ELLIPTIC CURVE CRYPTOGRAPHY

NANDANA MADHUKARA

Abstract. In this paper, we discuss cryptography using elliptic curves. We start with the basics of cryptography and then go on to the RSA Algorithm for context as why elliptic curve cryptography is so useful. Then we discuss discrete logarithms which is needed for the final section of this paper where we study elliptic curves.

Contents

1. Introduction

Cryptography is at the heart of most modern day privacy. It is what keeps most of the world's data secure and underlying most of it is number theory. In general, the feature that makes cryptosystems secure is one way operations which are hard to compute one way but easy to compute the other way. In the case of the famous RSA algorithm, this operation is the multiplication of primes which is relatively easy to do but factoring a large number into its prime factors is a tedious task.

However the multiplication of primes was not the first operation people thought of. The first famous use of cryptography was by Caesar with his Caesar Cipher. He would shift every his plaintext by 3 letters to get his ciphertext. For example, the name "Brutus" would be "Euxwxz." Then the recipient would take the encrypted message and decrypt it by shifting all the letters back by 3. This cipher is part of a larger class of ciphers called *substitution ciphers* where each letter is assigned to another letter. This map from letters to letters serves as the key and anyone with the this can reverse the encryption and decrypt the message. In Caesar's case, he was assigning each letter to the letter 3 after it.

For that time, this was a complicated enough cipher and got the job done but people started finding ways of breaking it. They found out that the way of breaking the cipher was through letter frequency analysis. Notice that if the letter "E" is assigned to the letter

Date: May 28, 2024.

"T," then "T" must occur in the message with roughly the same frequency as "E" does in normal english. Therefore, the letter occuring the most number of times is most likely an "E" in normal english. Doing this frequency analysis on the encrypted message is what helped people break the substitution ciphers.

As technology advanced, cryptography machines started to be formed. The most famous of these was the Enigma machine produced by the Nazis during World War II. This machine was essentially lots of substitution ciphers combined to essentially force the decrypter to brute force. (Even Alan Turing, the breaker of the Enigma had to brute force but his ingenuity was cutting down the number of possibilities). The thing that made Enigma so hard to crack was not only that it had lots of substitution ciphers combined together but that the same letter could be mapped to different letters. For example, "AAA" could be mapped to "ZLM." This meant that the actual substitution map changed for every letter. However, the one flaw as that a letter could not map to itself and Alan Turing was able to use this Achilles heal break the cipher.

Before we go on to different cryptosystems and the math behind them, we must first establish the basics of cryptography or one might call them our axioms. Even though they might seem obvious at first, they still must be stated since there is still some nuance to them.

2. Basics of Cryptography

The basic setup of cryptography is three people: Alice, Bob, and Eve. Alice wants to send a *plaintext* message to Bob securely. So, Alice uses a *key* to encrypt her message into *ciphertext* and sends this over to Bob. Bob then decrypts the message and changes the ciphertext back to plaintext using a decryption key. Now Eve might want to do anyone of the following things [TW05]:

(1) Read the message Alice is sending

(2) Find out the key for encryption so she can read all the messages coming from Alice

(3) Corrupt Alice's message so Bob reads something different than what Alice intended

(4) Act as Alice so Bob believes he is talking to Alice when in fact he is talking to Eve

The simplest of these is (1) and (2) which we will be focusing on in this paper.

Now there is three possible types of attacks Eve can perform, the first of which is only knowing the ciphertext which is the least amount of information Eve can have. Taking the example of the Caesar cipher, if Alice's message is "Brutus," all Eve will be able to see is "Euxwxz."

The next possibility is if Eve also has access to some part of the plaintext and the ciphertext that corresponds to this plaintext. This is the doom for many weak cryptosystems like substitution ciphers and is still a valuable tool to find the key of stronger systems like Enigma. Turing and his team were able to deduce that every German message ended with "Heil Hitler" and used this repeating phrase to deduce the key.

The third possibility is if Eve has access to the actual cryptosystem. This means that Eve knows how Alice encrypts her message and how Bob decrypts the ciphertext but she doesn't have access to the actual key. Therefore, Eve can send in whatever plaintext she wants and study the ciphertext that comes out to find the key. In this paper, we assume that Eve always has this amount of information. This is what is known as Kerckoff's Principle: *When assessing the security of a cryptosystem, one must always assume that the enemy knows the*

method being used. People can always join the enemy or be captured resulting in the enemy knowing the the method being used. This means that the security of a cryptosystem does not lie in how complicated the method is but in the key.

Now cryptosystems can be broken into two categories: *symmetric key encryption* and *asymmetric key encryption.* For the former, Alice and Bob must agree on a encryption and decryption key, but this is very hard to do if Alice and Bob are very far apart. Historically, this key exchange has been done with very secure transportation. For example, the Germans during WWII would transport a list of keys for Enigma on water dissolving paper so that if a ship was sunken by the Allies, they wouldn't get access to the keys. However, this is very unreliable and can easily go down, but the problem has been solved and a solution is called the *Diffie-Hellman Key Exchange.* We will go more into depth about this later in this paper.

Another way Alice and Bob can send secret messages only through public communication is through asymmetric key cryptography. This is when the encryption key is made public but the decryption key is not. This can be thought of as Bob sending lots of boxes with locks out to the world and any Alice that wants to send him a secret message puts the plaintext in the box and locks it. Now Bob is the only one with the key to the open the box, so the security of the cryptosystem lies entirely on the decryption key. One of the most famous examples of this is the RSA algorithm which we will go into in the next section.

The main advantage of this type of cryptosystem is the lack of communication between Alice and Bob. In a symmetric key cryptosystem, Alice and Bob need to communicate with each other through the Diffie-Hellman key exchange and this is inevitable since communication must be required when agreeing on a key. However, in a assymetric key cryptosystem Alice and Bob don't need to agree on a key so Bob can just make the encryption key public and any Alice can send their message. The main disadvantage, is that public cryptosystems require a lot of computation so the encryption of large amounts of data is reserved to symmetric encryption.

To recap, and in historical order, we first started with cryptosystems where the method was made a secret like in the Caesar Cipher where not a lot of people knew how Caesar was encrypting and decrypting messages. Then we realized that it is inevitable that the enemy will know the method of the cryptosystem so by Kerckoff's Principle, we must assume that Eve knows the method being used. Then we went to symmetric key encryption where the encryption and decryption keys were kept private and now, we currently mainly use assymetric key encryption where the encryption key is made public. Quite ironically, our cryptosystems have been made more secure with the publicity of more information.

3. The RSA Cryptosystem

The most famous assymetric key cryptosystem is the RSA cryptosystem which goes as follows:

(1) Bob chooses two distinct primes p and q and computes $n = pq$.

(2) Bob chooses e such that $\gcd(e, (p-1)(q-1)) = 1$.

(3) Bob computes the d such that $de \equiv 1 \pmod{(p-1)(q-1)}$. (Bob can use the Euclidean Algorithm for speed).

(4) Bob makes n and e public while keeping p, q, and d private.

(5) Alice encrypts her message $0 \leq m < n$ as $c \equiv m^e \pmod{n}$ where c is the ciphertext she sends to Bob. (If m is not in range, she breaks it into smaller blocks).

(6) Bob recovers the message by computing $c^d \equiv m \pmod{n}$

The main magic of the RSA algorithm is

Theorem 3.1. *Alice's encrypted message c can indeed be decrypted into m by computing*

$$c^d \pmod{n}.$$

Proof. First we claim that $c^d \equiv m \pmod{p}$ and $\pmod{q}$. WLOG, we only consider modulo p since we can just replace the p's with q's to get the modulo q case. Notice that $c^d = (m^e)^d = m^{de}$ so this is the expression we will be focusing on. Since $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1)$, we have

$$de \equiv 1 \pmod{\phi(n)}$$

so we can write

$$m^{de} = m^{1+k\phi(n)} = m \cdot m^{k\phi(p)\phi(q)} = m \cdot (m^{\phi(p)})^{k\phi(q)}$$

for some integer k. Now we start with the case when $\gcd(m, p) = 1$, by Euler's Theorem, we have

$$m \cdot (m^{\phi(p)})^{k\phi(q)} \equiv m \cdot 1^{k\phi(q)} \equiv m \pmod{p}.$$

If $\gcd(m, p) \neq 1$, since p is prime, we have $m = m'p$, so

$$m \cdot (m^{\phi(p)})^{k\phi(q)} \equiv 0 \equiv m \pmod{p}$$

completing our proof of our claim.

Now our claim tells us

$$c^d = k_1 p + m$$

and

$$c^d = k_2 q + m$$

for integers k_1 and k_2. Multiplying the first equation by q and the second by p and adding the two, we get

$$(p+q)c^d = (k_1 + k_2)pq + (p+q)m.$$

Another way of writing this is

$$(p+q)c^d \equiv (p+q)m \pmod{n}.$$

Now $p+q$ cannot be 1, p, q or n so this means that

$$c^d \equiv m \pmod{n}.$$

■

Let us look at an example:

Example. Say Bob chooses the primes $p = 5$ and $q = 13$ so $n = 65$. Let us say he also picks the encryption exponent $e = 7$. Now say Alice wants to send the message $m = 27$. She must compute

$$c \equiv m^e \equiv 27^7 \equiv 53 \pmod{n}.$$

Now Bob must calculate his decryption exponent with

$$de \equiv 1 \pmod{(p-1)(q-1)}.$$

He can use the Euclidean Algorithm to find $d = 7$. Now he calculates

$$c^d \equiv 53^7 \equiv 27 \equiv m \pmod{n}$$

and sure enough, he gets the right message.

Notice that all the computations Alice or Bob make are relatively easy, at least of a computer. When Bob finds the decryption exponent, he can just use the Euclidean Algorithm. When Alice finds c, she can just repeatedly multiply by m and take the result modulo n and do this e times rather than reducing m^e modulo n. Finally, Bob can do the same thing when he decrypts the ciphertext.

Now things start getting computationally hard when Eve tries to intervene. Now Eve has access to n, e, c but doesn't have access to p, q, d. One thing she can try do is trying to take the eth root of $c \equiv m^e \pmod{n}$ to find m. However, this isn't as simple as plugging the expression into a calculator. It is very likely that $c^{1/e}$ is not an integer so reducing this modulo n is impossible. Therefore a case-by-case search must be done through all possible residues modulo n which is inefficient especially for large n.

Another thing Eve can try doing is finding the decryption exponent with

$$de \equiv 1 \pmod{\phi(n)}.$$

This requires the knowledge of $\phi(n)$ and this is essentially the same as knowing p and q:

Proposition 3.2. *If n can be factored into $n = pq$ and we know $\phi(n)$, then it is easy to find p and q.*

Proof. Notice that

$$n - \phi(n) + 1 = pq - (p-1)(q-1) + 1 = p + q$$

so p and q by Vieta's Formulas are the roots of the quadratic

$$x^2 - (n - \phi(n) + 1)x + n$$

so

$$p, q = \frac{n - \phi(n) + 1 \pm \sqrt{(n - \phi(n) + 1)^2 - 4n}}{2}$$

which is easy to compute. ∎

Therefore finding the decryption exponent is equally as hard as factoring n and we already know that factoring is very hard for computers, especially for large numbers.

4. Discrete Logorithms

Just like how the difficulty of factoring can be used to create cryptosystems RSA, the difficulty of computing *discrete logarithms* also has applications in cryptography.

Definition 4.1. Let p be a prime and let α and β be nonzero integers modulo p. Additionally, let n be the smallest positive integer such that $\alpha^n \equiv 1 \pmod{p}$. The *discrete logarithm* of β with respect to α denoted with $L_\alpha(\beta)$ is the integer x modulo n such that

$$\alpha^x \equiv \beta \pmod{p}.$$

Definition 4.2. A *primitive root* of a modulo p is an α such that every β modulo p is a power of α.

If α is not a primitive root of mod p, then the discrete logarithm may not be defined for certain residues modulo p.

Example. Let $p = 13$, $\alpha = 2$, $\beta = 11$. Notice that

$$2^7 \equiv 2^{19} \equiv 2^{31} \equiv \cdots \equiv 2^{7+12k} \pmod{13}.$$

The reason we get this pattern is because 12 is the smallest positive integer n such that $2^n \equiv 1 \pmod{13}$. This is why in our definition we considered the discrete logarithm modulo n – so that we get one answer. Therefore $L_\alpha(\beta) = L_2(11) = 7$.

Now in this example, we just guessed that 7 is the answer and we got this from an exhaustive search. For small primes, this trial and error method suffices but as we increase the size of our primes, this method becomes infeasible. Although people have found various clever ways of attacking the discrete logarithm problem, this is still considered a computationally hard problem – just like factoring. This leads us to another definition:

Definition 4.3. A *one-way function* is a map $f(x)$ easy to compute $f(x)$ for some x but computationally infeasible to the x such that $f(x) = y$.

Example. In RSA, this function was multiplication of primes. It is very easy to multiply two primes together but on the other hand, it is very hard to break a number into two primes. We also saw in RSA that it is easy to calculate modular exponentiation like $\alpha^x \pmod{p}$. (This is what Alice and Bob did to encrypt and decrypt the messages). However, the discrete logarithm shows that it is much harder to find the x such that $\beta \equiv \alpha^x \pmod{p}$

4.1. **Diffie-Hellman Key Exchange.** We will take a slight digression and talk about the Diffie-Hellman Key Exchange we introduced in the introduction.

Here is roughly how the exchange works: Alice and Bob each have their own private keys and there is one public key that everyone has access to. Alice and Bob each combine their keys with the public key through some one-way function. A non-mathematical one-way function of this is paint mixing since it is easy to mix two different paints together but hard to find what paints make up a given color. If Alice and Bob's private key are some secret colors and the public key is another color, then Alice and Bob would mix their secret colors with the public color. Continuing this analogy, Alice and Bob send their new colors to each other. Notice that when Eve sees the new colors, she knows that one of the constituents of each color is the public color but since paint mixing is one-way, it is very hard and infeasible, definition, for Eve to figure out Alice or Bob's secret color. Now when Alice and Bob get each others new colors, they add their own secret color to the new color. Notice that Alice and Bob each have a mix of both secret colors and the public color so the colors Alice and Bob end up with must be the same.

Here is the mathematical version of this:

(1) A public prime p, preferably large, and a primitive root $\alpha \pmod{p}$ is fixed.

(2) Alice and Bob choose integers a, b, respectively, in the range $1 \leq a, b < \phi(p) = p - 1$.

(3) Alice sends $\alpha^a \pmod{p}$ to Bob and Bob sends $\alpha^b \pmod{p}$.

(4) Alice calculates the common key with $K \equiv (\alpha^b)^a \equiv \alpha^{ab} \pmod{p}$ and Bob does the same with $K \equiv (\alpha^a)^b \equiv \alpha^{ab} \pmod{p}$.

Practically, the whole common key is not usually used because with the size of p, it is usually a huge number. Therefore Alice and Bob can agree on a specific part of K to use. For example, in DES, Alice and Bob can use the middle 56 bits of the key.

Now the whole key exchange lies the following question:

Question 4.4 (Computational Diffie-Hellman)**.** *Let p be a prime and let α be a primitive root modulo p. Given $\alpha^a \pmod p$ and $\alpha^b \pmod p$, is it possible to find out $\alpha^{ab} \pmod p$ feasibly?*

When Eve attacks this exchange with the knowledge of $\alpha^a \pmod p$ and $\alpha^b \pmod p$, one obvious way she can do this is by finding a through the discrete logarithm and then raising α^b to the ath power, giving Eve the common key K. Therefore breaking the key exchange is at most as hard as computing a discrete logarithm but this not feasible for computers for very big p. The main question is "is there a faster or better way?" which no one has found yet.

Another related question is the following:

Question 4.5 (Decision Diffie-Hellman)**.** *Let p be a prime and let α be a primitive root modulo p. Given $\alpha^a \pmod p$ and $\alpha^b \pmod p$ and some c is it possible to check if $c \equiv \alpha^{ab} \pmod p$.*

The obvious way of doing this is by computing $\alpha^{ab} \pmod p$ with the Computation Diffie-Hellman Question and it is now easy to check if $c \equiv \alpha^{ab} \pmod p$. This is trivial and people have found faster solutions using elliptic curves which we will look at in the next section. Now an interesting thing to consider is trying to solve Question 4.4 with Question 4.5. However, the only solution we know of is brute force where we go through all possible c's and see which equals $\alpha^{ab} \pmod p$ but this is just a brute-force method and no better than just computing a discrete logarithm by brute force.

4.2. **ElGamal Cryptosystem.** Now we come back to discussing cryptosystems that use discrete logarithms as the security feature rather than factoring. An example of this the ElGamal Cryptosystem:

(1) Bob chooses a prime p and a primitive root α. He also chooses a secret number b and computes $\beta = \alpha^b \pmod p$. He then makes (p, α, β) public.

(2) Alice chooses a message $1 \leq m < p$ (breaking the message up if it is not in this range) and records Bob's public key.

(3) Alice chooses a secret integer a and computes $r \equiv \alpha^a \pmod p$.

(4) Alice also computes $t \equiv \beta^a m \pmod p$.

(5) Alice sends (r, t) to Bob.

(6) Bob decrypts by computing $tr^{-b} \equiv m \pmod p$. (He can compute the modular inverse quickly with the Euclidean Algorithm).

The reason this works is because

$$tr^{-b} \equiv \beta^a m(\alpha^a)^{-b} \equiv (\alpha^b)^a m\alpha^{-ab} \equiv m \pmod p.$$

Now if Eve finds out b, Bob's secret key, then she can just follow the process Bob goes through to find the message. However $\beta \equiv \alpha^b \pmod{p}$ so Eve must compute a discrete logarithm so this is what the cryptosystem relies on.

One thing to note is that Alice must choose a different secret integer a every time she sends a message because if Alice sends two messages m_1 and m_2 with the same a, Eve can find m_2 if she finds m_1. This is because r will be the same and Eve will know (r, t_1) and (r, t_2). Notice that

$$\frac{t_1}{m_1} \equiv \beta^a \equiv \frac{t_2}{m_2} \pmod{p}$$

so $m_2 \equiv t_2 m_1 / t_1 \pmod{p}$.

Now in the RSA cryptosystem, it is easy to check if a given message m corresponds to a given ciphertext c. All we need to do is compute $m^e \pmod{p}$ and see if that equals c. We can ask a similar question for the ElGamal cryptosystem: for a given message m can we check if it corresponds to a given ciphertext (r, t)? It turns out that this question is as hard as the Decision Diffie-Hellman Question.

Proposition 4.6. *A machine that checks the validity of modulo p ElGamal ciphertexts can be used to answer the Decision Diffie-Hellman question modulo p and a machine that answers the Decision Diffie-Hellman question modulo p can be used to check the validity of modulo p ElGamal ciphertexts.*

Proof. We first prove the forward direction. We call the machine that checks the validity of mod p ElGamal ciphertexts M_1. This means that M_1 takes in p, α, β, m, and (r, t) and it returns either "yes" or "no" depending on m corresponds to (r, t). Now given $\alpha^x \pmod{p}$ and $\alpha^y \pmod{p}$ and a c we want to check if $c \equiv \alpha^{xy} \pmod{p}$. Let $\beta = \alpha^x \pmod{p}$, $r \equiv \alpha^y \pmod{p}$, $t = c$, and $m = 1$. We plug this all in M_1 and see what it returns. Notice x takes the place of Bob's secret number b and α^y takes the place of $r \equiv \alpha^a$. This means the decryption of (r, t) is $tr^{-a} \equiv c(\alpha^y)^{-x} \equiv c\alpha^{-xy} \pmod{p}$. Now if the machine outputs "yes," since $m = 1$, we know that

$$1 \equiv c\alpha^{-xy} \implies c \equiv \alpha^{xy} \pmod{p}.$$

If the machine says "no" then we know this is not true. Therefore we can solved the Decision Diffie-Hellman question by plugging in these specific values into the ElGamal validity machine.

Now let us prove the converse. Let M_2 be the machine that solves the Decision Diffie-Hellman question. This means that we input p, α^x, α^y, and c and M_2 ouputs "yes" if $c \equiv \alpha^{xy}$ and "no" otherwise. Let $\beta \equiv \alpha^x \pmod{p}$ so x takes the place of b. Next let $r \equiv \alpha^y \pmod{p}$ so y takes the place of a. Finally, let $c = tm^{-1}$. Now we know that m is the correct plaintext for (r, t) iff $m \equiv tr^{-b} \equiv t\alpha^{-ab} \pmod{p}$ but this only happens if $tm^{-1} \equiv c \equiv \alpha^{xy} \pmod{p}$ which is when M_2 returns "yes." Therefore the ouput of M_2 can be used to check the validity of mod p ElGamal ciphertexts meaning that we are done. ∎

5. Elliptic Curves

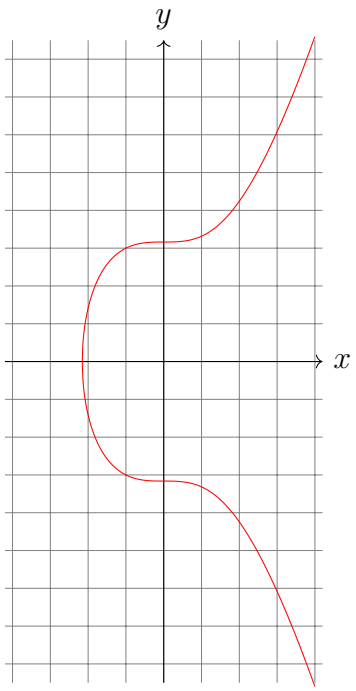
We finish the paper with a discussion of elliptic curves which is are heavily related to discrete logarithms. In fact we can create an analogous cryptosystem of discrete logarithms with elliptic curves and improve the cryptosystems. But first, we must define these curves [Sil09]:

Definition 5.1. Let K be any field of characteristic not 2 and let $a, b, c \in K$. An elliptic curve E is the set of points

$$E = \{(x, y) : x, y \in K, y^2 = x^3 + ax^2 + bx + c\}.$$

We also add the point (∞, ∞) to this set to represent the "point at infinity" in this curve. We denote this point with ∞.

Example. When working with $K = \mathbb{R}$ we can graph our elliptic curves. For example, here is the graph of $y^2 = x^3 + 10$:



Notice that in this example, the point at infinity is just the point at the top of the y-axis.

Example. We can also condier elliptic curves in modulo p since the set of integers modulo p is field of characteristic not 2. For example, E can be the set of points that satisfy

$$y \equiv x^3 + 2x - 1 \pmod 5.$$

This would mean that the elements of E are

$$E = \{(0, 2), (0, 3), (2, 1), (2, 4), (4, 1), (4, 4), \infty\}$$

where we have included the point at infinity. These are the elliptic curvues useful in cryptography.

One interesting thing we can do with elliptic curves is addition.

Definition 5.2. Let P_1 and P_2 be points on an elliptic curve E with $K = \mathbb{R}$. We define the sum of P_1 and P_2 to be the point P_3 with obtained through the following construction: we draw a line through P_1 and P_2 and see where it intersects E. We then take the reflection of this point across x-axis to get P_3.

Example. Let us take the elliptic curve $y^2 = x^3 + 73$ and let $P_1 = (2, 9)$ and $P_2 = (3, 10)$. This means that the line through these points is $y = x + 7$. Now we find the points of

intersection by plugging this into E:

$$(x+7)^2 = x^3 + 73$$

which gives us the cubic

$$x^3 - x^2 - 14x + 24 = (x-2)(x-3)(x+4) = 0.$$

Sure enough, two of our roots is $x = 2$ and $x = 3$ which are the x values of P_1 and P_2 which is what we expected. Now the new root is $x = -4$ which has a y value of $y = 3$. Therefore $P_1 + P_2 = (-4, -3)$.

Example. We can take this a step further by considering ∞ as one of the addends. Let $P = (x, y)$ and the line through P and ∞ is just a vertical line. Therefore the other point of intersection is $(x, -y)$. However, when we reflect this across the x axis, he just get $(x, y) = P$. Therefore

$$P + \infty = P$$

so ∞ acts as an additive identity.

This means that we can also define subtraction. We know that

$$(x, y) + (x, -y) = \infty$$

so we can define the additive inverse as

$$-(x, y) = (x, -y).$$

This means that we can subtract, $P - Q$, by adding the additive inverse, $P + (-Q)$.

However, when we are doing computations it is more useful to have a computation based definition rather than a geometric based definition:

Definition 5.3 (Addition Law)**.** Let E be the elliptic curve $y^2 = x^3 + bx + c$ and let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$. Then the sum

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

where

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

and

$$m = \begin{cases} (y_2 - y_1)/(x_2 - x_1) & \text{if } P_1 \neq P_2 \\ (3x_1^2 + b)/(2y_1) & \text{if } P_1 = P_2. \end{cases}$$

If the slope is undefined or infinite, $P_3 = \infty$. Finally, the last addition law is

$$P + \infty = P.$$

One quick side note is that we are not looking at the general elliptic curve because notice that the a is missing. This is because in most scenarios, through a change of variables, we can turn a general elliptic curve into $y^2 = x^3 + bx + c$. Therefore from this point on, these are the elliptic curve we will be considering.

Now let us discuss elliptic curve cryptosystems. It turns out that we can make analogous elliptic curve cryptosystems of discrete logarithm cryptosystems like ElGamal. Now the advantage of using elliptic curves is that it is harder to break since there is a pretty efficient method of computing discrete logarithms by factoring integers into primes known as the index calculus. This forces people to use very large primes but this is no longer the case in

elliptic curve versions since this cannot be done (there are not discrite logarithms). Therefore elliptic curves can have the same level of security as their counterparts but with less data required.

5.1. **Elliptic Curve ElGamal Cryptosystem.** Here is the procedure for this cryptosystem:

(1) Bob chooses an elliptic curve E modulo p, so $K = \mathbb{Z}\backslash p\mathbb{Z}$ and a point α on E. He then chooses a secret number b and computes

$$\beta = b\alpha = \alpha + \alpha + \cdots + \alpha$$

where we are adding α b times. Finally, Bob makes (E, α, β).

(2) Alice takes her message m and encodes it as a point on the elliptic curve (which we will dicsuss later). She chooses her secret number a and computes $r = a\alpha$ and $t = m + a\alpha$ and sends (r, t) to Bob.

(3) Bob takes this pair and decrypts the message by computing

$$t - ar = m.$$

Notice that this is essentially the ElGamal Cryptosystem but we have replaced all the modular exponentiations with elliptic curve addition. Therefore it is fairly easy to see why this algorithm works. Notice that this is just as hard a computing a discrete logarithm and arguably harder since we cannot use index calculus for small primes that we could use in normal ElGamal.

Now one thing we have not mentioned is how Alice encodes her plaintext as a point on an elliptic curve. Unlike before when we converted text to numbers, it is not a simple as ASCII where we just map characters to numbers. This is due to the fact that there is no algorithm of polynomial time that we can use to list out the points on an arbitrary elliptic curve E modulo p. However there is a probablistic algorithm formulated by Koblitz which Alice can use. Alice first converts her message into numbers like usual and embeds it on the x axis. However there is around a chance of $1/2$ that $x^3 + bx + c$ is actually a perfect square mod p so we adjust few of the end bits of m until it becomes a square mod p.

5.2. **Elliptic Curve Diffie-Hellman Key Exchange.** We also discussed the Diffie-Hellman Key Exchange which we can create an analagous elliptic curve version:

(1) Alice and Bob agree on a public common point G in the elliptic curve E modulo p.

(2) Alice chooses her secret number a and computes aG and Bob chooses his secret number b and computes bG and they send these to each other.

(3) Alice recieves bG and multiplies it by a to obtain abG. Bob recieves aG and multiplies it by b to also obtain abG.

Just like in the ElGamal cryptosystem we have replaced the modular exponentiation with elliptic curve addition which keeps the data even more secure.

References

[Sil09] J.H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer New York, 2009.

[TW05] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography with Coding Theory (2nd Edition)*. Prentice-Hall, Inc., USA, 2005.

Email address: sciencekid6002@gmail.com