as $\psi_l(x)$ has degree $(l^2 - 1)/2$, the degree of the elements is $O(l^2) = O(\log^2 p)$. In the Schoof-Elkies-Atkin algorithm, we will see that the division polynomials can be replaced with polynomials of lower degree, which results in considerable speedups over Schoof's algorithm.

2.2.4 Schoof-Elkies-Atkin Algorithm

In Schoof's algorithm, the characteristic equation of the Frobenius endomorphism is used to compute its trace t and thus $\#E(\mathbb{F}_p) = p + 1 - t$. Elkies and Atkin independently suggested improvements, depending on whether the characteristic equation splits over $\mathbb{F}_p$.

Theorem 2.2.9. *The group of automorphisms on $E[l]$ is isomorphic to $GL_2(\mathbb{Z}_l)$, the group of invertible 2-by-2 matrices over $\mathbb{Z}_l$.*

As $l \neq p$, ϕ_p is an automorphism on $E[l]$. Theorem 2.2.9 tells us that the action of the Frobenius endomorphism ϕ_p on $E[l]$ can be represented by $A_l \in \mathrm{GL}_2(\mathbb{Z}_l)$. Since the characteristic polynomial of ϕ_p restricted to $E[l]$ is $Z^2 - t_l Z + p_l$, it follows from the Cayley-Hamilton Theorem that A_l satisfies the polynomial and the eigenvalues of A_l are the roots of $Z^2 - t_l Z + p_l$. We will see later that A_l is diagonal, and the trace of A_l is the sum of its eigenvalues which is just t_l, hence justifying calling t the trace of the Frobenius endomorphism.

Definition 2.2.10. The **discriminant** of $Z^2 - tZ + p$ is $\Delta = t^2 - 4p$ and the polynomial has roots in $\mathbb{Z}_l$ if and only if Δ is a square in $\mathbb{Z}_l$. For each prime l, we call l an **Elkies** prime if Δ is a square modulo l; otherwise we call it an **Atkin** prime.

Suppose l is an Elkies prime. Let $\lambda, \mu \in \mathbb{Z}_l$ be the two roots of $Z^2 - t_l Z + p_l$. Then there exists nonzero vectors $v_P, v_Q \in \mathbb{Z}_l^2$ so that $A_l v_P = \lambda v_P$ and $A_l v_Q = \mu v_Q$. Likewise, there are nonzero points $P, Q \in E[l]$ such that $\phi_p(P) = \lambda P$ and $\phi_p(Q) = \mu Q$. Therefore l is an Elkies primes if and only if ϕ_p has a 1-dimensional eigenspace defined over $\mathbb{F}_p$.

If there exists $R \in E[l]$ such that $R = \alpha P$ and $R = \beta Q$ for some $\alpha, \beta \in \mathbb{Z}_l$, then $\lambda(\beta Q) = \lambda(\alpha P) = \alpha\lambda(P) = \alpha\phi_p(P) = \phi_p(\alpha P) = \phi_p(\beta Q) = \beta\phi_p(Q) = \beta\mu(Q) = \mu(\beta Q)$. Since $\lambda \neq \mu$, $R = \beta Q = \infty$. Since $E[l] \cong \mathbb{Z}_l \times \mathbb{Z}_l$, we have $E[l] \cong \langle P \rangle \times \langle Q \rangle$. Thus every point in $E[l]$ can be written as $aP + bQ$ for some $\alpha, \beta \in \mathbb{Z}_l$, i.e. $\{P, Q\}$ is a basis for the vector space $E[l]$.

Since t is not known, we cannot use the discriminant of the characteristic polynomial to check if a prime is Elkies or Atkin. Instead, we introduce modular polynomials as a means to do so.

2.2.5 Using Classical Modular Polynomials

Classical modular polynomials are used in this implementation for checking if a prime l is Elkies or Atkin, as they were the polynomials used in Atkin's theorem for classification of the primes. Since these polynomials exist independently of the curve parameters, they are usually pre-computed. As such the coefficients of the polynomials are stored in files and loaded by the implementation when needed.

The coefficients of the classical modular polynomials are very large, and the storage of the coefficients of the l-th classical modular polynomials for primes up to 199 requires more than 0.5GB. Fortunately, there are alternative types of modular polynomials with much smaller coefficients that can be adapted for use in Atkin's theorem. However, for the purposes of this work, we focus primarily on the classical modular polynomials. Repositories for the coefficients of the l-th classical modular polynomials exist online for primes $l \leqslant 199$. They are also included, for primes $l \leqslant 59$, in the distribution of MAGMA. Since the coefficients of these polynomials are very large, we would ideally choose to store as few of these polynomials as necessary. For the purposes of point counting for elliptic curves of cryptographic sizes, how large do we need the largest l to be?

In Section 4.4, we provide some estimates of the size of the largest prime l as well as the number of primes needed given the size of a random prime p. From the estimates, we can deduce that the online resources, with l up to 199, provide us with sufficient pre-computed polynomials for 521-bit prime fields when using the full SEA (see Section 5.1 for more discussion on this). For the completeness of the implementation, I have implemented routines for the computation of classical modular polynomials, allowing us to compute additional polynomials for counting points on curves over larger fields. In the next chapter, we will define the classical modular polynomials and examine how they can be computed.

Chapter 3

Classical Modular Polynomials via Isogeny Volcanoes

In this chapter we look at the recent advancements in computing modular polynomials, introduced at ECC2009 by Bröker, Lauter and Sutherland [2]. Using suitably chosen isogeny volcanoes, we can generate enough information to recover the coefficients of the classical modular polynomials via the Chinese Remainder Theorem. We proceed by first defining an isogeny on elliptic curves. From the relationship between isogenies and modular polynomials, we describe the relation between isogeny volcanoes and class groups. Within this framework, we can then present the algorithm to compute the modular polynomials.

The main source of material for this chapter is [2], which provides the justification for the choice of parameters to construct the l-isogeny volcano of interest, as well as the procedure to compute the modular polynomial using the isogeny volcanoes. As explained in Sutherland's expository paper [23] on isogeny volcanoes, David Kohel studied the structure of isogeny graphs for elliptic curves over finite fields in his thesis [12], while its application in algorithms came later [10][11]. Silverman's textbook [20] provides the background on isogenies and elliptic curves, while the material on the classical modular polynomial, the class group and the Hilbert class polynomial was referenced from [9] and [7].

3.1 Isogeny Volcanoes

3.1.1 Isogenies

Definition 3.1.1. An **isogeny** between two elliptic curves E_1 and E_2 is a group homomorphism $\psi : E_1 \to E_2$ (with respect to point addition) which is an algebraic map. The composition of two isogenies is also an isogeny.

Example 3.1.2. The scalar multiplication by n map, denoted by $[n] : E \to E$ with $[n]P = nP$, is an isogeny from E to E.

Proposition 3.1.3. *Every isogeny is either constant or surjective.*

The only constant isogeny is the zero isogeny $\psi : E \to \{\infty\}$, which is equivalent to the multiplication-by-zero map $[0]$.

Definition 3.1.4. The **degree** of a surjective isogeny $\deg(\psi)$ is the degree of ψ as an algebraic map, with $\deg[0] = 0$.

Theorem 3.1.5. *Let E_1, E_2 be elliptic curves defined over $\mathbb{F}$ and $\psi : E_1 \to E_2$ be a (non-constant) surjective isogeny of degree n. There exists a unique isogeny $\hat{\psi} : E_2 \to E_1$ so that*

$$\hat{\psi} \circ \psi = [n] = \psi \circ \hat{\psi}.$$

Definition 3.1.6. We refer to the map $\hat{\psi}$ as the **dual** isogeny of ψ.

Theorem 3.1.7. *Let E_1, E_2, E_3 be elliptic curves defined over $\mathbb{F}$, and $\psi : E_1 \to E_2$ and $\phi : E_2 \to E_3$ be (non-constant) surjective isogenies of degree n. The following hold:*

(i) $\widehat{\phi \circ \psi} = \hat{\psi} \circ \hat{\phi}$.

(ii) $\forall n \in \mathbb{Z}, \hat{[n]} = [n]$ and $\deg[n] = n^2$.

(iii) $\deg \hat{\psi} = \deg \psi$.

(iv) $\hat{\hat{\psi}} = \psi$.

3.1.2 Modular Polynomials

Definition 3.1.8. Let $\mathbb{H} := \{x + iy | y > 0; x, y \in \mathbb{R}\}$ denote the **upper half-plane**. The following functions are defined for $\tau \in \mathbb{H}$:

$$g_2(\tau) = 60 \sum_{(m,n)\neq(0,0)} (m + n\tau)^{-4} \tag{3.1}$$

$$g_3(\tau) = 140 \sum_{(m,n)\neq(0,0)} (m + n\tau)^{-6} \tag{3.2}$$

$$j(\tau) = 1728 \frac{g_2^3}{g_2^3 - 27g_3^2} \tag{3.3}$$

The function $j(\tau)$ is referred to as the j-invariant of the lattice $[1, \tau]$.

Theorem 3.1.9. *The elliptic curve*

$$E : y^2 = 4x^3 - g_2x - g_3$$

corresponds to the lattice $[1, \tau]$ *via the Weierstrass* $\wp$*-function for* $[1, \tau]$.

Definition 3.1.10. We define the **j-invariant** of an elliptic curve $E : y^2 = x^3 + ax + b$, where $a, b \in K$ a field with characteristic not 2 or 3, as:

$$j(E) = 1728 \frac{4a^3}{4a^3 + 27b^2}.$$

Note that for the elliptic curve $E : y^2 = x^3 + ax + b$ defined over $\mathbb{F}_p$ with $p \neq 2, 3$, we can use the isomorphism $(x, y) \mapsto (4x, 4y)$ to convert the equation to

$$E : y^2 = 4x^3 - (-a/4)x - (-b/16),$$

which corresponds to the lattice with $g_2 = -a/4$, $g_3 = -b/16$ and

$$j = 1728 \frac{(-a/4)^3}{(-a/4)^3 - 27(-b/16)^2} = 1728 \frac{4a^3}{4a^3 + 27b^2}.$$

This shows that two definitions for j-invariants are consistent.

The reason why $j(E)$ is rightly called an invariant is due to the result below, which can be proven by considering all possible isomorphisms on E. Note that the j-invariant lies in the same field K as that which the curve is defined over.

Proposition 3.1.11. *Two elliptic curves are isomorphic over $\mathbb{F}_q$ have the same j-invariant. Conversely, two elliptic curves with the same j-invariant are either isomorphic over $\mathbb{F}_q$ or are twists of each other.*

Definition 3.1.12. We define the set of matrices

$$C(l) = \left\{ \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} : ad = l, a > 0, 0 \leqslant b < d, \gcd(a, b, d) = 1 \right\}.$$

For $\sigma \in C(l)$, we define the action on $\mathbb{H}$ as

$$\sigma\tau = \frac{a\tau + b}{d}$$

Definition 3.1.13. The l-th **(classical) modular polynomial** $\Phi_l(X, Y) \in \mathbb{C}[X, Y]$ is a polynomial such that

$$\Phi_l(X, j(\tau)) = \prod_{\sigma \in C(l)} (X - j(\sigma\tau)). \tag{3.4}$$

Note that when l is prime, either $a = l, d = 1$ (and thus $b = 0$), or $a = 1, d = l$ with $b \in [0, l-1]$. This means $|C(l)| = l + 1$, and thus Φ_l is monic in X with degree $l + 1$.

Theorem 3.1.14. *Let l be a positive integer.*

1. $\Phi_l(X, Y) \in \mathbb{Z}[X, Y]$.
2. $\Phi_l(X, Y)$ *is irreducible as a polynomial in* X.
3. $\Phi_l(X, Y) = \Phi_l(Y, X)$.
4. *If l is not a perfect square, then $\Phi_l(X, X)$ is a polynomial of degree > 1 with leading coefficient ± 1.*
5. *If l is prime, then $\Phi_l(X, Y) \equiv (X^l - Y)(X - Y^l) \bmod l\mathbb{Z}[X, Y]$.*

Example 3.1.15. The 5-th classical modular polynomial $\Phi_5(X,Y) \in \mathbb{Z}[X,Y]$ is

$$\begin{aligned}
X^6 &- X^5Y^5 + 3720X^5Y^4 - 4550940X^5Y^3 + 2028551200X^5Y^2 - 246683410950X^5Y \\
&+ 1963211489280X^5 + 3720X^4Y^5 + 1665999364600X^4Y^4 \\
&+ 10787892818533680 0X^4Y^3 + 38308360977981121 5375X^4Y^2 \\
&+ 12854179890682881638400 0X^4Y + 12847331328414244562534 40X^4 - 4550940X^3Y^5 \\
&+ 10787892818533680 0X^3Y^4 - 44120696551291483524610 0X^3Y^3 \\
&+ 26898488858380731577417728000X^3Y^2 \\
&- 19245793461892829965510823116 8000X^3Y \\
&+ 28024477782843952780432156529786 8800X^3 + 2028551200X^2Y^5 \\
&+ 38308360977981121 5375X^2Y^4 + 26898488858380731577417728000X^2Y^3 \\
&+ 51109417775524180831107651993600 00X^2Y^2 \\
&+ 36554736583949629295706472332656640000X^2Y \\
&+ 66925000426279977084871494150150684672 00X^2 - 246683410950XY^5 \\
&+ 12854179890682881638400 0XY^4 - 19245793461892829965510823116 8000XY^3 \\
&+ 36554736583949629295706472332656640000XY^2 \\
&- 26407345707662059625971579024797878294937 6XY \\
&+ 53274330803424425450420160273356509151232000X + Y^6 + 1963211489280Y^5 \\
&+ 12847331328414244562534 40Y^4 + 28024477782843952780432156529786 8800Y^3 \\
&+ 66925000426279977084871494150150684672 00Y^2 \\
&+ 53274330803424425450420160273356509151232000Y \\
&+ 14135994715472135869775347469107136275100467 2000
\end{aligned}$$

3.1.3 Isogeny Volcanoes

Definition 3.1.16. Two elliptic curves E_1 and E_2 are said to be **isogenous** if there is a surjective isogeny $\psi : E_1 \to E_2$. E_1 and E_2 are said to be l-**isogenous** if the isogeny ψ has degree l.

It is easy to show that being isogenous is an equivalence relation.

Theorem 3.1.17. (Tate) *Elliptic curves $E_1/\mathbb{F}_p$ and $E_2/\mathbb{F}_p$ are isogenous over $\mathbb{F}_p$ if and only if $\#E_1(\mathbb{F}_p) = \#E_2(\mathbb{F}_p)$.*

Theorem 3.1.18. *Let E_1, E_2 be elliptic curves defined over $\mathbb{F}_p$, with j-invariants j_1, j_2 respectively. For l prime, E_1, E_2 are l-isogenous if and only if $\Phi_l(j_1, j_2) = 0$ over $\mathbb{F}_p$.*

Theorem 3.1.18 tells us that for a given elliptic curve $E(\mathbb{F}_p)$ with j as its j-invariant, the $l + 1$ curves l-isogenous to E have j-invariants which are roots of $\Phi_l(X, j) = 0$ over $\mathbb{F}_p$.

Definition 3.1.19. The **l-isogeny graph** G_l is an undirected graph with vertex set $\{j(E) : E/\mathbb{F}_p\}$ and edges $\{(j_1, j_2)\}$ for all $j_1, j_2 \in \mathbb{F}_p$ whenever $\Phi_l(j_1, j_2) = 0$ over $\mathbb{F}_p$. A connected component of G_l is referred to as an **l-volcano**.

Structure of an l-volcano Since isogenous curves have the same number of points over $\mathbb{F}_p$, they share the same trace for the Frobenius endomorphism t. Thus $t^2 - 4p$ is the same for all curves in an l-volcano. The neighbourhood $N(j)$ of j is made up of the roots of $\Phi_l(X, j) = 0$. From Theorem 4.1.2 in the next chapter, we see that the $|N(j)|$ can take four possible values.

1. If $t^2 - 4p$ is not a square modulo l, then $\Phi_l(X, j) = 0$ has no roots in $\mathbb{F}_p$, which means $|N(j)| = 0$.
2. If $t^2 - 4p$ is a nonzero square modulo l, then $\Phi_l(X, j) = 0$ has two roots in $\mathbb{F}_p$, which means $|N(j)| = 2$.
3. If $t^2 - 4p = 0 \pmod{l}$, then $\Phi_l(X, j)$ either splits over $\mathbb{F}_p$, which means $|N(j)| = l + 1$, or $\Phi_l(X, j) = 0$ has exactly one root in $\mathbb{F}_p$, i.e. $|N(j)| = l$.

3.2 Computing lth Classical Modular Polynomial

3.2.1 Modular Polynomials via Isogeny Volcanoes

From Theorem 3.1.18, we know that for a given elliptic curve E defined over $\mathbb{F}_p$, the roots of the l-th classical modular polynomial $\Phi_l(X, Y)$ over $\mathbb{F}_p$ evaluated at $Y = j$ are the j-invariants of curves that are l-isogenous with j. In the next chapter, we will see how the SEA algorithm uses $\Phi_l(X, j)$ to find curves that are l-isogenous with E.

Conversely, if we can find sufficiently many curves that are l-isogenous to an elliptic curve with its j-invariant equal to j, we can find the distinct roots of $\Phi_l(X, j) \pmod p$, and therefore compute the coefficients of $\Phi_l(X, j)$ over $\mathbb{F}_p$. To achieve this without knowing Φ_l, we might think that we can use Tate's theorem, since we just need to find l-isogenous curves that have the same number of points. But this requires us to know the number of points on E, which is what we are going to use SEA to find!

Thankfully, this is not a real conundrum. As $\Phi_l(X, Y)$ is defined over $\mathbb{Z}$, independently of E and p, we can construct a set of curves with j-invariant j_i defined over p_r for which we know the number of points a priori. For each p_r, with sufficiently many $\Phi_l(X, j_i) \pmod{p_r}$, we can interpolate for the expression $\Phi_l(X, Y) \pmod{p_r}$. With sufficiently many expressions for $\Phi_l(X, Y) \pmod{p_r}$, we can use the Chinese Remainder Theorem (CRT) to compute the coefficients of $\Phi_l(X, Y)$ over $\mathbb{Z}$.

To ensure that after the Chinese Remainder Theorem step, the coefficients obtained for $\Phi_l(X, Y)$ modulo $\prod_r p_r$ are precisely that of $\Phi_l(X, Y) \in \mathbb{Z}[X]$, we must find sufficiently many primes p_r so that $\prod_r p_r$ is greater than twice the maximum absolute value MAX_l of the coefficients of $\Phi_l(X, Y)$.

The key to this approach is constructing curves for which the number of points is known without the need for a point counting algorithm. This is achieved using the Complex-Multiplication (CM) Method, which we will describe in the subsection.

3.2.2 Complex-Multiplication Method

To construct CM curves, we need to define an imaginary quadratic field by choosing a fundamental discriminant $D < 0$. Let p be a prime for which the diophantine equation

$$4p = x^2 - Dy^2$$

has an integer solution (t, s). Then $N = p + 1 \pm t$ are the possible group orders of the elliptic curves over $\mathbb{F}_p$ that we can construct.

Lemma 3.2.1. *The following holds for elliptic curves over $\mathbb{F}_p$:*

- *Every element in $\mathbb{F}_p$ is the j-invariant of an elliptic curve over $\mathbb{F}_p$.*
- *If $D < -4$, then all elliptic curves, with j-invariant $j \neq 0, 1728$, over $\mathbb{F}_p$ are given by*

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

  where $k = j/(1728 - j)$ and c is any element in $\mathbb{F}_p$.
- *Suppose E and E' have the same j-invariant $j \neq 0, 1728$ but are not isomorphic. Then E and E' are quadratic twists of each other.*

Lemma 3.2.1 tells us that every j-invariant defines a unique elliptic curve over $\mathbb{F}_p$, up to curve isomorphism and twists, and that we can construct an elliptic curve with the specified j-invariant explicitly if $D < -4$. Furthermore, the j-invariants of the curves we construct by the CM method can be characterized precisely as the roots of a polynomial, determined by D, over $\mathbb{F}_p$. This allows us to construct the vertices of an l-volcano and thereby determine the roots of $\Phi_l(X, j)$.

As we are only interested in finding the j-invariants of l-isogenous elliptic curves so as to determine the roots of , rather than distinguish between an elliptic curve and its quadratic twist. Hence it is justified to use $j(E)$ and E interchangeably and refer to the j-invariants as curves when unambiguous. We will now narrow our focus to CM curves that are used in our computation. Further details on the CM method can be found in [21] and [4].

For our purpose of computing Φ_l, we seek integer parameters v, D, with $D < -4$, $D \equiv 0$ or $1 \bmod 4$, for which we can find sufficient integers t and corresponding prime p that satisfy the conditions

$$\begin{aligned} 4p &= t^2 - v^2 l^2 D, && (3.5)\\ p &\equiv 1 \pmod{l}, && (3.6)\\ l &\nmid v, && (3.7)\\ \left(\frac{D}{l}\right) &= 1, && (3.8)\\ h(D) &\geqslant l + 2 && (3.9) \end{aligned}$$

From 3.5 and 3.6, we gather that $t^2 \equiv 4 \pmod{l}$. We can fix $v = 2$, $t \equiv 2 \pmod{2l}$, so that

$$4p = t^2 - 4l^2 D = (2 + 2kl)^2 - 4l^2 D = 4(1 + 2kl + k^2 l^2 - l^2 D)$$

and thus ensure that $p \equiv 1 \pmod{l}$. This also ensures that equation 3.7 is met, since l is an odd prime. Note that this restriction does not prevent us from finding an appropriate D.

With these parameters in place, for each p, we can define a family of elliptic curves j_i over $\mathbb{F}_p$, with $p + 1 - t$ points, that are the vertices of an l-volcano. Furthermore, since $t^2 - 4p \bmod l = 0$, from our earlier discussion, we note that each j_i in the l-volcano has either $l + 1$ neighbours or only one neighbour. In fact, the choice of parameters guarantees a specific structure for the l-volcano.

Definition 3.2.2. We refer to an l-volcano, with parameters D, p, l, v, t satisfying conditions 3.5, 3.6, 3.7, 3.8, 3.9 as a **Bröker-Lauter-Sutherland (BLS) l-volcano**. For the rest of this thesis, we will further assume that $v = 2$ and $t \equiv 2 \pmod{2l}$.

Proposition 3.2.3. *The structure of a BLS l-volcano satisfies the following:*

- *The subgraph of vertices with $l + 1$ neighbours is a cycle.*
- *Every vertex with $l + 1$ neighbours is adjacent to exactly $l - 1$ vertices with one neighbour.*
- *Every vertex with one neighbour is adjacent to exactly a vertex with $l + 1$ neighbours.*

Definition 3.2.4. We refer to the set of vertices with $l+1$ neighbours as the **rim** of the l-volcano, and the set of vertices with one neighbour as the **floor** of the l-volcano. We define an n**-isogeny cycle** as an ordered set of j-invariants $j_1, j_2, \cdots, j_m$ where j_i, j_{i+1} are n-isogenous and j_1, j_m are n-isogenous. Thus the rim of an l-volcano naturally defines an l-isogeny cycle.

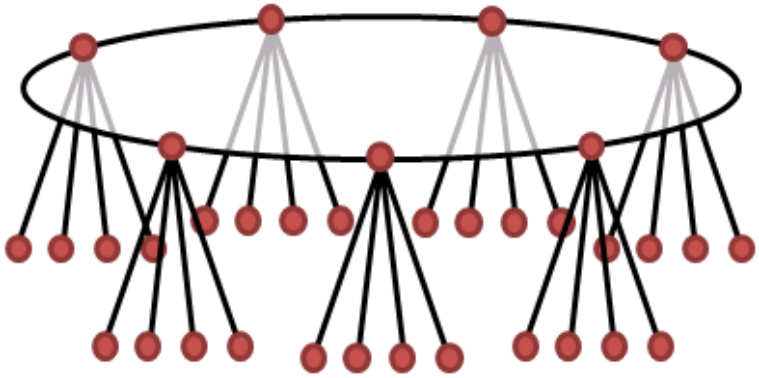

Figure 3.1: A 5-Isogeny Volcano

Example 3.2.5. In Figure 3.1, $l = 5$. There seven vertices with $6 = l + 1$ neighbours, forming the rim of the 5-volcano. The remaining 28 vertices each have one neighbour, and they form the floor of the 5-volcano.

For each $j(E_i)$ on the rim of an l-volcano there is a curve $E_{i,s}$ on the floor, and an isogeny $\phi_{i,s} : E_i \to E_{i,s}$ of degree l, for $s = 1, \cdots, l-1$. We use the l^2-isogeny $\psi_{i,s} := \widehat{\phi_{i,s}} \circ \phi_{i,s+1}$, for $s = 1, \cdots, l-1$ to define an l^2-isogeny cycle on the floor of an l-volcano for each E_i.

Example 3.2.6. For $l = 5$, we can choose $D = -151$, $v = 2$. Condition 3.5 becomes $p = (t/2)^2 + 3775$. The primes that we generate are

$$\begin{aligned} p \;=\; & 4451, 6911, 9551, 28111, 54851, 110051, 123491, 160591, \\ & 211711, 280451, 434111, 530851, 686051, 736511, \end{aligned}$$

corresponding to $t = 52, 112, 152, 312, 452, 652, 692, 792, 912, 1052, 1312, 1452, 1652, 1712$.

3.2.3 Details of Algorithm

We summarise the above as an algorithm before explaining the steps in greater detail.

ComputeModularPolynomial(l)
Input: Prime l
Output: $\Phi_l(X, Y)$ over $\mathbb{F}_p$.

1. Find suitable discriminant D, and compute the Hilbert class polynomial H_D.
2. Find suitable t, p.
3. While $\prod p < 2\,\mathrm{MAX}_l$ do:
   (a) Find suitable prime p.
   (b) Find a root j_1 of H_D over $\mathbb{F}_p$.
   (c) Enumerate the roots of H_D and identify the l-isogeny cycle on the rim.
   (d) Find a curve j_1' on the floor which is l-isogenous to j_1.
   (e) Enumerate the curves on the floor of the volcano, and identify the l^2-isogeny cycle.
   (f) Match each j_i on the rim to the $l-1$ curves on the floor it is l-isogenous to.
   (g) For each j_i, compute $\Phi_l(X, j_i) \in \mathbb{F}_p[X]$ via interpolation.
   (h) Interpolate $\Phi_l(X, Y)$ over $\mathbb{F}_p$ as a polynomial in $(\mathbb{F}_p[Y])[X]$, using $\Phi_l(X, j_i) \in \mathbb{F}_p[X]$.
4. Compute coefficients of $\Phi_l(X, Y) \in \mathbb{Z}(X, Y)$ via CRT using $\Phi_l(X, Y) \in \mathbb{F}_p[X, Y]$.
5. Return $\Phi_l(X, Y)$.

3.3 On the Rim

3.3.1 Hilbert Class Polynomials

Theorem 3.3.1. *Let $\mathcal{O}_D$ denote the imaginary quadratic order $\mathbb{Z}[\frac{1+\sqrt{D}}{2}]$ for $D < 0$. The j-invariant of an elliptic curve $\mathbb{C}/\mathcal{O}_D$ is an algebraic integer ω.*

Definition 3.3.2. The minimal polynomial of ω, as defined in Theorem 3.3.1, over $\mathbb{Z}$ is the **Hilbert class polynomial**, which we write as $H_D \in \mathbb{Z}[X]$. The **class number** $h(D)$ is defined as the degree of H_D.

Theorem 3.3.3. *$E/\mathbb{F}_p$ is an elliptic curve on the rim of a BLS l-volcano if and only if $j(E)$ is a root of H_D when defined over $\mathbb{F}_p$.*

The Hilbert class polynomial $H_D \in \mathbb{Z}[X]$ is first computed via a complex-analytic method (more details later in this chapter). We then express it as a polynomial over $\mathbb{F}_p$. Theorem 3.3.3 tells us that the roots of H_D over $\mathbb{F}_p$ will be precisely the j-invariants of curves on the rim of the BLS l-volcano. Instead of finding all the roots of H_D over $\mathbb{F}_p$, we begin with one such root, and use the class group of binary quadratic forms of discriminant D to enumerate the rest.

Example 3.3.4. The Hilbert class polynomial for $D = -151$ and $t = 52$, $p = 4451$ is

$$H_D(X) = X^7 + 2230X^6 + 1720X^5 + 3225X^4 + 2587X^3 + 2019X^2 + 2242X + 803$$

and has roots $351, 701, 901, 1582, 2215, 2501, 2872$, with $h(D) = 7$. These roots are the j-invariants of elliptic curves defined over $\mathbb{F}_p$ on the surface of the l-volcano, each with $p + 1 - t$ points.

3.3.2 Binary Quadratic Forms

Definition 3.3.5. A **binary quadratic form** is a quadratic polynomial $f(x, y) := ax^2 + bxy + cy^2$ where x, y are indeterminates. We write the form as $\langle a, b, c\rangle$ for brevity. We are interested in **integral** binary quadratic forms with discriminant D, where $a, b, c \in \mathbb{Z}$ and $b^2 - 4ac = D$. A binary quadratic form is **primitive** if $\gcd(a, b, c) = 1$ and **positive definite** if $a > 0$ and $D < 0$ (thus $c > 0$). A binary quadratic form is **reduced** if $0 \leqslant |b| \leqslant a \leqslant c$ and if $|b| = a$ or $a = c$, then $b \geqslant 0$.

For the remainder of this thesis, we will assume that any binary quadratic form will be integral, primitve and positive definite.

Proposition 3.3.6. *Any non-reduced binary quadratic form can be reduced to a unique reduced form by a combination of the following steps recursively:*

1. *If $a > c$, then swap a, c and set $b = -b$.*
2. *If $|b| > a$, set $b = b + 2\lfloor \frac{a-b}{2a} \rfloor a$ and recompute $c = \frac{b^2 - D}{4a}$.*

Note that the above steps leave the discriminant unchanged.

Definition 3.3.7. Two binary quadratic forms are **equivalent** if they share the same reduced form after reduction. Given two binary quadratic forms $\langle a_1, b_1, c_1 \rangle, \langle a_2, b_2, c_2 \rangle$, let

$$g = \gcd(a_1, a_2, \frac{b_1 + b_2}{2}), \qquad \varrho a_1 + \varsigma a_2 + \nu s = g$$

and

$$\beta = \frac{1}{g}\left(\varrho a_1 b_2 + \varsigma a_2 b_1 + \nu \frac{b_1 b_2 + D}{2}\right)$$

We define the Dirichlet composition as

$$\langle a_1, b_1, c_1 \rangle \circ \langle a_2, b_2, c_2 \rangle = \langle a_3, b_3, c_3 \rangle$$

with $a_3 = \frac{a_1 a_2}{g^2}, b_3 = \beta \bmod 2a_3, |b_3| \leqslant a_3, c_3 = \frac{b_3^2 - D}{4a_3}$. We write $\langle a, b, c \rangle^k$ to denote $\langle a, b, c \rangle$ composed with itself k times.

Proposition 3.3.8. *The set of reduced binary quadratic forms together with the Dirichlet composition forms a finite abelian group. The identity element is $\langle 1, 0, -D/4 \rangle$ when $D \equiv 0 \pmod 4$, and $\langle 1, 1, (1-D)/4 \rangle$ when $D \equiv 1 \pmod 4$. The inverse of $\langle a, b, c \rangle$ is $\langle a, -b, c \rangle$. We refer to this group as the class group of binary quadratic forms with discriminant D and denoted it by $\mathrm{cl}(D)$.*

Theorem 3.3.9. $\mathrm{cl}(D)$ *induces a free transitive group action on the roots of H_D.*

Corollary 3.3.10. *The order of the $\mathrm{cl}(D)$ is $h(D)$, the class number.*

Corollary 3.3.11. *The class number $h(D)$ is the number of reduced binary quadratic forms with discriminant D.*

Example 3.3.12. The class group $\mathrm{cl}(-151)$ consists of the reduced forms

$$\langle 1, 1, 38 \rangle, \langle 2, 1, 19 \rangle, \langle 2, -1, 19 \rangle, \langle 4, 3, 10 \rangle, \langle 4, -3, 10 \rangle, \langle 5, 3, 8 \rangle, \langle 5, -3, 8 \rangle$$

3.3.3 Finding Discriminants

Binary quadratic forms are easy to work with, since they are defined as integer tuples and require only basic arithmetic to manipulate. This gives us a concrete way to study the isogeny relations between the roots of H_D, in particular to check if our choice of D meets the condition $h(D) \geqslant l + 2$, before computing H_D itself. This is noteworthy, as computing H_D takes non-negligible time.

Furthermore, the class number $h(D)$ can be computed by counting the number of reduced forms directly. With $D = b^2 - 4ac$ and $|b| \leqslant a \leqslant c$, we have $a \leqslant \sqrt{-D/3}$, and thus only $-D/3$ possible pairs of (a, b) to check.

Theorem 3.3.3 states that the roots of H_D are the j-invariants on the rim of a BLS-volcano. With a root j_0 of H_D, Theorem 3.3.9 tells us that the action of $\mathrm{cl}(D)$ on j_0 will yield the remaining roots of H_D. To be specific, for distinct forms $\alpha_1, \alpha_2 \in \mathrm{cl}(D)$, $\alpha_1 j_0 \neq \alpha a_2 j_0$, and the identity element in $\mathrm{cl}(D)$ fixes j_0.

In the context of a BLS-volcano, we are interest in the fact that any l-isogeny on the rim corresponds to the action of either $\langle l, b, c\rangle$ or $\langle l, -b, c\rangle$. However, instead of using an explicit action of a form on the j-invariant, we will instead specify a prime parameter $l_0 \ll l$, for which there exists a form $\langle l_0, b_0, c_0\rangle \in \mathrm{cl}(D)$, and $\langle l_0, b_0, c_0\rangle^{k_{\mathrm{rim}}} = \langle l, b, c\rangle$ for some $0 \geqslant k_{\mathrm{rim}} < h(D)$. This relation between the forms guarantees that any l-isogeny on the rim is equivalent to the composition of some k_{rim} l_0-isogenies. Thus we can use the l_0-th classical modular polynomial, which is known since l_0 is very small, in place of the l-isogeny.

Therefore, when determining D for each l, we need to check that indeed such an l_0 exists. The approach that used in this implementation is to check that there exists a binary quadratic form $\langle l_0, b_0', c_0'\rangle$ which generates $\mathrm{cl}(l^2 D)$, the class group acting on the curves on the floor of the l-volcano. This condition ensures that there exists $\langle l_0, b_0, c_0\rangle$ that generates $\mathrm{cl}(D)$. As such, we can define parameters k_{rim} and k_{floor} where $\langle l_0, b_0, c_0\rangle^{k_{\mathrm{rim}}} = \langle l, b, c\rangle$ and $\langle l_0, b_0', c_0'\rangle^{k_{\mathrm{floor}}} = \langle l^2, b', c'\rangle$. As $\langle l, b, c\rangle$ is also a generator of the class group, k_{rim} will be coprime with $h(D)$. k_{floor} will be equal to $h(D)$.

Example 3.3.13. For the class group in Example 3.3.12, $\langle 2, 1, 19\rangle$ generates the sequence

$$\langle 2, 1, 19\rangle, \langle 4, -3, 10\rangle, \langle 5, 3, 8\rangle, \langle 5, -3, 8\rangle, \langle 4, 3, 10\rangle, \langle 2, -1, 19\rangle, \langle 1, 1, 38\rangle.$$

Hence $\langle 2,1,19\rangle$ is a generator with $\langle 2,1,19\rangle^3 = \langle 5,3,8\rangle$. Thus we can choose $l_0 = 2$ and $k_{\text{rim}} = 3$.

The class group on the floor is $\text{cl}(l^2 D) = \text{cl}(-3775)$, and has $(l-1)*h(D) = 28$ elements. We can check that $\langle 2,1,472\rangle$ generates the group

$$\langle 2,1,472\rangle, \langle 4,1,236\rangle, \langle 8,1,118\rangle, \langle 16,1,59\rangle, \langle 32,-31,37\rangle, \langle 19,5,50\rangle, \langle 25,-5,38\rangle,$$
$$\langle 29,13,34\rangle, \langle 17,-13,58\rangle, \langle 31,-21,34\rangle, \langle 22,3,43\rangle, \langle 11,3,86\rangle, \langle 22,-19,47\rangle, \langle 25,25,44\rangle,$$
$$\langle 22,19,47\rangle, \langle 11,-3,86\rangle, \langle 22,-3,43\rangle, \langle 31,21,34\rangle, \langle 17,13,58\rangle, \langle 29,-13,34\rangle, \langle 25,5,38\rangle,$$
$$\langle 19,-5,50\rangle, \langle 32,31,37\rangle, \langle 16,-1,59\rangle, \langle 8,-1,118\rangle, \langle 4,-1,236\rangle, \langle 2,-1,472\rangle, \langle 1,1,944\rangle$$

with $\langle 2,1,472\rangle^7 = \langle 5^2,-5,38\rangle$. Thus $k_{\text{floor}} = 7 = h(D)$.

As $\Phi_{l_0}(X,Y)$ has to be known, we require $l_0 \ll l$. In practice, we are able find $l_0 \leqslant 7$ for $l < 200$ that satisfies our needs. The table below shows a set of possible selection of parameters, along with the corresponding class number:

Table 3.1: Full Parameters for $l \leq 200$

l	D	$h(D)$	l_0	l	D	$h(D)$	l_0	l	D	$h(D)$	l_0
3	-71	7	2	59	-9431	91	2	131	-43711	147	2
5	-151	7	2	61	-10247	105	2	137	-47951	241	3
7	-271	11	5	67	-11783	95	2	139	-49919	189	2
11	-439	15	2	71	-15287	137	7	149	-57047	163	2
13	-599	25	2	73	-14431	85	5	151	-58967	253	3
17	-919	19	5	79	-16823	95	3	157	-63527	211	2
19	-1367	25	3	83	-17903	85	2	163	-67759	191	2
23	-1759	27	5	89	-20639	179	3	167	-74311	251	5
29	-2551	41	2	97	-25367	141	7	173	-76039	221	2
31	-2879	57	3	101	-26479	105	2	179	-81031	193	2
37	-3767	39	2	103	-27791	203	5	181	-83399	359	2
41	-4583	61	3	107	-29879	195	7	191	-94727	355	7
43	-5039	83	3	109	-30983	145	2	193	-95191	259	5
47	-6311	89	5	113	-33191	133	5	197	-98927	335	2
53	-7607	89	2	127	-41231	163	5	199	-99767	249	3

3.3.4 Isogeny Cycle

NTL has a `FindRoot()` function that helps find a root j_1 of H_D. From this first root, we compute $\Phi_{l_0}(X, j)$ and search its roots for the next root j_2 of H_D. We expect to find $l_0 + 1$ roots, but only two roots will correspond to j-invariants on the rim. To distinguish between them, we check the number of l_0-isogenies from each root. Exactly two roots will have $l_0 + 1$ l_0-isogenies, while the other $l_0 - 1$ roots will have only one l_0-isogeny defined from it. We continue with this process until we arrive back at j_1.

This cyclic sequence of roots of H_D gives us a l_0-isogeny cycle. Since we know that the composition of k_{rim} consecutive l_0-isogenies gives us an l-isogeny, we can reorder the l_0-isogeny cycle to obtain the l-isogeny cycle that we require.

Example 3.3.14. For $l = 5$, $D = -151$, and $h(D) = 7$. The first l-volcano has $t = 52, p = 4451$, and the first root of $H_D \in \mathbb{F}_p[X]$ is $j = 351$. From Example 3.3.13, we have $l_0 = 2$. The roots of $\Phi_{l_0}(X, j)$ are $65, 701, 2501$. We expect to find exactly two l_0-isogenies on the rim, and thus we need to distinguish between the three.

We accomplish this by finding the roots of $\Phi_{l_0}(X, 65)$, $\Phi_{l_0}(X, 701)$, $\Phi_{l_0}(X, 2501)$. Since $\Phi_{l_0}(X, 65)$ has only a single root (which must be 351) in $\mathbb{F}_p$, we conclude that 65 does not correspond to a j-invariant on the rim.

Repeating the process, we see that we have the l_0-isogeny cycle

$$351 \xrightarrow{2} 701 \xrightarrow{2} 2872 \xrightarrow{2} 2215 \xrightarrow{2} 901 \xrightarrow{2} 1582 \xrightarrow{2} 2501.$$

Since $k_{\text{rim}} = 3$, the l-isogeny cycle is

$$351 \xrightarrow{5} 2215 \xrightarrow{5} 2501 \xrightarrow{5} 2872 \xrightarrow{5} 1582 \xrightarrow{5} 701 \xrightarrow{5} 901.$$

3.4 On the Floor

3.4.1 Descending to the Floor

To find a curve E' on the floor that is l-isogenous to E on the rim, we need to define an l-isogeny $\phi : E \to E'$ explicitly. When l is prime, every l-isogeny is separable, and hence every isogeny ϕ with $|\ker \phi| = l$ has degree l. Furthermore, the kernel of a separable isogeny uniquely defines the isogeny. Thus for a given kernel (of a separable isogeny) C, we can use E/C in place of E' to denote the image of the isogeny. In our case, we only need to construct a subgroup of E of order l to define $E' = E/C$.

The order of $E(\mathbb{F}_p)$ is $p + 1 - t$. For a BLS l-volcano, we have chosen $p \equiv 1$ and $t \equiv 2 \pmod{l}$. Hence $p + 1 - t \equiv 0 \pmod{l}$, which means that $E(\mathbb{F}_p)$ contains a subgroup of order l. We can pick $P \in_R E$ until we get $Q := [\frac{p+1-t}{l}]P \neq \infty$, and use the cyclic group generated by Q as C, the kernel of an l-isogeny.

With the kernel C, we can use Vélu's formulae [24] to write down the Weierstrass equation for the curve E'_1, and thus the j-invariant $j'_1 := j(E'_1)$ that is l-isogenous to $j_1 := j(E)$ on the rim. If j'_1 is on the rim, we repeat the process with a new random point, until we obtain j'_1 not already on the rim.

3.4.2 Running along the Floor

Binary quadratic forms on the floor have discriminant l^2D and the class group on the floor has order $(l-1)h(D)$. To find the j-invariants on the floor of the l-volcano, we begin with $j'_1 := j(E')$ given by Vélu's formulae, and again find the roots $\tilde{j}$ of $\Phi_{l_0}(X, j'_1) \in \mathbb{F}_p[X]$ to enumerate the l_0-isogeny cycle, using the splitting of $\Phi_{l_0}(X, \tilde{j}) \in \mathbb{F}_p[X]$ to identity the curves on the floor.

Since the l_0-isogeny cycle on the floor can take either of two possible orientations, we need to check which orientation corresponds to that of the l-isogeny cycle on the rim of the l-volcano. Knowing that j_1 on the rim is l-isogenous to j'_1 on the floor, we check if the next curve in the l_0 isogeny cycles, namely j_2, j'_2, are l-isogenous. This can be accomplished by again finding an l-isogeny from j_2 that descends to the floor.

Once we have ascertained that the two l_0-isogeny cycles have the same orientation, we can traverse the cycles simultaneously to establish the incidence between the j-invariants on the rim and those on the floor. For each l_0-isogeny on the rim, we traverse the cycle on the floor by an l_0-isogeny on the floor to locate the next edge of the l-volcano. Every $h(D)$ consecutive l_0-isogenies on the floor will be an l^2-isogeny between two curves on the floor that are incident with the same curve on the rim.

Example 3.4.1. Continuing from Example 3.3.14, we find $j_1' := 2464$ to be a curve on the floor which is l-isogenous to $j_1 = 351$ on the surface. The roots of $\Phi_{l_0}(X, j_1')$ are $1180, 2138, 4221$. Since $\Phi_{l_0}(X, 2138)$ has only a single root in $\mathbb{F}_p$, 2138 does not correspond to a j-invariant on the floor. Suppose we choose $j_2' := 1180$ to be the next j-invariant in the l_0-isogeny cycle.

Repeating the process, we obtain the l_0-isogeny cycle

$$\begin{array}{ccccccccccccc}
2464 & \xrightarrow{2} & 1180 & \xrightarrow{2} & 3497 & \xrightarrow{2} & 2970 & \xrightarrow{2} & 676 & \xrightarrow{2} & 1502 & \xrightarrow{2} & 2843 & \xrightarrow{2} \\
3508 & \xrightarrow{2} & 3144 & \xrightarrow{2} & 945 & \xrightarrow{2} & 3188 & \xrightarrow{2} & 3341 & \xrightarrow{2} & 2087 & \xrightarrow{2} & 4397 & \xrightarrow{2} \\
2566 & \xrightarrow{2} & 3147 & \xrightarrow{2} & 291 & \xrightarrow{2} & 3328 & \xrightarrow{2} & 1868 & \xrightarrow{2} & 1064 & \xrightarrow{2} & 3345 & \xrightarrow{2} \\
2976 & \xrightarrow{2} & 2255 & \xrightarrow{2} & 3244 & \xrightarrow{2} & 1478 & \xrightarrow{2} & 2434 & \xrightarrow{2} & 4228 & \xrightarrow{2} & 4221 &
\end{array}$$

The seven l^2-isogeny cycles and the corresponding curve on the rim are

$$\begin{array}{ccccccccccc}
351 & \xrightarrow{5} & 2464 & \xrightarrow{25} & 3508 & \xrightarrow{25} & 2566 & \xrightarrow{25} & 2976 \\
2501 & \xrightarrow{5} & 1180 & \xrightarrow{25} & 3144 & \xrightarrow{25} & 3147 & \xrightarrow{25} & 2255 \\
1582 & \xrightarrow{5} & 3497 & \xrightarrow{25} & 945 & \xrightarrow{25} & 291 & \xrightarrow{25} & 3244 \\
901 & \xrightarrow{5} & 2970 & \xrightarrow{25} & 3188 & \xrightarrow{25} & 3328 & \xrightarrow{25} & 1478 \\
2215 & \xrightarrow{5} & 676 & \xrightarrow{25} & 3341 & \xrightarrow{25} & 1868 & \xrightarrow{25} & 2434 \\
2872 & \xrightarrow{5} & 1502 & \xrightarrow{25} & 2087 & \xrightarrow{25} & 1064 & \xrightarrow{25} & 4228 \\
701 & \xrightarrow{5} & 2843 & \xrightarrow{25} & 4397 & \xrightarrow{25} & 3345 & \xrightarrow{25} & 4221
\end{array}$$

3.5 Putting it Together

3.5.1 Evaluation on the Rim

Recall that the BLS l-volcano has $lh(D)$ vertices. There are $h(D)$ vertices on the rim, which we refer to as j_i, $i = 1, \cdots, h(D)$, that forms an l-isogeny cycle. Each vertex j_i on the rim is incident to $l-1$ vertices on the floor, which we label as $j_{i,s}$, with $s = 1, \cdots, l-1$. Upon establishing the j-invariants at each vertex of the l-volcano, we can write down $h(D)$ polynomials

$$f_i = (X - j_{i-1})(X - j_{i+1}) \prod_{s=1}^{l-1} (X - j_{i,s}), \tag{3.10}$$

for $i = 1, \cdots, h(D)$, taking $j_0 := j_{h(D)}$. Each of these polynomial has degree $l + 1$, and is precisely the value of $\Phi_l(X, Y) \in \mathbb{F}_p[X]$ evaluated at $Y = j_i$, i.e. $\Phi_l(X, j_i)$.

Since we have $h(D) \geqslant l + 2$, we have the values of $(\Phi_l(X))(Y) \in (\mathbb{F}_p[X])[Y]$ evaluated on at least $l+2$ instances of Y. The degree of $\Phi_l \in (\mathbb{F}_p[X])[Y]$ is $l+1$, so these are sufficient for interpolation. Thus we obtain $\Phi_l(X, Y) \in \mathbb{F}_p[X, Y]$.

Example 3.5.1. Continuing from Example 3.4.1, we write down the seven polynomials defined over $\mathbb{F}_{4451}$.

$$\begin{aligned}
\Phi_l(X, 351) &:= (X - 901)(X - 2215)(X - 2464)(X - 3508)(X - 2566)(X - 2976) \\
\Phi_l(X, 901) &:= (X - 351)(X - 701)(X - 2970)(X - 3188)(X - 3328)(X - 1478) \\
\Phi_l(X, 701) &:= (X - 901)(X - 1582)(X - 2843)(X - 4397)(X - 3345)(X - 4221) \\
\Phi_l(X, 1582) &:= (X - 701)(X - 2872)(X - 3497)(X - 945)(X - 291)(X - 3244) \\
\Phi_l(X, 2872) &:= (X - 1582)(X - 2501)(X - 1502)(X - 2087)(X - 1064)(X - 4228) \\
\Phi_l(X, 2501) &:= (X - 2872)(X - 2215)(X - 1180)(X - 3144)(X - 3147)(X - 2255) \\
\Phi_l(X, 2215) &:= (X - 2501)(X - 351)(X - 676)(X - 3341)(X - 1868)(X - 2434)
\end{aligned}$$

After interpolation, we obtain $\Phi_5(X, Y)$ over $\mathbb{F}_{4451}$:

$$\begin{aligned}
X^6 &+ 4450X^5Y^5 + 3720X^5Y^4 + 2433X^5Y^3 + 3499X^5Y^2 + 70X^5Y + 3927X^5 \\
&+ 3720X^4Y^5 + 3683X^4Y^4 + 2348X^4Y^3 + 2808X^4Y^2 + 3745X^4Y + 233X^4 \\
&+ 2433X^3Y^5 + 2348X^3Y^4 + 2028X^3Y^3 + 2025X^3Y^2 + 4006X^3Y + 2211X^3 \\
&+ 3499X^2Y^5 + 2808X^2Y^4 + 2025X^2Y^3 + 4378X^2Y^2 + 3886X^2Y + 2050X^2 \\
&+ 70XY^5 + 3745XY^4 + 4006XY^3 + 3886XY^2 + 905XY + 2091X \\
Y^6 &+ 3927Y^5 + 233Y^4 + 2211Y^3 + 2050Y^2 + 2091Y + 2108
\end{aligned} \tag{3.11}$$

3.5.2 Chinese Remainder Theorem

We use a_{ij} and $\bar{a}_{ij,r}$ to denote the coefficients of X^iY^j in $\Phi_l(X,Y)$ in $\mathbb{Z}[X,Y]$ and $\mathbb{F}_{p_r}[X,Y]$ respectively. In Section 3.2.1, we alluded to the need to find a set of p_r so that $\prod_r p_r \geqslant 2\,\mathrm{MAX}_l$, where $\mathrm{MAX}_l := \max_{0\leqslant i,j\leqslant l+1} |a_{ij}|$ denotes the maximum that the absolute value of the coefficients of $\Phi_l(X,Y)$ can take. The upper bound below, as proven in [3], allows us to decide on the number of p_r's needed a priori.

Theorem 3.5.2. *A logarithmic upper bound for the explicit height* MAX_l *is*

$$\log(\mathrm{MAX}_l) \leqslant 6l\log l + 16l + 14\sqrt{l}\log l \tag{3.12}$$

After the interpolation step, we have the system of congruence equations

$$\bar{a}_{ij,r} \equiv a_{ij} \pmod{p_r}$$

from the expressions for $\Phi_l(X,Y) \in \mathbb{F}_{p_r}[X,Y]$. Solving this system using the Chinese Remainder Theorem gives us the solution

$$\bar{a}_{ij} \equiv a_{ij} \pmod{\prod_r p_r},$$

restricting $|\bar{a}_{ij}| \leqslant \frac{1}{2}\prod_r p_r$. Since $|a_{ij}| < \mathrm{MAX}_l \leqslant \prod_r p_r$, we must have $a_{ij} = \bar{a}_{ij}$, which means that we recover the coefficient $a_{ij} \in \mathbb{Z}$.

Chapter 4

Schoof-Elkies-Atkin Algorithm

In this chapter we will describe and explain the workings of the Schoof-Elkies-Atkin algorithm [17][8], as well as discuss some of the considerations involved during the implementation. Most of the content follows the exposition in Chapter 17.2 of [5], with details of the Match-Sort algorithm from Chapter VII of [1]. Besides using NTL 5.5.2 [19] for its support for generic finite field and polynomial arithmetic, some routines for factorization and Frobenius computations [18][25] were used.

4.1 Use of Modular Polynomials

In this section, we will look at the algorithms that Atkin and Elkies used to extract information from the l-th modular polynomial to decide if a prime l is Atkin or Elkies.

Proposition 4.1.1. *Let $E/\mathbb{F}_p$ be an elliptic curve with j-invariant $j \neq 0$ or 1728. Then*

- *the polynomial $\Phi_l(X, j)$ has a zero $\tilde{\jmath} \in \mathbb{F}_{p^r}$ if and only if the kernel C of the isogeny $\psi : E \to E/C$ is a one-dimensional eigenspace of ϕ_p^r in $E[l]$*
- *the polynomial $\Phi_l(X, j)$ splits completely in $\mathbb{F}_{p^r}[X]$ if and only if ϕ_p^r acts as a scalar matrix in $E[l]$.*

Proposition 4.1.1 tells us that l is an Elkies prime if and only if $\Phi_l(X, j)$ has a zero in $\mathbb{F}_p$. Hence we can study the splitting of $\Phi_l(X, j)$ over $\mathbb{F}_p$ to classify primes as Elkies or Atkin. Atkin's theorem below takes this one step further by characterizing the possible factorizations, thus providing us with a sufficient condition to do the classification.

Remark We will be using classical modular polynomials in this thesis for SEA, even though there exist other types of modular polynomials that are used in practice, such as Weber and Atkin polynomials which can perform the same task. In fact, these alternatives have much smaller coefficients, and are thus easier to store. We have used the classical polynomials here as the theoretical justification for their use is more obvious.

4.1.1 Atkin's Classification Theorem

Theorem 4.1.2. (Atkin) *Let $E/\mathbb{F}_p$ be an ordinary elliptic curve with j-invariant $j \neq 0$ or 1728. Let $\Phi_l(X, j) = f_1 f_2 \cdots f_s$ be the factorisation of $\Phi_l(X, j) \in \mathbb{F}_p[X]$ into irreducible polynomials. Then we have three possible cases for the degrees of $f_1, f_2, \cdots, f_s$, depending on the relation between the discriminant $\Delta = t^2 - 4p$ and l.*

1. $\Delta \equiv 0 \pmod{l}$: *the degrees are (1,l) and $r = l$, or $(1, 1, \cdots, 1)$ and $r = 1$.*
2. $\Delta \in QR_l - \{0\}$: *the degrees are $(1, 1, r, r, \cdots, r)$ and $r \mid l - 1$, and ϕ_p acts on $E[l]$ as $\begin{bmatrix} \lambda & 0 \\ 0 & \mu \end{bmatrix}$, where $\lambda, \mu \in \mathbb{F}_l^*$ are the roots of $Z^2 - t_l Z + p_l$.*
3. $\Delta \in \overline{QR}_l$: *the degrees are $(r, r, \cdots, r)$ and $r > 1$, $r \mid l + 1$.*

Remark In cases 1 and 2, l is an Elkies prime and in case 3, l is an Atkin prime. Note that r in the above theorem is the order of the Frobenius endomorphism ϕ_p as an element of $\mathrm{PGL}_2 \mathbb{F}_l$, i.e. $\phi_p^r = \mathrm{id}$. We refer to r as the order of the Frobenius endomorphism on $\mathrm{Aut}(E[l])$. It can be shown that $r > 1$ is the smallest integer so that $\lambda^r = \mu^r$, where λ, μ are the roots of the characteristic polynomial for ϕ_p.

Propostion 4.1.1 tells us that ϕ_p has an eigenspace of dimension one defined over $\mathbb{F}_p$ if and only if $\Phi_l(X, j)$ has a root in $\mathbb{F}_p$. Theorem 4.1.2 tells us that the factorisation of $\Phi_l(X, j)$ indicates if l is an Elkies or Atkin prime.

To classify primes as Atkin or Elkies, it is not necessary to factorize $\Phi_l(X, j)$ over $\mathbb{F}_p$; it suffices to determine if $\Phi_l(X, j)$ has a root in $\mathbb{F}_p$. This is equivalent to finding

$$g(X) := \gcd(\Phi_l(X, j), X^p - X) = \gcd(\Phi_l(X, j), X^p - X \bmod \Phi_l(X, j)) \tag{4.1}$$

(similar to Equation 2.16). Note that the latter expression allows us to exploit existing speed ups in performing modular exponentiation of X over the ring $\mathbb{F}_p[X]/(\Phi_l(X, j))$.

We describe the algorithm for Atkin's classification below, assuming that we have the algorithm ModularExponentiation$(a(X), e, f(X))$ which returns $a(X)^e \bmod f(X)$.

Algorithm: AtkinClassification
Input: The l-th classical modular polynomial $\Phi_l(X, j)$ over $\mathbb{F}_p$ evaluated at $Y = j$.
Output: ATKIN or ELKIES.

1. Compute $h(X) = \text{ModularExponentiation}(X, p, \Phi_l(X, j))$.
2. Compute $g(X) = \gcd(h(X) - X, \Phi_l(X, j))$.
3. If $\deg(g) = 0$, return ATKIN.
4. Otherwise, return ELKIES.

Proposition 4.1.3. (Atkin) *Let r be as defined in Theorem 4.1.2. Then t_l satisfies the equation*

$$t_l^2 \equiv p(\xi + \xi^{-1})^2 \pmod{l}, \tag{4.2}$$

for some primitive r'-th root of unity $\xi \in \overline{\mathbb{F}}_l$, where $r' = r$ if r is odd, $r' = 2r$ if r is even.

Remark In some sources, such as equation 17.9 in [5] and Proposition 6.2 in [17], ξ has been defined as a r-th root of unity. However, if we have $r = 2$ for some Atkin prime l (i.e. case 3), if ξ is a 2nd root of unity, then $\xi^{-1} = \xi = -1$. That gives us $t_l^2 \equiv 4p \pmod{l}$, which means $\Delta = t_l^2 - 4p \equiv 0 \pmod{l}$, which is a contradiction. This motivates a second look at the proof for the above proposition.

Proof. Since λ, μ are the roots of $Z^2 - t_l Z + p_l$, we have $\lambda\mu \equiv p \pmod{l}$ and $\lambda + \mu \equiv t \pmod{l}$. Then

$$\lambda^{2r} = \lambda^r \mu^r = (\lambda\mu)^r \equiv p^r \pmod{l},$$

and so $\lambda^2 \equiv \zeta p \pmod{l}$ for some primitive r-th root of unity $\zeta \in \overline{\mathbb{F}}_l$. Therefore

$$t^2 \equiv (\lambda + \mu)^2 \equiv (\lambda + p/\lambda)^2 \equiv \lambda^2 + 2p + p^2\lambda^{-2} \equiv \zeta p + 2p + \zeta^{-1}p \equiv p(\zeta + 2 + \zeta^{-1}) \pmod{l}. \tag{4.3}$$

Let $\xi^2 = \zeta$, which implies that ξ is a primitive r'-th root of unity, where $r' = r$ or $r' = 2r$. Then we can rewrite the final equation as

$$t^2 \equiv p(\xi + \xi^{-1})^2$$

Case 1: r odd $\xi = \pm\zeta^{\frac{r+1}{2}}$, since these are two solutions to

$$\xi^2 = (\pm\zeta^{\frac{r+1}{2}})^2 = \zeta^{r+1} = \zeta^r\zeta = \zeta.$$

We then have $\xi^r = (\zeta^{\frac{r+1}{2}})^r = (\zeta^r)^{\frac{r+1}{2}} = 1$, i.e. ξ is an r-th root of unity. Since ξ is a primitive r'-th root, $r' \mid r$. But $r \mid r'$ in our definition of r', so $r' = r$ if r is odd.

Case 2: r even Suppose ξ is an r-th root of unity. Thus we can write $\xi = \zeta^i$ for some integer i, and

$$\zeta = \xi^2 = (\zeta^i)^2 = \zeta^{2i}.$$

Thus $\zeta^{2i-1} = 1$, which implies $r \mid 2i-1$. But this contradicts r being even. Therefore ξ is not an r-th root of unity, and so $r' = 2r$ if r is even.

It should be noted that $\varphi(r') = \varphi(r)$, since $\varphi(2^k) = 2^{k-1}$ for $k \geqslant 1$. Hence the error in the definition for ξ in 4.2 does not change the number of possible values for t_l. Alternatively, one can use 4.3 instead to avoid considering cases for odd and even r. □

Proposition 4.1.4. (Atkin) *Let $E/\mathbb{F}_p$ be an ordinary elliptic curve with j-invariant $j \neq 0$ or 1728. Let s be the number of irreducible factors of $\Phi_l(X, j)$ in $\mathbb{F}_p[X]$. Then s satisfies the equation*

$$(-1)^s = \left(\frac{p}{l}\right). \tag{4.4}$$

4.2 Elkies Primes

4.2.1 Elkies's Algorithm

We have explained at the beginning of this chapter that when l is an Elkies prime, the eigenspaces of ϕ_p acting on $E[l]$ are subgroups of order l that are stable under ϕ_p and correspond to eigenvalues in $\mathbb{F}_l$. For the λ-eigenspace C, there is a divisor $F(X) \in \mathbb{F}_p[X]$ of the l-th division polynomial $\Psi_l(X)$. We refer to this polynomial as the kernel polynomial. This divisor is of degree $(l-1)/2$ and its zeroes are the $(l-1)/2$ distinct x-coordinates of the points in C.

Given the λ-eigenspace C, we check, for each of $\lambda' = 1, \cdots, l-1$, if the relation

$$\phi_p(x, y) = (x^p, y^p) = \lambda' \cdot (x, y)$$

holds for all $(x, y) \in E[l]$. The value of t_l is then computed as

$$t_l = \lambda + p_l \cdot \lambda^{-1} \bmod l.$$

Instead of computing separately for each of the l points in C, the polynomial $F(X)$ will allow us to check across all points in C simultaneously. Note that we can use a Baby-Step Giant-Step (BSGS) approach to lower the number of computations from $O(l)$ to $O(\sqrt{l})$.

Computing torsion points The x-coordinates of the l-torsion points in the associated subgroup satisfies the kernel polynomial $F(X)$. Since the torsion points lie on $E : Y^2 = X^3 + aX + b$, their coordinates satisfy $Y^2 - X^3 - aX - b$ as well. Therefore we perform elliptic curve arithmetic on l-torsion points over the ring

$$R := \frac{\mathbb{F}_p[X, Y]}{(F(X), Y^2 - X^3 - aX - b)}. \tag{4.5}$$

We store a torsion point in the form $(x(X), y(X), z(X))$, where $x(X), y(X), z(X) \in \mathbb{F}[X]$ and $z(X)$ is 0 or 1. While Y is not stored explicitly, we define our point arithmetic with the assumption that the polynomial in the y-coordinate is $Y \cdot y(Y)$. For example, $(x, y, 1)$ is stored as $(X, 1, 1)$, and the point at infinity is stored as $(1, 1, 0)$.

Torsion point doubling To obtain $(x'(X), y'(X), z'(X)) := [2](x(X), y(X), z(X))$ (assuming not point at infinity or point of order two), we use the following equations, computed modulo $F(X)$:

$$\begin{aligned} y^{-2}(X) &= \left(y(X)^2 \cdot (X^3 + aX + b)\right)^{-1} && (4.6)\\ m(X) &= 3x(X)^2 + a && (4.7)\\ x'(X) &= \frac{m(X)^2 \cdot y^{-2}(X)}{4} - 2x(X) && (4.8)\\ y'(X) &= y(X) \cdot \left(\frac{m(X) \cdot y^{-2}(X)}{2} \cdot (x(X) - x'(X)) - 1\right) && (4.9)\\ z'(X) &= 1 && (4.10) \end{aligned}$$

Torsion point addition To obtain $(x''(X), y''(X), z''(X)) := (x'(X), y'(X), z'(X)) + (x(X), y(X), z(X))$ (assuming distinct points that are not inverses, and no points at infinity), we use the following equations, computed modulo $F(X)$:

$$\begin{aligned} m(X) &= \frac{y'(X) - y(X)}{x'(X) - x(X)} && (4.11)\\ x''(X) &= m(X)^2 \cdot (X^3 + aX + b) - x(X) - x'(X) && (4.12)\\ y''(X) &= m(X) \cdot (x(X) - x''(X)) - y(X) && (4.13)\\ z''(X) &= 1 && (4.14) \end{aligned}$$

There exists also, an isogenous elliptic curve $\tilde{E} = E/C$ and a separable isogeny of degree l between E and $\tilde{E}$. From Theorem 3.1.18, we see that $j(\tilde{E})$ is a root of $\Phi_l(X, j) = 0$ in $\mathbb{F}_p$, which we can efficiently compute. The following theorem provides an explicit equation for $\tilde{E}$.

Theorem 4.2.1. (Elkies) *Let $E/\mathbb{F}_p$ be an ordinary elliptic curve with j-invariant $j \neq 0$ or 1728, where $E : y^2 = x^3 + a_4x + a_6$. Let $\Phi_{l,X}$ and $\Phi_{l,Y}$ denote the partial derivative of $\Phi_l(X, Y)$ with respect to X and Y respectively. Suppose $\tilde{E}$ is l-isogenous to E over $\mathbb{F}_p$, with $\tilde{\jmath}$ being its j-invariant. Then a Weierstrass equation for $\tilde{E}$ is given by*

$$\tilde{E} : y^2 = x^3 + \tilde{a}_4 x + \tilde{a}_6,$$

where

$$\tilde{a}_4 = -\frac{1}{48}\frac{\tilde{\jmath}'^2}{\tilde{\jmath}(\tilde{\jmath} - 1728)} \qquad \text{and} \qquad \tilde{a}_6 = -\frac{1}{864}\frac{\tilde{\jmath}'^3}{\tilde{\jmath}(\tilde{\jmath} - 1728)},$$

$\tilde{\jmath}' \in \mathbb{F}_p$ *is given by*

$$\tilde{\jmath}' = -\frac{18}{l}\frac{a_6}{a_4}\frac{\Phi_{l,X}(j,\tilde{\jmath})}{\Phi_{l,Y}(j,\tilde{\jmath})}j.$$

Theorem 4.2.2. (Elkies) *Let* $E_4 = -48a_4, E_6 = 864a_6$ *and* $\tilde{E}_4 = -48\tilde{a}_4, \tilde{E}_6 = 864\tilde{a}_6$. *Then the sum* $p_1 \in \mathbb{F}_p$ *of the* x*-coordinates of the non-zero points in* C *is given by*

$$p_1 = \frac{l}{2}J + \frac{l}{4}\left(\frac{E_4^2}{E_6} - l\frac{\tilde{E}_4^2}{\tilde{E}_6}\right) + \frac{l}{3}\left(\frac{E_6}{E_4} - l\frac{\tilde{E}_6}{\tilde{E}_4}\right),$$

where J *is defined as*

$$J = \frac{j'^2\Phi_{l,XX}(j,\tilde{\jmath}) + 1lj'\tilde{\jmath}'\Phi_{l,XY}(j,\tilde{\jmath}) + l^2\tilde{\jmath}'^2\Phi_{l,YY}(j,\tilde{\jmath})}{j'\Phi_{l,X}(j,\tilde{\jmath})}$$

with $j' = -jE_6/E_4$ *and* $\tilde{\jmath}' = -\tilde{\jmath}\tilde{E}_6/\tilde{E}_4$; $\Phi_{l,XY}$ *is shorthand for* $\frac{\partial}{\partial X}\frac{\partial}{\partial Y}\Phi_l$.

Computing partial derivatives NTL does not have a definition of multivariate polynomials, and has only a function for differentiation for univariate polynomials. However, since we seek to compute the partial derivative of $\Phi_l(X,Y)$ with respect to X, followed by evaluation at $Y = \tilde{\jmath}$, we can obtain the desired partial derivative by computing

$$\Phi_{l,X}(X,\tilde{\jmath}) = \frac{d}{dX}\left(\Phi_l(X,\tilde{\jmath})\right) \tag{4.15}$$

We can compute $\Phi_{l,Y}(j,\tilde{\jmath}), \Phi_{l,XX}(j,\tilde{\jmath}), \Phi_{l,YY}(j,\tilde{\jmath})$ similarly.

$\Phi_l(X,Y)$ is a symmetric polynomial with degree $l+1$, and can be stored in roughly half the space since the coefficients of $X^{i_1}Y^{i_2}$ and $X^{i_2}Y^{i_1}$ are the same. As $\Phi_{l,XY}$ is the only partial derivative that remains symmetric, $\Phi_{l,XX}(j,\tilde{\jmath})$ is the only partial derivative that can be computed without increasing the number of coefficients stored.

Kernel polynomial Now that we have the Weierstrass form for the two isogenous curves E and $\tilde{E} = E/C$ and the value of p_1, we would like to compute the degree $(l-1)/2$ polynomial

$$g_l(X) = \prod_{\pm P \in C - \{\infty\}} (X - x(P)).$$

Instead of using the curves $E/\mathbb{F}_p$ and $\tilde{E}/\mathbb{F}_p$, we consider their analogues $E/\mathbb{C}$ and $\tilde{E}/\mathbb{C}$, as it is easier to define the isogeny between $E/\mathbb{C}$ and $\tilde{E}/\mathbb{C}$.

Let $\hat{a}_4 = l^4\tilde{a}_4$ and $\hat{a}_6 = l^6\tilde{a}_6$. We define the curve $\hat{E} : y^2 = x^3 + \hat{a}_4 x + \hat{a}_6$, to which we can associate the reduced Weierstrass $\wp$-function by

$$\wp(z) = \frac{1}{z^2} + \sum_{k=1}^{\infty} c_k z^{2k} \tag{4.16}$$

$$c_1 = -\frac{a_4}{5}, \quad c_2 = -\frac{a_6}{7}, \quad c_k = \frac{3}{(k-2)(k+3)} \sum_{j=1}^{k-2} c_j c_{k-1-j} \quad \text{for } k \geqslant 3.$$

A similar function $\hat{\wp}$ for $\hat{E}$ is defined.

Theorem 4.2.3. *Let $g_l(X) \in \mathbb{F}_p[X]$ be the divisor of $\Psi_l(X)$ having the x-coordinates of C as its zeroes. Let $\psi_l : E \to \tilde{E}$ be the isogeny with $\ker(\Psi_l) = C$. Then*

$$z^{l-1} g_l(\wp(z)) = \exp\left(-\frac{1}{2} p_1 z^2 - \sum_{k=1}^{\infty} \frac{\hat{c}_k - l c_k}{(2k+1)(2k+2)} z^{2k+2} \right). \tag{4.17}$$

We inductively compute the coefficients c_k and $\tilde{c}_k$ from a_4, a_6 and $\tilde{a}_4, \tilde{a}_6$ respectively. Although equations (4.16) and (4.17) involve infinite series, it suffices to compute the infinite series with k up to $d = (l-1)/2$, the degree of the kernel polynomial $g_l(X)$.

With the value of p_1 from Theorem 4.6, we can compute the polynomial on the RHS of (4.17), storing its coefficients as a vector b of length $d+1$. Next, we compute $z^{2(d-i)}\wp(z)^i$ for each $i = 0, \cdots, d$, and store its coefficients as the i-th row of a $(d+1)$-by-$(d+1)$ square matrix M. Note that M, by its definition, will always be invertible. $M^{-1}b$ gives us the vector with the coefficients of $g_l(X) \in \mathbb{F}_p[X]$ as its entries.

Note that the formula has terms of the form $(k-2)(2k+3)$ in the denominator, which means that the formula would fail if $k \geqslant (p-3)/2$. This means that for finite fields of small characteristic, alternative methods to compute $g_l(X)$ are required.

4.3 Atkin Primes

4.3.1 Atkin's Algorithm

In Atkin's classification theorem, we used r to the order of the Frobenius endormorphism on $\mathrm{Aut}(E[l])$, which is equivalently the smallest positive integer such that $\Phi_l(X, j)$ splits over $\mathbb{F}_{p^r}$. In the case where l is Atkin, we need to compute r to determine the set of possible values for t_l.

For each $i = 2, 3, \cdots$, we compute

$$g_i(X) := \gcd(\Phi_l(X, j), X^{p^i} - X)$$

until we find the value i such that $g_i(X) = \Phi_l(X, j)$, upon which we set $r = i$. Since r divides $l + 1$ and $s = (l + 1)/r$, we can limit the values of i to the divisors of $l + 1$ for which

$$(-1)^{(l+1)/i} = \left(\frac{p}{l}\right).$$

With r and thus r' determined, there are $\varphi(r')$ choices for ξ (where φ is Euler's totient function) in Equation 4.2. Since

$$\xi + \xi^{-1} = \left(\xi^{-1}\right) + \left(\xi^{-1}\right)^{-1},$$

there are $\varphi(r')/2$ choices for t_l^2 and thus

$$\rho_l := \varphi(r') < 2l$$

possible values for t_l.

To find the correct value of t, we repeat this computation for various small Atkin primes l and search for the correct value of t using a modified version of the Baby-Step Giant-Step algorithm called the Match-Sort algorithm, described in Section 4.4.

The Match-Sort algorithm combines the known values of t_l for Elkies primes and the set of possible t_l for Atkin primes, to solve a discrete logarithm problem on the restricted set of values.

4.3.2 Computing Order of Frobenius

The most obvious approach to computing r, the order of the Frobenius endormorphism on $\mathrm{Aut}(E[l])$ is to perform ModularExponentiation$(X, p^i, \Phi_l(X, j))$ for each i that divides $l+1$ to get $h_i(X) = X^{p^i}$, followed by computing $g_i(X) = \gcd(\Phi_l(X, j), h_i(X) - X)$.

Algorithm: OrderFrobenius1$(\Phi_l(X, j))$
Input: For an Atkin prime l, the l-th classical modular polynomial $\Phi_l(X, j)$ over $\mathbb{F}_p$.
Output: r so that $\gcd(\Phi_l(X, j), X^{p^r} - X) = \Phi_l(X, j)$.

1. compute $b := \left(\frac{p_l}{l}\right)$.
2. for $i = 1, \cdots, l+1$ do
3. if $i \mid l+1$ and $(-1)^i = b$ do
4. compute $h(X) = \text{ModularExponentiation}(X, p^i, \Phi_l(X, j))$
5. compute $g(X) = \gcd(\Phi_l(X, j), h(X) - X)$
6. if $g(X) = \Phi_l(X, j)$ do
7. return i.

Suppose we use Square-and-Multiply to perform modular exponentiation. We can see that OrderFrobenius1 repeats the computation of $X^2, (X^2)^2, \cdots$ for each factor of $l+1$ that is considered, resulting in a large number of repeated computations when the number of factors are large.

Instead of viewing each exponentiation p^i individually, we can consider them as performing a sequence of Frobenius maps, and by writing

$$X^{p^i} = \left(X^{p^k}\right)^{p^{i-k}}$$

we can perform the exponentiation iteratively for increasing factors of $l+1$, thus eliminating repeated computations. Furthermore, we can utilise the addition chain for p that we found for the classification algorithm to reduce the time for each Frobenius map.

Algorithm: OrderFrobenius2($\Phi_l(X, j)$)
Input: For Atkin prime l, l-th classical modular polynomial $\Phi_l(X, j)$ over $\mathbb{F}_p$.
Output: r so that $\gcd(\Phi_l(X, j), X^{p^r} - X) = \Phi_l(X, j)$.

1. compute $b := \left(\frac{p_l}{l}\right)$.
2. set $k = 0$ and $h(X) = X$.
3. for $i = 1, \cdots, l + 1$ do
4. if $i \mid l + 1$ and $(-1)^i = b$ do
5. for $c = 1, \cdots, i - k$ do
6. compute $h(X) = \text{ModularExponentiation}(h(X), p, \Phi_l(X, j))$.
7. set $k = i$;
8. compute $g(X) = \gcd(\Phi_l(X, j), h(X) - X)$.
9. if $g_i(X) = \Phi_l(X, j)$ do
10. return i.

In NTL, Victor Shoup has implemented the subroutine (Algorithm 3.1 in [25]) for computing iterated Frobenius maps, that is X^{p^r} modulo a fixed polynomial F defined over $\mathbb{F}_p$. The main idea behind the algorithm is to represent the Frobenius image X^p as an element β in the ring $\mathbb{F}_p[X]/(F)$, and compute $a(X)^p$ by evaluating $a(\beta)$. This subroutine provides us with an alternative approach to finding the Frobenius order r.

4.4 Match-Sort Algorithm

4.4.1 Combining Information

After we have computed the values of (t_l, l) for Elkies primes and (r', l) for Atkin primes, the Match-Sort algorithm is used combine the information gathered from these primes to compute the value of $t \in \mathbb{Z}$. We will use notations similar to that in [1] to describe the algorithm.

Let $\mathcal{E}, \mathcal{A}$ denote the set of Elkies and Atkin primes respectively. We define $m_E := \prod_{l \in \mathcal{E}} l$ and $t_E := t \bmod m_E$. We partition the Atkin primes into two sets $\mathcal{A}_1, \mathcal{A}_2$ (i.e. $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ and $\varnothing = \mathcal{A}_1 \cap \mathcal{A}_2$). For each $\mathcal{A}_i$, $i = 1, 2$, we define $m_i := \prod_{l \in \mathcal{A}_i} l$ and $t_i := t \bmod m_i$. To minimise the number of computations to generate the points in each list we will partition the Atkin primes so that $1 \leqslant |\mathcal{A}_1|/|\mathcal{A}_2| \leqslant 2$. Putting these definitions together, we can write

$$t = t_E + m_E(r_1 m_2 + r_2 m_1) \tag{4.18}$$

for some integers r_1, r_2. Since

$$t_1 \equiv t \equiv t_E + m_E m_2 r_1 \pmod{m_1} \quad \text{and} \quad t_2 \equiv t \equiv t_E + m_E m_1 r_2 \pmod{m_2}, \tag{4.19}$$

we have the relations

$$r_1 \equiv \frac{t_1 - t_E}{m_E m_2} \pmod{m_1} \quad \text{and} \quad r_2 \equiv \frac{t_2 - t_E}{m_E m_1} \pmod{m_2}, \tag{4.20}$$

We can show (see Lemma VII.10 of [1]) that since $t_E < m_E$, if $|r_1| \leqslant \frac{m_1 - 1}{2}$, then $|r_2| \leqslant m_2$.

For each Atkin prime $l \in \mathcal{A}_i$, $i = 1, 2$, the value of r' allows us to find a set of ρ_l possible values for t_l. We create the list τ_i of possible values for $t_i \bmod m_i$ for each $i = 1, 2$ via Chinese Remainder Theorem. From these we can compute a list R_i of $\prod_{l \in \mathcal{A}_i} \rho_l$ possible values for r_i using equation 4.20. We maintain both R_i's as increasing lists, sorted by the absolute value of r_i. Each possible pair for (r_1, r_2) gives us a possible value for t via 4.18, and we can find the correct value by performing Shanks' Baby-Step-Giant-Step (BSGS) algorithm with R_1 and R_2 as the set of possible values for the baby-step and giant-step respectively.

The order of a point $P \in E(\mathbb{F}_p)$ divides $p+1-t$, so we can use the condition $[p+1-t]P = \infty$ with a random point P to check if a value of t is correct.

4.4.2 Baby-Step Giant-Step

Baby-Step The baby-step in the algorithm consists of computing, for each possible value of r_1,

$$Q_{r_1} = [p + 1 - t_E]P - [r_1]([m_2 m_E]P). \tag{4.21}$$

In the above equation, the values $p + 1 - t_E$ and $m_2 m_E$ are constant for all values for r_1, so it is possible to compute $[p + 1 - t_E]P$ and $[m_2 m_E]P$ in advance. Therefore the computation of Q_{r_1} is subtracting r_1 copies of $[m_2 m_E]P$ from $[p + 1 - t_E]P$.

If the list R_1 is sorted, with the i-th entry denoted as $r_{1,i}$, then we can write

$$Q_{r_{1,i}} = Q - [r_{1,i}]([m_2 m_E]P) \tag{4.22}$$

and compute them by computing $r_{1,i} m_2 m_E P$ recursively. Each point Q_{r_1} will be converted to an affine point and stored in a list $\mathcal{Q}_1$ of (Q_{r_1}, r_1) sorted by their X-coordinates.

Giant-Step The giant-step in the algorithm consists of computing, for each possible value of r_2,

$$Q_{r_2} = [r_2]([m_1 m_E]P). \tag{4.23}$$

Like in the baby-step, the point $[m_1 m_E]P$ can be computed in advance. Also, with R_2 sorted, we can compute each $Q_{r_{2,i+1}}$ with a scalar multiplication of magnitude $|r_{2,i+1}| - |r_{2,i}|$. With the list $\mathcal{Q}_1$ sorted, we can use binary search to check if a match exists for each Q_{r_2}.

If no match exists for all Q_{r_2}, then we repeat the baby-step giant-step procedure with a different randomly generated point $P \in E$. Once a match is found, we can use 4.18 with the values of r_1 and r_2 corresponding to the match to determine the value of t.

4.4.3 Complexity of Match-Sort Algorithm

Full Match-Sort Algorithm
Input: t_E, m_E, (T_l, l) for each prime $l \in \mathcal{A}$.
Output: Trace of Frobenius t.

1. Compute $t_E = t \bmod m_E$ from $(\{t_l\}, l)$, $l \in \mathcal{E}$ via CRT.
2. Sort the pairs $(T_l, l), l \in \mathcal{A}$ by decreasing size.
3. Let $n_1 = n_2 = 1$, $\mathcal{A}_1 = \mathcal{A}_2 = \varnothing$.
4. For $l \in \mathcal{A}$, if $n_1 \leqslant n_2$, $\mathcal{A}_1 = \mathcal{A}_1 \cup l$, otherwise $\mathcal{A}_2 = \mathcal{A}_2 \cup l$.
5. Compute sets τ_i from $\{(T_l, l) : l \in \mathcal{A}_i\}$ for $i = 1, 2$.
6. Compute sets R_i from τ_i for $i = 1, 2$, restricting $|r_1| \leqslant \lfloor m_1/2 \rfloor$.
7. Sort each R_i by the absolute values of their entries, as an increasing sequence.
8. Choose random point $P \in E$ and set $\mathcal{Q}_1 = \varnothing$, $r_1 = 0$.
9. Compute $Q := [p + 1 - t_E]P$, $m_E P$, $m_1 m_E P$ and $m_2 m_E P$, in affine coordinates.
10. For $i = 1, \cdots, |R_1|$,
    (a) Compute $[r_{1,i}]m_2 m_E P$ and $Q_{r_{1,i}} := Q - [r_{1,i}]m_2 m_E P$.
    (b) Set $\mathcal{Q}_1 = \mathcal{Q}_1 \cup \{Q_{r_{1,i}}, r_{1,i}\}$.
11. Convert $\mathcal{Q}_1$ to affine coordinates via batch inversion, sorted by x-coordinate.
12. For $r_2 \in R_2$,
    (a) Compute $r_2 m_1 m_E P$.
    (b) if $r_2 m_1 m_E P$ equals $Q_{r_{1,i}}$,
        i. Set $r_1 = r_{1,i}$ and exit loop.
13. If no match is found, then repeat from step 7.
14. Compute t from (r_1, r_2).
15. Return t.

Definition 4.4.1. The **Chebyshev function** [13] $\vartheta(x)$ is defined as

$$\vartheta(x) = \sum_{l \leqslant x} \ln l = \ln \left(\prod_{l \leqslant x} l \right) \tag{4.24}$$

where l is a prime. This function is asymptotically linear, that is to say

$$\lim_{x \to \infty} \frac{\vartheta(x)}{x} = 1 \tag{4.25}$$

and it is bounded by the inequalities

$$x \ln 2 \leqslant \vartheta(x) \leqslant x \ln 4 \tag{4.26}$$

Number of primes For a random prime p, let n_p denote the number of primes l_i (starting with 2) so that $\prod_{i=1}^{n_p} l_i \geqslant 4\sqrt{p}$. Let L denote the largest prime; we take l_i as an increasing sequence, so $l_1 = 2$ and $l_{n_p} = L$. In other words, we seek to find L so that

$$\ln \left(\prod_{l \leqslant L} l \right) = \vartheta(L) \geqslant L \ln 2 \geqslant \ln 2 \cdot \left(2 + \frac{1}{2} \log p \right) = \ln(4\sqrt{p}) \tag{4.27}$$

Hence it suffices to choose $L > 2 + \frac{1}{2} \log p$ to guarantee sufficiently many primes are found. Asymptotically, we will expect L to be lower, approximately $\ln 2 \cdot \left(2 + \frac{1}{2} \log p\right)$. The number of primes can be estimated to be

$$n_p \approx \frac{L}{\ln L} \approx \frac{\ln 2 \cdot \left(2 + \frac{1}{2} \log p\right)}{\ln \left(\ln 2 \cdot \left(2 + \frac{1}{2} \log p\right) \right)} = \frac{2 + \frac{1}{2} \log p}{\log \left(\ln 2 \cdot \left(2 + \frac{1}{2} \log p\right) \right)} \tag{4.28}$$

which tends to $\frac{\log p}{2 \log \log p}$ for large p.

We proceed to determine the complexity for each step in Match-Sort.

Computing t_E Step 1 of the algorithm can be performed at the end of the loop for each Elkies prime without the need to store the value. It would be faster to compute t_E recursively for each subset $\mathcal{E}_1, \mathcal{E}_2$ where $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$, after all the Elkies primes and their respective traces t_l are determined. However, the savings would be marginal, and hence this latter approach has not been implemented.

Partitioning Atkin primes If we assume the number of Atkin primes to be half of the total number of primes, then the size of the sorting problem in step 2 is $n_p/2$, and thus has complexity $O(n_p \log n_p)$. Step 4 runs in n_p steps. The partitions $\mathcal{A}_1$, $\mathcal{A}_2$ will have sizes roughly $\sqrt{n_p/2}$.

Number of possible t_l Each T_l has size $\varphi(r)$ for each Atkin prime l, where $r \mid l+1$. We denote by $\nu(l+1)$ the number of factors of $l+1$. Assuming that each factor of $l+1$ is an equally likely candidate for r (besides $r=1$, since that will make l an Elkies prime), we expect the size of number of possible t_l to be

$$E(\varphi(r)) = \frac{\sum_{r \mid l+1, r \neq 1} \varphi(r)}{\nu(l+1)-1} = \frac{(l+1)-1}{\nu(l+1)-1} = \frac{l}{\nu(l+1)-1} \tag{4.29}$$

since $l+1$ is even and has at least four factors if $l \neq 2, 3$, $\nu(l+1)-1 \geqslant 3$. For simplicity we will use $l/3$ in place of $E(\varphi(r))$.

Computing τ_i We index $\mathcal{A}_i$ as the increasing sequence $(l_{i,k})$ and the values in $T_{l_{i,k}}$ as $(t_{l_{i,k}})$. Starting with $S_{i,1} := T_{l_{i,1}}$ and $m_{i,k} = l_{i,1}$, compute the set of solutions $S_{i,k+1} \bmod m_{i,k}$ to the congruence problem

$$\begin{array}{rcll} x & \equiv & s & \pmod{m_{i,k}} \\ x & \equiv & t_{l_{i,k}} & \pmod{l_{i,k}} \end{array} \tag{4.30}$$

via the Chinese Remainder Theorem for each $s \in S_{i,k}$ and $t_{l_{i,k}} \in T_{l_{i,k}}$. Note that the final $S_{i,k}$ has size n_i. Each use of CRT requires one inversions for each modulus $m_{i,k}$ and $l_{i,k}$, but since it depends only on the index k, only two inversions need to be computed for each k. Therefore the expected running time of step 5 is of the order of n_i, which is

$$O\left(\prod_{l \in \mathcal{A}_i} \varphi(r)\right) = O\left(\prod_{l \in \mathcal{A}_i} \frac{l}{3}\right) = O\left(\left(3^{-n_p} \prod_{l} l\right)^{\frac{1}{4}}\right) = O\left(3^{-\frac{n_p}{4}} p^{\frac{1}{8}}\right) \approx O(p^{\frac{1}{8}}). \tag{4.31}$$

Computing R_i Step 6 involves computing the possible values of r_1 and r_2 given the sets τ_1 and τ_2, and thus is of the order of n_i as well, which is $O(p^{\frac{1}{8}})$.

Sorting R_i If we use QuickSort in step 7, we will require $O(n_i \log(n_i))$ steps, which will be higher than $O(p^{\frac{1}{8}})$. Alternatively, we can use BucketSort with $O(m_i)$ buckets to performing sorting in $O(n_i + m_i) = O(p^{\frac{1}{8}})$ time. The storage needed is no bigger than $O(m_i)$ since the r_i's are distinct.

Computing $\mathcal{Q}_1$ Since R_1 is sorted by absolute values, for each $r_{1,i}, r_{1,i+1} \in R_1$, the difference $|r_{1,i+1}| - |r_{1,i}|$ is much smaller than m_1, the size of the interval of values $r_{1,i}$ can take. Assuming that m_1 and m_2 are of the same order of magnitude, and likewise for $m_1 m_2$ and m_E, m_1 and m_2 has sizes in the order of magnitude $O(p^{\frac{1}{8}})$. The size of $|r_{1,i+1}| - |r_{1,i}|$ is in the order of $O(m_1/n_1)$, which is less than $O(p^{\frac{1}{8}})$.

As we can compute successive $[r_{1,i}]m_2 m_E P$ recursively via scalar multiplication by $|r_{1,i+1}| - |r_{1,i}|$, the running time of these steps is dominated by the number of $|r_{1,i}|$'s, i.e. n_1). Thus the computation of $\mathcal{Q}_1$ is in the order of $O(p^{\frac{1}{8}})$ elliptic curve point doubling/additions.

Converting to affine coordinates To convert a point P_i on an elliptic curve from projective coordinates (X_i, Y_i, Z_i) to affine coordinates $(x_i, y_i, 1)$, we need to compute

$$x_i = X_i/Z_i, \qquad y_i = Y_i/Z_i, \tag{4.32}$$

which requires the inverse of Z_i. When we have a set of projective points $\{P_i : i = 1, \cdots, n\}$ that we wish to express in affine coordinates, we can compute the inverse for each Z_i by performing an inversion only once. We first compute $Z_0 := \prod_{1 \leqslant k \leqslant n} Z_i$, followed by its inverse over $\mathbb{F}_p$, Z_0^{-1}. Next, we compute $\bar{Z}_i := \prod_{1 \leqslant k \leqslant n, k \neq i} Z_i$, storing intermediate products so as to eliminate repeating the same computations. The inverse of Z_i can then be computed as $\bar{Z}_i \cdot Z_0^{-1}$.

The extra multiplications that we require to perform coordinate change for the set scales linearly in the size of the set, but this is more than offset by the cost savings from reducing the number of inversions required. Sorting the points by x-coordinate requires $O(n_1 \log(n_1))$ steps if we use QuickSort. As the x-coordinate lies between 0 and $p - 1$, and no more than two points can have the same x-coordinate, we can use bucket sort with $O(n_1)$ buckets to reduce the average case complexity to $O(n_1)$.

Finding a match As in step 10, the computation of the points $r_2 m_1 m_E P$ takes $O(p^{\frac{1}{8}})$ elliptic curve point doubling/additions. Since $\mathcal{Q}_1$ is sorted, we can use binary search to find a match with just $O(\log(n_1))$ comparisons. A match may not be found if P does not have order $\#E(\mathbb{F}_p)$, which occurs with expected probability

$$1 - E\left(\frac{\varphi(\#E(\mathbb{F}_p))}{\#E(\mathbb{F}_p)}\right) \approx 1 - \frac{6}{\pi^2} \approx 0.392.$$

Overall Complexity The largest complexity amongst all the steps of the algorithm is $O(p^{\frac{1}{8}})$, which occurs whenever we deal with arrays of size m_i or n_i. Amongst these, computing $\mathcal{Q}_1$ has its running time in terms of elliptic curve arithmetic, which is more costly than other operations. Hence it is important to use the efficient elliptic curve arithmetic for our purposes.

4.4.4 Coordinate Systems

We briefly explain the choice of coordinate systems here, as the Match-Sort algorithm is performed on explicit points rather than kernel polynomials. Further details can be found in Section 13.2 of [5]. The overriding consideration in this implementation is speed, within practical memory constraints (single laptop). This gives us the flexibility to use more coordinates for each point to achieve arithmetic speed ups. This is important as point arithmetic operations are expensive; the running times of the program have indeed shown that point arithmetic within the Match-Sort algorithm can dominate the SEA algorithm.

Besides the cost of point addition and doubling, we need to consider the cost of comparing two points. This is because we need to check if a point generated during the baby-steps matches one generated by the giant-steps. Affine coordinates provide us with the best speed, since we can compare two points (x_1, y_1), (x_2, y_2) by comparing the x_1 with x_2 and the y_1 with y_2. However point addition and doubling are expensive operations, requiring I + 2M + S and I + 2M + 2S respectively, where S and M denotes squaring and multiplication over $\mathbb{F}_p$. I denotes inversion over $\mathbb{F}_p$, which requires between 9M and 40M on average. S is faster than M, taking about 0.8M. In comparison, projective coordinates require 12M + 2S and 7M + 5S for addition and doubling respectively, while requiring at least 2M for each comparison as we need to compare $x_1 * z_2$ with $x_2 * z_1$ (and possibly $y_1 * z_2$ with $y_2 * z_1$) for points (x_1, y_1, z_1), (x_2, y_2, z_2).

In the Match-Sort algorithm, we need to compare entries in lists $\mathcal{Q}_1$ and $\mathcal{Q}_2$. With $\mathcal{Q}_1$ sorted, we expect to take $|\mathcal{Q}_2|/2 \log |\mathcal{Q}_1|$ comparisons to find a match. The cost of converting the points in the two lists from projective to affine coordinates is $|\mathcal{Q}_1|+|\mathcal{Q}_2|$ times I + 2M, while the cost of comparison of the lists in projective coordinates is $|\mathcal{Q}_2|/2 \log |\mathcal{Q}_1|$ times 2M. Therefore performing comparisons in affine coordinates is asymptotically better, which justifies our choice in the baby-step.

4.5 Full SEA Algorithm

4.5.1 Description of the Schoof-Elkies-Atkin Algorithm

Input: An ordinary curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$, $j := j(E)$ not 0 or 1728.
Output: $\#E(\mathbb{F}_p)$.

1. Compute $t_2 = t \bmod 2$ by checking if $x^3 + ax + b$ has a root in $\mathbb{F}_p$.
2. Let $l = 3$, $\Pi = 1$, $\Sigma_A = \varnothing$, $m_E = 2$, $\Sigma_E = \{(\{t_2\}, 2)\}$.
3. While $\Pi < \lceil 4\sqrt{p} \rceil$ do:
   (a) Evaluate the modular equation $\Phi_l(X, Y) \in \mathbb{F}_p[X, Y]$ at $Y = j$.
   (b) Compute $X^p \bmod \Phi_l(X, j)$.
   (c) Compute $\gcd(\Phi_l(X, j), X^p - X)$ and decide if l is an Elkies or Atkin prime.
   (d) If l is an Elkies prime do:
      i. Compute $j(\tilde{E})$ as a root of $\Phi_l(X, j)$ in $\mathbb{F}_p$.
      ii. Determine $\tilde{E}$ using Theorem 4.2.1 and compute g_l using Theorem 4.17.
      iii. Find an eigenvalue λ of ϕ_p in $\mathbb{F}_l$.
      iv. Set $t_l = \lambda + p/\lambda \bmod l$.
      v. Add the pair $(\{t_l\}, l)$ to Σ_E, and set $m_E = m_E \times l$ and $\Pi = \Pi \times l$.
   (e) Else l is an Atkin prime, so we do:
      i. Compute $r = \text{OrderFrobenius2}(\Phi_l(X, j))$.
      ii. Determine the set T_l of possible values for t_l from the primitive r-th roots of unity using Equation 4.2.
      iii. Add the pair (T_l, l) to Σ_A, and set $\Pi = \Pi \times l$.
   (f) Set l as the next higher prime.
4. Determine t from $\Sigma_E \cup \Sigma_A$ via Match-Sort Algorithm.
5. Return $\#E(\mathbb{F}_p) = p + 1 - t$.

4.5.2 Complexity of SEA

Schoof's. For Schoof's algorithm [17] we are working with $\Psi_l(X, Y)$, a polynomial of degree $(l^2 - 1)/2$. Thus the elements in the ring $\mathbb{F}_p[X, Y]/(\Psi_l(X, Y), Y^2 - X^3 - Ax - B)$ have size $b := O(l^2 \log p)$.

The amount of work required to compute $\phi(X, Y)$ and $\phi^2(X, Y)$ is $O(\log p M(b))$, where $M(n)$ is the complexity of multiplying two n-bit numbers. The amount of work required to compute $t_l\phi(X, Y)$ is $O(lM(b))$. Thus for each l the complexity is $O(\log p M(l^2 \log p))$. Since $n_p < \log p$, and $L = O(\log p)$, the total work required for Schoof's algorithm is $O(\log^2 p M(\log^3 p))$.

SEA. In contrast, for the SEA algorithm we are working with $\Phi_l(X, j)$ and $F(X)$ which are both polynomials of degree $O(l)$. Thus the elements we are working with have size $b' := O(l \log p)$. Therefore the total work required for the Schoof-Elkies-Atkin algorithm is $O(\log^2 p M(\log^2 p))$.

One of the fastest methods for multiplying two n-bit numbers is the Schönhage-Strassen algorithm [16], which runs in $O(n \log n \log \log n)$. Hence the complexity of SEA is

$$O(\log^4 p \log \log p \log \log \log p).$$

Note that although Match-Sort runs in exponential time, its complexity is subsumed into that of SEA. This is because we can choose to reject the use of 'bad' Atkin primes — those that increase the number of possible values of t by too much. By choosing our Atkin primes carefully, it is possible to gain information from these primes while keeping the Match-Sort component from growing too large. We will discuss a few variations of the Schoof-Elkies-Atkin algorithm next.

4.5.3 Variants of SEA

Elkies Only. A popular variant of SEA in use can be referred to as SE, since it is running the SEA without using any Atkin primes. Examples of usage of this variant are MIRACL, as well as Andrew Sutherland's implementation for current records of point counting on curves over prime fields.

The justification for this variant comes from our analysis of the complexity of the Match-Sort algorithm in Section 4.4.3, where we showed that its running time is in the order of $O(n_1) \approx O(p^{\frac{1}{8}})$, where n_1 approximately the square root of the number of possible values for t prior to the Match-Sort algorithm. If every Atkin prime was used for the Match-Sort algorithm, then the size of the Match-Sort problem will eventually become bigger than that of the full SEA algorithm.

Since SE outperforms SEA asymptotically, for record setting (the current record [22] having $p = 16219299585 * 2^{16612} - 1 \approx 2^{16646}$) it makes sense to use only Elkies primes. Assuming half of the primes are Atkin primes, and that the product of Atkin primes equals that of Elkies primes, then instead of having $\prod_l l > 4\sqrt{p}$ we will require $\prod_l l > 16p$. However the complexity of the overall algorithm remains the same as that of SEA, since the largest prime L is now approximately $\ln 2 \cdot (4 + \log p)$, which is still $O(\log p)$.

This justisfies us 'ignoring' the complexity of the Match-Sort algorithm. However, while the asymptotic performance of SE remains good, the constant factor growth in complexity from the doubling of the largest prime L is very large, as each larger prime contributes more to the complexity than the previous. Furthermore, our current understanding of analytic number theory is unable to prove sharp bounds on the worst-case distributions of Elkies prime, and so there is currently no guarantee that a given curve will have sufficient Elkies primes. Hence it is worthwhile considering using some Atkin primes.

Elkies and some Atkin. We refer to this as SEa; that is using all Elkies primes and some Atkin primes. From our discussion on the complexity of the Match-Sort algorithm in Section 4.4.3, we gain some understanding of which Atkin primes should be considered for use in Match-Sort. For convenience, we will refer to such Atkin primes as **good** (otherwise **bad**).

By including an Atkin prime l, we increase Π by a factor of l, but increase the size of the Match-Sort problem by a factor of $\varphi(r)$ for the corresponding $r \mid l + 1$. Hence a good Atkin prime should have a high ratio of l to $\varphi(r)$. A straightforward way is to compute the value of r for each Atkin prime l, and then $\varphi(r)$. If $\varphi(r)$ is too large relative to l, then we consider it as bad. However, computing r can be a costly operation, and we may incur computational costs for many bad Atkin primes before finding a good one.

Equation (4.29) offers a way round this problem. First, we obtain the prime factorization of $l+1$. The expected value of $\varphi(r)$ decreases if $l+1$ has more factors. Hence prior to computing r, the number of factors of $l+1$ can inform us if an Atkin prime is likely to be good or bad. Also, with more prime factors, the maximum value of $\varphi(r)$ is reduced. The number of factors and prime factors of $l+1$, thus $E(\varphi(r))$, can be precomputed and stored with each l.

Example 4.5.1. Consider $l = 2309$. We have $l + 1 = 2310 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11$, with $E(\varphi(r)) = l/(2^5 - 1) \approx 74.5$ and the maximum value of $\varphi(r)$ is 480. Although r may range from 2 to 2310, 2309 is likely to be a good Atkin prime, and should be considered for inclusion if Elkies primes larger than 2309 may be used.

Example 4.5.2. Consider $l = 2341$. We have $l + 1 = 2342 = 2 \cdot 1171$. To avoid the high cost of determining r for large values, we can just check if $r = 2$ to decide if l is good or bad. If $r \neq 2$ (i.e. $r = 1171$ or 2342), then l is bad. If $r = 2$ then l is good (in fact, very good, since $\varphi(r) = 1$). To generalize this example, if $(l + 1)/2$ is prime, then it is worthwhile to check whether $r = 2$.

Note that the definition of good or bad Atkin primes has been rather imprecise, offering only a vague notion. A better definition requires a detailed analysis of the computational cost of using a larger Elkies prime, versus the cost of increasing the size of the Match-Sort problem.

Chapter 5

Run-Time Performance

5.1 Complexity vs Run-Time

The complexity analysis tells us that asymptotically, using the Schoof-Elkies-Atkin algorithm with Elkies primes only is the best algorithm on average, while using all Atkin primes will be the worse.

For cryptographic purposes however, the size of the prime field is not very large (521-bit p to achieve 256-bit security), rendering the analysis much less helpful. The large hidden constants for SE, vis-a-vis that of SEA, means that for elliptic curves of interest to cryptographers, the running time of SE could be much worse than that of SEA and SEa. In [1], it was remarked that the best variant in practice will be one that judiciously selects Atkin primes.

In this chapter I will present the run-time analysis of my implementation. Five random curves are generated for each prime fields with bit-sizes 128, 160, 192, 224 and 256, and SEA, SEa and SE are used to count the points for each of them, and the running times compared.

5.1.1 Run-Time Performance

For this test, in SEa we choose Atkin primes when their value of r lies in the set $\{r \in \mathbb{N} : r \leqslant 66, r \neq 31, 37, 38, 41, 43, 44, 47, 49, 53, 55, 57, 59, 61, 62, 64, 65\}$. The excluded values of

r tend to occur when $\varphi(r)/l$ is high. For example, $r = 44$ may occur when $l = 43$ with $\varphi(r)/l = 20/43 > 0.45$, thus increasing the Match-Sort problem 20-fold while contributing only 5.4 bits to the product $\prod_l$. It should be noted that the choice of 66 as the threshold is somewhat arbitrary; it is optimized for the size of fields considered. A lower threshold may give better performances if larger prime fields are considered.

This criterion is a blunt tool as it makes no distinction between different l; an optimized criterion should favour larger l with the same r values. It suffices though, to highlight the importance of retaining the use of Atkin primes at some level.

The running times below are listed in terms of the logarithm of the CPU's output (Intel i5-2520M @ 2.50GHz), with the first table listing the logarithm of the average running times and the latter the average of the logarithm of the running times. This allows us to capture some information on the fluctuations in the running times.

Table 5.1: Logarithm of Average Running Time of SEA Variants (s)

bits	SEA	SEa	SE
128	12.02	12.06	13.94
160	13.47	13.44	15.20
192	17.31	16.85	16.58
224	-	16.96	17.20
256	-	16.84	18.07

Table 5.2: Average of Logarithmic Running Time of SEA Variants (s)

bits	SEA	SEa	SE
128	12.00	12.04	13.79
160	13.32	13.39	15.11
192	16.29	15.50	16.46
224	-	16.08	17.09
256	-	16.78	18.05

From the running times, we observe that for small fields of 128 and 160 bits, SEA and SEa perform comparably, as Match-Sort remains tractable even with many Atkin primes

for SEA. SEa at these field sizes omit very few Atkin primes, so savings in Match-Sort tends to be balanced out by the increases in using larger primes. In contrast, SE performs poorly at these levels, as the cost of each additional large primes can be as large as the whole Match-Sort problem.

At 192-bits the performance of SEA varied greatly, with five orders of magnitude difference in the running times. From 224-bits the run-time of SEA tends to be too large, and as such the computations were not completed. At these sizes, SEa tends to run twice as fast as SE, even though SE does run faster in some cases.

5.2 Improvements

To handle larger prime fields, we can prepare classical modular polynomials for $l > 200$ in our repository. As the size of the polynomial grows very quickly, a better approach will be to use either Atkin or Weber modular polynomials instead, so as to reduce the amount of storage needed.

Instead of restricting the modular polynomials to prime l, we can choose prime powers of l as well. This will delay the need for larger prime fields, and thus speed up the algorithm. Such an approach is described in [6].

Existing complexity analysis assumes each l to be of size $O(\log p)$, which does not capture the increasing costs of successive Elkies primes. When using SEa, choosing the right Atkin primes to compute and to use in Match-Sort requires a better understanding of the running time tradeoffs between using a higher Elkies prime and increasing the Match-Sort problem. It may be worthwhile to obtain tighter estimates for the expected sizes of $\varphi(r)$ for Atkin primes for a given curve so as to decide the threshold for r that allows the maximum use of Atkin primes while keeping the Match-Sort procedure tractable. These estimates can then be compared against estimates for the expected running time for each Elkies prime.

Chapter 6

Future Work

6.1 Partitioning Strategy

In this thesis, we have analysed the actual run-time of variants of the Schoof-Elkies-Atkin algorithm, each employing Atkin primes in varying degrees. For each Atkin prime l, we partition the set of possible r-values into two sets defined as such: if the actual value comes from the first set, then we will use this Atkin prime in Match-Sort, else we will not. In SEA the latter set is empty, while for SE the first set is empty. In this implementation of SEa, which gives the best average run-times compared to SE and SEA, the partitioning is independent of l and p.

I believe that the optimal strategy for partitioning the set of r-values should be dependent on both l and p. Two Atkin primes $l_1 < l_2$ may have the same r-value and thus contribute to the size of the Match-Sort problem identically, but the larger prime will have contribute more bits to the product $\prod_l l$ which we need to exceed $4\sqrt{p}$. Thus the set of acceptable r-values for l_1 should be less than that of l_2.

For $p_1 < p_2$, $4\sqrt{p_1} < 4\sqrt{p_2}$, so the number of primes l that is needed is greater for p_2. The Match-Sort problem is expected to grow exponentially in the number of Atkin primes, while the running time of the algorithm grows linearly. Hence to control the growth of the Match-Sort problem, we need to be more selective in our choice of smaller Atkin primes for p_2 than for p_1.

6.2 Run-Time Estimation

Better estimates of the actual running time of the Atkin classification (to determine if a prime is Elkies or Atkin), the BSGS step for Elkies prime and the Match-Sort step are needed. From these, we can obtain better predictions of the running times of SEa for each strategy for partitioning the r-values, and thus minimise the expected running time of the algorithm a priori.

The run-time results in the previous chapter were generated from a small sample for each size of p. Further run-time statistics should be gathered for each size, with multiple curves chosen for each prime as well.

The analysis of the Match-Sort algorithm in Chapter 4 assumed that the probability for each r-value is uniformly distributed. It will be worthwhile to study the actual distribution, as this has a significant impact on the likelihood that an Atkin prime will be good.

References

[1] Ian Blake, Gadiel Seroussi, and Nigel Smart. *Elliptic Curves in Cryptography*. LMS Lecture Note Series. Cambridge University Press, New York, first edition, 1999.

[2] Reinier Bröker, Kristin Lauter, and Andrew Sutherland. Modular polynomials via isogeny volcanoes. *Mathematics of Computation*, 81:1201–1231, 2012.

[3] Reinier Bröker and Andrew Sutherland. An explicit height bound for the classical modular polynomials. *Ramanujan J.*, 22(3):293–313, 2010.

[4] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics. Springer-Verlag, New York, third edition, 1996.

[5] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Douce, Tanja Lange, Kim Nguyen, and Frederik Vercautern. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC, New York, first edition, 2006.

[6] Jean Marc Couveignes and François Morain. Schoof's Algorithm and Isogeny Cycles. In *Algorithmic number theory (Ithaca, NY, 1994)*, volume 877 of *Lecture Notes in Comput. Sci.*, Berlin, 1994. Springer.

[7] David Cox. *Primes of the form x^2+ny^2*. A Wiley-Interscience Publication. John Wiley & Sons Inc., New York, 1989. Fermat, class field theory and complex multiplication.

[8] Noam Elkies. Elliptic and modular curves over finite fields and related computational issues. In *Computational perspectives on number theory (Chicago, IL, 1995)*, volume 7 of *AMS/IP Stud. Adv. Math.*, pages 21–76. Amer. Math. Soc., Providence, RI, 1998.

[9] Andreas Enge. The complexity of class polynomial computation via floating point approximations. *Math. Comp.*, 78(266):1089–1107, 2009.

[10] Mireille Fouquet. *Anneau d'endomorphismes et cardinalité des courbes elliptiques: aspects algorithmiques*. PhD thesis, École Polytechnique, 2001.

[11] Mireille Fouquet and François Morain. Isogeny volcanoes and the SEA algorithm. In *Algorithmic Number Theory (Sydney, 2002)*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 276–291. Springer, Berlin, 2002.

[12] David Kohel. *Endomorphism Rings of Elliptic Curves over Finite Fields*. ProQuest LLC, Ann Arbor, MI, 1996. Thesis (Ph.D.)–University of California, Berkeley.

[13] Melvyn Nathanson. *Elementary Methods in Number Theory*. Graduate Texts in Mathematics. Springer-Verlag, New York, first edition, 2000.

[14] Nicole Pitcher. *Efficient Point-Counting on Genus-2 Hyperelliptic Curves*. PhD thesis, University of Illinois at Chicago, January 2009.

[15] Takakazu Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *J. Ramanujan Math. Soc.*, (4):247–270.

[16] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing (Arch. Elektron. Rechnen)*, 7:281–292, 1971.

[17] René Schoof. Counting points on elliptic curves over finite fields. *J. Theor. Nombres Bordeaux*, 7:219–254, 1995.

[18] Victor Shoup. A new polynomial factorization algorithm and its implementation. *Journal of Symbolic Computation*, 20:363–397, 1996.

[19] Victor Shoup. *NTL: A Library for doing Number Theory*. `http://www.shoup.net/ntl/`, August 2009.

[20] Joseph Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1986.

[21] Joseph Silverman. *Advanced Topics in the Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer-Verlag, New York, first edition, 1994.

[22] Andrew Sutherland. *Genus 1 Point Counting Records over Prime Fields*. `http://math.mit.edu/~drew/SEArecords.html`, July 2010.

[23] Andrew Sutherland. Isogeny volcanoes. *ArXiv e-prints*, August 2012.

[24] Jacques Vélu. Isogénies entre courbes elliptiques. *C. R. Acad. Sci. Paris Sér. A-B*, 273:A238–A241, 1971.

[25] Joachim von zur Gathen and Victor Shoup. Computing frobenius maps and factoring polynomials. *Computational Complexity*, 2:187–224, 1992.