5-2022

AN EXPOSITION OF ELLIPTIC CURVE CRYPTOGRAPHY

Travis Severns

An Exposition of Elliptic Curve Cryptography

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

in

Mathematics

by

Travis Severns

May 2022

An Exposition of Elliptic Curve Cryptography

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

by

Travis Severns

May 2022

Approved by:

Abstract

Protecting information that is being communicated between two parties over unsecured channels is of huge importance in today's world. The use of mathematical concepts to achieve high levels of security when communicating over these unsecured platforms is cryptography. The world of cryptography is always expanding and growing. In this paper we set out to explore the use of elliptic curves in the cryptography of today, as well as the cryptography of the future.

We also offer our own original cryptosystem, CSDH. This system on its own offers some moderate level of security. It shares many similarities to the post-quantum, SIDH system. The parallels between these two systems can lead to deeper understanding of the systems offered for our post-quantum world.

Acknowledgements

First, I would like to thank all my friends and family for their constant love and support throughout my time as a student. A special thank you must be given to my parents. Their constant encouragement and words of affirmation were helpful "pick me ups" along my journey.

I would also like to express my deepest gratitude to Dr. Jeffrey S Meyer. Throughout my time working on my thesis I hit many of life's mile stones. I began working full-time, I got promoted to a position of higher responsibility, I moved out of my parents house, and most importantly I got married, all while going through a global pandemic. Although all of these life changes came with trials, Dr. Meyer was supportive every step of the way. I am extremely thankful for his consistent encouragement, understanding, and flexibility. His passion for mathematics was inspiring and I am better for having had the opportunity to work with him.

I would like to thank my wife Cameron Severns, without her support I would not only be where I am academically but I would not be the person I am today. She has been a constant source of encouragement throughout my time as a student. I am thankful for all the meals cooked, the pep-talks, the stern reminders to stay focused, and the grace shown to me when I could not give her the attention she deserved. This is only a small list of the ways she has supported me. This process would have been painful, maybe impossible, without her by my side.

Above all thanks be to God.

Table of Contents

List of Figures

Chapter 1

Introduction

In the world we live in today the ability to safely communicate information on insecure platforms is extremely necessary. Currently the field of cryptography is ramping up its development of new and improved protocols that not only provide security against classical computers, but also hold strong against quantum computers. In this paper we will discuss the use of elliptic curves in current protocols and also look to the future use of elliptic curves in the post-quantum world.

In Chapter 2 of this paper, we cover basics the of elliptic curves. We cover necessary definitions and theorems for classical elliptic curve cryptography. Background needed for classical elliptic curve cryptography, such as the definition an elliptic curve, and the group law associated with elliptic curve groups, with examples, are included. Information needed for post-quantum applications, such as isogeny and torsion subgroups will also be covered.

In Chapter 3, we cover a brief background of cryptography. We then cover the original Diffie-Hellman Key Exchange in $\mathbb{F}_p$. We also provide a simple example to further the readers understanding of the protocol. We also discuss the hard problem associated with the Diffie-Hellman Key Exchange, the Discrete Log Problem, and offer some ways to solve it. In addition to describing some algorithms, Sage sample code will be given.

In Chapter 4, we move to covering classical elliptic curve cryptography. Specifically we cover Elliptic Curve Diffie-Hellman Key Exchange (ECDH). We discuss the analogous hard problem Elliptic Curve Discrete Log Problem. We then give examples of the ECDH protocol. Also we discuss potential vulnerabilities within the key exchange and offer Sage code for possible attacks on the exchange.

Moving on to Chapter 5, we cover a post-quantum system based on elliptic curves, the Supersingular Isogeny Diffie-Hellman Key Exchange (SIDH). We cover the algorithm for the exchange as well as offer an example. We then move to explain the isogeny graphs that have use within the post quantum cryptographic world.

Finally in Chapter 6, we offer our own cryptosystem, the Cyclic Subgroup Diffie-Hellman Key Exchange (CSDH), that has strong parallels to the post-quantum SIDH. We cover the algorithm associated to the exchange and prove the associated theorems. We continue in our exploration by using Sage to compute many large scale trials to understand expected values and tendencies of our system. We also provide another analogy between subgroup graphs and isogeny graphs.

Throughout this exploration of elliptic curves we will continue to draw on the parallels between the original Diffie-Hellman, Elliptic Curve Diffie-Hellman, Supersingular Isogeny Diffie-Hellman, and Cyclic Subgroup Diffie-Hellman. We will discuss the idea of hard problems and their use in cryptography, specifically the hard problems associated with the protocols covered in the systems mentioned above.

This research was supported in part by the high performance computing resources provided by the Information Technology Services and Academic Technologies and Innovation at California State University San Bernardino [ATI22].

Chapter 2

Elliptic Curves

A complete and thorough resource for the study of elliptic curves is Silverman's, *The Arithmetic of Elliptic Curves* [Sil16]. We will cover only what is necessary for the purposes of this paper.

We take as a given a field k of characteristic *char* $k \neq 2, 3$ and the ring of polynomials $k[x, y]$.

Definition 2.1. *An* ***algebraic curve*** *in* k^2 *is the zero set of a polynomial* $f \in k[x, y]$. *In symbols,* $V(f) = \{(x, y) \in k^2 | f(x, y) = 0\}$.

Definition 2.2. *An algebraic curve defined over* k *is* ***smooth*** *at* $P \in V(f)$ *if* $\nabla f(P) \neq 0$. *If* $V(f)$ *is smooth at all points, then it is smooth (or nonsingular).*

Definition 2.3. *An* ***elliptic curve*** *is the solution set to a smooth cubic polynomial together with a base point* O *at infinity.*

Definition 2.4. *A elliptic curve curve* E *is in* ***Weierstrass form*** *if* $E = V(f)$ *where,*

$$f(x, y) = y^2 - (x^3 + Ax + B).$$

Going forward, we will write $E : y^2 = x^3 + Ax + B$ *when defining an elliptic curve.*

Given an elliptic curve defined over a field k with *char* $\neq 2, 3$, then there exists an isomorphism defined over $\bar{k}$, to another curve that is in Weierstrass form [Sil16, Proposition 3.1]. For most applications within this paper we will be using elliptic curves in Weierstrass form. By [Sil16, Proposition 3.1], we are justified in this restriction. Computations with curves in Weierstrass forms are much easier, and we can represent these curves by their coefficients $[A, B]$.

Definition 2.5. *The **discriminant** of an elliptic curve with Weierstrass form $f(x, y) = y^2 - (x^3 + Ax + B)$ is,*

$$\Delta f = 4A^3 + 27B^2$$

Proposition 2.6. *If $f(x, y) = y^2 - (x^3 + Ax + B)$ and $C = V(f)$, then C is smooth if and only if $\Delta f \neq 0$.*

A proof of Proposition 2.6 can be found in [Sil16, Proposition 3.1.4].

Example 2.7. *Elliptic curve over $\mathbb{R}$.*

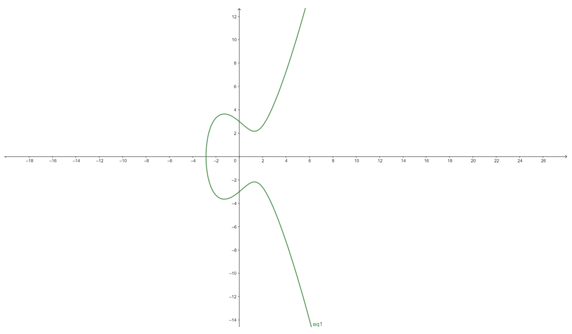

Figure 2.1: $E : y^2 = x^3 - 5x + 9$ is a nonsingular elliptic curve with $\Delta = 1687$

Example 2.8. *Singular cubic curve with cusps.*

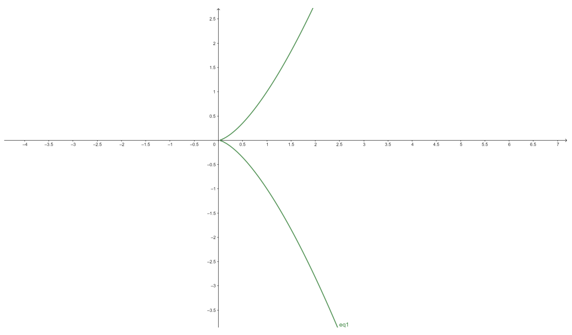

Figure 2.2: $E : y^2 = x^3$ is a singular curve with $\Delta = 0$ and singular point $(x, y) = (0, 0)$

Definition 2.9. *If k'/k is a field extension and E is an elliptic curve defined over k, then the set of points on E with coordinates in k' is the set*

$$E(k') = \{(x, y) | x, y \in k' \text{ satisfying } y^2 = x^3 + Ax + B\} \cup \{O\}.$$

*We call these the **k'-points** of E.*

Definition 2.10. *If k'/k is a field extension and E is an elliptic curve in Weierstrass form over k, then we define the* ***Group Law*** *as follows. Given two points P and Q on E we define the line through P and Q as L. Since E is a cubic we know, by Bezout's Theorem [Bix06, Theorem 11.5], L will intersect E at exactly three points (with multiplicity), we call this third point R. In the case were L is a vertical line, we let O, the point at infinity, be the third point of intersection. Since E is symmetric about the x-axis we can reflect R about the x-axis to $R' \in E$. For the case when $R' = -O$ we let $-O = O$. Define R' be the sum of P and Q, and we use $\oplus$ to represent this operation: $P \oplus Q := R'$.*

Note: P and Q need not be distinct, so in some cases L may be tangent to E. We will use $*$ to denote adding a point P to itself multiple times, for example $3 * P := P \oplus P \oplus P$.

Theorem 2.11. *If k'/k is a field extension and E is an elliptic curve defined over k, then the $E(k')$-points form a group with $\oplus$.*

The proof of Theorem 2.11 is straight forward, however showing the associative property is very lengthy. A proof using the Riemann–Roch Theorem can be found in Silverman's *The Arithmetic of Elliptic Curves* [Sil16].

In Figure 2.3 we represent this operation over $\mathbb{R}$. Although the picture is less clear over other fields, the definition works just as well.

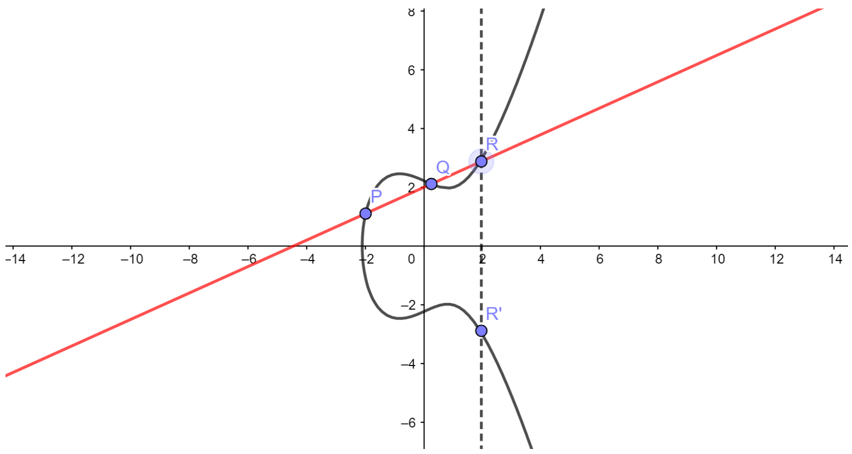

Figure 2.3: $P \oplus Q = R'$

This geometric process can be implemented by a computer using the following algorithm [HPS14, Theorem 6.6]:

Theorem 2.12. ***The Group Law Algorithm***
Let $E : y^2 = x^3 + Ax + B$ be an elliptic curve and let $P = (x, y)$ and $Q = (x', y')$ be points on E.

1. *If $P = O$ or $Q = O$, then $P \oplus Q = Q$ or $P \oplus Q = P$ respectively.*
2. *If $x = x'$ and $y = -y'$ then $P \oplus Q = O$.*
3. *Otherwise let*
$$\lambda = \begin{cases} \frac{y'-y}{x'-x} & \text{if } P \neq Q, \\ \frac{3x^2+A}{2y} & \text{if } P = Q. \end{cases}$$
and let $x'' = \lambda^2 - x - x'$ and $y'' = \lambda(x - x'') - y$ and $P \oplus Q = (x'', y'')$

Proof. Let $E : y^2 = x^3 + Ax + B$ be an elliptic curve and let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be points on E.

1. We must prove if $P = O$ or $Q = O$, then $P \oplus Q = Q$ or $P \oplus Q = P$ respectively. Let $P = O$. Then the line L through P and Q is a vertical line. Let the third intersection point of L and E be R. Since L is a vertical line, R is the reflection of Q across the x-axis. Then reflecting R across the x-axis will land you back at Q. So, $P \oplus Q = Q$. A similar argument can be made for the case when $Q = O$.
2. We must prove if $x_1 = x_2$ and $y_1 = -y_2$ then $P \oplus Q = O$. Let $P = (x_1, y_1)$ and $Q = (x_2, -y_2)$. The line L through P and Q is a vertical line. The third point of intersection between E and L will be the point at infinity O. Thus, $P \oplus Q = O$.
3. There are two cases to consider, $P \neq Q$ and $P = Q$.
Case 1: Suppose $P \neq Q$. We must prove if $P \neq Q$, then $P \oplus Q = (x_3, y_3)$ with $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and $\lambda = \frac{y_2-y_1}{x_2-x_1}$. Observe $x_1 \neq x_2$. This is because if $x_1 = x_2$, then either $y_1 = y_2$ and $P = Q$ (Case 2), or $y_1 = -y_2$ and we would use step 2 of the algorithm. The line L through P and Q is given by,
$$y - y_1 = \lambda(x - x_1).$$
$$y = \lambda x - \lambda x_1 + y_1.$$

We can let $b = -\lambda x_1 + y_1$, so $y = \lambda x + b$. Then to find the intersection of L and E we can substitute L into E. We then have,

$$(\lambda x + b)^2 = x^3 + Ax + B$$

$$(\lambda x)^2 + 2\lambda bx + b^2 = x^3 + Ax + B$$

We can then combine like terms,

$$0 = x^3 - \lambda^2 x^2 + (A - 2\lambda b)x - b^2 + B$$

We already know that x_1 and x_2 are roots of this cubic, we can let x_3 represent the third root. So,

$$(x - x_1)(x - x_2)(x - x_3) = x^3 - \lambda^2 x^2 + (A - 2\lambda b)x - b^2 + B$$

We then expand the left-hand side and we have,

$$x^3-(x_1+x_2+x_3)x^2+(x_1x_2+x_2x_3+x_3x_1)x-x_1x_2x_3 = x^3-\lambda^2x^2+(A-2\lambda b)x-b^2+B$$

If we look at the x^2 coefficient we can see that $-\lambda^2 = -(x_1 + x_2 + x_3)$. Then we can solve for x_3, so $x_3 = \lambda^2 - x_1 - x_2$. Then plugging x_3 back into the line through P and Q, we have $y_3 = \lambda x_3 + b$. Thus, $P \oplus Q = (x_3, y_3)$ with $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and $\lambda = \frac{y_2-y_1}{x_2-x_1}$.

Case 2: Suppose $P = Q$. We must prove if $P = Q$, then $P \oplus Q = (x_3, y_3)$ with $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and $\lambda = \frac{3x^2+A}{2y}$. A similar argument to case 1 can be made but λ will be the slope of the tangent line instead of the slope of the line through P and Q.

□

We mentioned earlier that for every elliptic curve, there exists an isomorphic curve that can be represented in Weierstrass form. There also is a shared value that will be invariant between isomorphic curves, the j-invariant.

Definition 2.13. *The j - **invariant** of an elliptic curve in Weierstrass form is defined by the equation,*

$$j = -1728\frac{(4A)^3}{\Delta}.$$

Theorem 2.14. *Two elliptic curves defined over k are isomorphic over $\bar{k}$, the algebraic closure of k, if and only if they both have the same j-invariant.[Sil16]*

A proof of Theorem 2.15 can be found in [Sil16, Chapter 3].

When using elliptic curves in cryptography it is particularly important to focus our attention on those defined over finite fields. We can show that taking the group law as defined earlier and applying it to elliptic curves over finite fields will result in the points being a finite group. We will now look at an example of an elliptic curve over a finite field.

Example 2.15. *We will first look at the curve $y^2 = x^3 + 1$ over $\mathbb{R}$.*

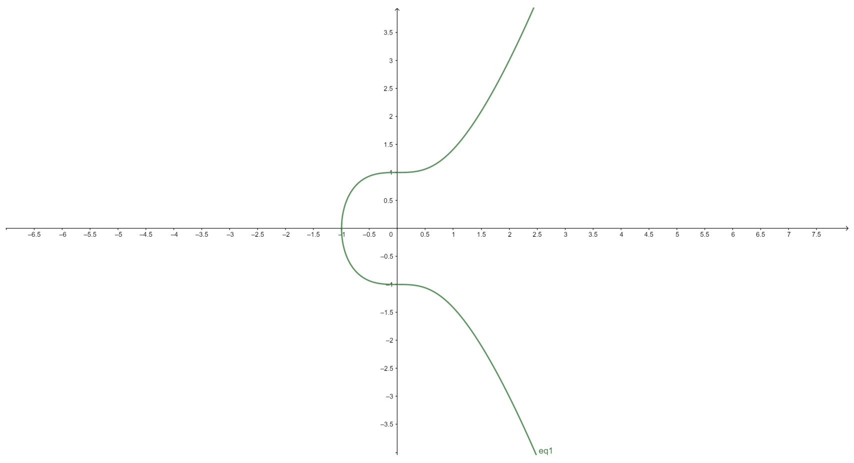

Figure 2.4: $E : y^2 = x^3 + 1$ over $\mathbb{R}$

Now we look at that same equation but look at its roots, $E(\mathbb{F}_{433})$. We wrote code in Sage to produce Figure 2.5 which shows the plot of $E(\mathbb{F}_{433})$ in $\mathbb{F}_{433}^2$.

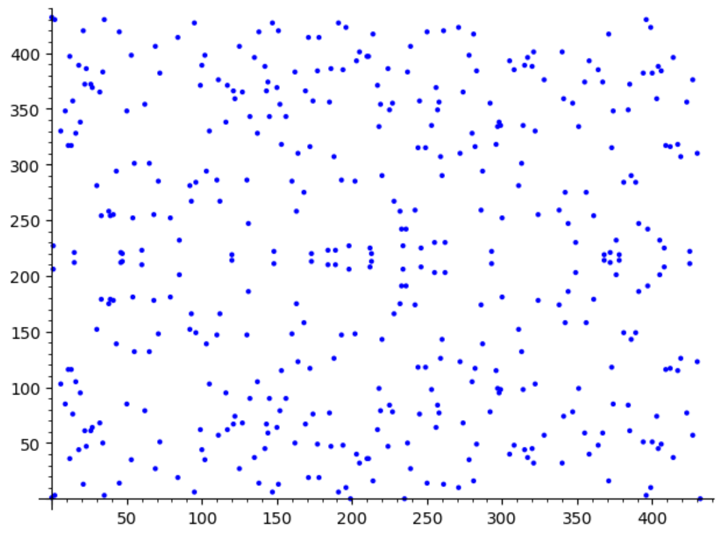

Figure 2.5: $E : y^2 = x^3 + 1$ defined over $\mathbb{F}_{433}$

You can see that the curve defined over $\mathbb{F}_{433}$ is much more chaotic than the curve over $\mathbb{R}$, which makes it much more useful in the world of cryptography.

We can find all points of $E(\mathbb{F}_{433})$ by checking all possible tuples (x, y) to find solutions with $x, y \in \mathbb{F}_{433}$. Let us consider two points that are in $E(\mathbb{F}_p)$: $P = (151, 13)$ and $Q = (162, 383)$. We can verify that these points are in the solution set by plugging in the values into the cubic.

$$13^2(\text{mod } 433) = 151^3 + 1(\text{mod } 433)$$

$$169(\text{mod } 433) = 3442951 + 1(\text{mod } 433)$$

$$169 = 168 + 1(\text{mod } 433)$$

$$169 = 169(\text{mod } 433).$$

So we can see P is in $E(\mathbb{F}_p)$. Following this same process we can show Q is in $E(\mathbb{F}_p)$ as well. We can now use the algorithm from the group law, Theorem 2.12, to find the value of $P \oplus Q$.

Since $P \neq Q$, we can proceed to step 3 of the algorithm. So $\lambda = \frac{(383-13)}{(162-151)} = 370/11 = 73$, since we are working in the finite field $\mathbb{F}_{433}$. Then,

$$x'' = 73^2 - 151 - 162 = 5016 \equiv 253(\text{mod } 433),$$

$$y'' = \lambda(x - x'') - y = 73(151 - 253) - 13 = -7459 \equiv 335 (\text{mod } 433).$$

So, $P \oplus Q = (253, 335)$.

The process of repeatedly adding a point to itself, $n * P$, on E is not quick even with a computer, however we can use an algorithm that will make adding a point to itself multiple times significantly more efficient. In Chapter 3, we present the Fast Powering Algorithm as described in [HPS14, Chapter 1.3].

Finding all points of the an elliptic curve is a cumbersome task that can take a long time, even for computer, if the field is large, i.e. p is large. However, Hemlut Hasse discovered an upper bound for the number of points in the finite group $E(\mathbb{F}_p)$.

Theorem 2.16. *Hasse Theorem [HPS14, Theorem 6.11]*
If E is an elliptic curve over $\mathbb{F}_p$, then

$$\#E(\mathbb{F}_p) = p + 1 - t_p \text{ with } t_p \text{ satisfying } |t_p| \leq 2\sqrt{p}.$$

Let $t_p = P + 1 - \#E(\mathbb{F}_p)$ we call t_p the trace of the Frobenius map (see Definition 2.17 below).

For a detailed proof of this theorem, see [Sil16, Theorem 1.1]. One definition used within that proof that will be useful for us later the *Frobenius Map.*

Definition 2.17. *If p is prime and $k \in \mathbb{Z}_+$, then the (p-power)* ***Frobenius*** *map τ is the map from $\mathbb{F}_{p^k}$ to itself defined by*

$$\tau : \mathbb{F}_{p^k} \to \mathbb{F}_{p^k}, \ \alpha \mapsto \alpha^p$$

Proposition 2.18. *The (p-power) Frobenius map τ is a ring homomorphism from $\mathbb{F}_{p^k}$ to itself.*

Proof. First we show that $\tau(1) = 1$, by observing $\tau(1) = 1^{p^k} = 1$. We now show τ preserves multiplication. Let $a, b \in \mathbb{F}_{p^k}$. Consider, $\tau(ab) = (ab)^p$. Since a, b are elements of a commutative ring then, $\tau(ab) = (ab)^p = a^p b^p = \tau(a)\tau(b)$. Thus, τ preserves multiplication.

We now show that τ preserves addition. Let $a, b \in \mathbb{F}_{p^k}$. Consider,

$$\tau(a + b) = (a + b)^p.$$

We know that, $(a+b)^n = \sum \binom{n}{i} a^i b^{n-i}$. So,

$$(a+b)^p = \sum \binom{p}{i} a^i b^{p-i} = \binom{p}{0} a^0 b^{p-0} + \binom{p}{1} a^1 b^{p-1} + ... + \binom{p}{p-1} a^{p-1} b^1 + \binom{p}{p} a^p b^0.$$

Notice $\binom{p}{0}$ and $\binom{p}{p}$ are both equal to 1. Also since $p | \binom{p}{i}$ for all $0 < i < p$, then $\binom{p}{i} \equiv 0 (\text{mod } p)$ for $0 < i < p$. Then,

$$\begin{aligned}(a+b)^p &= \sum \binom{p}{i} a^i b^{p-i} \\ &= \binom{p}{0} a^0 b^{p-0} + \binom{p}{1} a^1 b^{p-1} + ... + \binom{p}{p-1} a^{p-1} b^1 + \binom{p}{p} a^p b^0 \\ &= b^p + a^p \\ &= a^p + b^p.\end{aligned}$$

So,

$$\tau(a+b) = (a+b)^p = a^p + b^p = \tau(a) + \tau(b).$$

Thus τ preserves addition. Therefore, the set map $\tau : \mathbb{F}_{p^k} \to \mathbb{F}_{p^k}$, where $\alpha \mapsto \alpha^p$, is a ring homomorphism. $\square$

Let us look back at Example 2.17 to see the Hasse Theorem in practice. Let $E : y^2 = x^3 + 1$ be an elliptic curve defined over $\mathbb{F}_{433}$. If we apply the theorem we can see the upper bound $\#E(\mathbb{F}_{433}) \leq 2\sqrt{433} + 433 - 1$. This is a reasonable upper bound since the actual size $\#E(\mathbb{F}_p) = 432$. We will see later a special class of elliptic curves where the trace of the Frobenius map is always 0.

Knowing the upper bound for the number of points of an elliptic curve is useful because it narrows the number of prospects, however for cryptographic applications it is often very important for us to know exactly how many points are in $E(\mathbb{F}_p)$. By checking all possible tuples (x, y) with $x, y \in \mathbb{F}_p$ we can achieve this. However, as mentioned earlier this process is very inefficient and, once the size of the group is too big, counting this way is infeasible. Rene Schoof offered an algorithm for finding the number of points in the group in polynomial-time in [Sch85]. The algorithm was later improved and today the best known way at finding the number of points of an elliptic curve over a finite field is the Schoof-Elkies-Atkin Algorithm (SEA). The proof of the algorithm is beyond the scope of this paper but one can be found in [Sch95].

Another useful piece of information when working with elliptic curve groups is finding all points of a specific order. For example, if E is an elliptic curve defined over a finite field $\mathbb{F}_p$, and $m \in \mathbb{Z}_+$, then we may want to find all points of order m.

Definition 2.19. *Let E be an elliptic curve over k, k'/k a field extension and let $m \in \mathbb{Z}$ with $m > 1$. The m-**torsion points** of $E(k')$, denoted $E(k')[m]$, is the set of points of $E(k')$ of order m,*

$$E(k')[m] = \{P \in E(k') : [m]P = O\}.$$

The collection of these points actually form a group themselves.

Proposition 2.20. *[HPS14, Chapter 6.8.1] Let E be an elliptic curve and let $m \in \mathbb{Z}$ with $m > 1$. The m-torsion points form a subgroup.*

Remark 2.21. *The m-torsion subgroup may not be cyclic.*

Example 2.22. *Let $E : y^2 = x^3 + 1$ be an elliptic curve define over $\mathbb{F}_p$. Using the SEA algorithm in Sage we can compute that $\#E(\mathbb{F}_p) = 84$. Recall that Lagrange's theorem [Gal17, Theorem 7.1] states that the order of an element must divide the order of the group. We can then look to see if there exist an element of order 6 using Sage and find that $(2, 3) \in E$ has order 6. We can then define $E[6]$. Again we use Sage and see that $E[6] = ((2, 3), (2, 94), (25, 3), (25, 94), (70, 3), (70, 94))$.*

We have seen one map, the Frobenius map, between elliptic curves. We now look at isogenies.

Definition 2.23. *If E_1 and E_2 are elliptic curves, then an **isogeny** from E_1 to E_2 is a morphism*

$$\phi : E_1 \longrightarrow E_2 \text{ satisfying } \phi(O) = O$$

*The **degree**, ℓ, of an isogeny is the size of its kernel.*

Remark 2.24. *A morphism is a rational map that is defined at every point.*

Isogenies are a core concept of this paper. Researchers are using the difficulty of computing these maps as hard problems within both cryptosystems as well as hash functions. Later we will look at graphs in order to visualize isogenies and their complexity.

The following definitions will be useful when we discuss possible attacks on the Elliptic Curve Diffie-Hellman Key Exchange.

Definition 2.25. *Let E be an elliptic curve over $\mathbb{F}_p$ and let $m > 1$ be an integer with $p \nmid m$. The* ***embedding degree*** *of E with respect to m is the smallest value of k such that*

$$E(\mathbb{F}_{p^k})[m] \cong \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}.$$

Definition 2.26. *If E is an elliptic curve in Weierstrass form,*

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

and $f(x, y)$ is a nonzero rational function of two variables, then the ***divisor*** *is the sum*

$$div(f) = \sum_{P \in E} n_P[P].$$

Where n_P is the exponents of the zeros and poles of f.

Definition 2.27. *Let $P, Q \in E(k')[m]$. Let f_P and f_Q be rational functions on E satisfying*

$$div(f_P) = m[P] - m[O] \text{ and } div(f_Q) = m[Q] - m[O].$$

The ***Weil pairing*** *of P and Q is the quantity*

$$e_m(P, Q) = \frac{\frac{f_P(Q+S)}{f_P(S)}}{\frac{f_Q(P-S)}{f_Q(-S)}}.$$

Where S is any element in E such that $S \notin \{P, -Q, P \oplus -Q, O\}$

Remark 2.28. *By construction using Millers algorithm, Algorithm 4.9, we know f_P and f_Q exist. For a proof that the Weil pairing is well-defined we direct the reader to [Mil04].*

Definition 2.29. *If $B \in \mathbb{Z}$, then an integer n is called B-**Smooth** if all of its prime factors are less than or equal to B.*

Definition 2.30. *The* ***factor base*** *of B is the set of primes less than or equal to B.*

In Chapter 4 we will discuss the MOV algorithm. This attack uses the Weil pairing and the embedding degree, k, of E to reduce the elliptic curve discrete log problem the classical discrete log problem in $\mathbb{F}_{p^k}$. Elliptic curves with small embedding degrees are vulnerable to this type of attack. We will also see how researchers have taken these vulnerable elliptic curves with small embedding degrees, and used them to create new quantum resistant systems.

Chapter 3

Cryptography Background

3.1 Key Exchanges

Cryptography can be defined as the process of protecting communication between parties by encoding their messages. Within cryptography it is common to refer to the two parties attempting to communicate as Alice and Bob, and we refer to a third party attempting to intercept, alter, or steal information as Eve.

In cryptography one form of exchanging information is a private key cryptosystem. These schemes are referred to as symmetric key exchanges. The basic idea behind these systems is that Alice uses the private secret key to encrypt the data she wants to send to Bob. Bob then receives the data in its encrypted form. Bob then uses the same private, secret key to decrypt the data. Throughout history this was a very common way of exchanging sensitive information. However, one important aspect to notice is that in order for the system to work both Alice and Bob must have access to the private key. In the past the agreed upon private key may have been communicated in person, however in today's world a majority of communication happens on the internet. If Alice wants to prevent Eve from knowing the information she is sending to Bob, she cannot simply send the key to him. Alice and Bob need to find a way to agree upon a private key without running the risk of Eve stealing it. The problem of agreeing upon a shared private key is solved by key exchanges.

One of the most important discoveries in the world of cryptography was by Diffie and Hellman. In their paper, *New Directions in Cryptography* [DH76], they defined both *public key cryptography* and *one-way functions.* Public key Cryptography is also known as asymmetric cryptography. The important characteristic of public key cryptography is the ability of Alice and Bob to securely share information without having a prior discussion about a key. Asymmetric cryptography offers the ability for a private key to be created by using an algorithm and a public key.

One-way function are functions that are very easy to compute, but very hard to invert. It is important to note that for cryptographers, computations that are "easy" are those that are fast, and computations that are "hard" are those that are slow. These one way functions are related to *hard problems.* Cryptography is largely based on the concept of hard problems. Much of cryptography can be boiled down to simple concepts of number theory, however the security lies in the fact that some computations can take an extremely long time to complete. If we want communication between two parties, Alice and Bob, to be secure, then we want the process of decoding the messages between them to be hard. Even if the process for decoding is understood, we want it to be extremely time consuming. It is interesting fact that there is no proof of the existence of one-way functions, however the assumption that one-way functions exist is necessary for cryptography [HPS14, Chapter 1.7].

3.2 The Discrete Log Problem and Diffie-Hellman Key Exchange

Here we will define one of the most common hard problems that is used in cryptography today.

Definition 3.1. *Let g be a primitive root for $\mathbb{F}_p$ and let h be a nonzero element of $\mathbb{F}_p$. The* ***Discrete Log Problem (DLP)*** *is the problem of finding an exponent x such that*

$$g^x \equiv h (\text{mod } p).$$

The number x is called the discrete logarithm of h to the base g and is denoted by $\log_g(h)$.

Compared to logarithms over $\mathbb{R}$, the Discrete Log Problem is much harder. For real numbers a and b the $\log_a b$ is the real number x such that $a^x = b$. This is similar to the DLP, however the important thing about the Discrete Log Problem is that it is working over a finite group.

The Diffie-Hellman Key Exchange (DH) makes use of the Discrete Log Problem to allow Alice and Bob to exchange a secret key on an unsecured platform.

Before moving on to the Diffie-Hellman Key Exchange we will describe the Fast-Powering algorithm. We now present the fast powering algorithm as described in [HPS14, Chapter 1.3].

Algorithm 3.2. ***The Fast-Powering Algorithm***
We want to compute $g^A \pmod N$.

1. *Compute the binary expansion of* A,

$$A = A_0 + A_1 2 + ... + A_r 2^r \text{ with } A_0, ..., A_r \in \{0,1\}$$

2. *Compute the powers of* $g^{2^i} \pmod N$ *for* $0 \leq i \leq r$,

$$\begin{aligned}
&a_0 \equiv g && \pmod N \\
&a_1 \equiv a_0^2 \equiv g^2 && \pmod N \\
&\vdots \\
&a_r \equiv a_{r-1}^2 \equiv g^{2^r} && \pmod N
\end{aligned}$$

3. *Then,*

$$\begin{aligned}
g^A &= g^{A_0 + A_1 2 + ... + A_r 2^r} \\
&= g^{A_0} \cdot (g^2)^{A_1} \cdot (g^{2^2})^{A_2} \cdots (g^{2^r})^{A_r} \\
&= a_0^{A_0} \cdot a_1^{A_1} \cdots a_r^{A_r} \pmod N.
\end{aligned}$$

Key Exchange 3.3. ***Diffie-Hellman Key Exchange****.*

First a large prime p *and integer* $g \pmod p$ *with large order in* $\mathbb{F}_p^x$ *will either be provided by a trusted third party or chosen by Alice and Bob. Alice and Bob will both be using* p *and* g *for their computations so these values will be made public. Alice and Bob will both pick a secret integer, which we denote* a *and* b*, respectively. Alice computes* $A \equiv g^a \pmod p$ *and Bob computes* $B \equiv g^b \pmod p$*. Each of them then sends* *their computed values to one another. Next Alice computes* $A' \equiv B^a (\text{mod } p)$ *and Bob computes* $B' \equiv A^b (\text{mod } p)$*. We can see that* $A' = B'$*, since*

$$A' \equiv B^a (\text{mod } p) \equiv (g^b)^a (\text{mod } p) \equiv (g^a)^b (\text{mod } p) \equiv A^b (\text{mod } p) \equiv B' (\text{mod } p).$$

Thus, Alice and Bob now have their shared secret key.

If Eve now wants to attempt to steal this shared secret key she is left with the task of solving the Discrete Log Problem. To make this clearer let us look at an example.

Public Parameters Large prime p, and integer $g (\text{mod } p)$ with large order	
Alice	Bob
Alice picks random integer a Alice computes $A \equiv g^a (\text{mod } p)$	Bob picks random integer b Bob computes $B \equiv g^b (\text{mod } p)$
Alice sends A to Bob $\longrightarrow$	
	$\longleftarrow$ Bob sends B to Alice
Alice computes $A' \equiv B^a (\text{mod } p)$	Bob computes $B' \equiv A^b (\text{mod } p)$
Shared secret key is $B' = A'$	

Table 3.1: Summary of Diffie-Hellman Key Exchange

Example 3.4. *Public Parameters*

$p = 53$, $g = 21$

Alice	Bob
Alice picks random integer $a = 3$	Bob picks random integer $b = 7$
Computes $A \equiv 21^3 (\text{mod } 53) \equiv 39$	Computes $B \equiv 21^7 (\text{mod } 53) \equiv 35$
Alice sends A to Bob $\longrightarrow$	
	$\longleftarrow$ Bob sends B to Alice
Alice computes $(35^3)\text{mod } 53$	Bob computes $(39^7)\text{mod } 53$

Shared secret key is $(35^3)(\text{mod } 53) \equiv 51 \equiv (39^7)(\text{mod } 53)$

Consider the information available to Eve. Eve knows p, g, A, and B. For Eve to find $A' \equiv B' \equiv 51$, she must solve $21^a \equiv 39 (\text{mod } 53)$ and $21^b \equiv 35 (\text{mod } 53)$. This may seem simple because the parameters chosen for this example are small, however current standards of security require picking primes that are 1024 bits. If Eve was to attempt to attack a key exchange by using brute force trial and error it would take linear time in P,

$O(P)$. In terms of fast and slow, algorithms that run at linear in P are considered slow and algorithms that run at polynomial time are considered fast. This is what makes the DLP a hard problem.

3.3 Solving the DLP

Over the years people have uncovered algorithms that speed up the time it takes to solve the DLP, in certain circumstances dramatically. Shank's Babystep-Giantstep Algorithm is a collision algorithm that creates two lists and by searching each list and finding a match you can solve the DLP. The complexity of an algorithm like this is approximately is $O(\sqrt{N})$, where N is the order of the element g. This is far better than the linear time $O(N)$ of brute force. We then have the Pohlig–Hellman Algorithm which hinges on the use of the Chinese Remainder Theorem to solve a set of congruence and the index calculus method both of which have subexponetial running time [HPS14, Chapter 2]. The index calculus method is the fastest known way to solve the DLP, which can take less than $O(P^{\epsilon})$ steps with $\epsilon > 0$ [Sil16, Chapter 11]. We will next give a brief overview of the index calculus method which will be useful later in this paper.

Index Calculus Method.

We first pick a value B and the set of all primes $l \leq B$ will be our factor base. We then run a loop taking random powers of g and save the values that are B-smooth (Definition 2.29). We can then view these quantities as linear combinations of discrete log problems of primes in the factor base. Using these linear combinations we now have a set of linear equations that we can solve using Gaussian Elimination. We now how found x_i for congruences $g^{x_i} \equiv v_i (\text{mod } p)$ for all v_i in the factor base. We then take $h(g^{-k}) \text{mod } p$ for random k until we find a solution that is B-smooth. Since we have already solved the DLP for values in the factor base we can then combine the information to solve the original DLP.

Index Calculus Sage Code

```
def index_calculus(g,h,p,B):
    order = p - 1
    factbase = prime_range(B+1)
    print(factbase)
    L = []
    m = []
```

```
j = 1
while j<500 and (len(L))<(len(factbase)+1):
    x = ZZ.random_element(p)
    w = FPA(g,x,p)
    if w!=1 and w.factor()[-1][0]<= factbase[-1]:
        s = (g^(x))%p
        exp = []
        for n in factbase:
            e = 0
            temp = s
            while temp%n == 0:
                e+=1
                temp = temp // n
            exp.append(e)
        exp.append(x)
        L.append(exp)
    j += 1
ordfact = order.factor()
solutions = []
print(L)
for k in range (len(ordfact)):
    o = int(ordfact[k][0])
    A = matrix(IntegerModRing(o),L).rref()
    if A.rank() != len(factbase):
        print('Rank issue, automatic re_run')
        new = index_calculus(g,h,p,B)
        return(new)
    m.append(A)
for j in range (len(m)):
    x = []
    for i in range (len(L)):
        x.append(int(m[j][i][-1]))
    solutions.append(x)
mod = []
for i in range (len(ordfact)):
    mod.append(ordfact[i][0])
cong = []
for g1 in range (len(factbase)):
    h1=[]
    for v in range (len(solutions)):
        h1.append(solutions[v][g1])
    cong.append(h1)
Z = []
for y in range (len(cong)):
    sol = CRT_list(cong[y],mod)
    Z.append(sol)
t = 0
while t == 0:
    x2 = ZZ.random_element(p)
    pot = (h*(inverse_mod(g^(x2),p)))%p
    if pot !=0 and pot!=1 and pot.factor()[-1][0]<= factbase[-1]:
        t = 1
exp1 = []
for a2 in factbase:
    e1 = 0
    temp2 = pot
    while temp2%a2 == 0:
        e1+=1
        temp2 = temp2 // a2
    exp1.append(e1)
total = x2
for l in range (len(exp1)):
     total += exp1[l]*Z[l]
print(g,'^',total%(p-1),'=',h,'( mod',p,')')
return(total%(p-1))
```

We will now look at an example comparing the brute force computation time versus that of the Index Calculus Method.

Example 3.5. *Let* $p = 90239$*,* $g = 7$*, and* $7^x(\text{mod } 90239) = 51198$*. To solve this using brute force, we can use the following code:*

```
p = 90239
g = 7
h = 51198
start = time.time()
for i in range (p):
    if (g^i)%p == h:
        print(i)
        break
end = time.time()
x = end - start
print('Time taken for brute force is',x)
```

The output of this code:

```
41506
Time taken for brute force is 22.68076229095459
```

We can then solve for x using the Index Calculus Method.

```
p = 90239
g = 7
h = 51198
B = 11
start1 = time.time()
index_calculus(g,h,p,B)
end1 = time.time()
x1 = end1 - start1
print('Time taken using Index Calculus Method', x1)
```

The output for this code:

```
7 ^ 41506 = 51198 ( mod 90239 )
Time taken using Index 0.7332439422607422
```

We can see that using the Index Calculus Method significantly decreased the time it took to solve the DLP, even for this small example.

Chapter 4

Classical Elliptic Curve Cryptography

4.1 Elliptic Curve Discrete Log Problem

Similar to the hard problem used in the Diffie-Hellman key exchange, there exists an analogous hard problem within the field of elliptic curves.

Definition 4.1. *Let E be an elliptic curve over a finite field $\mathbb{F}_p$ and let P and Q be points in $E(\mathbb{F}_p)$. The **Elliptic Curve Discrete Log Problem** is the problem of finding an integer n such that $Q = n * P$.*

To emphasize the similarities between the DLP and ECDLP we can write this as

$$n = \log_p(Q).$$

Cryptographers have exploited the difficulty in finding n by creating elliptic curve based cryptosystems that have similar protocols to Diffie-Hellman and ElGamal [HPS14, Chapter 2.4]. In order to compute the values needed for the key exchange, Alice and Bob are required to compute $d * P = P \oplus P \oplus ... \oplus P$, d times. If we pick d to be big, finding the point $d * P$ can take a long time. The **Double-and-Add Algorithm** is an effective and can compute $n * P$ in $O(\log_2 d)$ time.

Algorithm 4.2. *Double-and -Add*
Input *Point $P \in E(\mathbb{F}_p)$*

1. *Set $Q = P$ and $R = O$.*

2. *Loop while $d > 0$.*

3. *If $n \equiv 1 (\bmod 2)$, set $R = R \oplus Q$.*

4. *Set $Q = 2 * Q$ and $d = \lfloor d/2 \rfloor$.*

5. *If $d > 0$, continue with loop at Step **2**.*

6. *Return the point R, which equals $d * P$.*

If the ECDLP is analogous to DLP and there also exist The Double-and-Add Algorithm which is analogous to The Fast Powering Algorithm (Algorithm 3.2), then you might ask why would we use elliptic curves over $\mathbb{F}_p^*$? It turns out that even the fastest algorithms today used to solve the ECDLP have a complexity approximately $O(\sqrt{p})$. This stands in contrast to DLP over $\mathbb{F}_p^*$, for which there exist algorithms that can run at subexponetial time, i.e. The Index Calculus Method.

Remark 4.3. *With the implementation of certain algorithms the Discrete Log Problem over $\mathbb{F}_p^*$ is easier to solve than the Elliptic Curve Discrete Log Problem. However, any cryptosystem could be made sufficiently secure if the parameters were picked accordingly, i.e. pick extremely large primes. The ECDLP is more secure than DLP over $\mathbb{F}_p^*$, when picking primes of similar size for both. Cryptographers must always balance security with representability and implementability.*

4.2 Elliptic Curve Diffie-Hellman

Leveraging the similarities the between ECDLP and the DLP over $\mathbb{F}_p^*$ we can easily define a key exchange using elliptic curves that is analogous to Diffie-Hellman Key Exchange 3.3.

Key Exchange 4.4. *Elliptic Curve Diffie-Hellman Key Exchange*
For Alice and Bob to begin exchanging information, a trusted third party must first publish public parameters. This trusted third party will publish a prime p, an elliptic curve $E(\mathbb{F}_p)$, typically via the Weierstrass coefficients $[A, B]$, and a base point $P \in E(\mathbb{F}_p)$. Then Alice will pick a secret key $a \in \mathbb{Z}$ and she will compute, using the Double-and-Add Algorithm,

$A = a * P$. She will then send her point A to Bob. Similarly, Bob will pick a secret key $b \in \mathbb{Z}$ and he will compute, using the Double-and-Add Algorithm, $B = b * P$. He will then send his point B to Alice. Alice will then take Bob's point B and compute $a * B$, and Bob will take Alice's point A and compute $b * A$. Since $a * B = a * (b * P) = b * (a * P) = b * A$, the shared secret point on the curve between Alice and Bob will be $a * B = b * A$. These are points in $\mathbb{F}^2$, for the shared secret key Alice and Bob will take the first coordinate of their shared secret point.

If Eve intends to steal the secret key that Bob and Alice now posses, Eve must figure out the values of either a or b. This boils down to Eve solving $n * P = A$ or $n * P = B$ for n which is precisely the ECDLP.

Public Parameters Prime p, Elliptic Curve $E(\mathbb{F}_p)$. Point $P \in E(\mathbb{F}_p)$	
Alice	Bob
Alice picks random integer a Alice computes $A = a * P$	Bob picks random integer b Bob computes $B = b * P$
Alice sends A to Bob $\longrightarrow$	
	$\longleftarrow$ Bob sends B to Alice
Alice computes $a * B$	Bob computes $b * A$
Shared secret point is $a * B = b * A$	

Table 4.1: Summary of ECDH Key Exchange

Example 4.5. *ECDH*

Public Parameters

$$p = 97,\ E(\mathbb{F}_{97}) : y^2 = x^3 + x + 4, P = (39, 80)$$

Alice	Bob
Alice picks random integer $a = 61$	Bob picks random integer $b = 86$
Alice computes $A = a * P = (0, 2)$	Bob computes $B = b * P = (26, 90)$
Alice sends A to Bob $\longrightarrow$	
	$\longleftarrow$ Bob sends B to Alice
Alice computes $a * B = (54, 73)$	Bob computes $b * A = (54, 73)$

Shared secret point is $a * B = (54, 73) = b * A$

Shared secret key is 54

4.3 Sending and Certifying Messages

Continuing to expand on the analogy between the DLP and ECDLP, we now present an ElGamal style cryptosystem based on elliptic curves.

4.3.1 Elliptic Curve ElGamal Public Key Cryptosystem

Similar to Elliptic Curve Diffie-Hellman, a trusted third party will publish a prime p, an elliptic curve $E(\mathbb{F}_p)$, and a base point $P \in E(\mathbb{F}_p)$. Alice will then choose a private key n_A and compute a point $Q_A = n_A * P$. Alice will then send the point Q_A to Bob. Bob can then choose a plain text $M \in E(\mathbb{F}_p)$, and a random element k. He then computes two values $C_1 = k * P$ and $C_2 = M \oplus k * Q_A$, and sends these values as an order pair, (C_1, C_2), to Alice. Using C_1 and C_2, Alice computes $C_2 - n_A * C_1 = M \in E(\mathbb{F}_p)$. [HPS14, Chapter 6.4.2]

Public Parameters	
a prime p, an elliptic curve $E(\mathbb{F}_p)$, and a base point $P \in E(\mathbb{F}_p)$	
Alice	Bob
Alice picks a private key n_A Alice computes $Q_A = n_A * P$	
Alice sends Q_A to Bob $\longrightarrow$	
	Bob chooses plain text $M \in E(\mathbb{F}_p)$. Bob chooses random element k Bob computes two values $C_1 = k * P$ and $C_2 = M \oplus k * Q_A$
	$\longleftarrow$ Bob sends (C_1, C_2) to Alice
Alice computes $C_2 - n_A * C_1 = M \in E(\mathbb{F}_p)$	

Table 4.2: Summary of Elliptic Curve ElGamal Key Exchange

4.3.2 Digital Signatures

One of the main uses of elliptic curves in the world of cryptography is in digital signatures. The digital currency, Bitcoin, uses the Elliptic Curve Digital Signature Algorithm for its security (ECDSA). Before we summarize the ECDSA we will first give some background on digital signatures.

What is a digital signature?

Digital signatures are used in a slightly different way than the way in which our regular handwritten signatures are used, they serve to certify the authenticity and fidelity of digital documents. This is different from the public key cryptosystems which allow Alice and Bob to share information, but the methods used in public key cryptosystems are also used in digital signature algorithms. Digital signature algorithms have been created in such a way that if Alice signs a digital document, Bob can then use Alice's public key to verify that Alice is the actual sender and the document has not been tampered with. This is important because it is possible that Eve may tamper with the information sent to Bob. Without digital signatures, Bob may unknowingly believe this information was from Alice.

Prior to the use of elliptic curves in cryptography the *Digital Signature Algorithm* (DSA) was the standard for signing digital documents. A complete explanation of the DSA can be found in [HPS14, Chapter 6, Section 5]. For our purpose we will focus on the ECDSA. Moving from Diffie-Hellman and ElGamal to ECDH and Elliptic curve ElGamal was useful because the level of security was higher while keeping the parameters (bitsize) at a manageable size. Similarly moving from DSA to ECDSA was an obvious transition. We now explain the steps in the Elliptic Curve Digital Signature Algorithm.

Algorithm 4.6. *Elliptic Curve Digital Signature Algorithm*

Public Parameters A finite field $\mathbb{F}_p$, an elliptic curve $E/\mathbb{F}_p$, and a point $P \in E(\mathbb{F}_p)$ of prime order N	
Alice	Bob
1) Alice picks a private key a 2) Alice computes $A = a * P \in E(\mathbb{F}_p)$ 3) Alice then chooses document $d(\text{mod } N)$ to sign and random integer $k(\text{mod } N)$ 4) Alice then computes: $s_1 \equiv X(k * P)(\text{mod } N)$ and $s_2 \equiv (d + as_1)k^{-1}(\text{mod } N)$ 5) Alice publishes (s_1, s_2)	
	1) Bob computes $v_1 \equiv ds_2^{-1}(\text{mod } N)$ and $v_2 \equiv s_1 s_2^{-1}(\text{mod } N)$ 2) Bob then computes $v_1 {*} P {+} v_2 {*} A$. 3) Bob then verifies $X(v_1 {*} P {+} v_2 {*} A) \equiv s_1(\text{mod } N)$

Table 4.3: Summary of Elliptic Curve Digital Signature Algorithm [Sil16, Chapter 11]

Remark 4.7. *We use X to denote taking the x-coordinate of a point (x, y), i.e. $X(x, y) = x$.*

We can show that if Alice follows the steps above then Bob will be able to verify her as the sender.

Proof. Let $P \in E(\mathbb{F}_p)$ be prime order N, and $A = a * P \in E(\mathbb{F}_p)$. For all computations we will be working mod N, so for simplicity we will drop mod N for some parts of the proof. Also let $s_1 \equiv X(k * P)(\text{mod } N)$ and $s_2 \equiv (d + as_1)k^{-1}(\text{mod } N)$ for random integer $k(\text{mod } N)$, and $v_1 \equiv ds_2^{-1}(\text{mod } N)$ and $v_2 \equiv s_1 s_2^{-1}(\text{mod } N)$. We must show $X(v_1 * P + v_2 * A) \equiv s_1(\text{mod } N)$. Consider $X(v_1 * P + v_2 * A)$. If we substitute for v_1, v_2, and A then,

$$X(v_1 * P + v_2 * A) = X[(ds_2^{-1}) * P + (s_1 s_2^{-1})(a) * P]$$

$$= X[s_2^{-1}(d + as_1)] * P.$$

We can then substitute in for s_2,

$$\begin{aligned} X(v_1 * P + v_2 * A) &= X[((d + as_1)k^{-1})^{-1}(d + as_1)] * P \\ &= X[((d + as_1)^{-1}k^{1})(d + as_1)] * P \\ &= X[k * P] \\ &= s_1. \end{aligned}$$

□

We can see that Bob can accurately verify the identity of the sender as Alice. This is secure because Alice never sends her private key so only she knows it. Also, if Eve was able to intercept the document and tamper with it this would obstruct Bob's ability to verify in step 3 of the algorithm, in which he would infer the message was tampered with and discard it.

4.3.3 Attacks on Elliptic Curve Cryptography

If Alice and Bob are exchanging information using Diffie-Hellman or ElGamal, Eve must find an effective way to solve the DLP. If, instead, Alice and Bob are using ECDH or Elliptic Curve ElGamal, Eve is then left with the task of solving the ECDLP. We have already discussed that in general the fastest known algorithms used to solve the ECDLP have a complexity approximately $O(\sqrt{p})$. However, if cryptographers are not careful in how they choose the curves being used, they may be leaving themselves vulnerable to faster attacks.

We will now discuss the special cases for which the ECDLP can be turned into the DLP over $\mathbb{F}_{p^k}$. In [MVO93], Menezes, Okamoto, and Vanstone were able to show that the ECDLP can be reduce to the DLP over F_{p^k} with k being the embedding degree of the elliptic curve.

We now summarize the algorithm described in that paper.

Algorithm 4.8. *MOV Algorithm*
Let E be an elliptic curve over $\mathbb{F}_p$ of embbedding degree k and $P \in E(\mathbb{F}_p)$ be a point of order ℓ. If $Q = n * P$ for some n, then given Q and P the following algorithm can produce n.

1. Compute $N = \#E(\mathbb{F}_{p^k})$

2. Pick a random point $T \in E(\mathbb{F}_{p^k})$ with $T \in E(\mathbb{F}_p)$.

3. Compute $T' = (N/\ell) * (T)$. If $T' = O$, then pick new random point T. Otherwise T' has order ℓ.

4. Compute Weil Pairing $\alpha = e_\ell(P, T')$ and $\beta = e_\ell(Q, T')$. Note: both α and β are in $\mathbb{F}_{p^k}^*$.

5. Solve DLP $\beta = \alpha^n$ for n.

6. Then $Q = n * P$.

A proof of the the MOV can be found in [Sil16, Chapter 11, Section 6] or [MVO93].

One issue that arises when attempting to implement the MOV Algorithm is the necessity to compute the Weil pairing. To compute the Weil pairing we must compute rational functions with specific divisors. Victor Miller offers an efficient way to find this value [Mil04]. Miller's Algorithm is described in [HPS14], and we describe it here.

Algorithm 4.9. *(Miller Algorithm) Let E be an elliptic curve and let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be nonzero points on E.*

(a) Let λ be the slope of the line connecting P and Q, or the slope of the tangent line to E at P if $P = Q$. Define a function $g_{P,Q}$ on E as follows:

$$g_{P,Q} = \begin{cases} \frac{y - y_P - \lambda(x - x_P)}{x + x_P + x_Q - \lambda^2} & \text{if } \lambda \neq \infty, \\ x - x_P & \text{if } \lambda = \infty. \end{cases}$$

Then

$$div(g_{P,Q}) = [P] + [Q] - [P + Q] - [O].$$

(b) (Miller's Algorithm) Let $m \leq 1$ and write the binary expansion of m as

$$m = m_0 + m_1 2 + m_2 2^2 + ... + m_{n-1} 2^{n-1}$$

with $m_i \in 0, 1$ and $m_{n-1} \neq 0$. The following algorithm returns a function f_p whose divisors satisfies

$$div(f_P) = m[P] - [mP] - (m - 1)[O],$$

where the functions $g_{T,T}$ and $g_{T,P}$ used by the algorithm are as defined in (a).

```
1) Set $T = P$ and $f = 1$
2) Loop $i = n - 2$ down to 0
3) Set $f = f^2(g_{T,T})$
4) Set $T = 2 * T$
5) If $m_i = 1$
6) Set $f = fg_{T,P}$
7) Set $T = T \oplus P$
8) Return the value $f$
```

In particular, if $P \in E[m]$, then $div(f_P) = m[P] - m[O]$.

We now offer a sample of Sage code to implement the MOV Algorithm

```
def gfunc(P,Q,p):
    F = GF(p)
    R.<x,y> = PolynomialRing(F)
    if P == Q:
        slope = (((3*(P[0])^2+a)/(2*P[1]))%p)
    elif P[0] == Q[0]:
        slope = 'infinity'
    else:
        slope = (((Q[1]-P[1])/(Q[0]-P[0]))%p)
    if slope == 'infinity':
        g = x-P[0]
    else:
        g = (y - P[1] - slope*(x-P[0]))/(x + P[0] + Q[0] - (slope)^2)
    return(g)
def Miller(m,P,Q,a,b,p):
    T = P
    f = 1
    m=m.digits(2)
    n = len(m)
    for i in range(n-2,-1,-1):
        f = (f^2)*gfunc(T,T,p)
        T = Group_Law(T,T,a,p)
        if m[i] == 1:
            f = f*gfunc(T,P,p)
            T = Group_Law(T,P,a,p)
    return(f)
def weil(m,P,Q,a,b,p,S):
    f_p = Miller(m,P,Q,a,b,p)
    num = (f_p(Group_Law(Q,S,a,p)))/(f_p(S))
    f_q = Miller(m,Q,P,a,b,p)
    negS = (S[0],-S[1])
    den = (f_q(Group_Law(P,negS,a,p)))/(f_q(negS))
    weil = (num/den)%p
    return(weil)
def MOV2(p,a,b,P,Q):
    E = EllipticCurve(GF(p), [a,b])
    l = E.cardinality()
    k = embedding_degree(P,p)
    if k > 6:
        print('--------------------------------------\
              ----------------------------------------\
              -------------')
        return('Embedding Degree is too large, \
               DLP over GF(P^K) is just a difficult\
               as ECDLP over E(GF(p))')
    E2 = EllipticCurve(GF(p^k),[a,b])
    N = E2.cardinality()
    P=E2(P)
    Q=E2(Q)
    T2 = (E2.random_element())*\
        (((E2.random_element()).additive_order())//l)
    while T2.additive_order() == 1:
        T2 = (E2.random_element())*\
            (((E2.random_element()).additive_order())//l)
    #print(T2)
    alpha = P.weil_pairing(T2,l)
    beta = Q.weil_pairing(T2,l)
    for i in range (p):
        if alpha^i == beta:
            return(i)
```

We can now look at an example of the MOV algorithm in action.

Example 4.10. *Let $p = 8111$, $E(\mathbb{F}_{8111}) : y^2 = x^3 + z + 300$, $P = (6116, 2715)$ and $Q = (3786, 7380)$. We want to find n such that $n * P = Q$. We will first offer Sage code to compute this by brute force.*

```
start =time.time()
for i in range (p):
    if P*i == Q:
        print(i)
        break
end =time.time()
print(end-start)
```

The output for this code:

```
4034
1.448946475982666
```

We now look at the same problem computed using the MOV algorithm.

```
start1 =time.time()
n = MOV(p,a,b,P,Q)
end1 =time.time()
print(n,end1 - start1)
```

The output for this code:

```
4034 0.05069732666015625
```

We can see that both brute force and MOV produce the same result 4034*, however the in the example the MOV algorithm was almost* 30 *times as fast. This difference in speed becomes even clearer as we pick larger primes.*

Remark 4.11. *Within the code we have offered for the MOV algorithm we use brute force to solve the DLP. If the index calculus method is used instead our code would solve the ECDLP even faster, possibly in subexponetial time.*

Chapter 5

Supersingular Isogeny Diffie-Hellman Key Exchange

As we mentioned earlier the growing danger of quantum computers has lead to the need to develop a cryptosystems that is secure against such computers. Many systems have been developed and proposed, one of which is based on isogenies between supersingular elliptic curves [DFJP11]. In this chapter we cover the SIDH public key system.

In Chapter 4 we discussed a vulnerability within classical ECC that allowed the attacker to reduce the Discrete Log Problem in $E(\mathbb{F}_p)$ to the Discrete Log Problem in $\mathbb{F}_p$. That vulnerability was supersingular curves, however for the Supersingular Isogeny Diffie-Hellman (SIDH) key exchange we make use of these curves.

5.1 Supersingular Elliptic Curves

There are many equivalent ways to define **supersingular** elliptic curves. We will define it in a way that is most relevant to this paper.

Definition 5.1. *An elliptic curve $E(\mathbb{F}_p)$ for $p > 3$ is* ***supersingular*** *if the embedding degree k of $E(\mathbb{F}_p)$ is $k \leq 6$.*

The MOV Algorithm discussed in Chapter 4 exploits the small embedding degree of supersingular curves in order to solve the ECDLP. Though these curves are avoided in classical elliptic curve cryptography, in post-quantum cryptography the properties of these curves can be used to strengthen security. One of the hard problems that researchers have proposed for the use in cryptography is the computing of isogenies. However, there already exists an algorithm to compute isogenies for ordinary curves in sub-exponential time on quantum computers. On the other hand, the endomorphism rings of supersingular curves are not necessarily commutative, which prevents the use of the sub-exponential algorithm to compute isogenies.

Another practical characteristic of supersingular elliptic curves is the size of $E(\mathbb{F}_p)$.

Proposition 5.2. *If $E(\mathbb{F}_p)$ is a supersingular elliptic curve, then $\#E(\mathbb{F}_p) = p + 1$.*

It is very useful for cryptographers to easily know $\#E(\mathbb{F}_p)$. When picking parameters for cyrpotsytems there are standard size of parameters that are chosen to add a certain level of security.

We now present the Supersingular Isogeny Diffie-Hellman Key Exchange. This key exchange is conducted by constructing isogenies between supersingular elliptic curves. Jacques Vélu, discovered formulas to construct these isogenies [Vé71]. The precise statement and proof of Velu's formula is outside the scope of this paper. As such we will not present it here, and rather, we refer the reader to [Vé71] or [MS16]. For our purposes, it is sufficient to understand that Velus formula allows us to input the equation of an elliptic curve over a field k and a point $P \in E(k')$, and in return we get an isogenous curve $E/\langle P \rangle$ and the isogeny $\phi : E \to E/\langle P \rangle$.

5.2 Supersingular Isogeny Diffie-Hellman Key Exchange

Key Exchange 5.3. *Supersingular Isogeny Diffie Hellman Key Exchange*

To begin some public parameters must be set. A trusted third party will publish a prime p. This prime will be of the form $p = l_a^{e_a} l_b^{e_b} f + 1$. Standard practice is to let $l_a = 2$ and $l_b = 3$ with $e_a, e_b \in \mathbb{Z}$. Along with p, the third party will publish a supersingular elliptic curve, E_0, in Weierstrass form defined over $\mathbb{F}_{p^2}$. Lastly, points $P_a, Q_a \in E[l_a^{e_a}]$ and $P_b, Q_b \in E[l_b^{e_b}]$ will be published.

Alice will then use $P_a, Q_a \in E[l_a^{e_a}]$ and two, secret, random integers m_a and n_a to form a subgroup. Using Velu's formula [MS16], Alice can now compute a secret isogeny ϕ_a to a new curve E_a with her constructed subgroup as the kernel of ϕ_a. She then sends

Bob E_a, $\phi_a(P_b)$ and $\phi_a(Q_b)$. Following these same steps Bob will send E_b, $\phi_b(P_a)$ and $\phi_b(Q_a)$ to Alice.

Alice will then take $\phi_b(P_a)$ and $\phi_b(Q_a)$ and her secret random integers m_a and n_a to form another subgroup. She then uses this subgroup as a kernel of an isogeny from E_b to a new curve E_{ba}. Bob will follow these same steps using the information Alice has sent to him and he will arrive at a curve E_{ab}. The key exchange relies on the result: The two elliptic curves E_{ba} and E_{ab} are isomorphic. This is summarized in Theorem 5.4. Alice and Bob then use the j-invariant of these curves as their shared secret key. Table 5.1 summarizes this process. [DFJP11]

Theorem 5.4. *[DFJP11,] If $p = l_a^{e_a} l_b^{e_b} f + 1$, $E_o = E(\mathbb{F}_{p^2})$ is a supersingular curve, $P_a, Q_a \in E[l_a^{e_a}]$ and $P_b, Q_b \in E[l_b^{e_b}]$, $m_a, n_a \in \mathbb{Z}/l_a^{e_a}\mathbb{Z}$, $m_b, n_b \in \mathbb{Z}/l_b^{e_b}\mathbb{Z}$ and;*

1. $\phi_a = E_o/\langle m_a * P_a \oplus n_a * Q_a\rangle$ *and* $\phi_b = E_o/\langle m_b * P_b \oplus n_b * Q_b\rangle$
2. $\phi_a : E_o \to E_a$ *and* $\phi_b : E_o \to E_b$
3. $\phi'_a = E_b/\langle m_a * \phi_b(P_a) \oplus n_a * \phi_b(Q_a)\rangle$ *and* $\phi'_b = E_a/\langle m_b * \phi_a(P_b) \oplus n_b * \phi_a(Q_b)\rangle$
4. $\phi'_a : E_b \to E_{ba}$ *and* $\phi'_b : E_a \to E_{ab}$

Then $E_{ba} \cong E_{ab}$.

Public Parameters Prime $p = l_a^{e_a} l_b^{e_b} f + 1$, supersingular elliptic curve $E(\mathbb{F}_{p^2})$ Points $P_a, Q_a \in E[l_a^{e_a}]$ and $P_b, Q_b \in E[l_b^{e_b}]$	
Alice	Bob
Computes $\phi_a = E/\langle m_a * P_a \oplus n_a * Q_a\rangle$ Sends E_a,$\phi_a(P_b)$,$\phi_a(Q_b)$ Computes $\phi'_a =$ $E_b/\langle m_a * \phi_b(P_a) \oplus n_a * \phi_b(Q_a)\rangle$ Arrives at E_{ba}	$\phi_b = E/\langle m_b * P_b \oplus n_b * Q_b\rangle$ Sends E_b,$\phi_b(P_a)$,$\phi_b(Q_a)$ Computes $\phi'_b =$ $E_a/\langle m_b * \phi_a(P_b) \oplus n_b * \phi_a(Q_b)\rangle$ Arrives at E_{ab}
Since $E_{ba} \cong E_{ab}$ the shared secret key is their j-invariant	

Table 5.1: Summary of SIDH Key Exchange

Let us now look at a simple example using Sage to do the computation.

Example 5.5. *Public Parameters: As is common practice we will let $l_a = 2$ and $l_b = 3$. We then let $e_a = 2, e_b = 3$ and $f = 1$. We then have $p = 2^2 3^3 - 1 = 107$. Let the beginning curve be $E_o : y^2 = x^3 + x$.*

$$P_a = (69i + 45, 37i + 65), Q_a = (69i, 39i + 106)$$

$$P_b = (69i + 48, 39i + 19), Q_b = (14i + 79, 106i + 33)$$

Note: $i = \sqrt{-1}$ in $\mathbb{F}_{p^2}$.

Alice:

Alice then picks her secret integers $m_a = 53$ and $n_a = 50$.

Then $E_a = y^2 = x^3 + (47i + 104)x + (99i + 98)$.

Alice sends Bob $(E_a, \phi_a(P_b), \phi_a(_b))$

Bob:

Bobs then picks his secret integers $m_b = 5$ and $n_b = 68$.

Then $E_b = y^2 = x^3 + (45i + 52)x + (83i + 82)$.

Bob sends Alice $(E_b, \phi_b(P_a), \phi_b(Q_a))$.

Alice:

Alice uses information sent from Bob to arrive at E_{ba}.

$E_{ba} : y^2 = x^3 + (30i + 33)x$ with j-invariant 16.

Bob:

Bob uses information sent from Alice to arrive at E_{ab}.

$E_{ab} : y^2 = x^3 + (30i + 33)x$ with j-invariant 16.

Thus the shared secret key is 16.

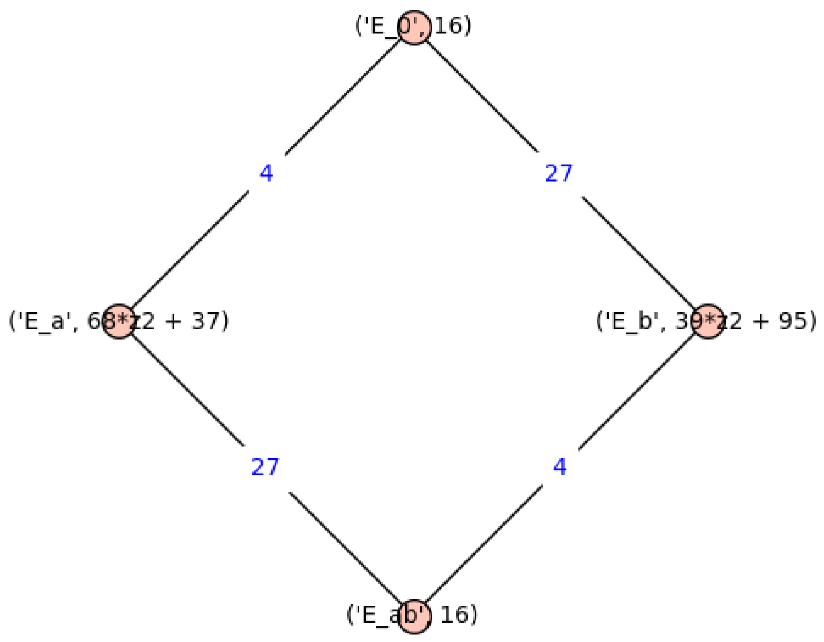

Figure 5.1: Pathway taken by Alice and Bob in Example 5.6

In Figure 5.1, the vertices of the graph are elliptic curves within the key exchange represented by their j-invariant. The edges of the graph are the isogenies between the elliptic curve labeled by their degree. We use this graph as a representation of the pathways taken by Alice and bob. The idea of pathways is convenient because we will later look at isogeny graphs that give a visual representation of all the possible paths Alice and Bob could have taken.

5.3 Hard Problem

Like all cryptosystems, the SIDH key exchange is based on a hard problem. Earlier in this paper we looked at the Diffie-Hellman Key Exchange which bases its security on the difficulty of solving the Discrete Log Problem. We then moved to the Elliptic Curve Diffie-Hellman Key Exchange, which similarly based its security on the Elliptic Curve Discrete Log Problem. What then is the hard problem associated with Supersingular-Isogeny based cryptosystems? Although the SIDH has strong connections with ECDH the hard problem is not a version of the DLP, instead it is rooted in the difficulty of finding isogenys between elliptic curves. This problem is known as the Computational Supersingular Isogeny, (CSSI), problem [CGL06]. Throughout the protocol Eve has the potential to steal information such as E_a and E_b, however knowing these curves as well as E_o does not give her any information about the isogenies between them, ϕ_a and ϕ_b. Since she has no information on ϕ_a or ϕ_b she cannot then have any information on Alice and Bob's secret keys, which will prevent her from reaching the final shared key. The complexity of CSSI is complicated and still being understood. In the world of quantum computers, cryptographers must consider attacks from both classical computers and quantum computer. In their paper on SIDH [DFJP11], Jao, De Feo and Plut state the time complexity of CSSI to be $O(p^{1/4})$ for classical computer and slightly faster for quantum computers at $O(p^{1/6})$. Early on, researchers believed that the best way to solve CSSI was a meet-in-the-middle algorithm, however recently it was showed in [ACVCD+19] that the collision algorithm offered by van Oorschot and Wiener in [vOW96] preformed better. The basic idea behind these attacks is to start at 2 elliptic curves and take random walks along edges of isogeny graphs until they meet. In the next section we cover the constrcution of these isogeny graphs.

5.4 Supersingular Isogeny Graphs

Super singular isogeny graphs serve many uses within the field of cryptography. These graphs have been considered for the use of creating attacks on isogeny based cryptosystems, as well as being used to create hash functions. They are examples of a type of highly connected graph called expander graphs. Further information for the use of expander graphs can be found in [CGL06]. We begin by going over necessary definitions.

Definition 5.6. *The* ***supersingular ℓ-isogeny graph over $\bar{\mathbb{F}}_p$,*** *$\mathcal{G}_\ell(\bar{\mathbb{F}}_p)$, is the graph whose vertices are the $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular elliptic curves labeled by their j-invariant over $\mathbb{F}_{p^2}$ and there is an edge between 2 vertices if there exists an isogeny between them of degree ℓ .[ACNL+19]*

Definition 5.7. *The* ***spine*** *$\mathcal{S}$ is the full subgraph of $\mathcal{G}_\ell(\bar{\mathbb{F}}_p)$ with vertices defined over $\mathbb{F}_p$ and edges being isogenies in $\mathcal{G}_\ell(\bar{\mathbb{F}}_p)$.[ACNL+19]*

Definition 5.8. *The* ***supersingular l-isogeny graph over*** *$\mathbb{F}_p$, $\mathcal{G}_\ell(\mathbb{F}_p)$, is the graph whose vertices are the $\mathbb{F}_p$-isomorphism classes of supersingular elliptic curves labeled by* *their j-invariant, and edges of the graph are isogenies of degree ℓ. [ACNL^{+}19]*

A detailed investigation of these graphs can be found in [ACNL^{+}19]. We now give sample Sage code to produce these graphs.

```
#automatically produce isogoney graph for isogenies of degree l
def SIG3(p,l):
    F = GF(p)
    graph2 = []
    V = []
    J = []
    L = []
    M = []
    W = []
    coef = []
    for i in range (p):
        for j in range (p):
            if (4*i^3 + 27*j^2)%p != 0:
                m = [i,j]
                if (EllipticCurve(F,m)).is_supersingular() == True:
                    M.append(m)
    for m in M:
        E = EllipticCurve(F,m)
        j = E.j_invariant()
        if j not in set(J):
            coef.append(m)
            V.append((m,j))
            W.append((m))
            J.append(j)
    #print(W)
    for a in W:
        for m in M:
            if EllipticCurve(F,a).j_invariant() ==
            EllipticCurve(F,m).j_invariant() and
            EllipticCurve(F,a).is_isomorphic(EllipticCurve(F,m))==False:
                V.append((m,EllipticCurve(F,m).j_invariant()))
                coef.append(m)
                break
    print(V,len(V))
    print('-----------------------------------------------')
    #print(coef)
    for i in range (len(V)):
        E1 = EllipticCurve(F, V[i][0])
        print(E1,V[i][1])
        affine_points = []
        for P in E1:
            if P.order() == l:
                affine_points.append(list(P)[:2])
        print(affine_points)
        for P1 in affine_points:
            E2 = E1.isogeny(E1(P1))
            E3 = E2.codomain()
            if E2.degree() == l :
                for y in coef:
                    if EllipticCurve(F,y).is_isomorphic(E3):
                        E3 = EllipticCurve(F,y)
                graph2.append(((E1.j_invariant(),
                (E1.a_invariants()[3],E1.a_invariants()[4])),
                (E3.j_invariant(),
                (E3.a_invariants()[3],E3.a_invariants()[4])),l))
    print('----------------------------------------------------\
--------------------------------------')
    print(graph2,len(graph2))
    if len(J) < 2:
        print("All supersingular curves over GF(",p,") are isomorphic")
    else:
        G = Graph(graph2,loops = True)
        return(G)
```

Figure 6.2 offers an example of the implementation of the above code for $p = 101$ and $\ell = 3$. Observe each vertex is labeled $(j, (A, B))$ where $[A, B]$ are the Weierstrass coefficients and j is the j-invariant of the associated curve.

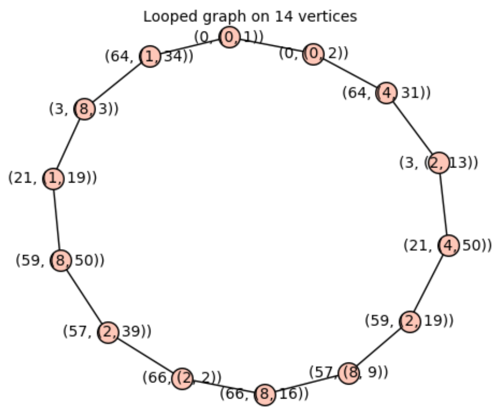

Figure 5.2: $\mathcal{G}_\ell(\mathbb{F}_p)$ for $p = 101$

We can also overlap two of these graphs to have the edges represent isogenies of two different degrees. We show an example of this in Figure 6.3.

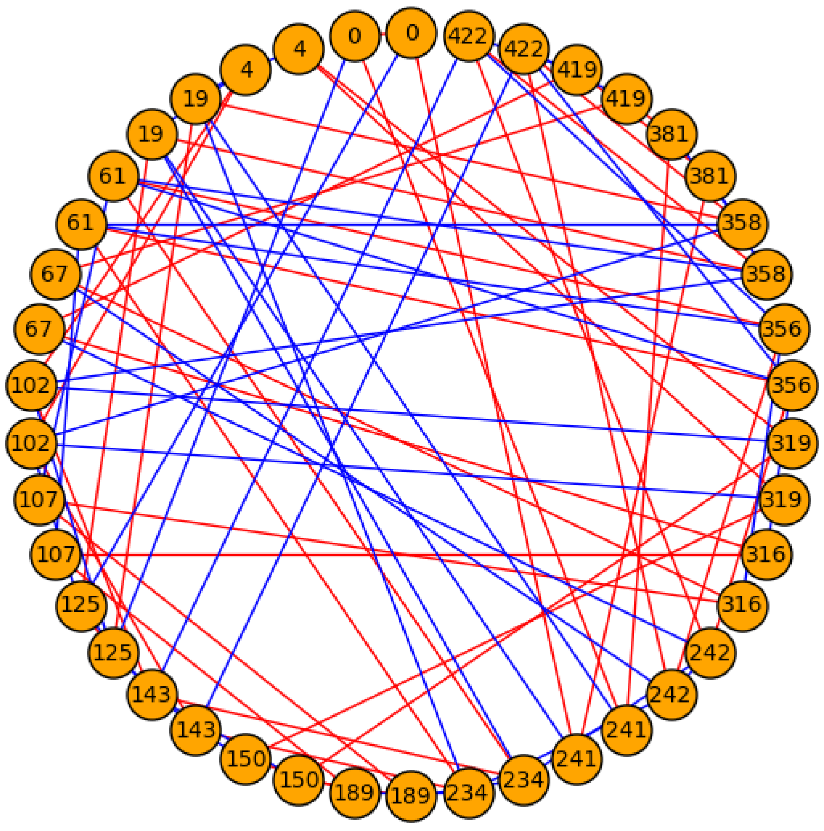

Figure 5.3: Supersingular Isogoney Graph $\mathcal{G}_{2,3}(\mathbb{F}_{431})$

In Figure 5.3, the vertices are all $\mathbb{F}_p$-isomorphism classes of supersingular elliptic curves over $\mathbb{F}_{431}$. The edges of this graph isogenies between these curves off degree 2 and 3, represented by blue and red respectively. This graph gives a visual representation of the possible paths taken by Alice and Bob within the SIDH key exchange. Eve will have access to E_a and E_b, the first paths taken by Alice and Bob respectively. She will not have access to the next paths taken by Alice and Bob, this is where the security of SIDH can be found. These graphs are highly connected, making the quantity of possible pathways for Alice and Bobs to take very high. The complexity and connectivity of these graphs will only grow as we pick larger primes. In Figure 5.3 we are using a $p = 431$, this is a relatively small prime, and an example offered by De Feo, Jao, and Plut in [DFJP11] uses $p = 37004441637405283255944010403058171248 63$.

The National Institute of Standards and Technology is conducting a competition to find a secure option for post-quantum cryptography. One of the submissions for the competions by David Jao et al. is the Supersingular Isogeny Key Encapsulation. The Supersingular Isogeny Diffie–Hellman key exchange is the basis for the Supersingular Isogeny Key Encapsulation (SIKE).

Chapter 6

Cyclic Subgroup Diffie-Hellman Key Exchange

6.1 CSDH

As way of better understanding Supersingular Isogeny Diffie-Hellman Key Exchange we offer an analogues cryptographic system that is rooted in cyclic groups. We call this Cyclic Subgroup Diffie-Hellman Key Exchange, (CSDH). The CSDH key exchange is very similar to the SIDH. In both systems we see Alice and Bob each making two moves along the way to an end shared secret key. We can now summarize the Cyclic Subgroup Key Eexchange.

Key Exchange 6.1. *CSDH*
First a trusted third party will publish $\mathbb{Z}_S$ and $P_a, Q_a, P_b, Q_b \in \mathbb{Z}$.

1. *Alice and Bob both pick secret integers m_a, n_a and m_b, n_b respectively.*

2. *Alice then computes $A = P_a(n_a) + Q_a(m_a)$ and Bob computes $B = P_b(n_b) + Q_b(m_b)$.*

3. *Alice then finds $\mathbb{Z}_S \big/ \langle A \rangle = G_a$ and Bob finds $\mathbb{Z}_S \big/ \langle B \rangle = G_b$.*

4. *Alice computes $\phi(P_b), \phi(Q_b)$ with $\phi : \mathbb{Z}_S \to G_a$ and Bob computes $\psi(P_a), \psi(Q_a)$ with $\psi : \mathbb{Z}_S \to G_b$.*

5. *Alice sends Bob $G_a, \phi(P_b), \phi(Q_b)$, and Bob sends Alice $G_b, \psi(P_a), \psi(Q_a)$.*

6. *Alice computes* $\bar{A} = \psi(P_a)(n_a) + \psi(Q_a)(m_a)$*, and Bob computes* $\bar{B} = \phi(P_b)(n_b) + \phi(Q_b)(m_b)$.

7. *Alice computes* $G_{endA} = G_b \big/ \langle \bar{A} \rangle$*, and Bob computes* $G_{endB} = G_a \big/ \langle \bar{B} \rangle$

8. $G_{endA} = G_{endB}$

9. *The shared secret key is* $|G_{endA}| = |G_{endB}|$

$$A = Pa(na) + Qa(ma), B = Pb(nb) + Qb(mb)$$
$$\bar{A} = \psi(Pa)(na) + \psi(Qa)(ma), \bar{B} = \phi(Pb)(nb) + \phi(Qb)(mb)$$

$$\begin{tikzcd}
 & \mathbb{Z}_S = G \arrow[dl, "\text{Alice}_1"', "\phi"] \arrow[dr, "\text{Bob}_1", "\psi"'] & \\
\mathbb{Z}_S \big/ \langle A \rangle = G_a \arrow[rr, bend left, "\text{Alice Sends } G_a, \phi(P_b), \phi(Q_b)"] \arrow[dr, "\text{Alice}"'] & & \mathbb{Z}_S \big/ \langle B \rangle = G_b \arrow[ll, bend left, "\text{Bob Sends } G_b, \psi(P_a), \psi(Q_a)"] \arrow[dl, "\text{Bob}"] \\
 & G_{endA} = G_b \big/ \langle \bar{A} \rangle = G_a \big/ \langle \bar{B} \rangle = G_{endB} &
\end{tikzcd}$$

Figure 6.1: Summary of Cyclic Subgroup Diffie-Hellman Key Exchange

For CSDH, the shared secret key is the order of the end cyclic group that Alice and Bob both compute separately. The current representation of the algorithm uses the language of cyclic groups, however we can reinterpret this into the language of number theory using modular arithmetic and gcd. This is useful because although the representation is more clean when using cyclic group notation, the computations are easier when boiled down to number theory. The following theorem allows us to move from the language of cyclic groups to basic number theory.

Theorem 6.2. *[Gal17, Theorem 4.2] Let a be an element of order n in a cyclic group and let k be a positive integer. Then $\langle ak\rangle = \langle (a)gcd(n,k)\rangle$ and $|ak| = n/gcd(n,k)$.*

Using Theorem 6.2 we can see from Key Exchange 6.1 $|\langle A\rangle| = S/\gcd(S,A)$. Then we have $|G_a| = |\mathbb{Z}_S\big/\langle A\rangle| = |\mathbb{Z}_S|\big/|\langle A\rangle| = S\big/S/\gcd(S,A) = \gcd(S,A)$. We now represent our CSDH using only number theory concepts.

Key Exchange 6.3. *A trusted third party publishes $S \in \mathbb{Z}$ and $P_a, Q_a, P_b, Q_b < S$.*

1. *Alice and Bob pick secret random integers $n_a, m_a, n_b, m_b \in \mathbb{Z}$.*
2. *Alice computes $A = P_a n_a + Q_a m_a$ and Bob computes $B = P_b n_b + Q_b m_b$.*
3. *Alice computes $newA = \gcd(S,A)$, and Bob computes $newB = \gcd(S,B)$.*
4. *Bob computes $ImP_a = Pa(\text{mod } newB)$ and $ImQ_a = Q_a(\text{mod } newB)$ and Alice computes $ImP_b = P_b(\text{mod } newA)$ and $ImQ_b = Q_b(\text{mod } newA)$*
5. *Alice sends Bob $newA, ImP_b, ImQ_b$, Bob sends Alice $newB, ImP_a, ImQ_a$.*
6. *Alice computes $\bar{A} = ImP_a n_a + ImQ_a m_a$ and Bob computes $\bar{B} = ImP_b n_b + ImQ_a m_b$.*
7. *Shared secret key is $\gcd(newB, \bar{A}) = \gcd(newA, \bar{B})$.*

$$\begin{tikzcd}
& S \arrow[dl, "\text{Alice}_1"'] \arrow[dr, "\text{Bob}_1"] & \\
newA = \gcd(S,A) \arrow[rr, shift left, "\text{Alice Sends } newA, ImP_b, ImQ_b"] \arrow[dr, "\text{Alice}"'] & & newB = \gcd(S,B) \arrow[ll, shift left, "\text{Bob Sends } newB, ImP_a, ImQ_a"] \arrow[dl, "\text{Bob}"] \\
& endA = gcd(newB, \bar{A}) = gcd(newA, \bar{B}) = endB &
\end{tikzcd}$$

Figure 6.2: Summary of CSDH

The following core result certifies that, if Alice and Bob follow the key exchange correctly, then they will always have the same shared secret key.

Theorem 6.4. *If $S \in \mathbb{Z}$ and $P_a, Q_a, P_b, Q_b < S$ and $n_a, m_a, n_b, m_b \in \mathbb{Z}$ and;*

1. $A = P_a n_a + Q_a m_a$ *and* $B = P_b n_b + Q_b m_b$
2. $newA = \gcd(S, A)$ *and* $newB = \gcd(S, B)$
3. $ImP_a = Pa\text{mod }(newB)$ *and* $ImQ_a = Q_a\text{mod }(newB)$
4. $ImP_b = P_b\text{mod }(newA)$ *and* $ImQ_b = Q_b\text{mod }(newA)$
5. $\bar{A} = (ImP_a)n_a + (ImQ_a)m_a = A + c_1(newB)$ *and* $\bar{B} = (ImP_b)n_b + (ImQ_b)m_b = B + c_2(newA)$

Then $\gcd(newB, (ImP_a)n_a + (ImQ_a)m_a) = \gcd(newA, (ImP_b)n_b + (ImQ_a)m_b)$.

Before we prove this theorem we must prove a lemma.

Lemma 6.5. *Using the notation above,* $\bar{A} = (ImP_a)n_a + (ImQ_a)m_a = A + c_1(newB)$ *and* $\bar{B} = (ImP_b)n_b + (ImQ_b)m_b = B + c_2(newA)$.

Proof. We will start by showing $\bar{A} = (ImP_a)n_a + (ImQ_a)m_a = A + c_1(newB)$. Since $ImP_a = P_a(\text{mod } newB)$, then $ImP_a = P_a - q_1(newB)$ for some $q_1 \in \mathbb{Z}$. Similarly, $ImQ_a = Q_a - q_2(newB)$ for some $q_2 \in \mathbb{Z}$. So we have,

$$\bar{A} = [P_a - q_1(newB)]n_a + [Q_a - q_2(newB)]m_a.$$

After rearranging and combining like terms, we have,

$$\bar{A} = P_a n_a + Q_a m_a + (-q_1 - q_2)newB.$$

Since $A = P_a n_a + Q_a m_a$ and $(-q_1 - q_2) \in \mathbb{Z}$, then $\bar{A} = A + c_1(newB)$ for some $c_1 \in \mathbb{Z}$. A similar argument can be made to show $\bar{B} = B + c_2(newA)$ for some $c_2 \in \mathbb{Z}$. □

We now prove Theorem 6.4.

Proof. We must show $\gcd(newB, \bar{A}) = \gcd(newA, \bar{B})$. We can show this by showing

$$\gcd(newB, \bar{A}) | \gcd(newA, \bar{B}) \text{ and } \gcd(newA, \bar{B}) | \gcd(newB, \bar{A}).$$

Since $\gcd(\gcd(a,b),c) = \gcd(a,b,c)$, then $\gcd(newB,\bar{A}) = \gcd(S,B,\bar{A})$ and $\gcd(newA,\bar{B}) = \gcd(S,A,\bar{B})$. So we must show $\gcd(S,B,\bar{A}) \mid \gcd(S,A,\bar{B})$.

We will first show $\gcd(S,B,\bar{A}) \mid \gcd(S,A,\bar{B})$. By definition $\gcd(S,B,\bar{A}) | S$. We now show $\gcd(S,B,\bar{A}) | A$. We can show that if $d|newB$ and $d|\bar{A}$, then $d|A$. Since $d|\bar{A}$, then by Lemma 5.5 $dz = \bar{A} = A + c(newB)$ for some $z \in \mathbb{Z}$. Since we know $d|newB$ then d must divide A. Since $\gcd(S,B,\bar{A})$ divides $newB$ and $\bar{A}$, then $\gcd(S,B,\bar{A})|A$.

We now show $\gcd(S,B,\bar{A})|\bar{B} = B + c(newA)$. Since $newA = \gcd(S,A)$ and $\gcd(S,B,\bar{A})$ divides S and A, then $\gcd(S,B,\bar{A})|newA$.

So $\gcd(S,B,\bar{A})|B$ and $\gcd(S,B,\bar{A})|newA$ and so $\gcd(S,B,\bar{A})|\bar{B}$. Thus $\gcd(S,B,\bar{A})$ divides S, A, and $\bar{B}$, so $\gcd(S,B,\bar{A}) \mid \gcd(S,A,\bar{B})$. By a similar argument, we can show $\gcd(S,A,\bar{B}) \mid \gcd(S,B,\bar{A})$.

So $\gcd(S,B,\bar{A}) = \gcd(S,A,\bar{B})$. □

Example 6.6. *Public Parameters*

$$S = 2310, P_a = 595, Q_a = 1825, P_b = 598, Q_b = 1947$$

Alice	Bob
Alice picks random integers $m_a = 42$, and $n_a = 66$	Bob picks random integers $m_b = 1155$ and $n_b = 770$
Alice computes:	Bob computes:
$A = 115920$	$B = 2709245$
$newA = 210$	$newB = 385$
$ImP_b = 1780$ and $ImQ_b = 57$	$ImP_a = 210$ and $ImQ_a = 285$
Alice sends $newA, ImP_b$ and ImQ_b to Bob $\longrightarrow$	
	$\longleftarrow$ Bob sends $newB, ImP_a$ and ImQ_a to Alice
Alice computes $\bar{A} = 25830$	Bob computes $\bar{B} = 202895$

Shared secret key is $gcd(newB,\bar{A}) = 35 = gcd(newA,\bar{B})$

From this example we can see that the shared secret key is 35. In order for Eve to uncover this information she would have to know either $newA$ and $\bar{B}$ or $newB$ and $\bar{A}$. For Eve to have that information she would need both A and B, however finding out A and B is extremely difficult for Eve because m_a, n_a, m_b and n_b are all secret and only Alice and Bob know them respectively.

However if Alice and Bob are not careful, then Eve can have a pretty good idea that the end shared secret key will be 1. The following example demonstrates what can go wrong with poor parameter selection.

Example 6.7. *Again we will let $S = 2310$. Let $P_a = 1339, Q_a = 1916, P_b = 1604$ and $Q_b = 1102$. Alice picks randomly her private keys $m_a = 2$ and $n_a = 7$. Bob picks his private keys $m_b = 462$ and $n_b = 770$. After doing their calculations Alice has $newA = 5$ and Bob has $newB = 15$. Then the end shared secret key is $endA = 1 = endB$. The exchange is summarized here in this diagram.*

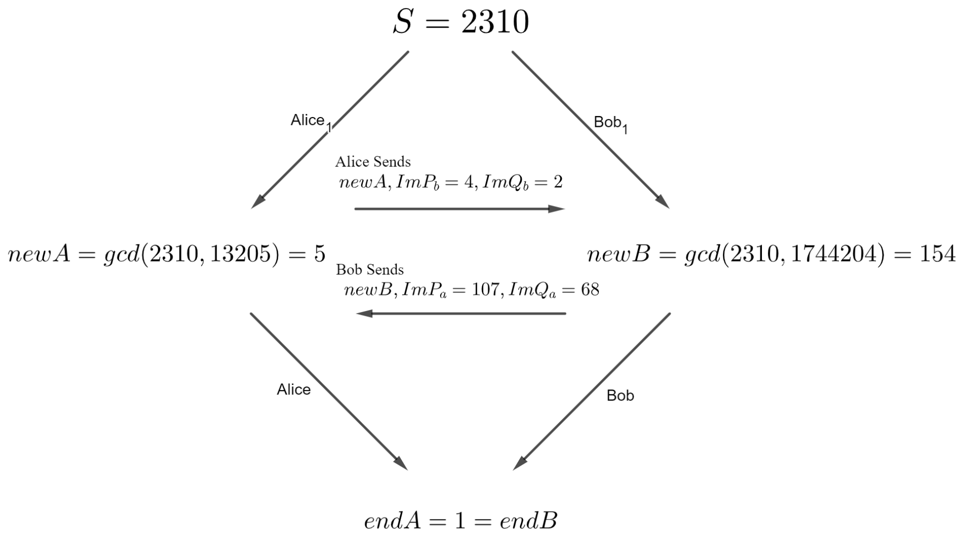
$S = 2310$
Alice$_1$
Bob$_1$
Alice Sends
$newA, ImP_b = 4, ImQ_b = 2$
$newA = gcd(2310, 13205) = 5$
$newB = gcd(2310, 1744204) = 154$
Bob Sends
$newB, ImP_a = 107, ImQ_a = 68$
Alice
Bob
$endA = 1 = endB$

Figure 6.3: CSDH with poor parameter selection.

Through experimentation we found that it was very common for the final shared secret key to be 1. This is a significant vulnerability in the key exchange because if Eve is interested in stealing information, she can with high certainty assume that the shared secret key is 1.

Proposition 6.8. *Following the steps from Key Exchange 6.3, if $A < S$, then the probability of $newA = 1$ is $\frac{\phi(S)}{S}$ where ϕ is Euler's totient function.*

Proof. Let $A < S$, and ϕ be Euler's totient function. Observe, $newA = \gcd(A, S)$. There are exactly $\phi(S)$ many positive integers less than S that are relatively prime to S. Thus, if $A < S$, then the probability of $newA = 1$ is $\frac{\phi(S)}{S}$. $\square$

We can however make some adjustments to the parameter selection in the key exchange that make it unlikely or even impossible for the end result to be 1. We now explain those adjustments.

6.2 Statistical Analysis

Recall from the original Key Exchange 6.1, Alice and Bob must construct values A and B. For Alice $A = P_a(m_a)+Q_a(n_a)$ and for Bob $B = P_b(m_b)+Q_b(n_b)$. The integers P_a, Q_a, P_b and Q_b are public parameters, so they are known to everyone, including Eve. In order to make values A and B secure from Eve we keep m_a, n_a, m_b, and n_b secret. We originally picked the private secret keys randomly, however this lead to vulnerabilities with our end result. If we pick these private keys more wisely, then we can have a much higher level of security. Notice that $newA, newB, endA$ and $endB$ are all constructed by computing the gcd. In order to prevent the end from boiling down to 1, we must have higher levels of shared divisibility between the integers for which we are computing the gcd. We now introduce the k-factor.

Definition 6.9. *If S is an integer and $d|S$, then the **k-factor** is the percent of prime factors of S which divide d.*

We use this k-factor when picking our private keys. When picking these integers if, instead of only picking randomly, we pick randomly with the restriction that it must be built out of high percent of primes in S, we can then have a higher shared divisibility between A, B and S.

Example 6.10. *Let $S = 210 = 7 \cdot 5 \cdot 3 \cdot 2$, a product of 4 primes. If we let $k = 50\%$ then Alice and Bob will pick their private secret keys out of a random combination of* 2 *primes.*

We wrote Sage Python code to compute this.

```
def point_gen(n,k):
    L = []
    i = 0
    while i<(n*k):
        x = random.choice(primes_first_n(n))
        if x not in set(L):
```

```
        L.append(x)
        i +=1
    h = prod(L)
    return(h)
```

Theorem 6.11. *If S be the product of the first N primes, $P_a, Q_a, P_b, Q_b < S$, $n_a, m_a, n_b, m_b \in \mathbb{Z}$ with k-factor, k, and;*

1. $A = P_a n_a + Q_a m_a$ *and* $B = P_b n_b + Q_b m_b$
2. $newA = \gcd(S, A)$ *and* $newB = \gcd(S, B)$
3. $ImP_a = Pa\text{mod }(newB)$ *and* $ImQ_a = Q_a\text{mod }(newB)$
4. $ImP_b = P_b\text{mod }(newA)$ *and* $ImQ_b = Q_b\text{mod }(newA)$
5. $\bar{A} = ImP_a n_a + ImQ_a m_a = A + c_1(newB)$ *and* $\bar{B} = ImP_b n_b + ImQ_a m_b = B + c_2(newA)$
6. $endA = \gcd(newB, ImP_a n_a + ImQ_a m_a) = \gcd(newA, ImP_b n_b + ImQ_a m_b) = endB.$

Then, at worst, the k-factor of $newA$ or $newB$ is $(2(\frac{k}{100\%}) - 1)\%$ and $endA = endB$ is $(4(\frac{k}{100\%}) - 3)\%$.

Proof. We will first show the k-factor of $newA$ or $newB$ is $(2(\frac{k}{100\%}) - 1)\%$. Let $\mathbb{M}_a = \{x \mid x \text{ is a prime factor of } m_a\}$ and $\mathbb{N}_a = \{y \mid y \text{ is a prime factor of } n_a\}$. Consider, $\mathbb{M}_a \cap \mathbb{N}_a = \{h \mid h \in \mathbb{M}_a \text{ and } h \in \mathbb{N}_a\}$. Since both $\mathbb{M}_a$ and $\mathbb{N}_a$ are sets containing $k\%$ of the primes in S, then by the inclusion–exclusion principle, $\mathbb{M}_a \cap \mathbb{N}_a$ is a set containing at least $(2(\frac{k}{100\%}) - 1)\%$ of the primes in S. We now consider $A = P_a n_a + Q_a m_a$. Let $L = \mathbb{M}_a - (\mathbb{M}_a \cap \mathbb{N}_a)$ and $G = \mathbb{N}_a - (\mathbb{M}_a \cap \mathbb{N}_a)$.

$$\begin{aligned}
A &= P_a n_a + Q_a m_a \\
&= P_a \prod_{x \in \mathbb{M}_a} x + Q_a \prod_{y \in \mathbb{N}_a} y \\
&= P_a \prod_{h \in \mathbb{M}_a \cap \mathbb{N}_a} h \prod_{l \in L} l + Q_a \prod_{h \in \mathbb{M}_a \cap \mathbb{N}_a} h \prod_{g \in G} g \\
&= \prod_{h \in \mathbb{M}_a \cap \mathbb{N}_a} h(P_a \prod_{l \in L} l + Q_a \prod_{g \in G} g).
\end{aligned}$$

Let $(P_a \prod_{l \in L} l + Q_a \prod_{g \in G} g) = w$. Since $w \in \mathbb{Z}$ and $\gcd(w, S)$ may be 1, then the $newA = \gcd(S, A) = \gcd(S, \prod_{h \in \mathbb{M}_a \cap \mathbb{N}_a} h(P_a \prod_{l \in L} l + Q_a \prod_{g \in G} g)$ is at worst $\prod_{h \in \mathbb{M}_a \cap \mathbb{N}_a} h$. Observe

$\prod_{h\in\mathbb{M}_a\cap\mathbb{N}_a} h$ is the product of all $h \in \mathbb{M}_a \cap \mathbb{N}_a$ which is the product of $(2(\frac{k}{100\%}) - 1)\%$. Thus, the k-factor of $newA$ is at least $(2(\frac{k}{100\%}) - 1)\%$. Using a similar argument we can show the k-factor of $newB$ is at least $(2(\frac{k}{100\%}) - 1)\%$.

We now prove the k-factor of $endA = endB$ is at least $(4(\frac{k}{100\%}) - 3)\%$. Since $endA = endB$ it is sufficient to provide the argument for either $endA$ or $endB$, we will show for $endA$. Recall, $endA = gcd(newB, \bar{A})$. By the above argument we know the k-factor of $newB$ is at least $(2(\frac{k}{100\%}) - 1)\%$. Also by a similar argument given for A we can show that the k-factor of $\bar{A}$ is at least $(2(\frac{k}{100\%}) - 1)\%$ also. We can then use the inclusion-exclusion principle again.
We then have,

$$newB \cup \bar{A} = newB + \bar{A} - newB \cap \bar{A}$$

In terms of k-factor this gives us,

$$1 = \left(2(\frac{k}{100\%}) - 1\right)\% + \left(2(\frac{k}{100\%}) - 1\right)\% + k\text{-factor of } \gcd(newB, \bar{A})$$

We then simplify terms.
Thus, k-factor of $\gcd(newB, \bar{A}) = endA = endB = (4(\frac{k}{100\%}) - 3)\%$.

$\square$

A consequence of the theorem is the following proposition.

Proposition 6.12. *Let $S =$the product of the first N primes. Let $k = 80\%$. If Alice and Bob follow the steps described in Key Exchange 6.1, then at worst the end shared secret key will be a product of 20% of the primes used to build S.*

After doing further experimentation with different values for k, we found that $k = 80\%$ seemed to be an optimal value. Eve knows that the end secret key is the product of some amount of primes that were used to build S. Let us say S is build out of the first N primes. Her success in stealing that shared secret key lies in choosing the correct combination of some set of primes in S. We know by the binomial distribution that the greatest number of possibilities for Eve to try occurs when the end result is built out of $N/2$ number of primes, $\binom{N}{\frac{N}{2}}$. Through our experimentation we consistently found that when $k = 80\%$, our expected number of primes in the end secret key was close to half of that in S.

We ran an experiment such that the number of primes to build S ranged from 10 to 300, moving in increments of 10. For each size S we ran 100 trials. We let $k = 80\%$, picked public parameters randomly, and picked private secret keys using the above code. The results are summarized in Table 6.1 and Figure 6.4.

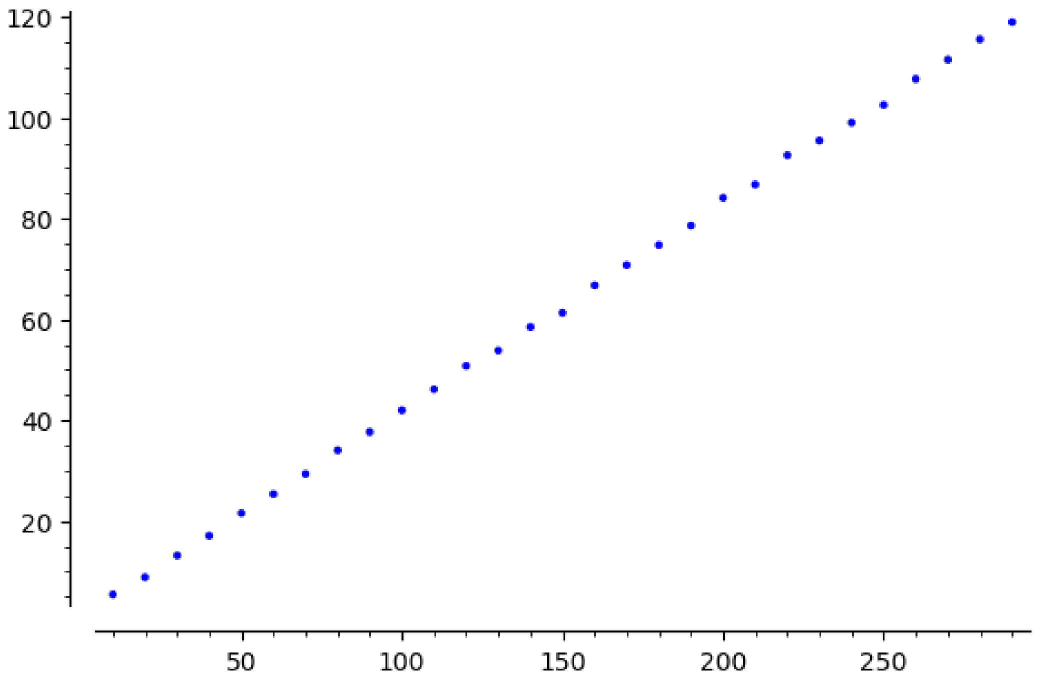

Figure 6.4: Random trials with S ranging from 10 - 290 and $k = 80\%$

Number of primes in S	Average number of primes in end shared secret key
10	5.46000000000000
20	8.90000000000000
30	13.2100000000000
40	17.1200000000000
50	21.6000000000000
60	25.4000000000000
70	29.3800000000000
80	34.0500000000000
90	37.7000000000000
100	41.9800000000000
110	46.1500000000000
120	50.8100000000000
130	53.8500000000000
140	58.5100000000000
150	61.3200000000000
160	66.7700000000000
170	70.7900000000000
180	74.7600000000000
190	78.6100000000000
200	84.1200000000000
210	86.7500000000000
220	92.5800000000000
230	95.5000000000000
240	99.0300000000000
250	102.530000000000
260	107.680000000000
270	111.520000000000
280	115.550000000000
290	118.960000000000

Table 6.1: Results from experiment of CSDH

Based of these trials, it seems that when the k-factor is set to be 80% the number of primes in the end shared key is approximately half of the number of primes that were used to build S. To further explore and understand this we used Sage to enumerate all possible combinations of private secret keys with fixed public parameters. Within the key exchange there are four private keys, two for Alice and two for Bob, so the number of all possible combinations is

$$\binom{\textbf{\# of primes in S}}{\textbf{\# of primes in S} \cdot (\frac{k}{100})}^4.$$

The number of possible combinations can be extremely large. Although having a large number of possibilities is very useful for the security of the exchange, enumerating all possible combinations can be time consuming, making analysis difficult.

Example 6.13. *For the purpose of these computations we will choose the public parameters to be small. We let $S = 2310$ which is a product of 5 primes and we build the private keys using with $k = 80\%$ or 4 primes. We can use the above equation to calculate the total number of combinations of private keys. We then have,*

$$\binom{\textbf{\textit{\# of primes in S}}}{\textbf{\textit{\# of primes in S}} \cdot (\frac{k}{100})}^4 = \binom{5}{(5 * (\frac{80}{100})}^4$$
$$= \binom{5}{4}^4$$
$$= 5^4$$
$$= 625$$

So we have 625 possible combinations. Using code we wrote in Sage, we ran through all 625 combinations and collected the data into a spread sheet. The collection of this data can be found in Appendix A. After analyzing the data, we found that, if we take the average number of primes in the end shared key, then we get 2.55, which is 51% of 5. This is consistent with our earlier findings that with $k = 80\%$ the amount of primes in the endA or endB is approximately half of S. P

We did many enumerations of this kind, computing all possible combinations of private keys. As a reminder of notation N is the number of primes used to construct S, and EV is the percentage of primes from S in the end shared secret key. We display some of those results here in the following table.

P_a	Q_a	P_b	Q_b	N	k	EV
2	3	4	5	5	80%	51%
2	3	4	5	5	90%	69%
216	949	1493	183	5	80%	48%
2	3	4	5	6	50%	14%
2	3	4	5	6	67%	28%
2	3	4	5	6	83%	56%
2	3	4	5	6	100%	100%
2	3	4	5	10	80%	46%
4747641764	3477213385	4437360853	11606517	10	80%	46%

Table 6.2: Results from enumerations of CSDH

Using access to a computer cluster we were able to compute some larger trials [ATI22]. For $N = 5$, we were able to enumerate all possible combinations of m_a, n_a, m_b and n_b while allowing the public keys to range from 1 to 10. This may seem small, however as mentioned above there are 625 combinations of secret keys. If we pair that with allowing each of the 4 public keys to range from 1 to 10, then the total number of possible combinations is 6250000.

Proposition 6.14. *Let $S = 2310$ the product of the first 5 primes, and let $k = 80\%$. If we compute all combinations of $P_a, Q_a, P_b,$ and Q_b, where $P_a, Q_a, P_b, Q_b \in \{1, ..., 10\}$, then the average number of primes in the endA and endB is 2.57 or 51% of those in S.*

If we take the above parameters and use the product of the first 10 primes instead of 5 we end up with similar results. The number of possible combinations for these parameters is much larger at 41006250000. Due to the size of this enumeration we could not compute it. We did however enumerate all possible combinations of secret keys for $N = 10$ with fixed public keys. Those results are shown in Proposition 6.15.

Proposition 6.15. *Let $S = 6469693230$ the product of the first 10 primes, and let $k = 80\%$. If we fix $P_a = 2, Q_a = 3, P_b = 4,$ and $Q_b = 5$, then the average number of primes in the endA and endB is 4.85 or 46% of those in S.*

To further our exploration of the trends of the key exchange we fixed S and picked the public and private parameters randomly with $k = 80\%$. For this example we fixed $S = 7799922041683461553249199106329813876687996789903550945093032474868511536164700810$ or the product of the first 46 primes. We wrote Sage code to run 300 trials. The results of those trials showed the minimum number of primes in the end shared key to be 14 and the maximum number of primes to be 26. We also plotted the results in a histogram.

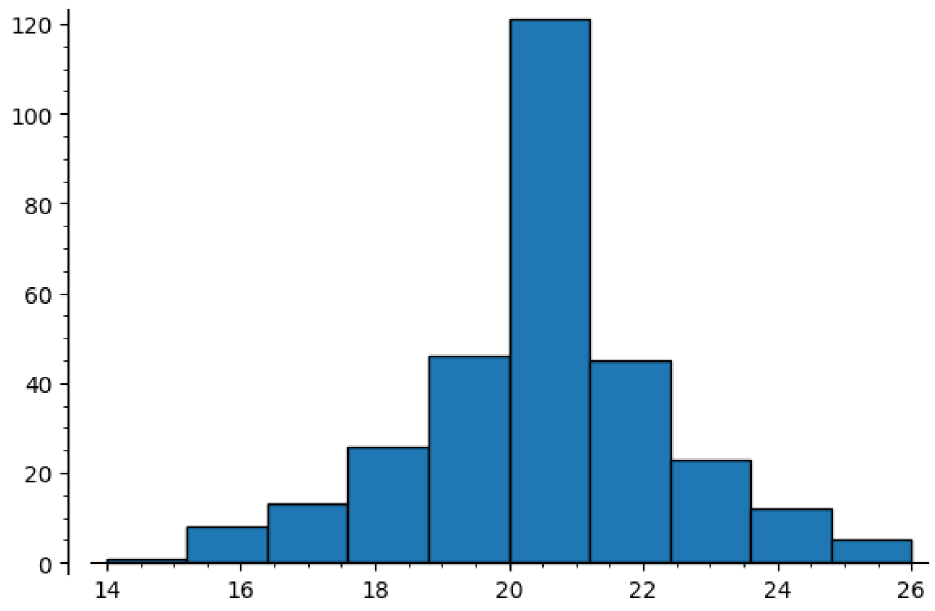

Figure 6.5: Histogram 300 trials

Looking at this histogram we can see the most common number of primes within the end shared key was somewhere between 20 and 22. This is fairly consistent with our previous findings that the end result on average would have approximately half the primes used to construct S. Also, the distribution of the results is very nice in the following sense. If Eve was interested in guessing by brute force the shared secret key she could not rely on checking only the the low and high areas because a majority of the results fell near the middle of the curve. Also as mentioned earlier guessing of the end secret key is most difficult for Eve when the end key is known to be made up of half the amount of primes in S.

Let us now consider the difficulty Eve would have if she attempted to brute force guess the shared secret key. If we made the simplifying assumption that all combinations of primes in S are equally as likely then the number of possible end shared secret keys Eve would have to try would be 2^i, where

$$S = p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_i.$$

Since we know *endA* and *endB* are a result of taking the gcd with S, we know that $endA = endB$ is made up of some combination of primes of S. We can interpret this as taking each prime in S and either turning it on or off. So,

$$endA = p_1^{e_1} \cdot p_2^{e_2} \cdot p_3^{e_3} \cdot ... \cdot p_i^{e_i} \text{ such that } e_i \in \{0, 1\}$$

Since there are i many primes in *endA* and each prime has two options for its exponent, this results in 2^i many possibilities. For large i this is impractical for eve to check all possibilities.

In summary, we ran experiments with large values for S and computed exact expected value for small parameters. Based off the results found we are sufficiently confident reasonable security is achieved when $k = 80\%$.

6.3 Implementation

Like any other cryptosytem, an important question is how fast the key exchange can be computed. Looking at Key Exchange 6.1, the most expensive step, as far a time goes, is the computation of the gcd. Within the exchange the gcd has to be computed three times by each Alice and Bob. The most efficient way to compute the gcd of two integers is the Euclidean Algorithm. So the CSDH time complexity is approximately $O(\log n)$. The speed of computation is beneficial for key generation and does not lower the security of the exchange. Since Eve knows that *endA* is a divisor of S but has no knowledge of $\bar{A}$ or $\bar{B}$ she has no reason to compute the gcd she instead can just look at divisors of S to find the shared secret key. By our analysis in Section 6.2, we concluded that a high level of security can be achieved when the k-factor was set to $k = 80\%$. This tended to produce shared secret keys built out $\frac{N}{2}$ primes, so we can pick parameters based on a the level of security we are trying to achieve.

6.4 Analogy between Cyclic Subgroup Graphs and Isogeny Graphs

The CSDH Key Exchange that we have constructed has its roots in strong analogies with the post-quantum SIDH Key Exchange. We continue here with another strong parallel that can be made between the two systems.

Consider the representation of the CSDH that is in cyclic groups Key Exchange 6.1. Following the steps within the algorithm Alice and Bob each make two stops along their way. First they stop at G_a and G_b respectively, and then they each reach their final destination $G_{endA} = G_{endB}$. Notice that by construction G_a, G_b, and $G_{endA} = G_{endB}$ are all subgroups of G. We can then use subgroup inclusion to represent possible pathways that Alice and Bob could have potentially made. Let us look back at our earlier Example 6.6 where $G = \mathbb{Z}_{2310}$. Figure 6.6 is the subgroup lattice.

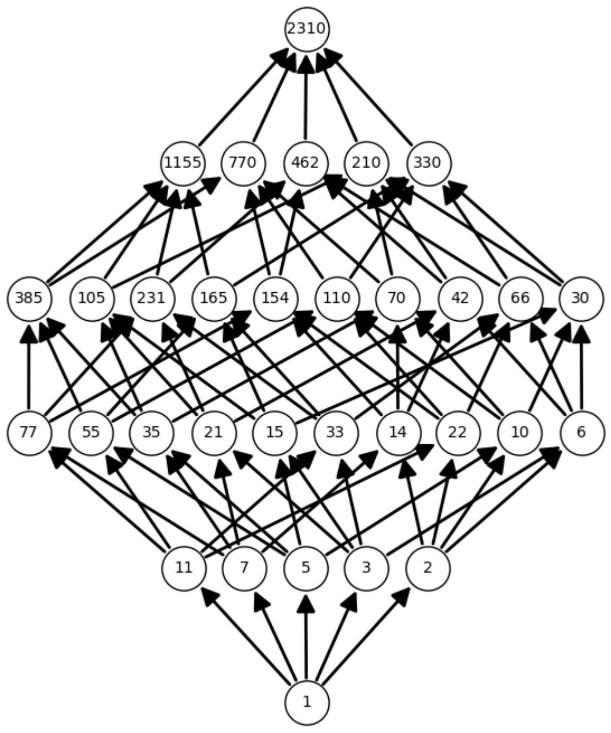

Figure 6.6: Subgroup Lattice for $\mathbb{Z}_{2310}$

The structure of the subgroup lattice is very nice. We can see that as you move from bottom to top each tier of the lattice is built of more primes. However, looking at the lattice we see that it is missing some edges. For example, in Example 6.6 Bob makes his first stop at 210, and Alice makes her first stop at 385. Looking at Figure 5.6 we see that there is an edge representing the path Bob has taken to get to $newB$, however the pathway Alice has taken is not represented, there is not an edge from 2310 to 385. In order to correct this we introduce the subgroup graph.

Definition 6.16. *The* ***Subgroup Graph*** *of a cyclic group G of order S is the graph whose vertices are all cyclic subgroups of G, labeled with their order. The edges of the graph being inclusion.*

Before returning to our previous example, let us look at a simple subgroup graph. Figure 5.7 shows the subgroup graph of $\mathbb{Z}_{30}$ whose cardinality is 30. The divisors, which make up all vertices, of 30 is the set $30, 15, 10, 6, 5, 3, 2, 1$. Using the Definition 6.16 edges should exist between any vertex that divides another. So the edge set should be $(1, 2), (1, 3), (1, 5), (1, 6), (1, 10), (1, 15), (1, 30), (2, 6), (2, 10), (2, 30), (3, 6), (3, 15), (3, 30),$ $(5, 10), (5, 15), (5, 30), (6, 30), (10, 30), (15, 30)$.

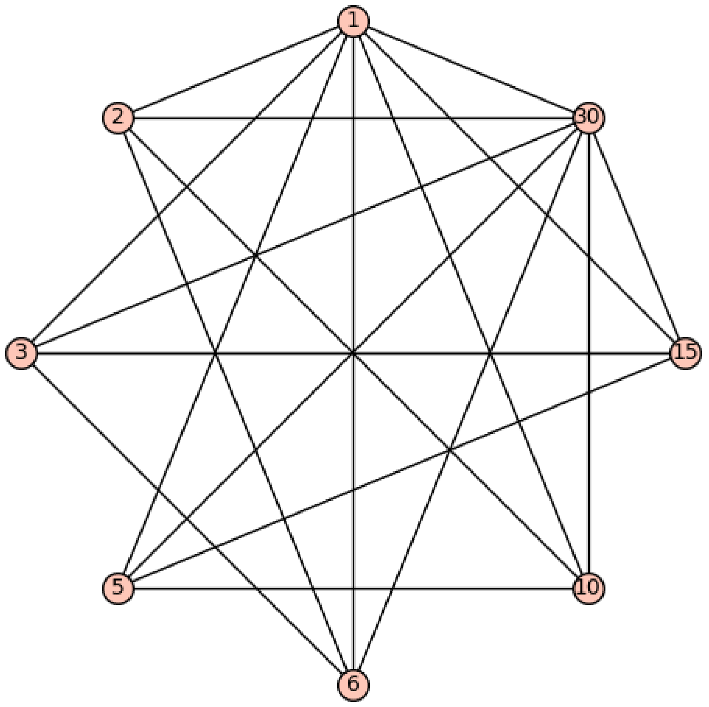

Figure 6.7: Subgroup Graph for $\mathbb{Z}_{30}$

We now look at a special case of the subgroup graph for groups of order prime power.

Example 6.17. *Let $G = \mathbb{Z}_{1024}$. The divisors of 1024 is the set $\{1, 2, 4, 8, 16, 32, 64, 128,$ $256, 512, 1024\}$. Notice that these are all of the form 2^k for k ranging from 1 to 10. This* *leads to all vertices having the same number of edges. So the subgroup graph for $\mathbb{Z}_{1024}$ is the complete graph on 11 vertices. Figure 6.8 shows this.*

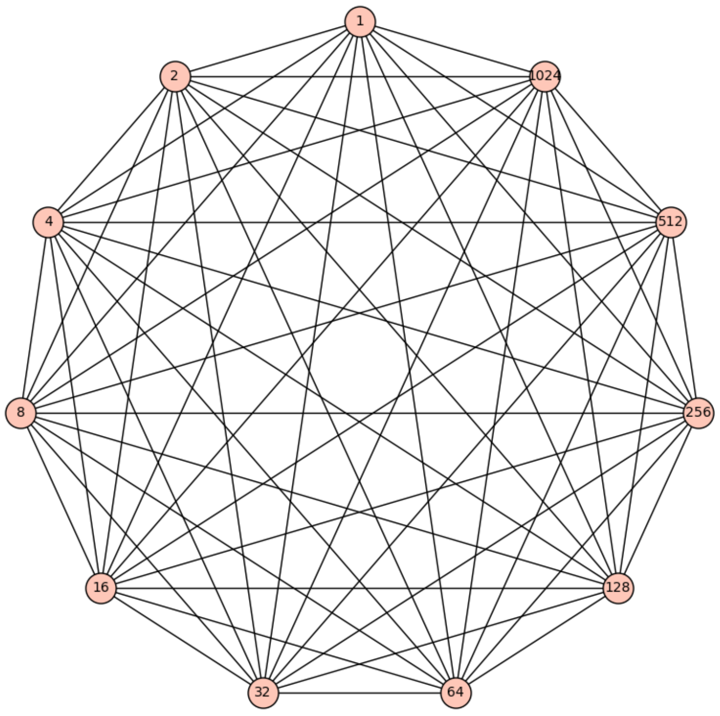

Figure 6.8: Subgroup Graph for $\mathbb{Z}_{1024}$

Proposition 6.18. *Let $G = \mathbb{Z}_{p^k}$ for some prime p and $k \in \mathbb{Z}$. The subgroup graph of G is isomorphic to the complete graph on $k+1$ vertices.*

We now turn back to our example using the product of the first 5 primes $S = 2310$. We can think of these vertices as all the possibilities Eve must try, so as mentioned earlier the number of vertices would be 2^N where N is the number of primes in S.

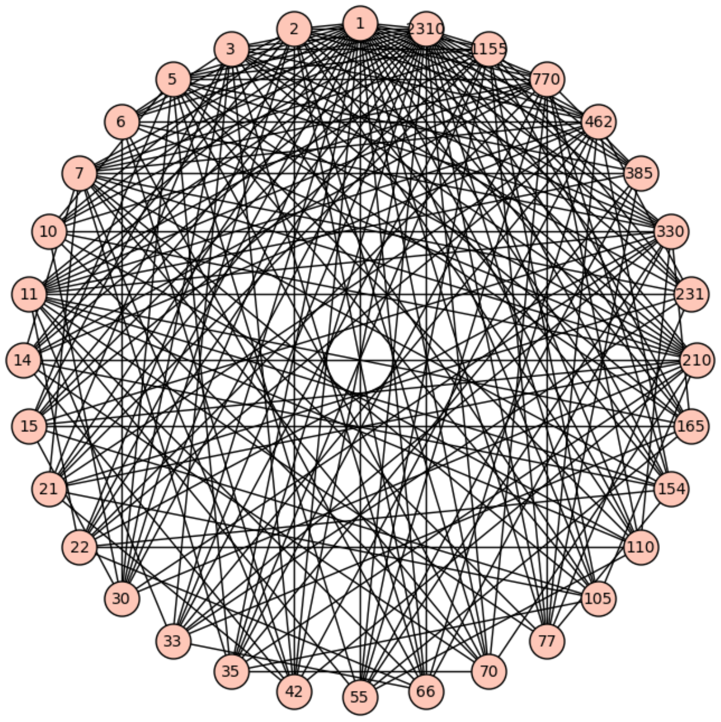

Figure 6.9: Subgroup Graph for $\mathbb{Z}_{2310}$

Looking at these graphs we can see that they are highly connected. The nature of these graphs reveals what makes this system secure, Alice and Bob have many potential pathways they can take throughout the exchange.

In summarry, the analogies between the CSDH and SIDH systems offer useful tools to help better understand the SIDH system. Further research needs to be done to better understand if CSDH is a viable cryptosystem in its own right.

Chapter 7

Conclusion

In this paper, we looked to investigate the use of elliptic curves in cryptography. We covered necessary definitions and theorems for the understanding of the use of elliptic curves in today's cryptosystems. We discussed hard problems associated with these cryptosystems, and covered some possible vulnerabilities associated to classical elliptic curve cryptography. As a means of better understanding the use of elliptic curves in the post-quantum world, we proposed our own cryptosystem, CSDH. We showed how our own system had strong parallels to the post-quantum system, SIDH. We then analyzed our algorithm and proved corresponding propositions. To further our research of CSDH, we enumerated many different cases and ran many large experiments on Sage. These calculations and trials lead us to make conjectures regarding expected outcomes of our algorithm.

Further research can be done to prove the expected number of primes in the end shared secret key of CSDH. Research can also be done to further analyze the relationship between N, the number of primes in S, and the k-factor. Research can be done to understand what size parameter selections offer sufficient security. Also, like any other cryptosystem, we must always ask what possible vulnerabilities are in CSDH that leaves it open to attacks? As we continue to advance technologically as a society, new and creative ways of protecting our information must be discovered.

Appendix A

Table A.1: All combinations of secret keys from Example 6.13

S	#	end	#	%	P_a	Q_a	P_b	Q_b	m_a	n_a	m_b	n_b
2310	5	210	4	0.8	2	3	4	5	210	210	210	210
2310	5	30	3	0.6	2	3	4	5	210	210	330	210
2310	5	210	4	0.8	2	3	4	5	210	210	462	210
2310	5	105	3	0.6	2	3	4	5	210	210	1155	210
2310	5	70	3	0.6	2	3	4	5	210	210	770	210
2310	5	30	3	0.6	2	3	4	5	210	210	210	330
2310	5	30	3	0.6	2	3	4	5	210	210	330	330
2310	5	30	3	0.6	2	3	4	5	210	210	462	330
2310	5	15	2	0.4	2	3	4	5	210	210	1155	330
2310	5	10	2	0.4	2	3	4	5	210	210	770	330
2310	5	42	3	0.6	2	3	4	5	210	210	210	462
2310	5	6	2	0.4	2	3	4	5	210	210	330	462
2310	5	42	3	0.6	2	3	4	5	210	210	462	462
2310	5	21	2	0.4	2	3	4	5	210	210	1155	462
2310	5	14	2	0.4	2	3	4	5	210	210	770	462
2310	5	210	4	0.8	2	3	4	5	210	210	210	1155
2310	5	30	3	0.6	2	3	4	5	210	210	330	1155
2310	5	210	4	0.8	2	3	4	5	210	210	462	1155
2310	5	105	3	0.6	2	3	4	5	210	210	1155	1155
2310	5	70	3	0.6	2	3	4	5	210	210	770	1155

2310	5	70	3	0.6	2	3	4	5	210	210	210	770
2310	5	10	2	0.4	2	3	4	5	210	210	330	770
2310	5	70	3	0.6	2	3	4	5	210	210	462	770
2310	5	35	2	0.4	2	3	4	5	210	210	1155	770
2310	5	210	4	0.8	2	3	4	5	210	210	770	770
2310	5	30	3	0.6	2	3	4	5	330	210	210	210
2310	5	30	3	0.6	2	3	4	5	330	210	330	210
2310	5	30	3	0.6	2	3	4	5	330	210	462	210
2310	5	15	2	0.4	2	3	4	5	330	210	1155	210
2310	5	10	2	0.4	2	3	4	5	330	210	770	210
2310	5	30	3	0.6	2	3	4	5	330	210	210	330
2310	5	30	3	0.6	2	3	4	5	330	210	330	330
2310	5	30	3	0.6	2	3	4	5	330	210	462	330
2310	5	15	2	0.4	2	3	4	5	330	210	1155	330
2310	5	10	2	0.4	2	3	4	5	330	210	770	330
2310	5	6	2	0.4	2	3	4	5	330	210	210	462
2310	5	6	2	0.4	2	3	4	5	330	210	330	462
2310	5	6	2	0.4	2	3	4	5	330	210	462	462
2310	5	3	1	0.2	2	3	4	5	330	210	1155	462
2310	5	2	1	0.2	2	3	4	5	330	210	770	462
2310	5	30	3	0.6	2	3	4	5	330	210	210	1155
2310	5	30	3	0.6	2	3	4	5	330	210	330	1155
2310	5	30	3	0.6	2	3	4	5	330	210	462	1155
2310	5	15	2	0.4	2	3	4	5	330	210	1155	1155
2310	5	10	2	0.4	2	3	4	5	330	210	770	1155
2310	5	10	2	0.4	2	3	4	5	330	210	210	770
2310	5	10	2	0.4	2	3	4	5	330	210	330	770
2310	5	10	2	0.4	2	3	4	5	330	210	462	770
2310	5	5	1	0.2	2	3	4	5	330	210	1155	770
2310	5	30	3	0.6	2	3	4	5	330	210	770	770
2310	5	42	3	0.6	2	3	4	5	462	210	210	210
2310	5	6	2	0.4	2	3	4	5	462	210	330	210

2310	5	42	3	0.6	2	3	4	5	462	210	462	210
2310	5	21	2	0.4	2	3	4	5	462	210	1155	210
2310	5	14	2	0.4	2	3	4	5	462	210	770	210
2310	5	6	2	0.4	2	3	4	5	462	210	210	330
2310	5	6	2	0.4	2	3	4	5	462	210	330	330
2310	5	6	2	0.4	2	3	4	5	462	210	462	330
2310	5	3	1	0.2	2	3	4	5	462	210	1155	330
2310	5	2	1	0.2	2	3	4	5	462	210	770	330
2310	5	42	3	0.6	2	3	4	5	462	210	210	462
2310	5	6	2	0.4	2	3	4	5	462	210	330	462
2310	5	42	3	0.6	2	3	4	5	462	210	462	462
2310	5	21	2	0.4	2	3	4	5	462	210	1155	462
2310	5	14	2	0.4	2	3	4	5	462	210	770	462
2310	5	42	3	0.6	2	3	4	5	462	210	210	1155
2310	5	6	2	0.4	2	3	4	5	462	210	330	1155
2310	5	42	3	0.6	2	3	4	5	462	210	462	1155
2310	5	21	2	0.4	2	3	4	5	462	210	1155	1155
2310	5	14	2	0.4	2	3	4	5	462	210	770	1155
2310	5	14	2	0.4	2	3	4	5	462	210	210	770
2310	5	2	1	0.2	2	3	4	5	462	210	330	770
2310	5	14	2	0.4	2	3	4	5	462	210	462	770
2310	5	7	1	0.2	2	3	4	5	462	210	1155	770
2310	5	42	3	0.6	2	3	4	5	462	210	770	770
2310	5	105	3	0.6	2	3	4	5	1155	210	210	210
2310	5	15	2	0.4	2	3	4	5	1155	210	330	210
2310	5	105	3	0.6	2	3	4	5	1155	210	462	210
2310	5	105	3	0.6	2	3	4	5	1155	210	1155	210
2310	5	35	2	0.4	2	3	4	5	1155	210	770	210
2310	5	15	2	0.4	2	3	4	5	1155	210	210	330
2310	5	15	2	0.4	2	3	4	5	1155	210	330	330
2310	5	15	2	0.4	2	3	4	5	1155	210	462	330
2310	5	15	2	0.4	2	3	4	5	1155	210	1155	330

2310	5	5	1	0.2	2	3	4	5	1155	210	770	330
2310	5	21	2	0.4	2	3	4	5	1155	210	210	462
2310	5	3	1	0.2	2	3	4	5	1155	210	330	462
2310	5	21	2	0.4	2	3	4	5	1155	210	462	462
2310	5	21	2	0.4	2	3	4	5	1155	210	1155	462
2310	5	7	1	0.2	2	3	4	5	1155	210	770	462
2310	5	105	3	0.6	2	3	4	5	1155	210	210	1155
2310	5	15	2	0.4	2	3	4	5	1155	210	330	1155
2310	5	105	3	0.6	2	3	4	5	1155	210	462	1155
2310	5	105	3	0.6	2	3	4	5	1155	210	1155	1155
2310	5	35	2	0.4	2	3	4	5	1155	210	770	1155
2310	5	35	2	0.4	2	3	4	5	1155	210	210	770
2310	5	5	1	0.2	2	3	4	5	1155	210	330	770
2310	5	35	2	0.4	2	3	4	5	1155	210	462	770
2310	5	35	2	0.4	2	3	4	5	1155	210	1155	770
2310	5	105	3	0.6	2	3	4	5	1155	210	770	770
2310	5	210	4	0.8	2	3	4	5	770	210	210	210
2310	5	30	3	0.6	2	3	4	5	770	210	330	210
2310	5	210	4	0.8	2	3	4	5	770	210	462	210
2310	5	105	3	0.6	2	3	4	5	770	210	1155	210
2310	5	70	3	0.6	2	3	4	5	770	210	770	210
2310	5	30	3	0.6	2	3	4	5	770	210	210	330
2310	5	30	3	0.6	2	3	4	5	770	210	330	330
2310	5	30	3	0.6	2	3	4	5	770	210	462	330
2310	5	15	2	0.4	2	3	4	5	770	210	1155	330
2310	5	10	2	0.4	2	3	4	5	770	210	770	330
2310	5	42	3	0.6	2	3	4	5	770	210	210	462
2310	5	6	2	0.4	2	3	4	5	770	210	330	462
2310	5	42	3	0.6	2	3	4	5	770	210	462	462
2310	5	21	2	0.4	2	3	4	5	770	210	1155	462
2310	5	14	2	0.4	2	3	4	5	770	210	770	462
2310	5	210	4	0.8	2	3	4	5	770	210	210	1155

2310	5	30	3	0.6	2	3	4	5	770	210	330	1155
2310	5	210	4	0.8	2	3	4	5	770	210	462	1155
2310	5	105	3	0.6	2	3	4	5	770	210	1155	1155
2310	5	70	3	0.6	2	3	4	5	770	210	770	1155
2310	5	70	3	0.6	2	3	4	5	770	210	210	770
2310	5	10	2	0.4	2	3	4	5	770	210	330	770
2310	5	70	3	0.6	2	3	4	5	770	210	462	770
2310	5	35	2	0.4	2	3	4	5	770	210	1155	770
2310	5	210	4	0.8	2	3	4	5	770	210	770	770
2310	5	30	3	0.6	2	3	4	5	210	330	210	210
2310	5	30	3	0.6	2	3	4	5	210	330	330	210
2310	5	30	3	0.6	2	3	4	5	210	330	462	210
2310	5	15	2	0.4	2	3	4	5	210	330	1155	210
2310	5	10	2	0.4	2	3	4	5	210	330	770	210
2310	5	30	3	0.6	2	3	4	5	210	330	210	330
2310	5	30	3	0.6	2	3	4	5	210	330	330	330
2310	5	30	3	0.6	2	3	4	5	210	330	462	330
2310	5	15	2	0.4	2	3	4	5	210	330	1155	330
2310	5	10	2	0.4	2	3	4	5	210	330	770	330
2310	5	6	2	0.4	2	3	4	5	210	330	210	462
2310	5	6	2	0.4	2	3	4	5	210	330	330	462
2310	5	6	2	0.4	2	3	4	5	210	330	462	462
2310	5	3	1	0.2	2	3	4	5	210	330	1155	462
2310	5	2	1	0.2	2	3	4	5	210	330	770	462
2310	5	30	3	0.6	2	3	4	5	210	330	210	1155
2310	5	30	3	0.6	2	3	4	5	210	330	330	1155
2310	5	30	3	0.6	2	3	4	5	210	330	462	1155
2310	5	15	2	0.4	2	3	4	5	210	330	1155	1155
2310	5	10	2	0.4	2	3	4	5	210	330	770	1155
2310	5	10	2	0.4	2	3	4	5	210	330	210	770
2310	5	10	2	0.4	2	3	4	5	210	330	330	770
2310	5	10	2	0.4	2	3	4	5	210	330	462	770

2310	5	5	1	0.2	2	3	4	5	210	330	1155	770
2310	5	30	3	0.6	2	3	4	5	210	330	770	770
2310	5	30	3	0.6	2	3	4	5	330	330	210	210
2310	5	30	3	0.6	2	3	4	5	330	330	330	210
2310	5	30	3	0.6	2	3	4	5	330	330	462	210
2310	5	15	2	0.4	2	3	4	5	330	330	1155	210
2310	5	10	2	0.4	2	3	4	5	330	330	770	210
2310	5	30	3	0.6	2	3	4	5	330	330	210	330
2310	5	330	4	0.8	2	3	4	5	330	330	330	330
2310	5	330	4	0.8	2	3	4	5	330	330	462	330
2310	5	165	3	0.6	2	3	4	5	330	330	1155	330
2310	5	110	3	0.6	2	3	4	5	330	330	770	330
2310	5	6	2	0.4	2	3	4	5	330	330	210	462
2310	5	66	3	0.6	2	3	4	5	330	330	330	462
2310	5	66	3	0.6	2	3	4	5	330	330	462	462
2310	5	33	2	0.4	2	3	4	5	330	330	1155	462
2310	5	22	2	0.4	2	3	4	5	330	330	770	462
2310	5	30	3	0.6	2	3	4	5	330	330	210	1155
2310	5	330	4	0.8	2	3	4	5	330	330	330	1155
2310	5	330	4	0.8	2	3	4	5	330	330	462	1155
2310	5	165	3	0.6	2	3	4	5	330	330	1155	1155
2310	5	110	3	0.6	2	3	4	5	330	330	770	1155
2310	5	10	2	0.4	2	3	4	5	330	330	210	770
2310	5	110	3	0.6	2	3	4	5	330	330	330	770
2310	5	110	3	0.6	2	3	4	5	330	330	462	770
2310	5	55	2	0.4	2	3	4	5	330	330	1155	770
2310	5	330	4	0.8	2	3	4	5	330	330	770	770
2310	5	6	2	0.4	2	3	4	5	462	330	210	210
2310	5	6	2	0.4	2	3	4	5	462	330	330	210
2310	5	6	2	0.4	2	3	4	5	462	330	462	210
2310	5	3	1	0.2	2	3	4	5	462	330	1155	210
2310	5	2	1	0.2	2	3	4	5	462	330	770	210

2310	5	6	2	0.4	2	3	4	5	462	330	210	330
2310	5	66	3	0.6	2	3	4	5	462	330	330	330
2310	5	66	3	0.6	2	3	4	5	462	330	462	330
2310	5	33	2	0.4	2	3	4	5	462	330	1155	330
2310	5	22	2	0.4	2	3	4	5	462	330	770	330
2310	5	6	2	0.4	2	3	4	5	462	330	210	462
2310	5	66	3	0.6	2	3	4	5	462	330	330	462
2310	5	66	3	0.6	2	3	4	5	462	330	462	462
2310	5	33	2	0.4	2	3	4	5	462	330	1155	462
2310	5	22	2	0.4	2	3	4	5	462	330	770	462
2310	5	6	2	0.4	2	3	4	5	462	330	210	1155
2310	5	66	3	0.6	2	3	4	5	462	330	330	1155
2310	5	66	3	0.6	2	3	4	5	462	330	462	1155
2310	5	33	2	0.4	2	3	4	5	462	330	1155	1155
2310	5	22	2	0.4	2	3	4	5	462	330	770	1155
2310	5	2	1	0.2	2	3	4	5	462	330	210	770
2310	5	22	2	0.4	2	3	4	5	462	330	330	770
2310	5	22	2	0.4	2	3	4	5	462	330	462	770
2310	5	11	1	0.2	2	3	4	5	462	330	1155	770
2310	5	66	3	0.6	2	3	4	5	462	330	770	770
2310	5	15	2	0.4	2	3	4	5	1155	330	210	210
2310	5	15	2	0.4	2	3	4	5	1155	330	330	210
2310	5	15	2	0.4	2	3	4	5	1155	330	462	210
2310	5	15	2	0.4	2	3	4	5	1155	330	1155	210
2310	5	5	1	0.2	2	3	4	5	1155	330	770	210
2310	5	15	2	0.4	2	3	4	5	1155	330	210	330
2310	5	165	3	0.6	2	3	4	5	1155	330	330	330
2310	5	165	3	0.6	2	3	4	5	1155	330	462	330
2310	5	165	3	0.6	2	3	4	5	1155	330	1155	330
2310	5	55	2	0.4	2	3	4	5	1155	330	770	330
2310	5	3	1	0.2	2	3	4	5	1155	330	210	462
2310	5	33	2	0.4	2	3	4	5	1155	330	330	462

2310	5	33	2	0.4	2	3	4	5	1155	330	462	462
2310	5	33	2	0.4	2	3	4	5	1155	330	1155	462
2310	5	11	1	0.2	2	3	4	5	1155	330	770	462
2310	5	15	2	0.4	2	3	4	5	1155	330	210	1155
2310	5	165	3	0.6	2	3	4	5	1155	330	330	1155
2310	5	165	3	0.6	2	3	4	5	1155	330	462	1155
2310	5	165	3	0.6	2	3	4	5	1155	330	1155	1155
2310	5	55	2	0.4	2	3	4	5	1155	330	770	1155
2310	5	5	1	0.2	2	3	4	5	1155	330	210	770
2310	5	55	2	0.4	2	3	4	5	1155	330	330	770
2310	5	55	2	0.4	2	3	4	5	1155	330	462	770
2310	5	55	2	0.4	2	3	4	5	1155	330	1155	770
2310	5	165	3	0.6	2	3	4	5	1155	330	770	770
2310	5	30	3	0.6	2	3	4	5	770	330	210	210
2310	5	30	3	0.6	2	3	4	5	770	330	330	210
2310	5	30	3	0.6	2	3	4	5	770	330	462	210
2310	5	15	2	0.4	2	3	4	5	770	330	1155	210
2310	5	10	2	0.4	2	3	4	5	770	330	770	210
2310	5	30	3	0.6	2	3	4	5	770	330	210	330
2310	5	330	4	0.8	2	3	4	5	770	330	330	330
2310	5	330	4	0.8	2	3	4	5	770	330	462	330
2310	5	165	3	0.6	2	3	4	5	770	330	1155	330
2310	5	110	3	0.6	2	3	4	5	770	330	770	330
2310	5	6	2	0.4	2	3	4	5	770	330	210	462
2310	5	66	3	0.6	2	3	4	5	770	330	330	462
2310	5	66	3	0.6	2	3	4	5	770	330	462	462
2310	5	33	2	0.4	2	3	4	5	770	330	1155	462
2310	5	22	2	0.4	2	3	4	5	770	330	770	462
2310	5	30	3	0.6	2	3	4	5	770	330	210	1155
2310	5	330	4	0.8	2	3	4	5	770	330	330	1155
2310	5	330	4	0.8	2	3	4	5	770	330	462	1155
2310	5	165	3	0.6	2	3	4	5	770	330	1155	1155

2310	5	110	3	0.6	2	3	4	5	770	330	770	1155
2310	5	10	2	0.4	2	3	4	5	770	330	210	770
2310	5	110	3	0.6	2	3	4	5	770	330	330	770
2310	5	110	3	0.6	2	3	4	5	770	330	462	770
2310	5	55	2	0.4	2	3	4	5	770	330	1155	770
2310	5	330	4	0.8	2	3	4	5	770	330	770	770
2310	5	42	3	0.6	2	3	4	5	210	462	210	210
2310	5	6	2	0.4	2	3	4	5	210	462	330	210
2310	5	42	3	0.6	2	3	4	5	210	462	462	210
2310	5	21	2	0.4	2	3	4	5	210	462	1155	210
2310	5	14	2	0.4	2	3	4	5	210	462	770	210
2310	5	6	2	0.4	2	3	4	5	210	462	210	330
2310	5	6	2	0.4	2	3	4	5	210	462	330	330
2310	5	6	2	0.4	2	3	4	5	210	462	462	330
2310	5	3	1	0.2	2	3	4	5	210	462	1155	330
2310	5	2	1	0.2	2	3	4	5	210	462	770	330
2310	5	42	3	0.6	2	3	4	5	210	462	210	462
2310	5	6	2	0.4	2	3	4	5	210	462	330	462
2310	5	42	3	0.6	2	3	4	5	210	462	462	462
2310	5	21	2	0.4	2	3	4	5	210	462	1155	462
2310	5	14	2	0.4	2	3	4	5	210	462	770	462
2310	5	42	3	0.6	2	3	4	5	210	462	210	1155
2310	5	6	2	0.4	2	3	4	5	210	462	330	1155
2310	5	42	3	0.6	2	3	4	5	210	462	462	1155
2310	5	21	2	0.4	2	3	4	5	210	462	1155	1155
2310	5	14	2	0.4	2	3	4	5	210	462	770	1155
2310	5	14	2	0.4	2	3	4	5	210	462	210	770
2310	5	2	1	0.2	2	3	4	5	210	462	330	770
2310	5	14	2	0.4	2	3	4	5	210	462	462	770
2310	5	7	1	0.2	2	3	4	5	210	462	1155	770
2310	5	42	3	0.6	2	3	4	5	210	462	770	770
2310	5	6	2	0.4	2	3	4	5	330	462	210	210

2310	5	6	2	0.4	2	3	4	5	330	462	330	210
2310	5	6	2	0.4	2	3	4	5	330	462	462	210
2310	5	3	1	0.2	2	3	4	5	330	462	1155	210
2310	5	2	1	0.2	2	3	4	5	330	462	770	210
2310	5	6	2	0.4	2	3	4	5	330	462	210	330
2310	5	66	3	0.6	2	3	4	5	330	462	330	330
2310	5	66	3	0.6	2	3	4	5	330	462	462	330
2310	5	33	2	0.4	2	3	4	5	330	462	1155	330
2310	5	22	2	0.4	2	3	4	5	330	462	770	330
2310	5	6	2	0.4	2	3	4	5	330	462	210	462
2310	5	66	3	0.6	2	3	4	5	330	462	330	462
2310	5	66	3	0.6	2	3	4	5	330	462	462	462
2310	5	33	2	0.4	2	3	4	5	330	462	1155	462
2310	5	22	2	0.4	2	3	4	5	330	462	770	462
2310	5	6	2	0.4	2	3	4	5	330	462	210	1155
2310	5	66	3	0.6	2	3	4	5	330	462	330	1155
2310	5	66	3	0.6	2	3	4	5	330	462	462	1155
2310	5	33	2	0.4	2	3	4	5	330	462	1155	1155
2310	5	22	2	0.4	2	3	4	5	330	462	770	1155
2310	5	2	1	0.2	2	3	4	5	330	462	210	770
2310	5	22	2	0.4	2	3	4	5	330	462	330	770
2310	5	22	2	0.4	2	3	4	5	330	462	462	770
2310	5	11	1	0.2	2	3	4	5	330	462	1155	770
2310	5	66	3	0.6	2	3	4	5	330	462	770	770
2310	5	210	4	0.8	2	3	4	5	462	462	210	210
2310	5	30	3	0.6	2	3	4	5	462	462	330	210
2310	5	210	4	0.8	2	3	4	5	462	462	462	210
2310	5	105	3	0.6	2	3	4	5	462	462	1155	210
2310	5	70	3	0.6	2	3	4	5	462	462	770	210
2310	5	30	3	0.6	2	3	4	5	462	462	210	330
2310	5	330	4	0.8	2	3	4	5	462	462	330	330
2310	5	330	4	0.8	2	3	4	5	462	462	462	330

2310	5	165	3	0.6	2	3	4	5	462	462	1155	330
2310	5	110	3	0.6	2	3	4	5	462	462	770	330
2310	5	42	3	0.6	2	3	4	5	462	462	210	462
2310	5	66	3	0.6	2	3	4	5	462	462	330	462
2310	5	462	4	0.8	2	3	4	5	462	462	462	462
2310	5	231	3	0.6	2	3	4	5	462	462	1155	462
2310	5	154	3	0.6	2	3	4	5	462	462	770	462
2310	5	210	4	0.8	2	3	4	5	462	462	210	1155
2310	5	330	4	0.8	2	3	4	5	462	462	330	1155
2310	5	2310	5	1.0	2	3	4	5	462	462	462	1155
2310	5	1155	4	0.8	2	3	4	5	462	462	1155	1155
2310	5	770	4	0.8	2	3	4	5	462	462	770	1155
2310	5	70	3	0.6	2	3	4	5	462	462	210	770
2310	5	110	3	0.6	2	3	4	5	462	462	330	770
2310	5	770	4	0.8	2	3	4	5	462	462	462	770
2310	5	385	3	0.6	2	3	4	5	462	462	1155	770
2310	5	2310	5	1.0	2	3	4	5	462	462	770	770
2310	5	21	2	0.4	2	3	4	5	1155	462	210	210
2310	5	3	1	0.2	2	3	4	5	1155	462	330	210
2310	5	21	2	0.4	2	3	4	5	1155	462	462	210
2310	5	21	2	0.4	2	3	4	5	1155	462	1155	210
2310	5	7	1	0.2	2	3	4	5	1155	462	770	210
2310	5	3	1	0.2	2	3	4	5	1155	462	210	330
2310	5	33	2	0.4	2	3	4	5	1155	462	330	330
2310	5	33	2	0.4	2	3	4	5	1155	462	462	330
2310	5	33	2	0.4	2	3	4	5	1155	462	1155	330
2310	5	11	1	0.2	2	3	4	5	1155	462	770	330
2310	5	21	2	0.4	2	3	4	5	1155	462	210	462
2310	5	33	2	0.4	2	3	4	5	1155	462	330	462
2310	5	231	3	0.6	2	3	4	5	1155	462	462	462
2310	5	231	3	0.6	2	3	4	5	1155	462	1155	462
2310	5	77	2	0.4	2	3	4	5	1155	462	770	462

2310	5	21	2	0.4	2	3	4	5	1155	462	210	1155
2310	5	33	2	0.4	2	3	4	5	1155	462	330	1155
2310	5	231	3	0.6	2	3	4	5	1155	462	462	1155
2310	5	231	3	0.6	2	3	4	5	1155	462	1155	1155
2310	5	77	2	0.4	2	3	4	5	1155	462	770	1155
2310	5	7	1	0.2	2	3	4	5	1155	462	210	770
2310	5	11	1	0.2	2	3	4	5	1155	462	330	770
2310	5	77	2	0.4	2	3	4	5	1155	462	462	770
2310	5	77	2	0.4	2	3	4	5	1155	462	1155	770
2310	5	231	3	0.6	2	3	4	5	1155	462	770	770
2310	5	42	3	0.6	2	3	4	5	770	462	210	210
2310	5	6	2	0.4	2	3	4	5	770	462	330	210
2310	5	42	3	0.6	2	3	4	5	770	462	462	210
2310	5	21	2	0.4	2	3	4	5	770	462	1155	210
2310	5	14	2	0.4	2	3	4	5	770	462	770	210
2310	5	6	2	0.4	2	3	4	5	770	462	210	330
2310	5	66	3	0.6	2	3	4	5	770	462	330	330
2310	5	66	3	0.6	2	3	4	5	770	462	462	330
2310	5	33	2	0.4	2	3	4	5	770	462	1155	330
2310	5	22	2	0.4	2	3	4	5	770	462	770	330
2310	5	42	3	0.6	2	3	4	5	770	462	210	462
2310	5	66	3	0.6	2	3	4	5	770	462	330	462
2310	5	462	4	0.8	2	3	4	5	770	462	462	462
2310	5	231	3	0.6	2	3	4	5	770	462	1155	462
2310	5	154	3	0.6	2	3	4	5	770	462	770	462
2310	5	42	3	0.6	2	3	4	5	770	462	210	1155
2310	5	66	3	0.6	2	3	4	5	770	462	330	1155
2310	5	462	4	0.8	2	3	4	5	770	462	462	1155
2310	5	231	3	0.6	2	3	4	5	770	462	1155	1155
2310	5	154	3	0.6	2	3	4	5	770	462	770	1155
2310	5	14	2	0.4	2	3	4	5	770	462	210	770
2310	5	22	2	0.4	2	3	4	5	770	462	330	770

2310	5	154	3	0.6	2	3	4	5	770	462	462	770
2310	5	77	2	0.4	2	3	4	5	770	462	1155	770
2310	5	462	4	0.8	2	3	4	5	770	462	770	770
2310	5	210	4	0.8	2	3	4	5	210	1155	210	210
2310	5	30	3	0.6	2	3	4	5	210	1155	330	210
2310	5	210	4	0.8	2	3	4	5	210	1155	462	210
2310	5	105	3	0.6	2	3	4	5	210	1155	1155	210
2310	5	70	3	0.6	2	3	4	5	210	1155	770	210
2310	5	30	3	0.6	2	3	4	5	210	1155	210	330
2310	5	30	3	0.6	2	3	4	5	210	1155	330	330
2310	5	30	3	0.6	2	3	4	5	210	1155	462	330
2310	5	15	2	0.4	2	3	4	5	210	1155	1155	330
2310	5	10	2	0.4	2	3	4	5	210	1155	770	330
2310	5	42	3	0.6	2	3	4	5	210	1155	210	462
2310	5	6	2	0.4	2	3	4	5	210	1155	330	462
2310	5	42	3	0.6	2	3	4	5	210	1155	462	462
2310	5	21	2	0.4	2	3	4	5	210	1155	1155	462
2310	5	14	2	0.4	2	3	4	5	210	1155	770	462
2310	5	210	4	0.8	2	3	4	5	210	1155	210	1155
2310	5	30	3	0.6	2	3	4	5	210	1155	330	1155
2310	5	210	4	0.8	2	3	4	5	210	1155	462	1155
2310	5	105	3	0.6	2	3	4	5	210	1155	1155	1155
2310	5	70	3	0.6	2	3	4	5	210	1155	770	1155
2310	5	70	3	0.6	2	3	4	5	210	1155	210	770
2310	5	10	2	0.4	2	3	4	5	210	1155	330	770
2310	5	70	3	0.6	2	3	4	5	210	1155	462	770
2310	5	35	2	0.4	2	3	4	5	210	1155	1155	770
2310	5	210	4	0.8	2	3	4	5	210	1155	770	770
2310	5	30	3	0.6	2	3	4	5	330	1155	210	210
2310	5	30	3	0.6	2	3	4	5	330	1155	330	210
2310	5	30	3	0.6	2	3	4	5	330	1155	462	210
2310	5	15	2	0.4	2	3	4	5	330	1155	1155	210

2310	5	10	2	0.4	2	3	4	5	330	1155	770	210
2310	5	30	3	0.6	2	3	4	5	330	1155	210	330
2310	5	330	4	0.8	2	3	4	5	330	1155	330	330
2310	5	330	4	0.8	2	3	4	5	330	1155	462	330
2310	5	165	3	0.6	2	3	4	5	330	1155	1155	330
2310	5	110	3	0.6	2	3	4	5	330	1155	770	330
2310	5	6	2	0.4	2	3	4	5	330	1155	210	462
2310	5	66	3	0.6	2	3	4	5	330	1155	330	462
2310	5	66	3	0.6	2	3	4	5	330	1155	462	462
2310	5	33	2	0.4	2	3	4	5	330	1155	1155	462
2310	5	22	2	0.4	2	3	4	5	330	1155	770	462
2310	5	30	3	0.6	2	3	4	5	330	1155	210	1155
2310	5	330	4	0.8	2	3	4	5	330	1155	330	1155
2310	5	330	4	0.8	2	3	4	5	330	1155	462	1155
2310	5	165	3	0.6	2	3	4	5	330	1155	1155	1155
2310	5	110	3	0.6	2	3	4	5	330	1155	770	1155
2310	5	10	2	0.4	2	3	4	5	330	1155	210	770
2310	5	110	3	0.6	2	3	4	5	330	1155	330	770
2310	5	110	3	0.6	2	3	4	5	330	1155	462	770
2310	5	55	2	0.4	2	3	4	5	330	1155	1155	770
2310	5	330	4	0.8	2	3	4	5	330	1155	770	770
2310	5	42	3	0.6	2	3	4	5	462	1155	210	210
2310	5	6	2	0.4	2	3	4	5	462	1155	330	210
2310	5	42	3	0.6	2	3	4	5	462	1155	462	210
2310	5	21	2	0.4	2	3	4	5	462	1155	1155	210
2310	5	14	2	0.4	2	3	4	5	462	1155	770	210
2310	5	6	2	0.4	2	3	4	5	462	1155	210	330
2310	5	66	3	0.6	2	3	4	5	462	1155	330	330
2310	5	66	3	0.6	2	3	4	5	462	1155	462	330
2310	5	33	2	0.4	2	3	4	5	462	1155	1155	330
2310	5	22	2	0.4	2	3	4	5	462	1155	770	330
2310	5	42	3	0.6	2	3	4	5	462	1155	210	462

2310	5	66	3	0.6	2	3	4	5	462	1155	330	462
2310	5	462	4	0.8	2	3	4	5	462	1155	462	462
2310	5	231	3	0.6	2	3	4	5	462	1155	1155	462
2310	5	154	3	0.6	2	3	4	5	462	1155	770	462
2310	5	42	3	0.6	2	3	4	5	462	1155	210	1155
2310	5	66	3	0.6	2	3	4	5	462	1155	330	1155
2310	5	462	4	0.8	2	3	4	5	462	1155	462	1155
2310	5	231	3	0.6	2	3	4	5	462	1155	1155	1155
2310	5	154	3	0.6	2	3	4	5	462	1155	770	1155
2310	5	14	2	0.4	2	3	4	5	462	1155	210	770
2310	5	22	2	0.4	2	3	4	5	462	1155	330	770
2310	5	154	3	0.6	2	3	4	5	462	1155	462	770
2310	5	77	2	0.4	2	3	4	5	462	1155	1155	770
2310	5	462	4	0.8	2	3	4	5	462	1155	770	770
2310	5	105	3	0.6	2	3	4	5	1155	1155	210	210
2310	5	15	2	0.4	2	3	4	5	1155	1155	330	210
2310	5	105	3	0.6	2	3	4	5	1155	1155	462	210
2310	5	105	3	0.6	2	3	4	5	1155	1155	1155	210
2310	5	35	2	0.4	2	3	4	5	1155	1155	770	210
2310	5	15	2	0.4	2	3	4	5	1155	1155	210	330
2310	5	165	3	0.6	2	3	4	5	1155	1155	330	330
2310	5	165	3	0.6	2	3	4	5	1155	1155	462	330
2310	5	165	3	0.6	2	3	4	5	1155	1155	1155	330
2310	5	55	2	0.4	2	3	4	5	1155	1155	770	330
2310	5	21	2	0.4	2	3	4	5	1155	1155	210	462
2310	5	33	2	0.4	2	3	4	5	1155	1155	330	462
2310	5	231	3	0.6	2	3	4	5	1155	1155	462	462
2310	5	231	3	0.6	2	3	4	5	1155	1155	1155	462
2310	5	77	2	0.4	2	3	4	5	1155	1155	770	462
2310	5	105	3	0.6	2	3	4	5	1155	1155	210	1155
2310	5	165	3	0.6	2	3	4	5	1155	1155	330	1155
2310	5	1155	4	0.8	2	3	4	5	1155	1155	462	1155

2310	5	1155	4	0.8	2	3	4	5	1155	1155	1155	1155
2310	5	385	3	0.6	2	3	4	5	1155	1155	770	1155
2310	5	35	2	0.4	2	3	4	5	1155	1155	210	770
2310	5	55	2	0.4	2	3	4	5	1155	1155	330	770
2310	5	385	3	0.6	2	3	4	5	1155	1155	462	770
2310	5	385	3	0.6	2	3	4	5	1155	1155	1155	770
2310	5	1155	4	0.8	2	3	4	5	1155	1155	770	770
2310	5	210	4	0.8	2	3	4	5	770	1155	210	210
2310	5	30	3	0.6	2	3	4	5	770	1155	330	210
2310	5	210	4	0.8	2	3	4	5	770	1155	462	210
2310	5	105	3	0.6	2	3	4	5	770	1155	1155	210
2310	5	70	3	0.6	2	3	4	5	770	1155	770	210
2310	5	30	3	0.6	2	3	4	5	770	1155	210	330
2310	5	330	4	0.8	2	3	4	5	770	1155	330	330
2310	5	330	4	0.8	2	3	4	5	770	1155	462	330
2310	5	165	3	0.6	2	3	4	5	770	1155	1155	330
2310	5	110	3	0.6	2	3	4	5	770	1155	770	330
2310	5	42	3	0.6	2	3	4	5	770	1155	210	462
2310	5	66	3	0.6	2	3	4	5	770	1155	330	462
2310	5	462	4	0.8	2	3	4	5	770	1155	462	462
2310	5	231	3	0.6	2	3	4	5	770	1155	1155	462
2310	5	154	3	0.6	2	3	4	5	770	1155	770	462
2310	5	210	4	0.8	2	3	4	5	770	1155	210	1155
2310	5	330	4	0.8	2	3	4	5	770	1155	330	1155
2310	5	2310	5	1.0	2	3	4	5	770	1155	462	1155
2310	5	1155	4	0.8	2	3	4	5	770	1155	1155	1155
2310	5	770	4	0.8	2	3	4	5	770	1155	770	1155
2310	5	70	3	0.6	2	3	4	5	770	1155	210	770
2310	5	110	3	0.6	2	3	4	5	770	1155	330	770
2310	5	770	4	0.8	2	3	4	5	770	1155	462	770
2310	5	385	3	0.6	2	3	4	5	770	1155	1155	770
2310	5	2310	5	1.0	2	3	4	5	770	1155	770	770

2310	5	70	3	0.6	2	3	4	5	210	770	210	210
2310	5	10	2	0.4	2	3	4	5	210	770	330	210
2310	5	70	3	0.6	2	3	4	5	210	770	462	210
2310	5	35	2	0.4	2	3	4	5	210	770	1155	210
2310	5	70	3	0.6	2	3	4	5	210	770	770	210
2310	5	10	2	0.4	2	3	4	5	210	770	210	330
2310	5	10	2	0.4	2	3	4	5	210	770	330	330
2310	5	10	2	0.4	2	3	4	5	210	770	462	330
2310	5	5	1	0.2	2	3	4	5	210	770	1155	330
2310	5	10	2	0.4	2	3	4	5	210	770	770	330
2310	5	14	2	0.4	2	3	4	5	210	770	210	462
2310	5	2	1	0.2	2	3	4	5	210	770	330	462
2310	5	14	2	0.4	2	3	4	5	210	770	462	462
2310	5	7	1	0.2	2	3	4	5	210	770	1155	462
2310	5	14	2	0.4	2	3	4	5	210	770	770	462
2310	5	70	3	0.6	2	3	4	5	210	770	210	1155
2310	5	10	2	0.4	2	3	4	5	210	770	330	1155
2310	5	70	3	0.6	2	3	4	5	210	770	462	1155
2310	5	35	2	0.4	2	3	4	5	210	770	1155	1155
2310	5	70	3	0.6	2	3	4	5	210	770	770	1155
2310	5	70	3	0.6	2	3	4	5	210	770	210	770
2310	5	10	2	0.4	2	3	4	5	210	770	330	770
2310	5	70	3	0.6	2	3	4	5	210	770	462	770
2310	5	35	2	0.4	2	3	4	5	210	770	1155	770
2310	5	70	3	0.6	2	3	4	5	210	770	770	770
2310	5	10	2	0.4	2	3	4	5	330	770	210	210
2310	5	10	2	0.4	2	3	4	5	330	770	330	210
2310	5	10	2	0.4	2	3	4	5	330	770	462	210
2310	5	5	1	0.2	2	3	4	5	330	770	1155	210
2310	5	10	2	0.4	2	3	4	5	330	770	770	210
2310	5	10	2	0.4	2	3	4	5	330	770	210	330
2310	5	110	3	0.6	2	3	4	5	330	770	330	330

2310	5	110	3	0.6	2	3	4	5	330	770	462	330
2310	5	55	2	0.4	2	3	4	5	330	770	1155	330
2310	5	110	3	0.6	2	3	4	5	330	770	770	330
2310	5	2	1	0.2	2	3	4	5	330	770	210	462
2310	5	22	2	0.4	2	3	4	5	330	770	330	462
2310	5	22	2	0.4	2	3	4	5	330	770	462	462
2310	5	11	1	0.2	2	3	4	5	330	770	1155	462
2310	5	22	2	0.4	2	3	4	5	330	770	770	462
2310	5	10	2	0.4	2	3	4	5	330	770	210	1155
2310	5	110	3	0.6	2	3	4	5	330	770	330	1155
2310	5	110	3	0.6	2	3	4	5	330	770	462	1155
2310	5	55	2	0.4	2	3	4	5	330	770	1155	1155
2310	5	110	3	0.6	2	3	4	5	330	770	770	1155
2310	5	10	2	0.4	2	3	4	5	330	770	210	770
2310	5	110	3	0.6	2	3	4	5	330	770	330	770
2310	5	110	3	0.6	2	3	4	5	330	770	462	770
2310	5	55	2	0.4	2	3	4	5	330	770	1155	770
2310	5	110	3	0.6	2	3	4	5	330	770	770	770
2310	5	14	2	0.4	2	3	4	5	462	770	210	210
2310	5	2	1	0.2	2	3	4	5	462	770	330	210
2310	5	14	2	0.4	2	3	4	5	462	770	462	210
2310	5	7	1	0.2	2	3	4	5	462	770	1155	210
2310	5	14	2	0.4	2	3	4	5	462	770	770	210
2310	5	2	1	0.2	2	3	4	5	462	770	210	330
2310	5	22	2	0.4	2	3	4	5	462	770	330	330
2310	5	22	2	0.4	2	3	4	5	462	770	462	330
2310	5	11	1	0.2	2	3	4	5	462	770	1155	330
2310	5	22	2	0.4	2	3	4	5	462	770	770	330
2310	5	14	2	0.4	2	3	4	5	462	770	210	462
2310	5	22	2	0.4	2	3	4	5	462	770	330	462
2310	5	154	3	0.6	2	3	4	5	462	770	462	462
2310	5	77	2	0.4	2	3	4	5	462	770	1155	462

2310	5	154	3	0.6	2	3	4	5	462	770	770	462
2310	5	14	2	0.4	2	3	4	5	462	770	210	1155
2310	5	22	2	0.4	2	3	4	5	462	770	330	1155
2310	5	154	3	0.6	2	3	4	5	462	770	462	1155
2310	5	77	2	0.4	2	3	4	5	462	770	1155	1155
2310	5	154	3	0.6	2	3	4	5	462	770	770	1155
2310	5	14	2	0.4	2	3	4	5	462	770	210	770
2310	5	22	2	0.4	2	3	4	5	462	770	330	770
2310	5	154	3	0.6	2	3	4	5	462	770	462	770
2310	5	77	2	0.4	2	3	4	5	462	770	1155	770
2310	5	154	3	0.6	2	3	4	5	462	770	770	770
2310	5	35	2	0.4	2	3	4	5	1155	770	210	210
2310	5	5	1	0.2	2	3	4	5	1155	770	330	210
2310	5	35	2	0.4	2	3	4	5	1155	770	462	210
2310	5	35	2	0.4	2	3	4	5	1155	770	1155	210
2310	5	35	2	0.4	2	3	4	5	1155	770	770	210
2310	5	5	1	0.2	2	3	4	5	1155	770	210	330
2310	5	55	2	0.4	2	3	4	5	1155	770	330	330
2310	5	55	2	0.4	2	3	4	5	1155	770	462	330
2310	5	55	2	0.4	2	3	4	5	1155	770	1155	330
2310	5	55	2	0.4	2	3	4	5	1155	770	770	330
2310	5	7	1	0.2	2	3	4	5	1155	770	210	462
2310	5	11	1	0.2	2	3	4	5	1155	770	330	462
2310	5	77	2	0.4	2	3	4	5	1155	770	462	462
2310	5	77	2	0.4	2	3	4	5	1155	770	1155	462
2310	5	77	2	0.4	2	3	4	5	1155	770	770	462
2310	5	35	2	0.4	2	3	4	5	1155	770	210	1155
2310	5	55	2	0.4	2	3	4	5	1155	770	330	1155
2310	5	385	3	0.6	2	3	4	5	1155	770	462	1155
2310	5	385	3	0.6	2	3	4	5	1155	770	1155	1155
2310	5	385	3	0.6	2	3	4	5	1155	770	770	1155
2310	5	35	2	0.4	2	3	4	5	1155	770	210	770

2310	5	55	2	0.4	2	3	4	5	1155	770	330	770
2310	5	385	3	0.6	2	3	4	5	1155	770	462	770
2310	5	385	3	0.6	2	3	4	5	1155	770	1155	770
2310	5	385	3	0.6	2	3	4	5	1155	770	770	770
2310	5	70	3	0.6	2	3	4	5	770	770	210	210
2310	5	10	2	0.4	2	3	4	5	770	770	330	210
2310	5	70	3	0.6	2	3	4	5	770	770	462	210
2310	5	35	2	0.4	2	3	4	5	770	770	1155	210
2310	5	70	3	0.6	2	3	4	5	770	770	770	210
2310	5	10	2	0.4	2	3	4	5	770	770	210	330
2310	5	110	3	0.6	2	3	4	5	770	770	330	330
2310	5	110	3	0.6	2	3	4	5	770	770	462	330
2310	5	55	2	0.4	2	3	4	5	770	770	1155	330
2310	5	110	3	0.6	2	3	4	5	770	770	770	330
2310	5	14	2	0.4	2	3	4	5	770	770	210	462
2310	5	22	2	0.4	2	3	4	5	770	770	330	462
2310	5	154	3	0.6	2	3	4	5	770	770	462	462
2310	5	77	2	0.4	2	3	4	5	770	770	1155	462
2310	5	154	3	0.6	2	3	4	5	770	770	770	462
2310	5	70	3	0.6	2	3	4	5	770	770	210	1155
2310	5	110	3	0.6	2	3	4	5	770	770	330	1155
2310	5	770	4	0.8	2	3	4	5	770	770	462	1155
2310	5	385	3	0.6	2	3	4	5	770	770	1155	1155
2310	5	770	4	0.8	2	3	4	5	770	770	770	1155
2310	5	70	3	0.6	2	3	4	5	770	770	210	770
2310	5	110	3	0.6	2	3	4	5	770	770	330	770
2310	5	770	4	0.8	2	3	4	5	770	770	462	770
2310	5	385	3	0.6	2	3	4	5	770	770	1155	770
2310	5	770	4	0.8	2	3	4	5	770	770	770	770

Table A.1:

Bibliography

[ACNL$^+$19] S Arpin, C Camacho-Navarro, K Lauter, J Lim, K Nelson, K Scholl, and J Sotakova. *Adventures in Supersingularland.* arXiv, 2019.

[ACVCD$^+$19] G ADJ, D Cervantes-Vazquez, Jesus-Javier Chi-Dominguez, A Menezes, and F Rodriguez-Henriquez. *On the Cost of Computing Isogenies Between Supersingular Elliptic Curves.* Springer, 2019.

[ATI22] California State University San Bernardino ATI. *High Performance Computing Initiative.* https://www.csusb.edu/academic-technologies-innovation/xreal-lab-and-high-performance-computing/high-performance-0, 2022.

[Bix06] R Bix. *Conics and Cubics.* Springer, 2006.

[CGL06] D Charles, E Goren, and K Lauter. *Cryptographic Hash Functions from Expander Graphs.* Microsoft Research, 2006.

[DFJP11] L De Feo, D Jao, and J Plut. *Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies.* Springer, 2011.

[DH76] H Diffie and M.E. Hellman. *New Directions in Cryptography.* IEEE, 1976.

[Gal17] J Gallian. *Contemporary Abstract Algebra.* Cengage Learning, 2017.

[HPS14] J Hoffstein, J Pipher, and J Silverman. *An Introduction to Mathematical Cryptography.* Springer, 2014.

[Mil04] V Miller. *The Weil Pairing, and its Efficient Calculation.* Journal of Cryptology, 2004.

[MS16] D Moody and D Shumow. *Analogues of Velus Formulas For Isogenies on Alternate Models of Elliptic Curves.* Mathematics of Computation, 2016.

[MVO93] A Menezes, S Vanstone, and T Okamoto. *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field.* IEEE, 1993.

[Sch85] R Schoof. *Elliptic curves over finite fields and the computation of square roots mod p.* Mathematics of Computation, 1985.

[Sch95] R Schhof. *Counting Points on Elliptic Curves Over Finite Fields.* Journal de Théorie des Nombres de Bordeaux, 1995.

[Sil16] J Silverman. *The Arithmetic of Elliptic Curves.* Springer, 2016.

[vOW96] P.C. van Oorschot and M.J. Wiener. *Parallel Collision Search with Cryptanalytic Application.* Journal of Cryptology, 1996.

[Vé71] J Vélu. *Isogenies between Elliptic Curves.* Academie des Sciences des Paris, 1971.