Creating New Elliptic Curves for Uses in Cryptography

By

Mollie J. Zechlin

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTERS IN SCIENCE

(MATHEMATICS)

at the

CALIFORNIA STATE UNIVERSITY - CHANNEL ISLANDS

2019

© 2019

Mollie Johanna Zechlin

ALL RIGHTS RESERVED

APPROVED FOR THE MATHEMATICS PROGRAM

[illegible] 03/28/2019

Dr. Ivona Grzegorczyk, Advisor Date

[illegible] 14 March 2019

Dr. Brian Sittinger Date

[illegible] 03/14/2019

Dr. Michael Soltys Date

APPROVED FOR THE UNIVERSITY

[illegible] 4/4/2019

Dr. Osman Özturgut Date

Dedication

I dedicate this work to my parents, my sister and brother-in-law who always supported me in every decision and whom I look up to greatly. And to Kevin, who always believes I can do anything I decide to, even when I don't. You have all supported me during this long process and it has given me tremendous strength, thank you so much.

Acknowledgements

I would like to thank my advisor, Dr. Ivona Grzegorczyk from California State University Channel Islands who has been a great mentor and incredibly supportive. I would also like to thank the rest of my committee, Dr. Michael Soltys and Dr. Brian Sittinger for helpful comments and assistance while completing this thesis.

Camarillo, California

April 15, 2019

ABSTRACT

Public key cryptography, is the basis of modern cryptography, allows us to send and receive messages over public channels secretly, without requiring a meeting beforehand. Most public key cryptosystems, such as the Diffie-Hellman Key Exchange, rely on the difficulty of solving the Discrete Logarithm Problem (DLP). We can translate public key cryptosystems that rely on the DLP to Elliptic Curve cryptosystems as the Elliptic Curve Discrete Logarithm Problem (ECDLP) is believed to be more difficult and therefore harder to break. There are certain precautions we need to take when using Elliptic Curve Cryptography to safeguard against particular attacks on the cryptosystem. Therefore, picking a curve that is secure enough is crucial to a good cryptosystem. Unfortunately, there are only a handful of secure elliptic curves that are publicly known and used. The goal of this thesis is to generate more elliptic curves that are useful for our security systems.

TABLE OF CONTENTS

0.1 Notation

S	Set of primes from 1 to 2^{300} (only for Sage Code uses)
p	Prime used in $\mathbb{Z}_p$
LFS	Set of large factors of group order
F	The field $\mathbb{Z}_p$ (only for Sage Code uses)
b	The constant added to $y^2 = x^3 + ax + b$
E	The elliptic curve $y^2 = x^3 - 3x + b$ over $\mathbb{Z}_p$ (only for Sage Code uses)
N	Number of elements in an elliptic curve group
D	Set of divisors of N (only for Sage Code uses)
l	The largest factor of N, i.e. $N = l \cdot f$ for some number f
s	Order of $\{P, 2P, ..., sP\}$ (order of point P)
m	$m = \frac{s}{l}$ where s, l are defined above

CHAPTER 1

INTRODUCTION

1.1 Public Key Cryptography and Elliptic Curves

Public key cryptography, is the basis of modern cryptography, allows us to send and receive messages over public channels secretly, without requiring a meeting beforehand. Most public key cryptosystems, such as the Diffie-Hellman Key Exchange, rely on the difficulty of solving the Discrete Logarithm Problem (DLP). We can translate public key cryptosystems that rely on the DLP to Elliptic Curve cryptosystems as the Elliptic Curve Discrete Logarithm Problem (ECDLP) is believed to be more difficult and therefore harder to break. We describe an elliptic curve as "secure" if it passes all the tests in the program written at the end of this paper and thereffore is resistant to the attacks described.

1.1 Curves in Projective Space

We consider our elliptic curves in a projective space $\mathbb{P}^2$, hence we start by explaining basic concepts for this situation.

Definition 1.1.1. *The projective space $\mathbb{P}^2(X,Y,Z)$ is the set of equivalence classes of lines passing though the origin in $\mathbb{R}^3$. Each line in $\mathbb{R}^3$ that goes through the origin can be represented as the point it hits in the plane $z=1$. Every line in the plane $z=0$ passing the origin but parallel to $z=1$ is sent as a point to the projective line at infinity known as P^1. To be precise: $\mathbb{P}^2 = \mathbb{R}^3 - \{(0,0,0)\}/\sim$ where $\sim$ is defined as $(a,b,c)\sim(\lambda a,\lambda b,\lambda c)$*

Remark 1.1.2. *Notice $\mathbb{R}^2 \cup P^1 = \mathbb{P}^2$ and therefore $\mathbb{R}^2(x,y) \subset \mathbb{P}^2(X,Y,Z)$, is a Zariski-open subset (i.e. very large), given by $z \neq 0$ (or equivalently $z=1$). In this paper, we consider $\mathbb{R}^2(x,y)$ as a quasi-projective algebraic set.*

Remark 1.1.3. *Primarily, we refer to the point at infinity O as the point that lies on every vertical line in the projective plane.*

1.2 What are Elliptic Curves?

Elliptic curves are interesting objects in projective space as they have a useful group structure. In the following section, we describe what they are.

Definition 1.2.1. *An elliptic curve is the set of solutions to $E \subset \mathbb{R}^2$ that can be described by the following equation:*

$$E : y^2 = x^3 + ax + b \text{ for some } a, b \in \mathbb{R}$$

such that $4a^3 + 27b^2 \neq 0$ (see Section 2.1).

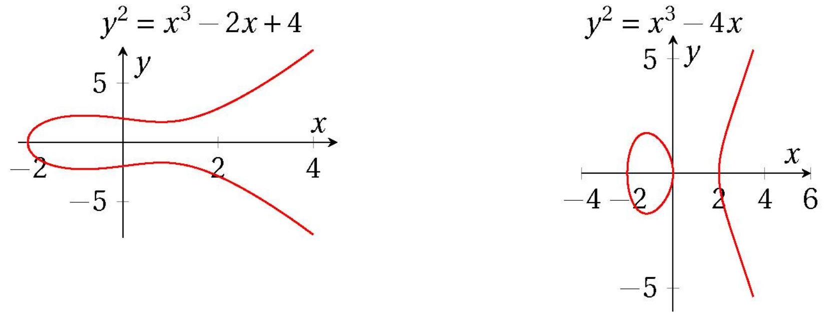


Figure 1.1: Examples of Elliptic Curves

1.3 Addition on Elliptic Curves

In this section, we show how we can “add” two points on a given elliptic curve. This operation of addition is crucial to understanding how the modern cryptosystems work. In the following paragraph we describe how this addition works geometrically.

Suppose that points P and Q are on the elliptic curve E. We first draw a line L through points P and Q. By Bézout's Theorem, there exists a third point on L that is also on E; we denote this point R. [8] Finaly, we define $P+Q=R^*$, the reflection of R across the x-axis. This is depicted in figure 1.2 below.

Note that when the points give a vertical line they add to $\boldsymbol{O}$ in an elliptic curve sense, i.e. the point at infinity, as we consider $E \subset \mathbb{R}^2$, i.e. $R+R^*=\boldsymbol{O}$.

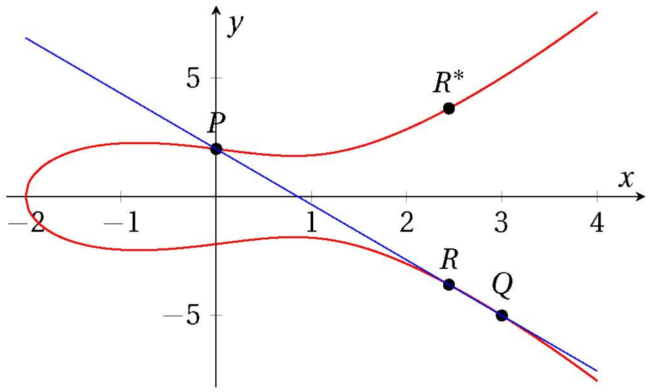


Figure 1.2: Addition of Points on an Elliptic Curve

Theorem 1.3.1. *The set E of rational points on an elliptic curve has the following properties under the addition operation where $P, Q, R \in E$ and $\boldsymbol{O}$ is the point at infinity.*

1. *Identity: $P+\boldsymbol{O}=P$*

2. *Commutative:* $P+Q=Q+P$

3. *Inverse:* $P+P^*=\mathcal{O}$ *where* P^* *is the reflection of* P.

4. *Associative:* $(P+R)+Q=P+(R+Q)$

In other words, the set of points on the curve E with respect to "+" is an abelian group. The proof of the associative property in particular is computationally long and can be found in [12].

We can find algebraic equations for $R=P+Q$ by first defining $P=(x_1,y_1)$ and $Q=(x_2,y_2)$. We start with the equation $y^2=x^3+ax+b$. Let's write the line L that goes through the points P and Q, in slope-intercept form, $y=mx+c$. If your line is vertical, it goes through two points that are additive inverses of each other. Therefore, they add to the point at infinity. We therefore eliminate this from our possible lines. We can find where the above equations intersect by substitution of the line into the elliptic curve. This gives us the equation in terms of x:

$$x^3-m^2x^2+(a-2mc)x+b-c^2=0.$$

The roots of this equation, x_1,x_2,x_3, are the x-coordinates of the points that are on both the elliptic curve and our line.

To find the formula for R, observe that

$$x^3 - m^2x^2 + (a - 2mc)x + b - c^2 = 0 = (x - x_1)(x - x_2)(x - x_3).$$

Equating coefficients of x^2 yields, x^2,

$$-m^2 = -(x_1 + x_2 + x_3) \text{ and thus } x_3 = m^2 - x_1 - x_2.$$

Now, we can solve for y_3 by substituting x_3 into the equation of our line where $R = (x_3, y_3)$:

$$y_3 = mx_3 + c$$

It follows from some algebra that

$$y_3 = m(x_1 - x_3) - y_1.$$

If P and Q are not equal, we can find the slope m of the line L since we have two points on the line. When adding a point to itself (or doubled), this is to be thought of as drawing a tangent line to the curve at the point to be doubled. We determine the slope for this line by using implicit differentiation to take the derivative of $x^3 + px + q = y^2$ and plugging in the point $P = (x_1, y_1)$. Therefore,

$$m = \frac{3x_1^2 + p}{2y_1} \text{ when } P = Q.$$

To get the point $R = P + Q$, we can substitute slope m and points P, Q into the equations from before.

By Bézout's Theorem, each line in the plane intersects with the elliptic curve in 3 points as long as you allow complex numbers, count multiplicities correctly and count $\mathcal{O}$ if necessary. [8]

1. Although, in a traditional picture of $\mathbb{R}^2$ we do not see the complex coordinates as this would be a 4-dimensional graph over $\mathbb{R}$, we still need to count complex points.

2. If an x-coordinate is a double root, then it should be counted twice since it is a point of multiplicity 2.

3. The point at infinity $\mathcal{O}$ is the point lying on every vertical line at infinity. Thus any 2 points on the curve creating a vertical line also intersects at the third point $\mathcal{O}$.

Remark 1.3.2. *We write $2P$ for $P + P$, $3P$ for $P + P + P$ and so on.*

Algorithm 1.3.3 (Addition Algorithm)**.** *Given an elliptic curve $E : y^2 = x^3 + ax + b$ and $P_1, P_2 \in E$ over $\mathbb{R}^2$, we can find $P_1 + P_2$.*

1. If $P_1 = \mathcal{O}$, then $P_1 + P_2 = P_2$.

2. *Let* $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$.

3. *If* $x_1 = x_2, y_1 = -y_2$, *then* $P_1 + P_2 = \mathcal{O}$. $\quad (P_1 = -P_2)$

4. *Let* $m = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2 \\ \\ \dfrac{3x_1^2 + p}{2y_1} & \text{if } P_1 = P_2. \end{cases}$

5. *Let* $x_3 = m^2 - x_1 - x_2$ *and* $y_3 = m(x_1 - x_3) - y_1$.

Then, $P_1 + P_2 = (x_3, y_3)$.

Lemma 1.3.4. *If* P *and* Q *are points on an elliptic curve and have rational coordinates, then* $P + Q$ *must have rational coordinates.*

Proof. Assume rational points P and Q are on the elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{R}^2$. Write $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. We use the Addition Algorithm to determine what the coordinates of $P + Q$ look like and therefore if they are in fact still rational.

1. If $P = \mathcal{O}$, then $P + Q = Q$, which are known to be rational.

2. If $P = -Q$, then $P + Q = -Q + Q = \mathcal{O}$. Therefore, we get the identity.

3. If $P \neq Q$, then $m = \frac{y_2 - y_1}{x_2 - x_1}$. Since $x_1, x_2, y_1, y_2 \in \mathbb{Q}$ then $m \in \mathbb{Q}$. Therefore, $x_3 = m^2 - x_1 - x_2 \in \mathbb{Q}$ and $y_3 = m(x_1 - x_3) - y_1 \in \mathbb{Q}$.

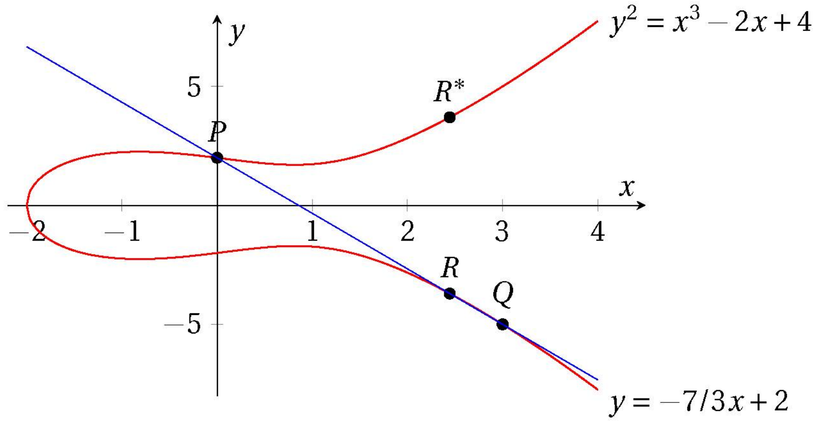


Figure 1.3: Addition of Points over a Finite Field

4. If $P = Q$, then $m = \frac{3x_1^2+a}{2y_1}$. If $a \in \mathbb{Q}$, then $m \in \mathbb{Q}$. Therefore, $x_3 = m^2 - x_1 - x_2 \in \mathbb{Q}$ and $y_3 = m(x_1 - x_3) - y_1 \in \mathbb{Q}$.

   Claim: $a, b \in \mathbb{Q}$.

   Since P and Q have rational coordinates and are solutions to the equation $y^2 = x^3 + ax + b$, we conclude that a and b must be rational.

□

1.3.1 Example

Consider the curve $E : y^2 = x^3 - 2x + 4$.

We apply this algorithm to find $P + Q$ on the curve $E : y^2 = x^3 - 2x + 4$, where $P = (0, 2)$ and $Q = (3, -5)$. First, we notice that our points do not fulfill the requirements for parts 1 and 3 of the algorithm.

Therefore, our first step is to find the slope of the line connecting P and Q. Since $m = \dfrac{-5-2}{3-0} = -\dfrac{7}{3}$, $\quad x_3 = (-\frac{7}{3})^2 - 0 - 3 = \frac{22}{9}$ and $y_3 = -\frac{7}{3}(0 - \frac{22}{9}) - 2 = \frac{100}{27}$.

Therefore, $R^* = (22/9, 100/27)$.

1.4 Elliptic Curves over Finite Fields

Instead of viewing an elliptic curve over $\mathbb{Q}, \mathbb{R}$ or $\mathbb{C}$, it proves useful to consider an elliptic curve over a finite field such as $\mathbb{Z}_p$, where p is prime. The set of points on the elliptic curve E over the finite field $\mathbb{Z}_p$ is expressed as follows:

$$E = \{(x, y) \mid x, y \in \mathbb{Z}_p,\ \ y^2 = x^3 + ax + b\} \cup \{\boldsymbol{O}\}.$$

In particular, E over $\mathbb{Z}_p$ has a finite number of points. The order of an elliptic curve group is the number of elements in it. Since there are a finite number of points it has a finite order N.

1.4.1 Example

Let's consider the curve $E: y^2 = x^3 - 2x$ over $\mathbb{Z}_{13}$.

First, we must determine what values of y^2 are possible when $y \in \mathbb{Z}_{13}$:

y	0	1	2	3	4	5	6	7	8	9	10	11	12
y^2	0	1	4	9	3	12	10	10	12	3	9	4	1

Now, find the points on E over $\mathbb{Z}_{13}$. X in the y row below represents y being undefined for that value of x.

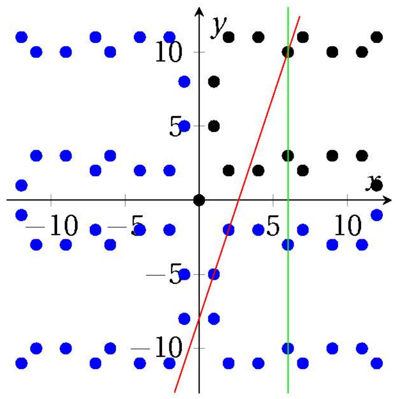
x	0	1	2	3	4	5	6	7	8	9	10	11	12
y^2	0	12	4	8	4	11	9	4	8	9	5	9	1
y	0	5,8	2,11	X	2,11	X	3,10	2,11	X	3,10	X	3,10	1,11

Therefore, the set of points in the group of the curve E over $\mathbb{Z}_{13}$ is given by:

$$E(\mathbb{Z}_{13}) = \{(0,0),(1,5),(1,8),(2,2),(2,11),(4,2),(4,11),(6,3),(6,10),$$

$$(7,2),(7,11),(9,3),(9,10),(11,3),(11,10),(12,1),(12,11),\mathcal{O}\}.$$

There are 18 points in this elliptic curve group, so our order is $N = 18$. Graphically, these points can be repeated infinitely and can be reflected across the x- and y-axis as shown below.



We can use the same methods as before to show our addition algorithm works for finite fields using this example.

As an example, we find $(2,-2)+(1,-5)$ on our elliptic curve (over $\mathbb{Z}_{13}$). As shown in the picture above, the line that goes through these points also goes through $(6,10)$. Then, reflecting this point over x-axis yields $(2,-2)+(1,-5) = (6,-10) = (6,3)$.

Next, we find $(2,-2)+(1,-5)$.

We can ignore step 1 and 3 since they do not apply here. Now, we find the slope of our line:

$$m = \frac{-2+5}{2-1} = 3.$$

Substituting our slope we obtain

$$x_3 = 3^2 - 1 - 2 = 6 \qquad y_3 = -(-5+3(6-1)) = -10.$$

Therefore, we get $(2,-2)+(1,-5) = (6,-10)$ by this method as well.

CHAPTER 2

PROPERTIES OF ELLIPTIC CURVES

An elliptic curve in $\mathbb{P}^2(X,Y,Z)$ has a general equation of the form

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

where $X, Y, Z \in \mathbb{R}$ are homogeneous coordinates. In our case, we use $[0,1,0]$ as our base point, meaning that the curve is written as the locus in $\mathbb{P}^2$ of a cubic equations with $\boldsymbol{O}$ as the only point on the line at infinity (as we can use transformations of $\mathbb{P}^2$ to move it to the right position).

Substituting $x = X/Z$ and $y = Y/Z$ where $Z \neq 0$ reduces this to an equation in $\mathbb{R}^2$:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

We can simplify this even further when we assume we have characteristic not equal to 2 or 3. This gives us the *Short Weierstrass* form that we are going to use throughout this paper:

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

For this form, the *discriminant* is given by $\Delta = -16(4a^3 + 27b^2)$, and the *$j$-invariant* is given by $j = -1728\frac{(64a^3)}{\Delta}$. [8]

Remark 2.0.1. *Whenever $p > 3$, we can convert any elliptic curve over $\mathbb{Z}_p$ to Short Weierstrass form.*

Definition 2.0.2. *A curve is* nonsingular *if every point on the curve has only one tangent line to the curve, i.e. for $f(x, y) = y^2 - x^3 + ax + b$, $\left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]_{(x,y)} \neq [0,0]$ for all $(x, y) \in E$.*

Proposition 2.0.3. *(i) A curve is nonsingular if and only if $\Delta \neq 0$.*

(ii) Two elliptic curves are isomorphic over $\overline{K}$ if and only if they have the same j-invariant. [8]

2.1 Isogenies

Definition 2.1.1. *A* morphism *is a function $\phi : X \to Y$ such that $\phi(a \cdot b) = \phi(a) \cdot \phi(b)$ where $a, b \in X$.*

Definition 2.1.2. *Let E_1 and E_2 be elliptic curves. An* isogeny *is a morphism*

$$\phi : E_1 \to E_2 \textit{ satisfying } \phi(\mathcal{O}_{E_1}) = \{\mathcal{O}_{E_2}\}.$$

Two elliptic curves, E_1, E_2 are isogenous *if there exists an isogeny between the two curves such that $\phi(\mathcal{O}_{E_1}) = \{\mathcal{O}_{E_2}\}$.*

Definition 2.1.3. *The endomorphism ring is the set of isogenies from an elliptic curve to itself:*

$$\mathrm{End}(E) = \{\phi \ : \ E \to E, \ \phi \textit{ is an isogeny}\}.$$

We can now define the *multiplication by m* isogeny, where we fix some $m \in \mathbb{Z}$:

$$[m] : E \to E \text{ such that when } m > 0,$$

$$[m](P) = \underbrace{P + P + ... + P}_{m \text{ times}}.$$

Definition 2.1.4. *The m-*torsion subgroup *of E is the set containing points of E of order m:*

$$E[m] = \{P \in E \ : \ [m]P = \mathcal{O}\}.$$

The torsion subgroup *of E is the set of points of finite order:*

$$E_{tors} = \bigcup_{m=1}^{\infty} E[m].$$

Fix $Q \in E$; we now define a *translation-by-Q-map*:

$$\tau_Q : E \longrightarrow E, \qquad P \longmapsto P + Q.$$

This map is an isomorphism since τ_{-Q} gives an inverse. Let $F : E_1 \to E_2$ be an arbitrary isomorphism of elliptic curves. Notice that $\phi = \tau_{-F(O)} \circ F$ is an isogeny, since $\phi(\boldsymbol{O}) = \boldsymbol{O}$.

Therefore, we know that any morphism F between elliptic curves can be written as a composition of an isogeny and a translation:

$$F = \tau_{F(O)} \circ \phi.$$

Remark 2.1.5. *We pick a specific P of order m on our E later to build our cryptosystem.*

Lemma 2.1.6. *Let E/K be an elliptic curve and $m \in \mathbb{Z}$ and $m \neq 0$ in K. Then the multiplication-by-m map on E (as described earlier) is a separable endomorphism.* [8]

2.2 The Weil Pairing

The following definitions regarding divisors are necessary for an understanding of the Weil Pairing.

Definition 2.2.1. *A divisor $D \in \mathrm{Div}(C)$ is a formal sum of points*

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

where $n_P \in \mathbb{Z}$ and $n_P = 0$ for all but finitely many points $P \in C$.

Definition 2.2.2. *Let f is a function of a curve C over an algebraically closed field. We examine on the divisor*

$$\mathrm{div}(f) = \sum_{P \in C} \mathrm{ord}_P(f)(P)$$

where $\mathrm{ord}_P(f)(P)$ is the order of the f at point P.

Definition 2.2.3. *The divisor group of a curve C is the free abelian group generated by the points of C. This is given by $\mathrm{Div}(C)$.*

The Weil pairing takes in a pair of points from the group $E[m]$, the group of points of order m on the curve E. It gives an output of an m^{th} root of unity. This is denoted as $e_m(P, Q)$.

Definition 2.2.4. *Let f_P and f_Q be rational functions on E such that*

$$\operatorname{div}(f_P) = m[P] - m[\boldsymbol{O}]$$

and

$$\operatorname{div}(f_Q) = m[Q] - m[\boldsymbol{O}].$$

The Weil Pairing of P and Q is

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

where $S \in E$ is any point such that $S \notin \{\boldsymbol{O}, P, -Q, P-Q\}$. [13]

The Weil pairing is bilinear and a number of other beneficial properties that we do not discuss here. These properties are what make the pairing easy to manipulate and use for purposes such as the MOV attack in Chapter 4

2.3 Endomorphism Rings

Theorem 2.3.1. *Let K be a field of prime characteristic q, and E/K be an elliptic curve. For every $r \geqslant 1$, let*

$$\phi_r : E \longrightarrow E^{(q^r)} \text{ defined by } \phi_r(P) = \phi_r((x,y)) = (x^{q^r}, y^{q^r})$$

the q^r-power Frobenius map and its dual

$$\hat{\phi}_r : E^{(q^r)} \longrightarrow E.$$

The following are equivalent:

1. $E[q^r] = 0$ *for one (all)* $r \geqslant 1$.
2. $\hat{\phi}_r$ *is inseparable for one (all)* $r \geqslant 1$.
3. *The map* $[q] : E \to E$ *is purely inseparable and* $j(E) \in F_{q^2}$.
4. *End(E) is a subalgebra of the quaternions.*
5. *The formal group* $\hat{E}/K$ *associated to* E *has height 2.* [8]

Definition 2.3.2. *A curve that satisfies the above requirements is refered to as* supersingular.

Remark 2.3.3. *Supersingular curves are important to understand since we must avoid using them in elliptic curve cryptography as is explained in Chapter 4.*

Definition 2.3.4. *An algebra of the form*

$$\mathcal{K} = \mathbb{Q} + \mathbb{Q}\alpha + \mathbb{Q}\beta + \mathbb{Q}\alpha\beta$$

where the multiplication satisfies

$$\alpha^2, \beta^2 \in \mathbb{Q}, \quad \alpha^2 < 0, \quad \beta^2 < 0, \quad \beta\alpha = -\alpha\beta$$

is known as a definite quaternion algebra *over* $\mathbb{Q}$ *(however we refer to it as just a* quaternion algebra *since it is the only kind we encounter in this paper).*

Definition 2.3.5. *Let $\mathscr{K}$ be a $\mathbb{Q}$-algebra that is finitely generated over $\mathbb{Q}$. An order $\mathscr{R}$ of $\mathscr{K}$ is a subring of $\mathscr{K}$ that is finitely generated as a $\mathbb{Z}$-module and satisfies $\mathscr{R} \otimes_{\mathbb{Z}} \mathbb{Q} = \mathscr{K}$.*

Theorem 2.3.6. *Let $\mathscr{R}$ be a ring of characteristic 0 having no zero divisors, and assume that it has the following properties:*

(I.) $\mathscr{R}$ has at most rank 4 as a $\mathbb{Z}$-module.

(II.) $\mathscr{R}$ has an anti-involution $\alpha \mapsto \hat{\alpha}$ satisfying

$$\widehat{\alpha + \beta} = \hat{\alpha} + \hat{\beta}, \quad \widehat{\alpha\beta} = \hat{\alpha}\hat{\beta}, \quad \hat{\hat{\alpha}} = \alpha, \quad \hat{a} = a \quad \textit{for } a \in \mathbb{Z} \subset \mathscr{R}.$$

(III.) For $\alpha \in \mathscr{R}$, the product $\alpha\hat{\alpha} \in \mathbb{Z}_{\geqslant 0}$ and $\alpha\hat{\alpha} = 0$ if and only if $\alpha = 0$.

The $\mathscr{R}$ is one of the following types of rings:

(i.) $\mathscr{R} \cong \mathbb{Z}$

(ii.) $\mathscr{R}$ is an order in an imaginary quadratic extension of $\mathbb{Q}$.

(iii.) $\mathscr{R}$ is an order in a quaternion algebra over $\mathbb{Q}$. [8]

Lemma 2.3.7. *The endomorphism right of an elliptic curve E/K is either $\mathbb{Z}$, an order in an imaginary quadratic field, or an order in a quaternion algebra. (If* $\operatorname{char}(K) = 0$, *then only the first two are posibilities). [8]*

2.4 Elliptic Curves over Finite Fields

We wish to determine the number of points in an elliptic curve group E over $\mathbb{Z}_p$. This is the number of solutions to the elliptic curve equation, plus one for the point at infinity, written as N. We can determine that each x-value with give us at most two y-values, resulting in the upper bound $N \leqslant 2p+1$.

Theorem 2.4.1 (Hasse)**.** *Let E be an elliptic curve defined over a finite field Z_p. Then $|N-p-1| \leqslant 2\sqrt{p}$. [8]*

Remark 2.4.2. *This theorem gives us a bound for the number of points, but does not point to an efficient way to calculate these points when p is a large number. This results in the elliptic curve discrete logarithm problem (ECDLP) that is used in public key cryptosystems.*

CHAPTER 3

CRYPTOSYSTEMS USING ELLIPTIC CURVES

In this chapter, we discuss how we use elliptic curves in in cryptography through examples of cryptosystems. In these examples we use Alice and Bob as the parties who are trying to communicate securely.

3.1 Discrete Logarithm Problem

Definition 3.1.1 (Discrete Logarithm Problem(DLP))**.** *Find an integer x such that $h \equiv g^x \bmod p$ where $h, g \in \mathbb{Z}_p^*$.*

This problem can be solved by a computer in subexponential time using the fastest algorithm currently at our disposal.

Definition 3.1.2 (Elliptic Curve Discrete Logarithm Problem (ECDLP))**.** *Find an integer n such that $Q = nP$ given $P, Q \in E$ over the field $\mathbb{Z}_p$.*

This problem can be best solved in exponential time as long as we take the precautions described in Chapter 4 and 5. Therefore, this problem is harder and hence more secure.

3.2 Diffie-Hellman Key Exchange

An example of a cryptosystem that uses the ECDLP is the Diffie-Hellman Key Exchange. In this section we describe this process.

To start off, Alice and Bob choose an elliptic curve E over a finite field $\mathbb{Z}_p$, and a point on that curve P. This information is public. From here, Alice chooses a secret integer n_A and computes $Q_A = n_A P$, while Bob chooses a secret integer n_B and computes $Q_B = n_B P$. Alice and Bob then publicly exchange their new points Q_A, Q_B. Alice then calculates $n_A Q_B = R_A$ and Bob calculates $n_B Q_A = R_B$. Alice and Bob now have a shared secret value, since

$R_A = n_A Q_B = n_A(n_B P) = n_B(n_A P) = n_B Q_A = R_B$. This algorithm is represented in the following table. [5]

Diffie-Hellman Key Exchange		
Alice's Actions	Bob's Actions	Public / Secret / Sent
	Bob chooses E over $\mathbb{Z}_p$ and point P	Public (known to Alice)
Chooses n_A	Chooses n_B	Secret
Computes $Q_A = n_A P$	Computes $Q_B = n_B P$	Alice $\underset{\longrightarrow}{Q_A}$ Bob (Public) Alice $\underset{\longleftarrow}{Q_B}$ Bob (Public)
Computes $n_A Q_B = R_A$	Computes $n_B Q_A = R_B$	Shared Secret ($R_A = R_B$)

3.3 ElGamal Public Key Cryptosystem

Another example that uses the ECDLP is the ElGamal Public Key Cryptosystem. We now describe the process involved in this cryptosystem.

Alice first chooses a large prime p, an elliptic curve E over $\mathbb{Z}_p$, and a point $P \in E$. This information is public knowledge. She then chooses a secret number n_A, known as her private key. Alice computes and publishes $Q = n_A P$. This is known as her public key. Now Bob wishes to send Alice a message, $M \in E$. He randomly chooses an integer k. Bob computes and sends Alice

$C_1 = kP$ and $C_2 = M + kQ$. To decrypt the message, Alice computes $C_2 - n_A C_1$. Since $C_2 - n_A C_1 = (M + kQ) - n_A(kP) = M + k(n_A P) - n_A(kP) = M$, this computation gives Bob's original message to Alice. This cryptosystem is also described in the table below. [3]

ElGamal Public Key Cryptosystem		
Alice's Actions	Bob's Actions	Public / Secret / Sent
Chooses a large prime p, elliptic curve E over $\mathbb{Z}_p$ and point $P \in E$		Public (known to Bob)
Alice chooses n_A		Secret
Computes $Q = n_A P$		Public
	Chooses k	Secret
	Computes $C_1 = kP$ and $C_2 = M + kQ$	Bob $\xrightarrow{(C_1, C_2)}$ Alice (Public)
Computes $C_2 - n_A C_1 = M$		Secret

Some other cryptosystems we encounter that use elliptic curves are Elliptic Curve Digital Signature (ECDSA) and Pairing-based Cryptography which uses pairings like the Weil Pairing. [13]

CHAPTER 4

ATTACKS ON ECDLP

Elliptic curve cryptography is susceptible to certain types of attacks. We prevent these attacks finding out what makes certain curves vulnerable to them, and then avoiding those curves. In this chapter we discuss these attacks and what needs to be done to avoid them.

4.1 Pollard's ρ Algorithm

Let N be the order or number of points in the Elliptic Curve subgroup. Given N, Pollard's ρ method can solve ECDLP with an exponential running time relying on p. Therefore, large prime numbers are required for security.

Algorithm 4.1.1 (Pollard's ρ Algorithm)**.** *Let* $N = |G|$. *Choose a function* f :

$G \longrightarrow G$ that behaves relatively "randomly". Start with a random point P_0. Compute $f(P_0) = P_1, f(P_1) = P_2$ and so on, so that $P_{i+1} = f(P_i)$. Since there are a finite number of points in our set, for some $i < j$ such that $P_i = P_j$. Therefore, $P_{i+1} = f(P_i) = f(P_j) = P_j$. It follows that $P_{i+n} = P_{j+n}$ for some $i < j$ for all $n \geqslant 0$. Therefore, the randomly chosen function f is periodic.

We must choose a function that is random, but it also needs to give us useful information, otherwise the point to the algorithm is lost. There are many ways to create this f, however, we do not go into the details here. [13]

4.2 The Pohlig-Hellman Method

Given P, Q in our elliptic curve group, to solve the ECDLP, we want k such that $Q = kP$. We can find the order s of our point P in our group. (We can also find the factorization of s.) Since this method utilizes the factorization of s, if we us a point P whose order s has at least one large prime factor, this curve is secure against this attack. We can design a point in the following way:

1. We first start with a group having a large prime order factor l.

2. Though trial an error, we should be able to quickly find a point P_0, whose order s, is divisible by l. The probability of picking a point like this is $1 - 1/l$.

3. Write the order of our point as $s = lm$.

4. Calculate $P = mP_0$. The point P now has order l.

Given the order of our Elliptic Curve group, N, we are able to solve the ECDLP in subexponential time if N factors into small primes. If N has a large prime factor, then this method is inefficient. We therefore need to have two large primes for our cryptosystem, p and a large prime factor of N. [13]

4.3 The MOV Attack

Definition 4.3.1. *Consider $E[m]$ to be the group of points of order m on the elliptic curve E over $\mathbb{Z}_p$. Let $m \geqslant 1$ such that $p \nmid m$. The* embedding degree *of E with respect to m is the smallest positive integer k such that $E[m] \cong \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$ where we view E over $\mathbb{Z}_{p^k}$. [5]*

The MOV Algorithm is able to convert a ECDLP on E over $\mathbb{Z}_p$ to a DLP on a multiplicative group E over $\mathbb{Z}_{p^k}^{\times}$ using the Weil Pairing. Therefore, if our embedding degree k is large, the DLP is just as difficult as the ECDLP and we need not be concerned. However, the embedding degree may be very small, and as we need a much higher number when working with the DLP, the curve may not be secure.

Notice the field $\mathbb{Z}_{p^k}^{\times}$ is cyclic of order $p^k - 1$. This leads to the second definition of an embedding degree being the smallest positive integer k that satisfies $p^k \equiv 1 \bmod m$ where m is defined as above. [8]

Algorithm 4.3.2 (The MOV Algorithm)**.** *Let us assume that E is an elliptic curve over $\mathbb{Z}_p$ and $a \neq p$ be the order of a point in E that is prime.*

1. *Compute N, the number of points in E over $\mathbb{Z}_{p^k}$. Notice that $a|N$ since there exists a point on our curve that is of order a.*

2. *Pick a random point $T \in E$ over $\mathbb{Z}_{p^k}$ where $T \notin E$ over $\mathbb{Z}_p$.*

3. *Compute $T' = \left(\frac{N}{a}\right) T$. If this gives you $\mathbf{O}$, then choose a different T. If T' is of order a, then continue.*

4. *Compute the Weil pairing values $\alpha = e_a(P, T') \in \mathbb{Z}_{p^k}^{\times}$ and $\beta = e_a(Q, T') \in \mathbb{Z}_{p^k}^{\times}$.*

5. *Find an exponent n such that $\beta = \alpha^n$ in $F_{p^k}^{\times}$. This can be done using index calculus when p^k is not too large.*

6. *It can be shown that now, $Q = nP$, so the ECDLP is solved.*

[13]

Remark 4.3.3. *When we use a curve that is supersingular, we end up with embedding degrees as small as 2, 3, 4, and 6. This is clearly a problem for the security of our curve; therefore, supersingular curves must be avoided at all costs.*

Remark 4.3.4. *As long as the embedding degree m is large enough the curve is resistant to this attack. The number 3 should be large enough.* [13]

Remark 4.3.5. *Many proposed the use of anomalous curves where the number of points in the elliptic curve subgroup over $\mathbb{Z}_p$ is equal to p. However, this makes solving the ECDLP very easy.*

4.3.1 Index Calculus

We can use Index Calculus to solve the discrete log problem. This works when the group is cyclic. Therefore, this method can be used when we use the MOV attack to transfer the ECDLP to the DLP.

Definition 4.3.6. *Consider $g^k \equiv h \bmod p$. The* discrete logarithm *of h with respect to g and p is k represented as $L(h) = k$.*

To find the discrete log for h, we go through the process of index calculus.

Note: $g^{L(h_1h_2)} \equiv h_1h_2 \equiv g^{L(h_1)+L(h_2)} \bmod p$, so

$$L(h_1h_2) \equiv L(h_1) + L(h_2) \bmod (p-1).$$

1. First, we choose a set of small primes B.
2. Let $(m_1, m_2, ..., m_n)$ be a set of numbers that are all products of primes from our set B. Create relations where $g^x \equiv m_i \bmod p$. Note that since the m_i's are still relatively small, the DLP for these relations are not hard to figure out.
3. Write these relations as discrete logs $L(m_i) = x \bmod (p-1)$.
4. We can use these relations together to find the discrete logs for each in our set B.
5. Compute $g^j \cdot h \bmod p$ for several random values of j until we find a number that can be factored into primes from our set B. Fix this number j.
6. We now have $L(h) \equiv L(m_i) - j \bmod (p-1)$.

7. Therefore, we have found $L(h) = x$ and solved the DLP.

This method solves the DLP in subexponential time. [5]

CHAPTER 5

WRITING A PROGRAM

The goal of this thesis is to create new curves by writing a program that eliminates the curves that are not secure through the necessary restrictions found. In Chapters 1 through 4 we have examined elliptic curves, their cryptosystems and the attacks on those cryptosystems. Now that we have gathered information, we are able to write a program based on that information. This chapter lays out the restrictions and decisions we have made and outlines a program to accomplish our goal. It also discusses the secure elliptic curves found using this program.

5.1 Necessary Restrictions on our Curves

The following conditions ensure secure curves:

1. The order of our elliptic curve group should have at least one large prime factor to prevent the Pohlig-Hellman Attack.

2. Curves should not be supersingular, as these give attackers the opportunity to use the MOV attack.

3. To avoid the MOV attack, we avoid elliptic curves such that the number of points on the curve is equal to p where the elliptic curve is over $\mathbb{Z}_p$. These curves are called anomalous, and although they can be attractive because of faster calculations, the ECDLP becomes quickly solvable. Therefore, these curves should also be removed from our search.

5.1.1 Choices made

- We use Short Weierstrass equations for elliptic curves

$$y^2 = x^3 + ax + b,$$

  because they are easier to compute and work with. Every elliptic curve over field $\mathbb{Z}_p$, with a $p > 3$, can be written in Short Weierstrass form. [8]

- We pick $a = -3$ so that our equation is of the form $y^2 = x^3 - 3x + b$. There is good evidence from past findings that these create safe curves. [2]

- We pick $\mathbb{Z}_p$ where p is prime. We start with primes larger than 2^{200}.

5.2 The Program

Since we are using the equation $y^2 = x^3 - 3x + b$ over the field $\mathbb{Z}_p$, the first two things we have to pick are the values of b and the prime p. The number b can technically be any number in $\mathbb{Z}_p$, but we prefer it to be prime as it produces more useful results. We choose to pick a prime p for the field $\mathbb{Z}_p$, and look at curves varying b over the same field. Theoretically, the p's should loop as well, but computational time to go through all the b's made it too time consuming to loop on both of them. Once we have chosen our p, we create the field $\mathbb{Z}_p$. We use the set of primes created by Sage to find a prime we for $\mathbb{Z}_p$ and to check primality later. This can also be done (possibly faster) using the Rabin-Miller primality test with the possibility of getting a pseudoprime.

5.2.1 Outlining the Program

This program has two parts. This first part identifies all the potential curves that stand up to all the attacks without checking that the large factor l of the group order is prime. In the second part we take a list of the l's and find the curves with the prime factors.

We use Sage, an open-source mathematical software that allows programming in python. It has many pre-made functions for elliptic curves making it extremely useful for this project. It is considered to be the top choice for coding in the field of elliptic curves.

- Pick a large prime for our field $\mathbb{Z}_p$ and define the field in Sage.
- Create an empty set of numbers which we fill with the primes we need to check (LFS).
- Start loop on different b's.
  - First we check a number of conditions with if statements that progresses the loop if any of them are true, thereby skipping any curves that have properties we don't want.
    - ∗ Check that the current b does not make the curve singular. (Check that the discriminant does not equal 0).

    * Define the smooth elliptic curve in Sage.
    * Check that the curve is not supersingular.
    * Record the order of the elliptic curve group over our field.
    * Check that the order is large enough ($N > 2^{160}$).
    * Check that the order is not prime.
    * Record the largest divisor of the order N, besides itself, (l) where $N = l \cdot f$ where f is some other number.
    * Check that l is large enough to stay secure.
    * Check the curve is not anomalous (the number of points does not equal our prime p).

  – This way we have narrowed it down to the elliptic curve groups that possibly can work securely. Now we have to pick points and determine if those points generate secure cryptosystems on these curves. For this, we need to create specific points and not just pick them.

    * Go through a loop to pick points.

      · Take a random point in our elliptic curve group and find its order.

        · This loop continues if the order of our point, s, cannot be divided by l, our chosen large prime divisor of our group order.

        · This loop terminates the curve if it goes through 20 points and does not get a good result. The probability that a point is found that follows these requirements within 20 steps is $1 - 1/l$, so extremely close to 1. [13]

    - Now we have a point, P.

    - Rewrite P to be $P = mP$ where $m = \frac{s}{l}$.

    - Now, check if $p^k \equiv 1 \bmod m$ for small values of k. If it does, then those points won't work because they may be weak to the MOV attack, and we must try again.

    - If the curve and point combination meet all the requirements, the program writes the curve, point, and field to the correct file and puts the order factor, l in the list, LFS of large factors to be checked for primality later.

- After this process, the loop starts again with a different prime b.

- The program runs a "while loop" on the list LFS.

- We first check if l is divisible by 2,3,5, and 11 all of which have easy algorithms to check if they divide a number.
- If the number is not divisible 2,3,5, or 11, then the program checks if it is in the list of primes, S. If it is, it adds the index number of the l in LFS and the factor l. The index number is also printed in the document containing the "good curves", and therefore serves as a means to find the correct curve.
- If it is still not in the prime list, then we loop again moving on to the next l.

The process of first checking if l is divisible by 2,3,5, and 11 reduced the time for primality checking tremendously.

5.3 The Curves

We have run this program using the field $\mathbb{Z}_p$ where $p = 3213876088517980551083924184682325205044405987565585670603103$. This is the first prime after 2^{201} according to the list of primes Sage has stored. The following are the curves that we have found and a discussion of their usefulness in security. Note that each time you run the program, your results may be different because points are picked at random from the set of points on the curve. We checked the curves where b is prime between $b = 2$ to $b = 20,000$. The equation for the elliptic curve we are using is

$$y^2 = x^3 - 3x + b \text{ over the field } \mathbb{Z}_p \text{ where}$$

$p = 3213876088517980551083924184682325205044405987565585670603103.$

We have found a number of curves that satisfy the necessary restrictions. We have summarized possible choices for b and P (generating a group of a well-behaving order over $\mathbb{Z}_p$) that we have found running our program for 14 hours in Appendix B. The code for finding these elliptic curves can be found in the Appendix A.

We have chosen the elliptic curve with $b = 5527$ as our best curve from the ones found above because the order of the elliptic curve group has the most complex factorization while still staying within the requirements set earlier in this chapter. The large factor of the order N is

$$l = 1507533122631831297281312207689323741312095553143.$$

Therefore we propose our new elliptic curve is $y^2 = x^3 - 3x + 5527$ over the field $\mathbb{Z}_{3213876088517980551083924184682325205044405987565585670603103}$ with the point

$P = (2475619237154351348941711661330269851359583809320067168435708,$

$2061556466751915919872761946030764831871366015842079315253 90)$

to be used for secure communications.

The following picture gives a visualization of the curve over $\mathbb{R}^2$. It is smooth and well defined for $x \in (-17.737138, \infty)$ in $\mathbb{R}^2$.

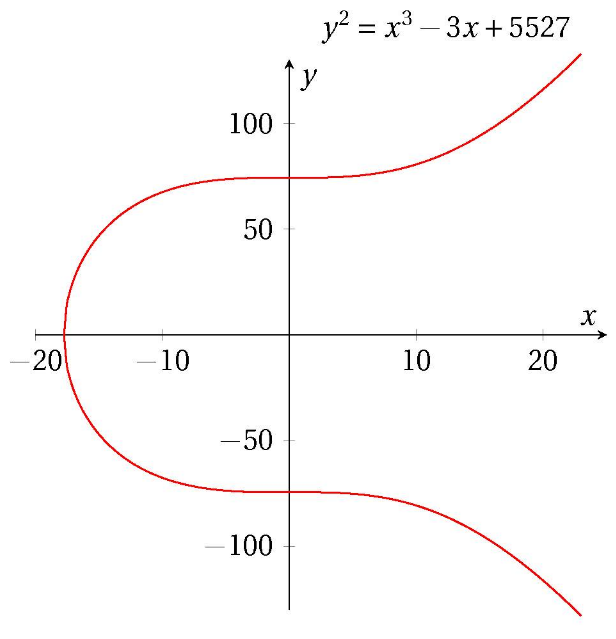


Figure 5.1: New Curve Graphed on $\mathbb{R}^2$

Note that over $\mathbb{Z}_p$ it appears as a collection of dense points. Due to the enormity of the numbers being used, attempts to graph this set only yield overflow errors. An example of the graph of an elliptic curve over a small finite field can be found in Chapter 1, Figure 1.3.

CHAPTER 6

CONCLUSIONS

We have clarified what properties an elliptic curve should satisfy to be useful for secure communications. We wrote a program that finds useful curves, generates appropriate points, and evaluates their security by finding largest factors for the groups and selecting a useful point.

We found a good elliptic curve that we may use in cryptography, with relatively reasonable integer parameters, hence computable in a short time. Our program may be used to find more curves that can be useful as well, this just requires more computational time. These curves may be applied to elliptic curve cryptography that is currently a base for most crypto interactions to date, and may be kept secret by the users. The difficulty is in generating large enough groups of points on the chosen curve with large prime factors of their orders, as this ensures additional security from known types of attacks.

There is a current list of useful, secure curves and we can add our elliptic curve to that list. Certicom sells and regulates the patents for these elliptic curves. We are currently looking into getting patents for our new elliptic curves. This thesis also gives a method for generating new secure curves enabling us to try to find even better curves and points over $\mathbb{Z}_p$ that provide even more efficient computational time and security. One may even consider looking at other algebraic varieties V that have group structures on them.

Sage Code

Appendix

A

```
In [1]: ###### THIS IS THE CODE THAT PRODUCES CURVES THAT PASS ALL THE TESTS ######



        #Set up curve files
        good = open('goodcurves.txt', 'w')
        bad = open('badcurves.txt', 'w')

        S = Primes(2^300)

        # Start primes for field, p, as the next prime after 2^200
        p = S.next(2^201)

        j=0 # index count on order large factors
        LFS = []      # Set of order large factors

        # Begin loop on different p's (for now this loop just runs on the one p,
        # however it can be made to run for a longer)
        while p == S.next(2^201):

            # Define the field
            F = GF(p)

            #Establish beginning b
            b=1
            # Begin loop on different b's in our EC
            while  b < 2^50:                              # < 2^200:

                # Make sure to progress b
                b = S.next(b)

                # The bulk of the program will run here!!!
                bad.write("\n")

                dis = -4*27+27*b^2
                # Throws out curve if the discriminant is 0
                if dis == 0:
                    bad.write("\n \n")
                    bad.write("Our field is ")
                    bad.write(str(p))
                    bad.write(" and our b is ")
                    bad.write(str(b))
                    bad.write("The discriminant is 0!")
                    continue

                # Define the Elliptic Curve
                E = EllipticCurve(F,[-3, b])
```

```
N = E.order()


# Throws out curve if it is supersingular
if E.is_supersingular() :
    bad.write("\n \n")
    bad.write(str(E))
    bad.write("\n Curve is supersingular!")
    continue

# Throws out an order that's computationally too small
if N <= 2^160:
    bad.write("\n \n")
    bad.write(str(E))
    bad.write("\n The elliptic curve group order is too small!")
    continue

# Throws out if the order  a large prime
if N in S:
    bad.write("\n \n")
    bad.write(str(E))
    bad.write("\n The elliptic curve group order is prime!")
    continue

# Finding prime factors of order N
D = divisors(N)
x = len(D)
l = D[x-2]




# Throws out if the order does not have a large prime
if l <= 2^160:
    bad.write("\n \n")
    bad.write(str(E))
    bad.write("\n The elliptic curve order does not have a large factor!")
    continue

# Number of points on the curve is M
M = E.cardinality()

# Throws out curve if the curve is anomalous
if M == p:
    bad.write("\n \n")
    bad.write(str(E))
    bad.write("\n Curve is anomolous!")
    continue
```

```
# Check that our order is a divisor of M
if M%N != 0:
    bad.write("\n \n")
    bad.write(str(E))
    bad.write("\n EC order is not a prime divisor of M!")
    continue


# The curve has passed the first series of tests
# and must now be checked for the Pohlig-Hellman and MOV attacks
else:
    s=1
    i = 0
    # Pick points and checks to see if l divides s
    while s%l != 0 or i==20:

        # Picks point
        P = E.random_element()

        # Determines order of point
        s = P.order()
        if s%l != 0:
            continue

        # Create Point P based on the requirements for Pohlig-Hellman Method
        m = int(s/l)
        P = m*P

        # Check that p^k != 1 mod m for small k
        k=1

        secure = true

        while k < 7 and secure:
            if p^k % m == 1:
                bad.write("\n \n")
                bad.write(str(E))
                bad.write("\n Point:")
                bad.write(str(P))
                bad.write("\n This point and curve are not
                 secure under the MOV attack!")
                secure = false
            else:
                if k == 6:
                    good.write("\n \n")

                    good.write("index number: ")
```

```
                                good.write(str(j))
                                good.write("\n")

                                good.write(str(E))
                                good.write("\n Point:")
                                good.write(str(P))
                                good.write("\n The large factor of the
                                 order used here is:")
                                good.write(str(l))
                                good.write("\nThis point and curves are secure
                                 under the MOV attack!")

                                LFS.append(l)
                                j = j+1

                            k=k+1
                    i=i+1



            # Make sure to progress p to the next prime
            p = S.next(p)

        bad.close()
        good.close()
        print "Finished"
```

In [2]:

```
###### THIS CODE FINDS OUT IF THE LARGE FACTORS ARE PRIME ######

primes = open('goodprimes.txt', 'w')
        nprimes = open('nonprimes.txt', 'w')


        i=0
        while i < len(LFS):

            if LFS[i] % 2 == 0:
                nprimes.write("\n")
                nprimes.write("Curve index: ")
                nprimes.write(str(i))
                i=i+1
                continue

            if LFS[i] % 3 == 0:
                nprimes.write("\n")
                nprimes.write("Curve index: ")
                nprimes.write(str(i))
            i=i+1
            continue

        if LFS[i] % 5 == 0:
            nprimes.write("\n")
            nprimes.write("Curve index: ")
            nprimes.write(str(i))
            i=i+1
            continue

        if LFS[i] % 11 == 0:
            nprimes.write("\n")
            nprimes.write("Curve index: ")
            nprimes.write(str(i))
            i=i+1
            continue

        if LFS[i] in S:
            primes.write("\n \n")
            primes.write("Curve index: ")
            primes.write(str(i))
            primes.write("\n Order: ")
            primes.write(str(LFS[i]))
            i=i+1
        else:
            nprimes.write("\n")
            nprimes.write("Curve index: ")
            nprimes.write(str(i))
            i=i+1

primes.close()
nprimes.close()
print "Finished"
```

APPENDIX B

TABLE OF SECURE ELLIPTIC CURVES

The following table summarizes possible choices for b and P (generating a group of a well-behaving order over $\mathbb{Z}_p$) that we have found running our program for 12 hours.

b	Point P
857	(3178588086911723544242454296504606116101050210724987221874964, 907956198569152150290046940783498501669922195530655379460958)
1283	2967154625500906619839186017326346389573031378676924739974761, 823774285742339083236540223859422724851023471819866396237 43)
1511	(2894360525755509241409613153896573920155592695994757446366768, 307442827015881509017073180815421923868206842911094067420567)
2251	(1245429110882716907039077583797792846417882732350635967874886, 1381981478677052427838387368196150605318621851392861256584341)
2347	(3125714564716729980589109022696096844031872148324486922319070, 2160055933963293507654549618316264061941542608726503350342138)
2417	(762608121924818757768874093030868930175990773135055540791562, 2640199891873252567769622504365720417061654086057674005902350)
2437	(2053039100121356198915329149448229814961066017670843460583300, 2610337719409883260067411807403650269269229168892279562185284)
4001	(1929439211694721520236131975896453620139529632231465967466729, 925257436973203193519713997377588072615835018901856660052821)

b	Point P
5527	(2475619237154351348941711661330269851359583809320067168435708, 206155646675191591987276194603076483187136601584207931525390)
7549	(326219443848737183970409758499070627821137422014050788026190, 2360686820493664174362142037674710986151338192895383770084274)
7621	(21367230997181428168838744361231643117577614074580702369913, 936100571395570713211432736761059602038168639906282969602611)
8423	(1179877346366415173927723556202579035924039838576971082504343, 2702417119217992291447159012248116603643556964128282097317502)
9677	(53414278254572422947478434631037065600824729551301450673870, 1875539777727734994560313089767977755806943394817924708212096)
12253	(342339405329874674324499496744694912492229050827687889848435, 283974993145053602385738234317088233092687857025552961378861)
12809	(1958139153402084855321090992423961718140167845177301286814565, 1294151137507414163332231654463455130812791287465565536804242)
12889	(6202861911197984427994240643553067894918895989639793937139577, 2892801266925618397688353679685564560216861073679879538032832)

b	Point P
13781	(20327119891968079581468896673793547054160436313166707044017 02, 29088205232810424774851055434441984863134743336736577744239 98)
13907	(10918412629625734219877392194887900880972653602951719155803 6, 93802347586959942611069939086401676428872506305327651225516 6)
14197	(25307742182252343378478198398507193996240929394262446463522 79, 26259597673595700483835718366167683728942293619419496735882 77)
15091	(11976438226482100275355036095690142280611334780137698987500 38, 15124347347868758947773240129245727249669388696543667002331 72)
16061	(29017654957783398848631162552343596124288395338405344659265 08, 22384522260111086587255737998863169094279424827909462812367)
16631	(20351720229693589598946019381454335727909040336358118795871 1, 13514385340425939837975406089661460409256590288843504931828 47)
19237	(14552113460642965950164596761206741451665704310384964776113 15, 27592334879544120940531492677834826610201337547116574787406 34)
19471	(25519537817601074326325074185339613015455683313787125296689 32, 17887741511664467978049523515688718292600251343352932205772 96)

List of Figures

BIBLIOGRAPHY

[1] *Elementary number theory.* http://doc.sagemath.org/html/en/constructions/number_theory.html. Accessed: 10.02.2018.

[2] D. J. BERNSTEIN AND T. LANGE, *Safecurves: choosing safe curves for elliptic-curve cryptography.* https://safecurves.cr.yp.to. Accessed: 10.02.2018.

[3] B. FINE AND G. ROSENBERGER, *Number Theory: An Introduction via the Density of Primes*, Springer International Publishing AG., 2 ed., 2016.

[4] F. HIVERT AND F. SALIOLA, *Tutorial: Programing in python and sage.* http://doc.sagemath.org/html/en/thematic_tutorials/tutorial-programming-python.html. Accessed: 10.02.2018.

[5] J. HOFFSTEIN, J. PIPHER, AND J. SILVERMAN, *Introduction to Mathematical Cryptography*, Springer-Verlag, 2010.

[6] F. LEPREVOST, J. MONNERAT, S. VARRETTE, AND S. VAUDENAY, *Generating anomalous elliptic curves*, Information Processing Letters, 93 (2005).

[7] A. SGHAIER, M. ZEGHID, AND M. MACHHOUT, *Differential fault attacks and countermeasures in elliptic curve cryptography*, International Journal of Computer Applications, 140 (2016), pp. 1–6.

[8] J. SILVERMAN, *The Arithmetic of Elliptic Curves*, Springer, 2 ed., 1986.

[9] M. L. SOMMERSETH, *Pohlig-hellman applied in elliptic curve cryptography*, 2015.

[10] W. STEIN, R. BRADSHAW, J. CREMONA, AND M. LENOX, *Elliptic curves over finite fields*. http://doc.sagemath.org/html/en/reference/curves/sage/schemes/elliptic_curves/ell_finite_field.html. Accessed: 10.02.2018.

[11] W. STEIN, R. BRADSHAW, J. CREMONA, M. LENOX, T. NAGEL, M. MARDAUS, AND D. HANSEN, *Points on elliptic curves*. http://doc.sagemath.org/html/en/reference/curves/sage/schemes/elliptic_curves/ell_point.html. Accessed: 10.02.2018.

[12] I. STEWART AND D. TALL, *Algebraic Number Theory and Fermat's Last Theorem*, A K Peters, 3 ed., 2002.

[13] L. C. WASHINGTON, *Elliptic Curves: Number Theory and Cryptography*, Chapman & Hall/CRC, 2008.

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author(s) retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author(s) or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Creating New Elliptic Curves for Uses in Cryptography
Title of Item

Elliptic Curves, Algebraic Geometry, Cryptography
3 to 5 keywords or phrases to describe the item

Mollie Zechlin
Author(s) Name (Print)

Mollie Zechlin — 4/12/19
Author(s) Signature — Date

This is a permitted, modified version of the Non-exclusive Distribution