

January 2016

An Elliptic Exploration

David Curtis Swart
Eastern Kentucky University

Follow this and additional works at: <https://encompass.eku.edu/etd>

 Part of the [Applied Mathematics Commons](#), and the [Defense and Security Studies Commons](#)

Recommended Citation

Swart, David Curtis, "An Elliptic Exploration" (2016). *Online Theses and Dissertations*. 433.
<https://encompass.eku.edu/etd/433>

This Open Access Thesis is brought to you for free and open access by the Student Scholarship at Encompass. It has been accepted for inclusion in Online Theses and Dissertations by an authorized administrator of Encompass. For more information, please contact Linda.Sizemore@eku.edu.

An Elliptic Exploration

By

David Swart

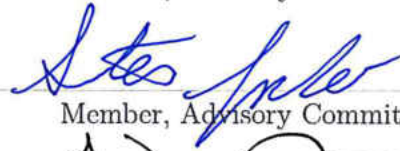
Thesis Approved:



Chair, Advisory Committee



Member, Advisory Committee



Member, Advisory Committee



Dean, Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for an M.A. degree at Eastern Kentucky University, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made. Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or, in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature *David Sweet*

Date 4/11/16

Elliptic Explorations

By

David Swart

Bachelor Of Arts, Mathematics

Berea College

Berea, Kentucky

2014

Submitted to the Faculty of the Graduate School of

Eastern Kentucky University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

May, 2016

Copyright ©David Swart, 2016
All rights reserved

DEDICATION

To all of my friends in the Math and Statistics Tutoring Center here at EKU.
You are all fantastic people, and I probably would have gone insane working
through this without you.

ACKNOWLEDGEMENTS

In addition to my fantastic friends at the Math and Statistics Tutoring Center, I would also like to thank the wonderful faculty of the ECU Math department for the great educational experience I have had here. I would like to give a special thanks to Dr. Pat Costello for working with me on this thesis, and to Dr. Szabo and Dr. Xu for serving on my committee and for all of their helpful feedback and advice.

ABSTRACT

In this paper I will be giving an introduction to an interesting kind of equation called elliptic curves, and how they can be used to protect our national security through Cryptology. We will explore the unique operation for adding points on elliptic curves and the group structure that it creates, as well as the ECC method, which stands among the RSA and AES methods as one of the modern day's most secure systems of cryptography. In addition, I will also introduce several algorithms and methods that are useful for working with ECC such as Schoof's Algorithm, and I will also provide examples of working with these algorithms. At the end of this paper, I will also demonstrate the ECC encryption and decryption process.

TABLE OF CONTENTS

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 2 | Working With Elliptic Curves | 6 |
| 2.1 | Adding Points on Elliptic Curves | 6 |
| 2.2 | The Group Axioms | 11 |
| 2.3 | A Note About the j -Invariant | 13 |
| 3 | Application to Cryptology | 14 |
| 3.1 | Hasse's Theorem | 14 |
| 3.2 | Using Schoof's Algorithm to Find $\#E(F)$ | 15 |
| 3.3 | Baby Step, Giant Step | 19 |
| 4 | Our Algorithms | 22 |
| 4.1 | Elliptic Curve Generation Algorithm | 22 |
| 4.2 | P Through kP Algorithm | 24 |
| 4.3 | kP Algorithm | 27 |
| 5 | Elliptic Curve Cryptography | 31 |
| 5.1 | Koblitz Method | 31 |
| 5.2 | The ECC System | 32 |
| 5.3 | Sample Encryption and Decryption | 33 |
| | Bibliography | 37 |

Chapter 1

Introduction

Simply put, an elliptic curve is represented by an equation where y^2 is set equal to a cubic polynomial in x . One example of a practical application is given in the introduction of Lawrence E. Washington's *Elliptic Curves: Number Theory and Cryptography*, such as the pyramid of cannonballs problem. The pyramid of cannonballs problem asks: if we have a collection of cannonballs piled in the shape of a square pyramid with x layers, where the number of cannonballs on the i th layer from the top is i^2 , is it possible to rearrange the balls into a square array if the pyramid were to collapse? [4, 1] Figure 1.1 shows an example of a pyramid of cannonballs with a 4×4 base.

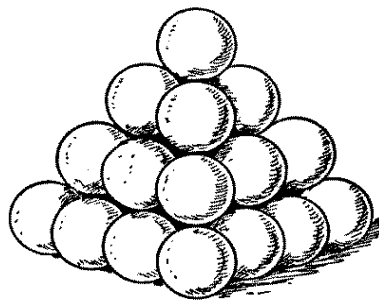


Figure 1.1: Pyramid of Cannonballs

Using an inductive proof, we can show that, for a square pyramid of height n , the total number of cannonballs can be found with the equality

$$1^2 + 2^2 + \cdots + x^2 = \frac{x(x+1)(2x+1)}{6}.$$

Proof. The case where $x = 1$ is easy to see, since the fraction on the right becomes

$$\frac{1(1+1)(2+1)}{6} = \frac{6}{6} = 1,$$

and $1^2 = 1$.

Now we suppose that the equality holds for a pyramid of height x . Starting on the left side of the equality,

$$1^2 + 2^2 + \cdots + x^2 + (x+1)^2 = \frac{x(x+1)(2x+1)}{6} + (x+1)^2$$

by the inductive hypothesis. Multiplying $(x+1)^2$ by $\frac{6}{6}$, we now have

$$\frac{x(x+1)(2x+1)}{6} + (x+1)^2 = \frac{x(x+1)(2x+1) + 6(x+1)^2}{6}.$$

Multiplying everything out gives us

$$\frac{2x^3 + 9x^2 + 13x + 6}{6},$$

which factors into

$$\frac{(x+1)((x+1)+1)(2(x+1)+1)}{6}.$$

Therefore,

$$1^2 + 2^2 + \cdots + x^2 = \frac{x(x+1)(2x+1)}{6}$$

by mathematical induction. □

Once this pyramid collapses, we want to be able to arrange it into a perfect square, so the total number of cannonballs must be a perfect square. This gives us the equation

$$y^2 = \frac{x(x+1)(2x+1)}{6}.$$

This equation matches the type described at the start of this introduction, so we are dealing with an elliptic curve.

When used in cryptology, we often define these elliptic curves over finite fields and focus on the integral points; points where the x and y coordinates are both integers. Finite fields, as the name suggests, are fields of finite order. An interesting fact that will be relevant throughout this paper is that any finite field has order equal to a prime-power. Further, there is only one unique field for each prime-power order up to isomorphism. [4, 408] This paper will focus on fields of prime order, since they tend to behave more nicely when used in cryptosystems. Graphically, elliptic curves look like the following image. Depending on the function, the shape can vary, but ultimately they share the same major qualities: symmetry about the x -axis, and y approaches ∞ as x approaches ∞ . An example of an elliptic curve graphed on a cartesian coordinate plane is given in Figure 1.2.

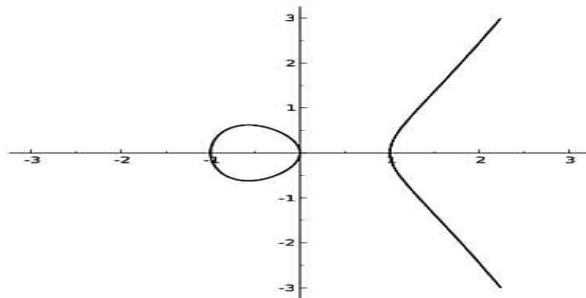


Figure 1.2: Sample Elliptic Curve

There are two common forms of elliptic curve equations. The first and simpler of the two is the Weierstrass equation for an elliptic curve,

$$y^2 = x^3 + Ax + B,$$

where A and B are constants from the field over which the elliptic curve is defined. The other form is the Generalized Weierstrass equation for an elliptic curve,

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

The generalized equation is more flexible and is more efficient for use in fields of characteristic 2 or 3. For fields of any other characteristic, we can simply complete the square and perform a change of variables to switch from the generalized equation to the standard Weierstrass equation. [2, 179-182] Fields of characteristic 2 or 3 aren't very useful for building cryptosystems, so we won't spend much time working with them. As a result, we won't need to worry about the generalized equation.

The last important detail to note is that, in addition to all of the points on the curve, we also include an extra point at (∞, ∞) , which will be denoted simply as ∞ for the sake of convenience. [1, 55] ∞ exists at the very top of the y -axis, but it also exists at the bottom, both of them being the same point. To imagine this, picture the coordinate plane drawn on the face of a globe where the y -axis is the Prime Meridian and the x -axis is the Equator. The y -axis and all other vertical lines travel up through the North Pole, loop around the other side of the globe, through the South Pole, and back up to their point of origin. ∞ , rather than being at one or both of the poles, is somewhere on the opposite side of the globe from the observer. A line will pass through the point at ∞ if and only if it is a vertical line. Once we start working with finite fields, there may not be a meaningful ordering to the points on the curve, so the placement of ∞ becomes

irrelevant. The point at ∞ serves as a formal symbol and will have several useful implications in the sections to come.

Chapter 2

Working With Elliptic Curves

2.1 Adding Points on Elliptic Curves

As stated in the previous section, we will focus on working with the integral points on elliptic curves as defined over finite fields. To do this, we perform addition with integer moduli, but first we need to define how exactly we will be adding these points together. This addition is very nonstandard, but it will give us a very nice Abelian group structure to work with. A visual aid can help when dealing with a general elliptic curve, but once we start working with a modulus, attempting to create a visual interpretation proves to be futile. [4, 15]

First, we define our two points, $P_1 = (x_1, y_1)$, and $P_2 = (x_2, y_2)$. For this first case, we suppose that $x_1 \neq x_2$. We draw the line that connects these two points, and our goal is to find the third point where this line intersects the elliptic curve. To do this, we need to calculate the slope of the line,

$$m = \frac{y_2 - y_1}{x_2 - x_1}.$$

Writing the equation of the line in point-slope form gives

$$y = m(x - x_1) + y_1.$$

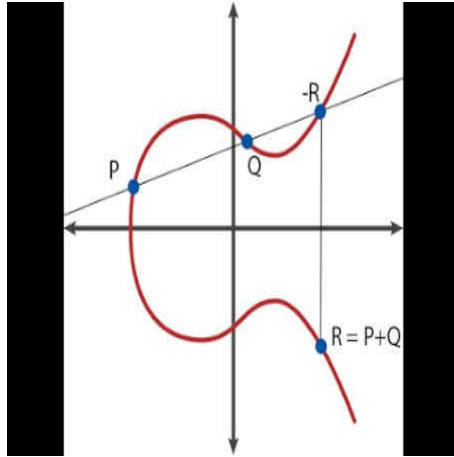


Figure 2.1: Adding Two Different Points on an Elliptic Curve

To find the point of intersection, we substitute the equation of this line into our elliptic curve equation for y ,

$$(m(x - x_1) + y_1)^2 = x^3 + Ax + B$$

Which can be rewritten as

$$0 = x^3 - m^2x^2 + \dots$$

Solving for the roots of this cubic would be extremely difficult, but we already have two roots, x_1 and x_2 , and a very helpful equation. For a cubic polynomial $x^3 + ax^2 + bx + c$ with roots r, s, t ,

$$x^3 + ax^2 + bx + c = (x - r)(x - s)(x - t) = x^3 - (r + s + t)x^2 + \dots$$

This tells us that for any cubic polynomial of the form $x^3 + ax^2 + bx + c$,

$$r + s + t = -a.$$

Given the roots r and s , we can find $t = -a - r - s$. From the above expansion of our elliptic curve equation, $a = -m^2$, and we already have two roots, so we can

find our third root, x_3 ,

$$x_3 = m^2 - x_1 - x_2,$$

And the corresponding y -value,

$$y = m(x_3 - x_1) + y_1.$$

We add one final step to our addition by reflecting the point across the x -axis, giving us the point $P_3 = (x_3, y_3)$, where

$$x_3 = m^2 - x_1 - x_2, \quad y_3 = m(x_1 - x_3) - y_1.$$

This last step may seem strange, but that reflection has an important implication: it causes this addition of points to have a group structure, which will be demonstrated in the next section. [3, 13]

So what about the case where $x_1 = x_2$? Since the x -coordinates are the same, the line connecting the two points is a vertical line. As defined previously, all vertical lines pass through the point at ∞ . Since ∞ is at the top and bottom of the y -axis simultaneously, reflecting ∞ across the x -axis yields ∞ . So for $P_1 = (x, y_1)$ and $P_2 = (x, y_2)$, $P_1 + P_2 = \infty$.

When $P_1 = P_2$, the line passing through them is the line tangent to the elliptic curve at point P_1 .

Using implicit differentiation on $y^2 = x^3 + Ax + B$, we get

$$2y * \frac{dy}{dx} = 3x^2 + A,$$

and

$$m = \frac{dy}{dx} = \frac{3x^2 + A}{2y}.$$

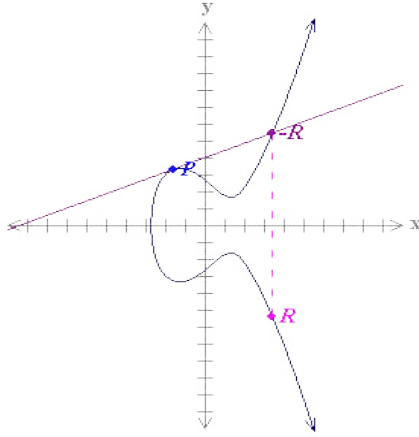


Figure 2.2: Doubling a Point on an Elliptic Curve

When $y = 0$, the tangent line is vertical, so $P_1 + P_2$, or $2P = \infty$. Otherwise, we use point slope form to write the equation of the tangent line as

$$y = m(x - x_1) + y_1.$$

Like in the earlier case, when we plug this into our elliptic curve equation we have a cubic polynomial that can be rearranged into the form

$$0 = x^3 - m^2x^2 + \dots .$$

Using the fact that the sum of the roots is equal to m^2 like in the earlier case, and the fact that the root x_1 is a double root, we can calculate

$$x_3 = m^2 - 2x_1, \quad y_3 = m(x_1 - x_3) - y_1.$$

Notice that we negate the y -coordinate, just like in the other case. [4, 13]

Lastly, consider the case where $P = (x, y)$, and we want to calculate $P + \infty$. Only vertical lines pass through the point at infinity, so the line connecting the two points is vertical and goes straight up through ∞ , loops back around from the bottom, and intercepts the elliptic curve at $(x, -y)$. Reflecting across the x -axis

brings us back to P . So for all P on the elliptic curve,

$$P + \infty = P.$$

Now that we have defined our operation, we can show that it forms an Abelian group, which has some interesting applications both in and out of Cryptography.

In summary, addition of points on an elliptic curve E defined by $y^2 = x^3 + Ax^2 + B$, where $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, and $P_3 = P_1 + P_2 = (x_3, y_3)$ is defined as follows.

1. If $x_1 \neq x_2$, then

$$x_3 = m^2 - x_1 - x_2, \quad y_3 = m(x_1 - x_3) - y_1,$$

where

$$m = \frac{y_2 - y_1}{x_2 - x_1}.$$

2. If $x_1 = x_2$, but $y_1 \neq y_2$, then $P_1 + P_2 = \infty$.
3. If $P_1 = P_2$, and $y_1 \neq 0$, then

$$x_3 = m^2 - 2x_1, \quad y_3 = m(x_1 - x_3) - y_1,$$

where

$$m = \frac{3x_1^2 + A}{2y_1}.$$

4. If $P_1 = P_2$ and $y_1 = 0$, then $P_1 + P_2 = \infty$.
5. $P + \infty = P$ for all P on E .

Given an integer k and point P , $kP = P + P + \cdots + P$, where there are k P s in the sum.

Example: Suppose that we are working with the curve $y^2 = x^3 + 3x + 2 \pmod{11}$, $P_1 = (2, 4)$, and $P_2 = (3, 7)$. Since $P_1 \neq P_2$ and $x_1 \neq x_2$, we calculate $P_1 + P_2$ using method 1. from the above list.

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{7 - 4}{3 - 2} = 3.$$

$$x_3 = m^2 - x_1 - x_2 = 9 - 2 - 3 = 4.$$

$$y_3 = -(m(x_1 - x_3) - y_1) = -(3(2 - 4) - 4) = -(3(-2) - 4) = -(-6 - 4) = -(-10) = 10.$$

So $P_1 + P_2 = (4, 10)$.

2.2 The Group Axioms

Let E be an elliptic curve over a finite field F plus the point at ∞ , where the characteristic of F is greater than 2. We denote $E(F)$ as the group of all points on E over F . Because of the way addition of points on elliptic curves is defined, the following group axioms are satisfied in $E(F)$:

1. (Closure) If P_1 and P_2 are on E , then $P_1 + P_2$ is also on E .
2. (Additive Identity) $P + \infty = P$ for all P on E .
3. (Additive Inverses) If P is on E , then there exists a $-P$ on E with $P + (-P) = \infty$.
4. (Associativity) For all P_1, P_2 , and P_3 on E , $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$.
5. (Commutativity) For all P_1 and P_2 on E , $P_1 + P_2 = P_2 + P_1$. [3, 15]

For the case where $x_1 = x_2$, it is clear that $P_1 + P_2 = \infty$ is an element of $E(F)$ since we appended ∞ to E , and by extension $E(F)$. Similarly, when doubling P and when $y = 0$, $2P = \infty$ is also an element of $E(F)$. When both P_1 and P_2 are not ∞ , then $P_1 + P_2 = (x_3, y_3)$, where $x_3 = m^2 - x_1 - x_2$. This works

for the case where $P_1 = P_2$, and the case where $P_1 \neq P_2$ and $x_1 \neq x_2$. Since the coordinates x_1, x_2, y_1, y_2 are elements of F , $m = \frac{y_2 - y_1}{x_2 - x_1}$ is an element of F . When $P_1 = P_2$, $m = \frac{3x_1^2 + A}{2y_1}$ is still an element of F since A is also an element of F . Thus $m^2 - x_1 - x_2$ is an element of F , therefore x_3 is also an element of F . Since x_3 is an element of F , it follows that $y_3 = m(x_1 - x_3) - y_1$ is also an element of F . Since x_3 and y_3 are both in F and (x_3, y_3) is on E as demonstrated in the previous section, it follows that $P_1 + P_2$ is an element of $E(F)$, and so $E(F)$ is closed under this addition.

Since we defined $P + \infty = P$ for all P in $E(F)$, it is clear that the additive identity holds.

If $P = (x, y)$, let $-P = (x, -y)$. By the definition of our addition, $P + (-P) = \infty$, so the property of additive inverses holds.

The proof of commutivity is a bit cumbersome for this paper and can be found in Lawrence Washington's book on elliptic curves. [4, 31-32]

Since the addition satisfies all of these axioms, $E(F)$ is a group. To be an Abelian group, if P_1, P_2 are in $E(F)$, then $P_1 + P_2 = P_2 + P_1$, which should be clear if we look at E rather than narrowing our focus to the finite field we want to define it over. Notice that the line through P_1 and P_2 is the same as the line through P_2 and P_1 , all of the calculations will still work out the same regardless of ordering. Therefore, $E(F)$ is an Abelian group under the addition of points on elliptic curves. An important theorem about the structure of the $E(F)$ states that, no matter what E or F may be, $E(F)$ will always be either cyclic, or the direct product of a cyclic group and \mathbb{Z}_2 . [2, 194]

2.3 A Note About the j -Invariant

There is a whole lot that could be said about twists and the concept of the j -invariant, but they are not very important for this paper. The j -invariant of E is defined as

$$j = j(E) = 1728 \frac{4A^3}{4A^3 + 27B^2}. [4, 42]$$

All we need to take from this for this paper is that, given an elliptic curve $y^2 = x^3 + Ax + B$ over a finite field modulo a prime p , if

$$4A^3 + 27B^2 \equiv 0 \pmod{p},$$

the set created by the points on the curve does not necessarily behave according to our Abelian group structure, since the denominator of the j -invariant would be 0. To get around this, we simply avoid elliptic curves that satisfy this condition for the entirety of this paper.

Chapter 3

Application to Cryptology

3.1 Hasse's Theorem

AES, RSA, and El Gamal are some of the bigger commercial cryptosystems currently in use, but elliptic curves can be used as an effective basis for encryption as well. To set this system up, we need an elliptic curve E and a finite field F where the group $E(F)$ has many points. Finding such a curve is much more complicated than it might sound, especially when working with finite fields of very large order. For this paper, we will focus on fields of the form \mathbb{Z}_p , where p is a prime.

So how do we find an elliptic curve that has many points with coordinates in a given finite field? Hasse's Theorem gives us one method.

Theorem 3.1 (Hasse's theorem). *Let E be an elliptic curve over the finite field \mathbb{Z}_p . Then the order of $E(\mathbb{Z}_p)$ satisfies*

$$|p + 1 - \#E(\mathbb{Z}_p)| \leq 2\sqrt{p},$$

where $\#E(\mathbb{Z}_p)$ is the order of $E(\mathbb{Z}_p)$. [2, 197]

Hasse's Theorem gives us a range of $4\sqrt{p}$ that the order of $E(\mathbb{Z}_p)$ can fall within. This method also requires that we know a point in $E(\mathbb{Z}_p)$ with order greater than $4\sqrt{p}$. By a corollary to Lagrange's theorem, we know that the order of any element in a group must divide the order of the group. If the order of this point is greater than $4\sqrt{q}$, then there will be exactly one multiple of that element's order within the range given by Hasse's theorem. That multiple must be the order of $E(\mathbb{Z}_p)$. This is important when applied to Cryptology since, in order to build a secure cryptosystem based on an elliptic curve E over \mathbb{Z}_p , $E(\mathbb{Z}_p)$ must have a very large order, and we can use Hasse's theorem to filter out groups that have insufficient order.

Now we have a way to find $\#E(\mathbb{Z}_p)$, but another question has arisen: how can we accurately find the order of an element in the first place? One effective method is the Baby Step, Giant Step algorithm, which will be discussed in section 3.3.

3.2 Using Schoof's Algorithm to Find $\#E(F)$

Another method that we can use to find $\#E(F)$ is Schoof's Algorithm. Schoof created this method in 1985, and its speed was unmatched for elliptic curves over large fields \mathbb{Z}_p . [3] The algorithm proved too difficult for us to replicate using available programming, so instead we have provided an example that illustrates the processes used in the algorithm.

Consider the elliptic curve E to be defined as $y^2 = x^3 + 3x + 9 \pmod{23}$. Thanks to Hasse's Theorem, we know that

$$\#E(\mathbb{Z}_{23}) = 23 + 1 - a,$$

where $|a| \leq 2\sqrt{p}$. Ultimately, our goal is to find this mystery number a . For the first step of Schoof's algorithm, we want to choose a set of primes $S = \{l_i\}$

where $\prod_{i \in S} l_i > 4\sqrt{p}$, and $l_i \neq p$ for all i . For this example, we let $S = \{2, 3, 11\}$. $2*3*11 = 66 > 4\sqrt{23}$, so this set is acceptable. Ultimately, our goal is to use these primes as moduli in the Chinese Remainder Theorem in order to solve for a . Since the focus of this paper is going to be on a cryptological application of Schoof's algorithm rather than the nitty gritty details of the algorithm itself, explanations of many of the algorithm's more technical aspects will be omitted for the sake of brevity. The interested reader can find a very thorough and detailed breakdown in Washington's book. [4, 113-118]

For $l = 2$, we do a different procedure from the one we use on the other primes in S . Since we are working $\pmod{2}$, there are only two possible values for a : 0 and 1. To figure out which one it is, we check to see whether or not $x^3 + 3x + 9$ has a root e in F_p . If $e \in \mathbb{Z}_p$, then $(e, 0) \in E(\mathbb{Z}_p)$ and $(e, 0)$ has even order, meaning that $E(\mathbb{Z}_p)$ has even order. This means that $p + 1 - a \equiv 0 \pmod{2}$, so a must be even. If $x^3 + 3x + 9$ does not have a root in \mathbb{Z}_p , then there are no points of order 2, therefore a must be odd. [4, 114]

To determine whether or not this root exists, we use a fact from Washington's work stating that the roots of $x^p - x$ are exactly the elements of \mathbb{Z}_p . [3, 114] This means that, in our example if $\gcd(x^3 + 3x + 9, x^{23} - x) = 1$, then $x^3 + 3x + 9$ does not have any roots in \mathbb{Z}_{23} . We could easily calculate this gcd with Mathematica, but we can also make x^{23} more functional and easier to work with by computing $x_{23} \equiv x^{23} \pmod{x^3 + 3x + 9}$. This is even easier to compute with Mathematica, and we can use this form for the other primes in the algorithm. Using Mathematica,

$$\text{PolynomialMod}[x_{23} - x, x^3 + 3x + 9, \text{Modulus} \rightarrow 23] = x + 21.$$

Therefore, $x^3 + 3x + 9$ has a root in \mathbb{Z}_{23} , so $a \equiv 0 \pmod{2}$.

Now we move on to the next prime in S , which in this example is 3. The following procedure is used on the remaining primes in S . We compute $q_3 \equiv 23 \pmod{3}$, and the result is $q_3 \equiv 2 \pmod{3}$. Next, we compute the x -coordinate x' of

$$(x', y') = (x^{p^2}, y^{p^2}) + q_l(x, y) \pmod{\psi_l},$$

where ψ_l is the l th division polynomial. This paper will not go into the details on division polynomials, but Washington's book has an entire section dedicated to them for the interested reader. [4, 76-81] $23^2 = 529$, so the equation for this case would be

$$(x', y') = (x^{529}, y^{529}) + 2(x, y) \pmod{\psi_3}.$$

Using the fact that $2 \equiv -1 \pmod{3}$, we can compute $-(x, y) = (x, -y)$ instead of doubling the point. Using the formula for adding points on an elliptic curve, we calculate x' ,

$$x' = \left(\frac{y^{529} + y}{x^{529} - x} \right)^2 - x^{529} - x.$$

Using the fact that $y^2 = x^3 + 3x + 9$, we can rewrite the above equation as

$$x' = (x^3 + 3x + 9) \left(\frac{(x^3 + 3x + 9)^{264} + 1}{x^{529} - x} \right)^2 - x^{529} - x.$$

This would be an extremely difficult mess of polynomials to simplify, but Washington provides us with an interesting trick for dealing with it. We know that $\psi_3 = 3x^4 + 6Ax^2 + 12Bx - A^2 = 3x^4 + 18x^2 + 108x - 9$ which reduces to $3x^4 + 18x^2 + 16x + 14 \pmod{23}$, so with the help of Mathematica we can compute $\gcd(x^{529} - x, \psi_3) = \psi_3$. Factoring ψ_3 , we can see that $x = 6$ is a root of ψ_3 , and our program for computing k multiples of P verifies that the points $(6, 6)$ and $(6, 17)$ in $E(\mathbb{Z}_{23})$ have order 3, so

$$\#E(\mathbb{Z}_{23}) = 23 + 1 - a \equiv 0 \pmod{3},$$

therefore $a \equiv 0 \pmod{3}$.

The last prime in our list S is 11. First, we compute $q_{11} \equiv 23 \pmod{11} \equiv 1 \pmod{11}$. As in the previous case, our goal is to compute the x -coordinate of

$$(x', y') = (x^{529}, y^{529}) + q_l(x, y) \pmod{\psi_l} = (x^{529}, y^{529}) + (x, y) \pmod{\psi_{11}}.$$

Using the formula for adding points on elliptic curves,

$$x' = \left(\frac{y^{529} - y}{x^{529} - x} \right)^2 - x^{529} - x.$$

Again, we have an extremely nasty polynomial on our hands, and $\gcd(x^{529} - x, \psi_{11}) = 1$, so we cannot use the same trick that we used during the last step. However, for this example we can get away with avoiding working with 11 all together.

We know that $a \equiv 0 \pmod{2}$ and $a \equiv 0 \pmod{3}$, therefore, $a \equiv 0 \pmod{6}$. This means that a could be any multiple of 6, but we also know that $|a| \leq 2\sqrt{23} = 9.59$ approximately. -6 and 6 are the only possible values of a that satisfy this condition, so it must be true that $a = 6$ or $a = -6$. This means that the order of the group is either 18 or 30. If we can find an element of order 9, then the order of the group must be 18 and $a = 6$, and if we can find an element of order 10, then the order of the group must be 30 and $a = -6$. Using the program provided by Kyle Martin at Wolfram, we can compute the 10th division polynomial in Mathematica. Through Mathematica, we also find that $x = 1$ is a root of ψ_{10} . Using our second program for computing $2P$ through kP , we can verify that the points

$(1, 6)$ and $(1, 17)$ in $E(\mathbb{Z}_{23})$ have order 10, so $\#E(\mathbb{Z}_{23}) = 30$, so $a = -6$. With the help of our first program, we can verify that $\#E(\mathbb{Z}_{23})$ is in fact 30.

Schoof's Algorithm has a few more technical intricacies that did not come up due to the simplicity of this example, but this was only meant to serve as a testament to what the algorithm is capable of. Due to the level of complexity in the program, Schoof's Algorithm is not very effective at finding the orders of curves large enough for commercial use, but it is nevertheless a powerful procedure if properly programmed. Unfortunately, at my current level of programming experience and with my current resources, we were unable to build a functional replica of the algorithm. The interested reader can find more information in just about all of the resources cited in this paper.

3.3 Baby Step, Giant Step

Let P be a point in $E(\mathbb{Z}_p)$, and we want to know the order of P . First, we find any integer k such that $kP = \infty$. Let $\#E(\mathbb{Z}_p) = N$. By Lagrange's theorem, $NP = \infty$. By Hasse's theorem, we know that $p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}$. The algorithm goes as follows.

1. Compute $Q = (p + 1)P$.
2. Choose an integer m with $m > p^{\frac{1}{4}}$. Compute and store the points jP for $j = 1, 2, \dots, m$.
3. Compute the points $Q + k(2mP)$ for $k = -m, -(m - 1), \dots, m$ until $Q + k(2mP) = \pm jP$ for one of the jP s on the stored list.
4. Since $Q + k(2mP) = \pm jP$, $Q + k(2mP) \mp jP = \infty$, which can be rewritten as $(p + 1 + 2mk \mp j)P = \infty$. Let $M = p + 1 + 2mk \mp j$.
5. Factor M . Let p_1, p_2, \dots, p_r be the distinct prime factors of M .

6. Compute $(\frac{M}{p_i})P$ for $i = 1, \dots, r$. If $(\frac{M}{p_i})P = \infty$ for some i , replace M with $(\frac{M}{p_i})$ and go back to step 5. If $(\frac{M}{p_i}) \neq \infty$ for all i , then M is the order of P .
[2, 348]

So to find $\#E(\mathbb{Z}_p)$, first we use Hasse's theorem to find the range that $\#E(\mathbb{Z}_p)$ could fall within. Next, we choose points that we know are in $E(\mathbb{Z}_p)$ and use the Baby Step, Giant Step algorithm to compute their orders, until we find a point with order greater than $4\sqrt{p}$. Finally, we find the multiple of that order that falls within the range given by Hasse's theorem, and that number is $\#E(\mathbb{Z}_p)$.

Example: Suppose that we are working with the elliptic curve $y^2 = x^3 + 25x + 25 \pmod{53}$, and we want to know the order of the point $(0, 5)$. Note that all of the computations in this example were done using one of our programs, which will be detailed later in this paper. First, we compute $(p + 1)P = 54P$, which can be shown to equal $(46, 39)$.

Next, we choose an $m > 53^{\frac{1}{4}} = 2.698$, so for simplicity we can just use $m = 3$. Now we have to compute jP for $j = 1$ to $j = 3$. $P = (0, 5)$, and it can be shown that $2P = (46, 39)$ and $3P = (36, 26)$.

For the next step, we compute $Q + k(2mP)$ for $k = -m, -(m - 1), \dots, m$ until we get one of the jP 's computed in the previous step. As you may have noticed already, $2P = 54P = (46, 39)$, so let us just cut to the chase and use $k = 0$. This means that $(p + 1 + 2mk - j)P = (53 + 1 + 0 - 2)P = 52P = \infty$, so we let $M = 52$, which factors into $2^2 * 13$.

Next we calculate $\frac{M}{p_i}P$ for $i = 1, 2$, where the p_i are the unique prime factors of $M = 52$. It can be shown that $26P = (18, 0)$ and $4P = (42, 50)$. Since $26P$ and $4P$ are not the identity, $M = 52$ must be the order of $P = (0, 5)$.

The Baby Step, Giant Step algorithm could also be used to find $\#E(\mathbb{Z}_p)$. To do this, run through the algorithm with random points in $E(\mathbb{Z}_p)$ until the least common multiple of the orders of those points only divides one number within our $4\sqrt{q}$ range, and that number will be the order of $E(\mathbb{Z}_p)$. [3, 103]

Chapter 4

Our Algorithms

4.1 Elliptic Curve Generation Algorithm

We have come up with Mathematica programs for carrying out several essential operations for Elliptic Curve Cryptography.

The first program enables us to find the order of groups generated by elliptic curves over finite fields of a given prime order, and it also lists all of the points in the groups. This program can run through several elliptic curves at once for a given field, but it is admittedly inefficient. It is nowhere near fast enough to handle the numbers needed for commercial use, but it has been useful as a source of examples and basic data for us to use and check for patterns. Also included is a sample output from the algorithm.

```
cnts= {};  
p = 11 ;  
For[a = 3, a < 4 , a ++,  
  For[b = 2, b < 3 , b ++,  
    If[Unequal[Mod[4a3 + 27b2, p], 0],  
      Print["curve is y2 = x3 +", a, "x + ", b, " MOD ", p];
```

```

cnt= 0;
For[x = 0, x < p, x ++,
z =Mod[x3 + ax + b, p];
For[y = 0, y < p, y ++,
If[Mod[y2 - z, p] == 0, cnt++];
Print["point is x = ",x," y = ",y]]];
Print["# of points =", cnt+1];
AppendTo[cnts, cnt +1]];
Print[Max[cnts]]]

```

Curve is $y^2 + x^3 + 3x + 2 \pmod{11}$.

point is $x = 2 \ y = 4$

point is $x = 2 \ y = 7$

point is $x = 3 \ y = 4$

point is $x = 3 \ y = 7$

point is $x = 4 \ y = 1$

point is $x = 4 \ y = 10$

point is $x = 6 \ y = 4$

point is $x = 6 \ y = 7$

point is $x = 7 \ y = 5$

point is $x = 7 \ y = 6$

point is $x = 10 \ y = 3$

point is $x = 10 \ y = 8$

of points = 13

13

First we define a prime p to serve as the order of the finite field we are defining the elliptic curves over. The first two For loops are where we define the coefficients A and B (from the Weierstrass equation $y^2 = x^3 + Ax + B$), and this is where we

set how many curves we want the program to check by setting an upper bound for A and B . The first If loop is where the algorithm filters out any curves where we would run into issues with the j-invariant ($4a^3 + 27b^2 \equiv 0 \pmod{p}$). After deciding that an equation will not be filtered out, the algorithm prints the equation for the sake of organization, and then the next three loops check for pairs of x and y values that satisfy the equation with the given modulus and then prints the points one-by-one. One final step was added on to state the order of the group, adding one to the total number of points found in order to account for the point at ∞ .

4.2 P Through kP Algorithm

We built the second program to calculate kP for some starting point P and some integer k . This algorithm is not as efficient as our third program for calculating a specific high-value kP , but what this addition program has over our other addition program is that it also prints all intermediate values between P and kP . This is useful because it can easily be used to determine whether or not P is a generator for $E(\mathbb{Z}_p)$ and it allows us to find the exact order of the point without having to resort to trial and error with the other algorithm. The sample output here is the output used to find the multiples of $P = (0, 5)$ in the example of the Baby Step, Giant Step algorithm earlier in this paper.

$x = 0$;

$y = 5$;

$p = 53$;

$a = 25$;

$b = 25$;

Print["curve is $y^2 = x^3 +$ ", a , " $x +$ ", b , " MOD ", p];

$c = \text{Mod}[-2 * x + ((3 * x^2 + a) * \text{PowerMod}[2y, -1, p])^2, p]$;

$d = \text{Mod}[-y + ((3 * x^2 + a) * \text{PowerMod}[2y, -1, p]) * (x - c), p]$;

```

Print["2P = (", c, ", ", "d, ")"];
For[k = 3, k ≤ 52, k++, If[c == x, Print[k, " P = ∞"],
g = Mod[((y-d)*PowerMod[x-c, -1, p])2-c-x, p]; h = Mod[(y-d)*PowerMod[x-
c, -1, p] * (c - g) - d, p]; Print[k, " P = (", g, ", ", h, ")"];
c = g;
d = h;]]

```

curve is $y^2 = x^3 + 25x + 25 \pmod{53}$ $2P = (46, 39)$

$$3P = (36, 26)$$

$$4P = (42, 50)$$

$$5P = (40, 43)$$

$$6P = (9, 5)$$

$$7P = (44, 48)$$

$$8P = (5, 13)$$

$$9P = (23, 43)$$

$$10P = (17, 13)$$

$$11P = (21, 35)$$

$$12P = (7, 38)$$

$$13P = (52, 30)$$

$$14P = (43, 10)$$

$$15P = (50, 20)$$

$$16P = (28, 29)$$

$$17P = (16, 4)$$

$$18P = (31, 40)$$

$$19P = (38, 41)$$

$$20P = (25, 2)$$

$$21P = (32, 37)$$

$$22P = (22, 26)$$

$$23P = (41, 45)$$

$$24P = (29, 21)$$

$$25P = (48, 27)$$

$$26P = (18, 0)$$

$$27P = (48, 26)$$

$$28P = (29, 32)$$

$$29P = (41, 8)$$

$$30P = (22, 27)$$

$$31P = (32, 16)$$

$$32P = (25, 51)$$

$$33P = (38, 12)$$

$$34P = (31, 13)$$

$$35P = (16, 49)$$

$$36P = (28, 24)$$

$$37P = (50, 33)$$

$$38P = (43, 43)$$

$$39P = (52, 23)$$

$$40P = (7, 15)$$

$$41P = (21, 18)$$

$$42P = (17, 40)$$

$$43P = (23, 10)$$

$$44P = (5, 40)$$

$$45P = (44, 5)$$

$$46P = (9, 48)$$

$$47P = (40, 10)$$

$$48P = (42, 3)$$

$$49P = (36, 27)$$

$$50P = (46, 14)$$

$$51P = (0, 48)$$

$$52P = \infty$$

In the above program, x and y are the coordinates of P , p is the modulus for the finite field E is defined over, and a and b are the coefficients in the equation for E . First, the program prints the equation for E for the sake of organization. Since this program calculates repeated addition of P , we had to add a few preliminary steps for calculating $2P$ (remember that the addition works differently when we are adding two different points and when we are doubling a single point). Values c and d are the x and y coordinates of $2P$. From there the program moves into a For loop, where we start at $k = 3$ ($3P$), and then the program calculates all values of kP up to a specified upper bound. If we know the order of $E(\mathbb{Z}_p)$, then we can set the upper bound to be the order of the group. The If loop is there to check if the x -coordinates of two consecutive points are equal. If they are, the program returns ∞ for the next k -value, and for every k -value thereafter. Finally, g and h are the x and y values of kP for the k th iteration of the loop, and the program prints them as an ordered pair. The program then redefines c and d , and then begins the process all over again to find the next point in the sequence. This program was mostly useful for working with individual points in $E(\mathbb{Z}_p)$ once we had figured out the group's order and were ready to start applying them to cryptosystems.

4.3 kP Algorithm

Our third program is somewhat of a shortcut for the previous program. Where that program calculated and printed kP and every intermediate value, this program uses a variation on the square-multiply method of modular exponentiation to calculate kP much more quickly. It doesn't help us find the order of P , but once we get into the next sections, it should become clear that the second program will be much too slow for finding specific multiples of P that will be practically useful. An important note is that this program will return an error if kP is greater than or equal to the order of P . This felt like poor programming, but it did not

seem worth fixing since this program was not meant to find the order of P , but to quickly find kP for large values of k . If we wanted to find the order of P with this program, we could always try different values of k until an error occurred, and then try $(k - 1)P$. If $(k - 1)P$ yields an ordered pair, then k is the order of P . If the program returns an error again, then we could try $(k - 2)P$, skip down a few values of k at a time until we get an actual result and then increment k until we run into an error. At that point, however, it would probably be much quicker to just use the second program. As an example, I provided a sample output for when $k = 26$ and $P = (0, 5)$ on the elliptic curve $y^2 = x^3 + 25x + 25 \pmod{53}$. The first few lines of the output are the enumerated binary digits of k , but the last two lines are the only important ones. The second to last line of the output gives the x -coordinate of kP , and the last line of the output gives the y -coordinate of kP .

```

k = 26;
intstr = IntegerString[k, 2];
len = StringLength[intstr];
list = Characters[intstr];
For[i = 1, i ≤ len, i ++, Print[i, " ", list[[i]]]];
x = 0 ;
y = 5 ;
p = 53 ;
a = 25 ;
b = 25 ;
c = x;
d = y;
For[i = 2, i ≤ len, i ++,
f = Mod[((3 * c^2 + a)*PowerMod[2 * d, -1, p])^2 - 2 * c, p];
g = Mod[(3 * c^2 + a)*PowerMod[2 * d, -1, p] * (c - f) - d, p];
If[list[[i]] == "1",

```

```

m =Mod[((y - g)*PowerMod[x - f, -1, p])2 - f - x, p];
n =Mod[(((y - g)*PowerMod[x - f, -1, p]) * (f - m)) - g, p];
c = m;
d = n,
c = f;
d = g]];
Print["P = (",x, ", ", y, ")"];
Print[k "P = (", c, ", ",d, ")"]

```

```

1 1
2 1
3 0
4 1
5 0
P = (0, 5)
26P = (18, 0)

```

The `intstr` command converts our integer k into a string of binary digits. The program saves the length of this string as `len` as a reference for how many times the `for` loop needs to run. For the last part of the initial step, the program saves the string as a list of individual digits, again to be used as a reference during the `for` loop. For the definitions, x and y are the coordinates of P , p is the prime order of the field over which the elliptic curve is defined, a is the coefficient on x in the Weierstrass form of our elliptic curve, b is the constant term, and c and d are the x and y coordinates of kP at each step during the calculation.

As opposed to squaring and multiplying numbers like in modular exponentiation, this program doubles and adds under elliptic curve addition. Each time through the loop, the program doubles the point (c, d) . If the i th element of the

list generated by our binary string is 1, the program performs an additional step by adding (x, y) to (c, d) . Note that the program starts on the second digit of the binary string, ignoring the leading 1, so technically the i th element of the list is referenced on the $(i-1)$ th loop, but for the sake of clarity let us avoid that distinction.

Suppose, for example, we wanted to calculate $5P$. 5 can be represented in binary as 101. Ignoring the leading 1, we are left with 01. The first time through the loop, the program doubles P . Since the binary digit for this loop is 0, we do not add P , so the first loop is complete. The next digit is a 1, so for the second loop, we double the result from the first loop, and then add P to that result. After the first loop, we have $2P$. The second loop doubles that result, yielding $4P$, and then adds P , yielding $5P$, our desired value.

Chapter 5

Elliptic Curve Cryptography

5.1 Koblitz Method

So far we have taken elliptic curves E defined over finite fields \mathbb{Z}_p , and built groups of points $E(\mathbb{Z}_p)$. We have a reliable way to check the order of $E(\mathbb{Z}_p)$ and the order of individual points. Before we can start building cryptosystems though, there is still one key component that we need: how to translate messages into points on our elliptic curve.

We cannot just pick random x -values and find the y -value that goes with them. The likelihood of a particular x -value giving a point on the curve is no more than a 50/50 chance. [3, 355-356] To remedy this, we use the Koblitz Method, created by Neal Koblitz. Note that, like the programs we have built, the Koblitz method deals with finite fields limited to prime order.

To start, we let E be defined $y^2 = x^3 + ax + b \pmod{p}$ for some prime p and we represent the message to be encrypted as an integer m . Next, we pick a large integer k , supposing that $(m + 1)k < p$. For $j = 0$ to $k - 1$,

Compute $x = mk + j$.

Compute $z = x^3 + ax + b \pmod p$.

If z has a square root $\pmod p$, say y , then use the ordered pair (x, y) to represent m . If z does not have a square root $\pmod p$, move on to the next j [3, 355-356]. An example of this method will be provided in 5.3 where we will actually use it to encode a message.

5.2 The ECC System

Now that we have a way to reliably find elliptic curve and finite field pairings where $E(\mathbb{Z}_p)$ has a large number of points and to calculate the order of individual points in a reasonable time, as well as a way to convert messages to points in $E(\mathbb{Z}_p)$, we are finally able to build a cryptosystem.

Elliptic Curve Cryptography (ECC) was discovered by Neal Koblitz and Victor Miller. It functions similarly to the El Gamal cryptosystem, but uses multiplication (repeated addition) of points on elliptic curves as opposed to exponentiation. Elliptic curve variations of other cryptosystems can increase security and potentially save on processing hardware due to the difficulty of the discrete log problem and differences in key operations. (If $a^x \equiv b \pmod p$, it is very difficult to solve for x .) [3, 363]

Like RSA and AES, ECC is a public key system, meaning that the information required to send an encrypted message to a person is available in a public file that anyone can access. Each person in the system is assigned four pieces of information in the public file:

1. A suitably large prime p .
2. An elliptic curve E_p that has a large number of points $\pmod p$.
3. A point α on E_p with a large order.
4. A point $\beta = \Delta\alpha$, where Δ is a large secret key known only to the individual person.

To encode a message, the sender looks up the receiver's entry (p, E_p, α, β) in the public file. They then choose an integer K such that $\frac{p-1}{2} < K < p-2$. Next, the sender converts their text message to a number of equal-sized numerical blocks. Using Koblitz method, they convert these numerical blocks to points M_i on E_p . Lastly, the sender computes $K\alpha$ and $M + K\beta$ and sends those two points to the receiver. Once the recipient has been given $K\alpha$ and $M + K\beta$, they use their secret key Δ to compute

$$\Delta * (K\alpha) = K * (\Delta\alpha) = K\beta.$$

Next they compute $-K\beta$ and add it to $M + K\beta$, which yields M as the result. Finally the recipient takes this point M and converts it back to the text message.

An important thing to note is that, even though α and $\Delta\alpha$ are available in the public file, finding κ is no easy task due to the difficulty of the discrete log problem.

5.3 Sample Encryption and Decryption

Now, the moment you have been waiting for, we have used our programs to find a fairly large prime and an elliptic curve that, together, create a group $E(\mathbb{Z}_p)$ with many points and our friend Bob is ready to encrypt a message to send to Alice. As described earlier in the section on ECC, Alice's entry in the public file would look like

$$(p, E_p, \alpha, \beta),$$

where p is his chosen prime, E_p is an elliptic curve with many points mod p , α is a point with very large order, and $\beta = \Delta\alpha$, where Δ is a very large secret key. For this example, Alice's entry in the public file will be

$$(3001, x^3 + 31x + 94, (2, 769), (2897, 2434)).$$

Bob has written his message and decomposed it into numerical values ($A= 00, B= 01, \dots, Z= 25, = 26$) resulting in the following string of numbers: 190704260104181926190704180818260304050413180426081826000714140326190704180818261405050413180427.

Next, Bob breaks the message up into evenly sized blocks, two digits each in this example, and then he uses Koblitz Method with $k = 11$ to convert those blocks into points in $E(\mathbb{Z}_{3001})$. He sends $k = 11$ to Alice, and then converts his message to the series of points

$$\begin{aligned} &(211, 672), (81, 1035), (46, 747), (286, 1384), (11, 1380), (46, 747), (200, 714), (211, 672), \\ &(286, 1384), (211, 672), (81, 1035), (46, 747), (200, 714), (88, 1136), (200, 714), (286, 1384) \\ &(33, 698), (46, 747), (55, 920), (46, 747), (143, 895), (200, 714), (46, 747), (286, 1384) \\ &(88, 1136), (200, 714), (286, 1384), (2, 769), (286, 1384), (67, 94), (157, 689), (157, 689) \\ &(33, 698), (286, 1384), (211, 672), (81, 1035), (46, 747), (200, 714), (88, 1136), (200, 714) \\ &(286, 1384), (157, 689), (55, 920), (55, 920), (46, 747), (143, 895), (200, 714), (46, 747). \end{aligned}$$

Bob chooses $K = 2000$, and then computes the points 2000α and $M_i + 2000\beta$ for each block M_i .

$$K\alpha = 2000(2, 769) = (1458, 2332),$$

$$K\beta = 2000(2897, 2434) = (1843, 2164).$$

For $M_1 = (211, 672)$, we use elliptic curve addition to find

$$M_1 + K\beta = (211, 672) + (1843, 2164) = (394, 595).$$

Using that same procedure, Bob encodes the rest of the points in his message and sends them to Alice, as well as the points $K\alpha$ and $K\beta$. Alice now has the following points:

$$\begin{aligned} &(394, 595), (1755, 1544), (2648, 755), (2994, 1758), (943, 1078), (2648, 755), (1540, 1393), \\ &(394, 595), (2994, 1758), (394, 595), (1755, 1544), (2648, 755), (1540, 1393), (746, 2273), \\ &(1540, 1393), (2994, 1758), (1778, 2929), (2648, 755), (2727, 1276), (2648, 755), (821, 862), \\ &(1540, 1393), (2648, 755), (2994, 1758), (746, 2273), (1540, 1393), (2994, 1758), (1884, 2348), \\ &(2994, 1758), (1759, 1381), (261, 1299), (261, 1299), (1778, 2929), (2994, 1758), (394, 595), \\ &(1755, 1544), (2648, 755), (1540, 1393), (746, 2273), (1540, 1393), (2994, 1758), (261, 1299), \\ &(2727, 1276), (2727, 1276), (2648, 755), (821, 862), (1540, 1393), (2648, 755), \end{aligned}$$

as well as the points $K\alpha = (1458, 2332)$ and $K\beta = (1843, 2164)$. An important thing to note here is that while using Koblitz Method to generate the M_i in this example, Bob used the same value j every time the same pair of digits appeared in the message. He could, however, use a different j value each time if more than one gave a point in $E(\mathbb{Z}_{3001})$, that way one letter could encrypt to a different point each time, thus thwarting frequency counting.

Now that Alice has $K\alpha$, $K\beta$, and all of the M_i , the first thing she does is apply her secret key, which is 3009, to $K\alpha$. Remember that $\Delta * K\alpha = K * (\Delta\alpha) = K\beta$.

She computes

$$\Delta * K\alpha = 3009(1, 2332) = (1843, 2164).$$

Alice then computes

$$-K\beta = -(1843, 2164) = (1843, -2164) = (1843, 837).$$

For the next step, Alice adds $-K\beta$ to $M_i + K\beta$ for each block of M . For $M_1 = (394, 595)$,

$$M_1 + K\beta + (-K\beta) = (394, 595) + (1843, 837) = (211, 672),$$

To convert the decrypted point back to the original message, Alice divides the x -coordinate from M_1 by the k that Bob used during the Koblitz Method, in this case 11, and keeps the integer part of the result. The result is 19, which is 'T'. Decrypting the rest of the M_i , Alice gets the original message, which is

THE BEST THESIS DEFENSE IS A GOOD THESIS OFFENSE

Bibliography

- [1] Avner Ash and Rober Gross. Elliptic Tales. New Jersey: Princeton and Oxford, 2012.
- [2] Abhijit Das. Computational Number Theory. Florida: CRC, 2013.
- [3] Wade Trappe and Lawrence Washington. Introduction to Cryptography. New Jersey: Pearson Prentice Hall, 2006.
- [4] Lawrence Washington. Elliptic Curves: Number Theory and Cryptography. Florida: CRC, 2003.