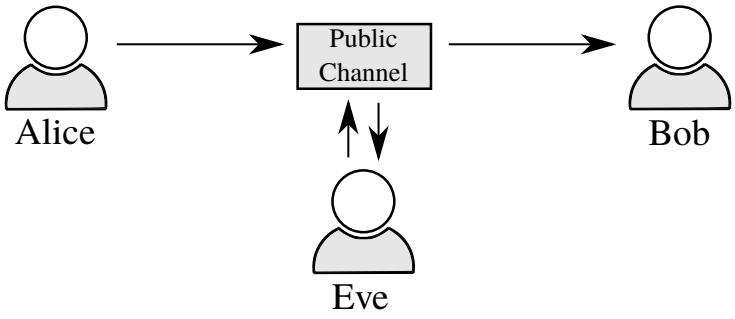
A SURVEY OF MATHEMATICAL CRYPTOGRAPHY

SON CHAU, WEIXIAN LAN, AND JONATHAN SHAW

ABSTRACT. Cryptography refers to the set of practices by which parties can exchange communications in such a way that those communications are only useful to their intended recipient. Cryptography has been used for millenia—indeed, the Roman military general Julius Caesar once used a primitive cryptographic system which involved shifting letters by a fixed amount. In fact, for most of history, all cryptographic systems were variations of this principle: they involved shifting and/or permuting glyphs in some way, and trusted parties needed to have previously obtained a formula (or *secret key*) for reversing these transformations. In the 1970s, however, an entirely new paradigm was introduced: asymmetric cryptography. Asymmetric cryptosystems rely on interesting number-theoretic properties to provide cryptosystems that do not require secret keys to be exchanged. In this paper, we will take a brief look into the origins of cryptography. We will then introduce the basic principles of asymmetric cryptography before taking an in-depth look at elliptic curves, which form the basis of many modern asymmetric cryptosystems.

1. AN INTRODUCTION TO CRYPTOGRAPHY

What is cryptography? To answer this question, we'll present a classic scenario. Suppose Alice wants to send a secret to her friend Bob over an insecure channel (be it the internet, a telephone line, the postal system, or any other public method of communication). Alice wants to be sure, however, that her enemy Eve—who may be monitoring the channel—cannot retrieve the secret that is meant for Bob's eyes only.



Date: December 13, 2019.

Key words and phrases. cryptography, asymmetric cryptosystem, elliptic curves.

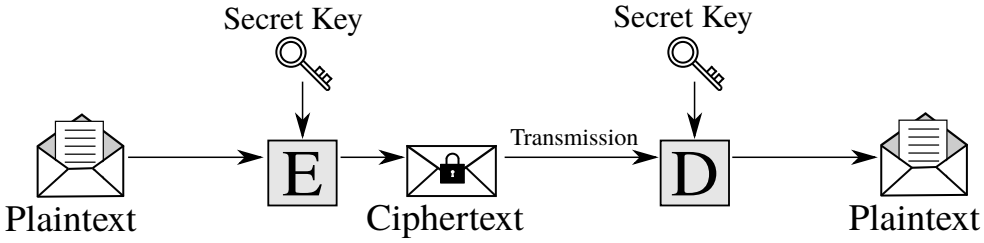
How can Alice achieve her goal? This is the motivating question behind the study of cryptography. Cryptography, then, is defined as the study of techniques by which private information can be broadcast over public channels while ensuring that only the intended recipient can actually receive the information.

1. **Symmetric Ciphers.** Throughout history, the vast majority of cryptosystems have fallen under the category of *symmetric ciphers*. Such cryptosystems contain the following components [9, Section 3.1]:

- A *encryption algorithm*, which converts an ordinary message, or *plaintext*, to an obfuscated version of the message, or *ciphertext*.
- A *decryption algorithm*, which converts ciphertext back to its original plaintext.
- A *secret key* which is shared between two parties and is required as input to both the encryption algorithm and the decryption algorithm. The encryption algorithm produces a different output ciphertext for each different key used.

Suppose Alice wishes to transmit a message secretly to Bob using a symmetric cipher. Assuming that Alice and Bob have, in the past, exchanged a secret key, the following exchange could occur [6, Chapter 1]:

(1) Alice feeds her plaintext message along with her copy of the secret key into the encryption algorithm to generate ciphertext.
(2) Alice transmits the ciphertext to Bob over an insecure channel.
(3) Bob receives the ciphertext and feeds it into the decryption algorithm along with his copy of the secret key, which returns the plaintext.



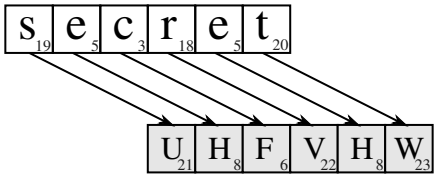
We will take a brief look at some symmetric ciphers before introducing asymmetric ciphers.

1.1. *Caesar Cipher.* An early user of cryptography was Roman general Julius Caesar, who used a very simple cryptographic system to send military communications in a secure manner. The system, known as the Caesar Cipher, involves shifting each letter of the alphabet by a fixed amount [9, Section 3.2]. If we think of the plaintext characters p as the numbers corresponding to their

respective positions in the English alphabet, the corresponding ciphertext character C is obtained by computing

$$C = p + s \text{ (least positive residue modulo 26)}$$

where s is the *shift parameter* corresponding to the number of positions the plaintext character is shifted. For example, with a shift parameter of 3, we can encode the word "secret":



The shift parameter is the key for this cipher—that is, the shift cipher is the piece information needed to encrypt and decrypt the message. This technique may have been useful to Julius Caesar while it was novel. However, it is not difficult to see that this cipher is very easily broken: given a long enough body of ciphertext, a quick statistical analysis of letter frequency would reveal the correspondence between plaintext characters and their ciphertext counterparts. [6, Section 1.2] shows in detail how one would go about breaking the Caesar Cipher.

1.2. *Rotor Machines.* The Caesar cipher is weak because it maps plaintext to ciphertext in a predictable manner. In the 19th and 20th centuries, mechanical devices were created to encrypt and decrypt messages according to much more complex schemes. During the second world war, the German and Japanese militaries [9, Section 3.4] used *rotor machines*: mechanical devices that relied on rotating cylinders to choose a different substitution for each letter in a message. The initial settings of the rotor machine acted as the secret key. Compared to the simple Caesar Cipher, the ciphers implemented in rotor machines were very difficult—but not impossible—to break.

1.3. *Modern Symmetric Ciphers.* Symmetric cryptosystems remain a staple of modern cybersecurity, and are used in conjunction with asymmetric cryptosystems. Modern symmetric encryption schemes are implemented as computer algorithms. Examples include Advanced Encryption Standard (AES) and Triple Data Encryption Standard (3DES). These systems are very complex: they obfuscate the ciphertext's relationship to the plaintext and the secret key such that it is computationally infeasible to recover either, no matter how much ciphertext is available for analysis [6, Section 3.7].

2. **Asymmetric Ciphers.** As we have seen, all symmetric cryptosystems are essentially the same in principle: they involve the use of an invertible function E_K (for a particular key K) which

performs permutations, substitutions, and perhaps other obfuscations to produce a ciphertext [9, Chapter 9].

While modern symmetric cryptosystems are very secure, they are not appropriate in all circumstances. This is because they assume that participants have already securely exchanged a secret key, which is not always the case. *Asymmetric Cryptosystems*, invented in the 1970s, introduce a radically new paradigm in which each participant has two keys: a *private key* which the participant safeguards, and a *public key* which he/she shares publicly. When a message is encrypted using an individual's public key, it can only be encrypted using that same individual's private key [9, Chapter 9].

Suppose Alice wishes to securely send a message to Bob using an asymmetric cryptosystem. The following might occur:

(1) Alice finds Bob's public key, which he has released in some accessible location such as the internet. She uses this key as an input to the encryption algorithm along with her plaintext.
(2) Alice transmits the encrypted message to Bob over an insecure channel. Her message cannot be decrypted by anybody other than Bob, as only Bob has access to his private key.
(3) Bob recieves the ciphertext and decrypts it using his private key.

How can asymmetric cryptosystems be implemented? To answer this question, we will begin by introducing the fundamental building blocks of such systems. We will then look at some early developments in asymmetric cryptography.

2. Asymmetric Cryptosystems

The symmetric cryptosystems discussed in the introductory section relied on permutations and substitutions performed on the plaintext in various complex ways. Asymmetric cryptosystems, however, take a different approach: they treat plaintext messages as numbers, and then use results from number theory to map these numbers to other numbers (and back again). In practice, asymmetric cryptosystems usually operate on characters encoded as integers (represented in binary) according to the ASCII or UNICODE schemes.

1. One-way Functions and Trapdoors.

We wish to map our "plaintext numbers" to "ciphertext numbers" in a way where only our desired recipient can do the reverse. In particular, we need a

one-way function—a function that is computationally easy to compute in one direction, but whose inverse is computationally difficult to compute[1].

We may also wish that our one-way function would have a *trapdoor*: a piece of information that makes its inverse computation computationally easy. If such a piece of information exists for a one-way function, the function is called a *trapdoor function*.

Do one-way functions and trapdoor functions exist? There is no definitive answer to this question [4, Section 2.1], but we do know of some functions that satisfy our desired properties to the best of our knowledge.

2. **Discrete Logarithm Problem.** A particular problem at the intersection of number theory and abstract algebra has been central to the development of many cryptosystems. This is the Discrete Logarithm Problem (DLP). Its most general form, stated in group-theoretic language, is the following:

Definition 2.1. *(Generalized discrete logarithm problem) Let G be a group containing elements g, h. The Discrete logarithm problem for G is to determine an integer x such that $g^x = h$* [6, Definition 8.3.2].

One form of the discrete logarithm problem used in various public-key cryptosystems is the discrete logarithm problem in the multiplicative group $U(p)$, where p is a prime number. Its statement is the following:

Definition 2.2. *(Discrete logarithm problem in $U(p)$) Suppose that g is a primitive root modulo p and $h \in U(p)$. The discrete logarithm problem for $U(p)$ is to find $x \in \mathbb{N}$ such that*

$$g^x \equiv h \pmod{p}.$$

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

The discrete logarithm problem over $U(p)$ appears to have the properties of a one-way function. Indeed, given g and x, we know of efficient algorithms (such as the Fast Modular Exponentiation discussed in [8, Section 4.1]) to determine h. However, given only h and g, it is not clear how one would determine x other than computing g^x for every possible x until the correct result is found. For very large values of p, it is computationally infeasible to compute x via this method. There exist other, more efficient methods for solving the discrete logarithm problem in $U(p)$, which

[1] Interestingly, the existence of such functions is not confirmed, and remains the holy grail of unsolved problems in computer science [4, Section 2.1]. For certain functions, we have not found an efficient way to compute the inverse—so we assume that no such way exists.

are discussed in [4, Section 2.6]. Right now, there are no known polynomial-time algorithms for solving the DLP in $U(p)$.

3. **Diffie-Hellman Key Exchange.** The discrete logarithm problem seems to have some one-way characteristics: given g and x, it is computationally easy to compute h, whereas it is computationally hard to compute x given g and h. This leads us to the first ever asymmetric cryptographic scheme, which was introduced by Whitfield Diffie and Martin Hellman in 1976 [6, Section 8.1].

Suppose that Alice and Bob wish to share a secret key, but they must communicate over a public/ monitored channel in order to do so. Is it possible for Alice and Bob to exchange a key without the eavesdropper Eve also obtaining the key? As it turns out, the difficulty of the discrete logarithm problem provides a way of performing this seemingly-impossible task. Consider the following procedure:

(1) Alice and Bob agree on a large prime p and an integer $g \in \{2, 3, ..., p-2\}$. They do so over the public channel, so Eve is remains aware of these numbers.
(2) Alice secretly chooses an integer a, which she keeps private. Similarly, Bob chooses a private integer b.
(3) Alice computes the least positive residue $A = g^a \pmod p$ and Bob computes the least positive residue $B = g^b \pmod p$. Alice and Bob then share these numbers with each other.
(4) Alice uses her private integer a to compute $B^a \pmod p$. Similarly, Bob computes $A^b \pmod p$. These least positive residues are equal, and serve as the shared key.

The correctness of the Diffie-Hellman exchange follows [4, Section 2.3]:

Proof. $A^b \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv B^a \pmod p$. □

Since Eve does not have access to Bob and Alice's private integers a and b, she is left with the problem of computing the value of ab in $g^{ab} \pmod p$. This is known as the *Diffie-Hellman Problem*.

Definition 2.3. *Given a prime number p and an integer g, the Diffie-Hellman Problem is the problem of computing $g^{ab} \pmod p$ using the known values of $A = g^a \pmod p$ and $B = g^b \pmod p$.*

The Diffie-Hellman problem is certainly no more computationally difficult than the discrete logarithm problem, but it is currently unknown whether it is computationally easier [4, Section 2.3].

The Diffie-Hellman key exchange is a very useful technique, but it does not allow us to encrypt messages. In the sequel, we will look at an extension of the Diffie-Hellman protocol that allows for encryption and decryption of arbitrary messages. We will then take a detour from discrete

logarithm-based techniques to look at a different encryption scheme that relies heavily on results from number theory.

4. **ElGamal Cryptosystem.** In 1985, Egyptian-American cryptographer Taher Elgamal introduced a method of sending encrypted messages based on the discrete logarithm problem in . Its use is outlined in the following example [4, Section 2.4] featuring Alice and Bob.

First, we set up the cryptosystem by doing the following:

- We choose a large prime number p and an element g modulo p. These numbers are shared among all participants.
- Alice chooses a secret number a as her private key. She also computes $A \equiv g^a \pmod{p}$ as her public key.

Suppose that Bob wants to use Alice's public key to securely send her a message. In particular, Bob's message m is encoded as a number between 2 and p. The exchange would proceed as follows:

(1) Bob first chooses a random number k modulo p, which we call his *ephemeral key*. He uses this key only one.
(2) Bob computes the following two quantities using g and Alice's public key A:

$$c_1 \equiv g^k \pmod{p} \qquad\qquad c_2 = mA^k$$

He sends Alice the pair (c_1, c_2), which serves as his ciphertext.
(3) Alice computes the number $y \equiv c_1^a \pmod{p}$ and its multiplicative inverse $y^{-1} \pmod{p}$. She multiplies the number c_2 by y^{-1}, which yields Bob's message m.

The proof of correctness for the ElGamal Cryptosystem over $U(p)$ can be shown easily [4, Section 2.4].

Proof.

$$\begin{aligned} y^{-1}c_2 &\equiv (c_1^a)^{-1}(mA^k) && \pmod{p} \\ &\equiv (g^{ak})^{-1}(m(g^a)^k) && \pmod{p} \\ &\equiv (g^{ak})^{-1}(g^{ak})m && \pmod{p} \\ &\equiv m && \pmod{p} \end{aligned}$$

□

In order to decode the message, Eve would need to compute y—but $y \equiv c_1^a$, so Eve would have for solve the discrete logarithm problem for $U(p)$.

The Elgamal Cryptosystem is versatile, as it can be based on the discrete logarithm problem in groups other than $U(p)$. In particular, we will revisit the Elgamal Cryptosystem in our discussion of elliptic curve cryptography. Before we proceed to this discussion, however, we will discuss RSA, which was the first public key encryption-decryption system ever developed.

5. **RSA.**

5.1. *The History of RSA.* RSA was the first public-key encryption-decryption system, and is still the most widely-used. It was introduced by three American computer scientists—Ronald Rivest, Adi Shamir, and Leonard Adleman—in 1977 [6, Chapter 7]. The algorithm is based on the difficulty of computing the Euler phi function $\phi(n)$ for a composite number n compared to the ease of computing $\phi(p) = p - 1$ for a prime p.

5.2. *The RSA Algorithm.* The basic encryption and decryption steps are the following [6, Chapter 7]:

(1) Choose two large prime numbers p and q.
(2) Compute their product $n = pq$.
(3) Compute $\phi(n) = (p-1)(q-1)$.
(4) Choose $e = k_{pub} \in \{1, 2, \ldots, \phi(n) - 1\}$, where $\gcd(e, \phi(n)) = 1$.
(5) Compute $k_{pr} = d$ such that $d\,e \equiv 1 \pmod{\phi(n)}$.
(6) Encryption: Given the public key $k_{pub} = (n, e)$ and original message $x \in \mathbb{Z}_n = \{0, 1, \ldots, n-1\}$, we can encrypt into $y = e_{k_{pub}}(x) \equiv x^e \pmod{n}$.
(7) Decryption: Given $k_{pr} = d$, we can decryp by $x = d_{k_{pr}}(y) \equiv y^d \pmod{n}$.

5.3. *How RSA Works.* The security of RSA cryptosystem relies on the computational difficulty of $\phi(n)$ for a large n, where n is the product of two prime numbers. The mathematical theory behind the RSA can be demonstrated with the help of a few critical theorems from elementary number theory.

Lemma 2.1 (Euler's phi function)**.** *For any number $n \in \mathbb{N}$ with $n = pq$, where p and q are two primes, the Euler's phi function can be expressed as $\phi(n) = (p-1)(q-1)$.*

Proof. $\phi(n)$ is a multiplicative function, so we can write $\phi(n) = \phi(p)\,\phi(q)$. Since p and q are prime, $\phi(p) = p - 1$ and $\phi(q) = q - 1$. Hence, we have $\phi(n) = (p-1)\,(q-1)$. □

Lemma 2.2 (Fermat's little theorem)**.** *Let p be a prime number and $a \in \mathbb{Z}$ with $gcd(a, p) = 1$. Then, $a^{p-1} \equiv 1 \pmod{p}$.*

Theorem 2.1 (Theory of RSA)**.** *Given two distinct prime numbers p, q, let $n = pq$, and let e be such that $(e, \phi(n)) = 1$ and d be such that $d\,e \equiv 1 \pmod{\phi(n)}$. Denote the original numeric value x, and let $y \equiv x^e \pmod{n}$ be the encryped value. Then, we can find the original value x by $x \equiv y^d \pmod{n}$.*

Proof. From the previous lemma, we know that $\phi(n) = (p-1)\,(q-1)$. Since $d\,e \equiv 1 \pmod{\phi(n)}$, $ed = 1 + k\,\phi(n) = 1 + k\,(p-1)\,(q-1)$. Now, $y^d = (x^e)^d \equiv x^{1+k(p-1)(q-1)} \pmod{n}$. If $p \nmid x$ and $q \nmid x$, by Fermat's little theorem, we have $x^{(p-1)} \equiv 1 \pmod{p}$ and $x^{(q-1)} \equiv 1 \pmod{q}$. Next, it yields $x^{1+k(p-1)(q-1)} \equiv x \pmod{p}$ and $x^{1+k(p-1)(q-1)} \equiv x \pmod{q}$. On the other hand, if we have either $p \mid x$ or $q \mid x$, we obtain the same results: $x^{1+k(p-1)(q-1)} \equiv x \pmod{p}$ and $x^{1+k(p-1)(q-1)} \equiv x \pmod{q}$. Notice that $pq \nmid x$ as $x < n$. Because p and q are relatively prime, in all the cases, we will have $x^{1+k(p-1)(q-1)} \equiv x \pmod{pq}$. That is, $y^d \equiv x \pmod{n}$, as required.

We also need to verify the uniqueness of our solution. Suppose we have both x_1 and x_2 as our solutions, where $x_1, x_2 \in \mathbb{Z}_n$. Hence, we have $x_1^e \equiv y \pmod{n}$ and $x_2^e \equiv y \pmod{n}$. Then, $x_1^e \equiv x_2^e \pmod{n}$, which implies $x_1^{de} \equiv x_2^{de} \pmod{n}$. Recall that $d\,e = 1 + k(p-1)(q-1)$ with $k \in \mathbb{N}$, and so $x_1 x_1^{k(p-1)(q-1)} \equiv x_2 x_2^{k(p-1)(q-1)} \pmod{pq}$. Also notice that $x_1^{k(p-1)(q-1)} \equiv 1 \pmod{(pq)}$ and $x_2^{k(p-1)(q-1)} \equiv 1 \pmod{(pq)}$ because $(x_1, pq) = 1 = (x_2, pq)$. This gives us $x_1 \equiv x_2 \pmod{n}$. As $x_1,\, x_2 \in \mathbb{Z}_n$, the only possibility is $x_1 = x_2$. So there is only one unique solution to $x^e \equiv t \pmod{n}$.

Putting together the first and second parts of our proof, we can safely conclude that $x \equiv y^d \pmod{n}$ is the unique solution to $x^e \equiv y \pmod{n}$. □

5.4. *Security of RSA.* To the best of our knowledge, the most efficient way to break RSA encryption is by attempting to factor the modulus n into its constituent primes p and q. RSA has historically used a 1024-bit modulus as a standard, but recent advancements in factoring algorithms and computer hardware have indicated that we may soon be capable of factoring such numbers. Hence, we should migrate to larger moduli: a 4096-bit modulus should remain safe for a long time to come [6, Section 7.8].

With increasingly large moduli being required for RSA, we might wish to look for alternative cryptosystems with shorter operands and shorter ciphertexts. The search for such an appropriate algebraic structure brings us to elliptic curves, which we will discuss in detail.

3. ELLIPTIC CURVE CRYPTOGRAPHY

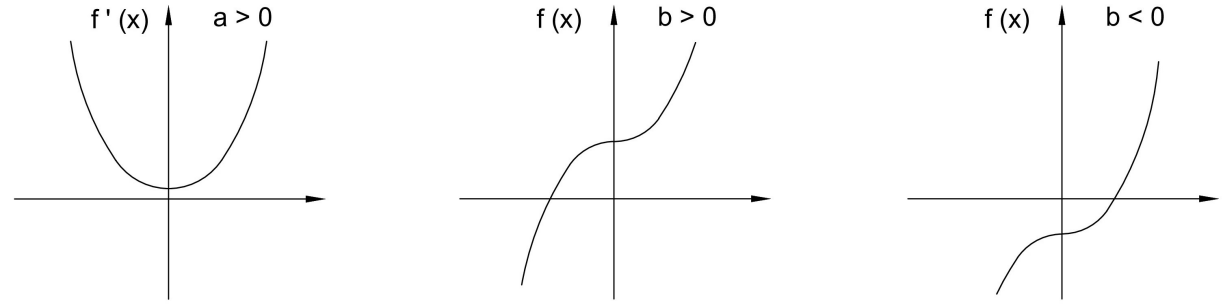
1. **Elliptic Curves.** Before we study elliptic curve cryptography, we first introduce elliptic curves. It is important to remember that elliptic curves are not ellipses. A formal definition of an elliptic curve will require much in-depth background in algebraic geometry, so instead we use the following simplified definition for our purpose:

Definition 3.1 (Elliptic Curve)**.** *An elliptic curve is a curve of the form*

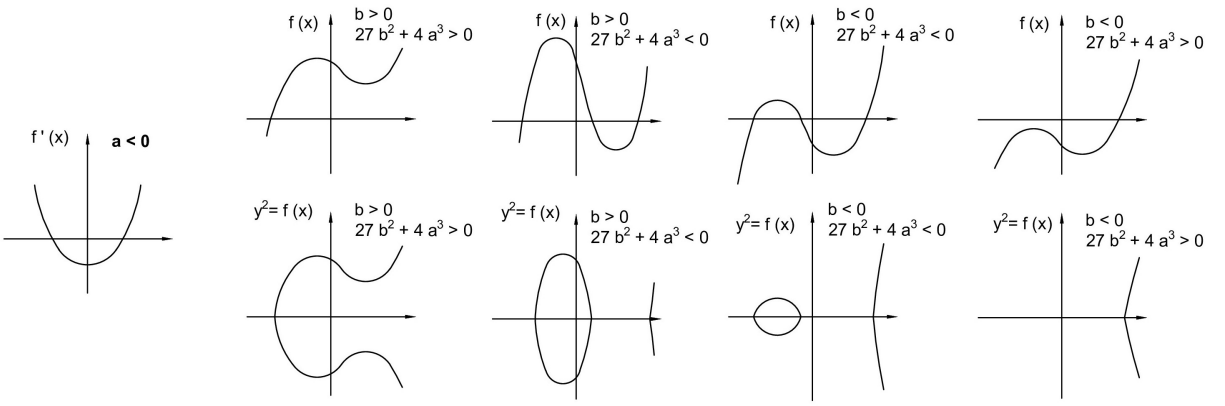
$$y^2 = x^3 + ax + b, \tag{1}$$

where $4\,a^3 + 27\,b^2 \neq 0$ *and* $a, b \in \mathbb{R}$ [4, Section 5.1].

The condition $4\,a^3 + 27\,b^2 \neq 0$ is to ensure that the elliptic curve has three distinct roots including complex roots. In order to visually grasp elliptic curves over $\mathbb{R}$, we generated some examples in Figure 1 and drew some observations from them.



(a) case I: $a > 0$



(b) case II: $a < 0$

Figure 1. $f(x) = x^3 + ax + b$

Here are some observations:

(1) An elliptic curve is continuously differentiable on $\mathbb{R}^2$, so geometrically, the curve is smooth, and it does not have any cusps.
(2) A curve is symmetric along $x-$axis, because $(-y)^2 = y^2 = x^3 + ax + b$.
(3) It is unbounded, because we can always have large x and y that satisfy equation 1.

Now, we introduce the algorithm to sketch an elliptic curve by hand.

(1) Given an elliptic curve $y^2 = x^3 + ax + b$, let $f(x) = x^3 + ax + b$.
(2) We differentiate $f(x)$ to obtain $f'(x) = 3x^2 + a$, which is a parabola. So the sign of a determines the behavior of $f'(x)$. If a is non-negative, then $f'(x) \geq 0$ for all $x \in \mathbb{R}$, and so $f(x)$ is non-decreasing. And with the sign of b, we can easily sketch $f(x)$ (see Fig 1a).
(3) If a is negative, then $f'(x) < 0$ for some interval, then the trend of $f(x)$ will be increasing, decreasing and increasing again. Depending on the value of b, the curve will have positive or negative $y-$intercept; and depending on the value of $27b^2 + 4a^2$, the curve might or might not have a negative local minimum (see Fig 1b).
(4) Lastly, since y^2 is non-negative, we let $y^2 = f(x)$ for $f(x) \geq 0$. Then, for the upper branch, we approximate $y \sim f(x)$, and for the lower branch, we have $y \sim -f(x)$ because the curve is symmetric across $x-$axis. This is how we obtained six types of elliptic curves in Fig 1. It is not accurate, but a good approximation of the shape of a curve.

2. **The Abelian group on elliptic curves.** In terms of group theory, we are interested in an *Abelian group* defined on the elliptic curves [7, Chapter 14]. Let the points on elliptic curves be elements of the group, and the point at infinity be the identity, which is assumed on every vertical line. We also define the group operation addition as follows: given two points A and B on the curve, the line connecting A and B will intersect with the curve at another point C. The reflection C' of C across the x-axis is $A + B$. Let's first explore the property of all the points in $\mathbb{R}^2$. We will denote the set and its operation by $(E, +)$, and we will show that it is a well-defined Abelian group.

(1) *The operation is commutative*
Clearly, when joining two points on the curve with a line, the order of (x_1, y_1) and (x_2, y_2) does not matter, ie, $(x_1, y_1) + (x_2, y_2) = (x_2, y_2) + (x_1, y_1)$.
(2) *The operation $+$ is closed in E.*
Let (x_1, y_1) and (x_2, y_2) be any two elements in E. There are two cases. First, if $x_1 = x_2 = c$, then the line joined by the two elements $x = c$ and $y^2 = x^3 + ax + b$ gives $y^2 = c^3 + ac + b$, so only y_1 and y_2 satisfy the equation. In other words, the line does not intersect at another finite point other than (x_1, y_1) and (x_2, y_2). Recall that the point

infinity lies on every vertical line, so in this case the addition of $(x_1,\, y_1)$ and $(x_2,\, y_2)$ would have to be the reflection of infinity, which is infinity itself. Second, if $x_1 \neq x_2$, then we would expect the line $y = kx + c$ joined by $(x_1,\, y_1)$ and $(x_2,\, y_2)$ to have another point $(x_3,\, y_3)$ on the elliptic curve, because a cubic equation with two distinct real roots must have another distinct real root. Now, we combine equations $y^2 = x^3+ax+b$ and $y = kx+c$ to obtain $x^3 - k^2x^2 + (a - 2kc)\, x + b - c^2 = 0$. The three distinct roots are related as $x_1 + x_2 + x_3 = k^2$. Thus, we have $(x_3,\, y_3) = (k^2 - (x_1 + x_2),\, -k^4 + k^2(x_1 + x_2) - c)$. In either case, we showed that $+$ is a closed operation for E.

(3) *An identity is included.*

For any point $(x,\, y)$ on the curve, if we connect it with infinity, we create a vertical line, which intersects at another finite point. The reflection of that intersecting point gives back the point (x, y). This verifies that infinity is the identity element.

(4) *An inverse is included.*

For ant point $(x,\, y)$ on the curve, we claim that $(x,\, -y)$ is the inverse. Notice that $(x,\, -y)$ is on the elliptic curve since it is symmetric along $x-$ axis. Also, the vertical line formed by $(x,\, y)$ and $(x,\, -y)$ intersect with the curve at infinity. So $(x,\, y) + (x,\, -y)$ is the identity, which is infinity.

(5) *The operation is associative.*

The proof by horrific computation using the formula of addition is skipped there. We will just accept that $+$ is associative. For those interested in a complete proof, please refer to [5].

A special case occurs when we wish to double a point P. The line connecting P to P is defined as the tangent line at P. Using this information, we are able to find a formula for the addition of points on elliptic curves.

Theorem 3.1 (Elliptic curve addition formula)**.** *Given a point $P : (x_0, y_0)$ and $Q : (x_1, y_1)$ on the elliptic curve $y^2 = x^3 + ax + b$, we have $P + Q = (\lambda^2 - x_0 - x_1,\, \lambda^3 - 2\lambda x_0 - \lambda x_1 + y_0)$, where*

$$\lambda = \begin{cases} \dfrac{3x_0^2 + a}{2y_0},\ P = Q \\ \dfrac{y_1 - y_0}{x_1 - x_0},\ P \neq Q \end{cases} .$$

Proof. • *Case one ($P = Q$):* Differentiating $y^2 = x^3 + ax + b$ on both sides with respect to x gives $y' = \dfrac{3x_0^2 + a}{2y_0}$, which is the slope of the tangent line $y = \dfrac{3x_0^2 + a}{2y_0}\, x + \dfrac{2y_0^2 - 3x_0^3 - ax_0}{2y_0}$. Let $\lambda = \dfrac{3x_0^2 + a}{2y_0}$, then $y = \lambda\, x + y_0 - \lambda\, x_0$. Substitute the line into the curve to obtain $(\lambda\, x+y_0-\lambda\, x_0)^2 = x^3+ax+b$, and thus, $x^3-\lambda^2x^2+[a-2\lambda(y_0-\lambda x_0)]\, x+$

$b-(y_0-\lambda x_0)^2=0$. Since we already know that x_0 is a repetitive root, we expect the cubic equation in form of $(x-x_0)^2(x-x_2)=0$. From here, we see the other root $x_2=\lambda^2-2x_0$. Then $y_2=\lambda^3-3\lambda x_0+y_0$. So $P+Q=2P=(x_2,\,y_2)=(\lambda^2-2x_0,\,\lambda^3-3\lambda x_0+y_0)$, where $\lambda=\dfrac{3x_0^2+a}{2y_0}$.

- *Case two ($P\neq Q$):* The line joined by P and Q is $y=\lambda x+y_0-\lambda x_0$, and we combine it with $y^2=x^3+ax+b$ to obtain $x^3-\lambda^2x^2+[a-2\lambda(y_0-\lambda x_0)]\,x+b-(y_0-\lambda x_0)^2=0$. Since $(x_0,\,y_0)$ and $(x_1,\,y_1)$ are solutions, we have $(x-x_0)(x-x_1)(x-x_2)=0$, and so $x_2=\lambda^2-x_0-x_1$. Then, $y_2=\lambda x_2+y_0-\lambda x_0=\lambda^3-2\lambda x_0-\lambda x_1+y_0$. Hence, $P+Q=(x_2,\,y_2)=(\lambda^2-x_0-x_1,\,\lambda^3-2\lambda x_0-\lambda x_1+y_0)$.

□

3. **Elliptic curves over finite fields.** Previously, we studied elliptic curves over $\mathbb{R}$. Now, let us turn to elliptic curves over a finite field $\mathbb{F}_p$. We introduce finite fields as follows in order to study elliptic curves over $\mathbb{F}_p$.

Definition 3.2. *A finite field is a field with a finite, prime number of elements. We denote a finite field by $\mathbb{F}_p$. A finite field has operations that satisfy the basic rules as we see in basic algebra. Moreover, it has four operations: addition, subtraction, multiplication, and division are well defined.*

Definition 3.3. *An elliptic curve E over $\mathbb{F}_p$ is an equation in form of $y^2=x^3+ax+b$ with $A,B\in\mathbb{F}_p$ and $4a^3+27b^2\neq 0$, and we denote the curve $E(\mathbb{F}_p)=\{(x,\,y)\in\mathbb{F}_p\times\mathbb{F}_p:\,y^2\equiv x^3+ax+b\pmod p\}\cup\{\infty\}$.*

Similarly, we define an elliptic curve over rational numbers as $E(\mathbb{Q})$= $\{(x,\,y)\in\mathbb{Q}\times\mathbb{Q}\colon\,y^2=x^3+ax+b\}$ and over integers as $E(\mathbb{Z})$ = $\{(x,\,y)\in\mathbb{Z}\times\mathbb{Z}:\,y^2=x^3+ax+b\}$.

Before, we explored elliptic curves and their properties over the field $\mathbb{R}$. Now we are interested in elliptic curves over finite fields $\mathbb{F}_p$. It is worth mentioning that the points on an elliptic curve over $\mathbb{F}_p$ is still a group, and the proof is analogous to the previous one. All the group properties and operations defined previously on an elliptic curve still apply, but we cannot expect any geometric interpretation for them. We will stick to the same formula for addition of two points. Because addition is well defined in $\mathbb{F}_p$, the four operations of two points will still be on $E(\mathbb{F}_p)$. Here are some examples generated by SageMath.

Example. We have an elliptic curve E defined by $y^2=x^3+ax+b$. We code the following:

```
E = EllipticCurve(GF(p),[a,b])
```

```
E.points()
```

Now, we have the following examples:

(1) $E : y^2 = x^3 + 1$, $E(\mathbb{F}_5) = \{\infty, (0,1), (0,4), (2,2), (2,3), (4,0)\}$
(2) $E : y^2 = x^3 - 2x + 3$, $E(\mathbb{F}_7) = \{\infty, (1,3), (1,4), (2,0), (6,2), (6,5)\}$
(3) $E : y^2 = x^3 + 11x - 15$,
$E(\mathbb{F}_{13}) = \{\infty, (1,6), (1,7), (5,3), (5,10), (8,0), (10,4), (10,9), (12,5), (12,8)\}$

From the above, we observe in $E(\mathbb{F}_p)$ that each x coordinate corresponds to two y coordinates, say y_1 and y_2, which are identical if and only if $y_1 = y_2 = 0$. Moreover, $y_1 + y_2 \equiv 0 \pmod{p}$.

Next, our concern is the number of points on an elliptic curve $y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ [4, Section 5.2]. Let us start with number probability. There are p possibilities for the value of $x^3 + ax + b$, and x_0 is a solution if $x_0^3 + ax_0 + b$ is a perfect square modulo p. So the number of points on the curve is related to the number of perfect squares in $\mathbb{F}_p$. If we square every element in $\mathbb{F}_p$, we will get $\frac{p-1}{2} + 1$ distinct perfect squares, because n and $p - n$ induce the same square, except when $n = 0$. So in about 50% of the time, $x^3 + ax + b$ has square roots, and most likely two distinct ones. Including a point at infinity, we have about $2\,(\frac{p-1}{2} + 1) + 1 \sim p + 1$ solutions to the curve. The following theorem gives a more precise estimation, but its proof is beyond the scope of the project.

Theorem 3.2 (Hasse)**.** *Let E be an elliptic curve over $\mathbb{F}_p$. Then the number of points on the curve $\#E(\mathbb{F}_p) = p + 1 - t_p$ with $|t_p| \leq 2\sqrt{p}$* [4, Section 5.2].

4. **Elliptic Curve Discrete Logarithm Problem.** The *elliptic curve discrete logarithm problem (ECDLP)* is a computational problem analogous to the discrete logarithm problem discussed earlier. Its formal definition is the following.

Definition 3.4. *(Discrete logarithm problem over elliptic curves) Let E by an elliptic curve over a finite field $\mathbb{F}_p$, and let A and B be points on E. The Elliptic Curve Discrete Logarithm Problem (ECDLP) is the problem of finding an integer n such that*

$$A = nB$$

The number n is called the elliptic discrete logarithm of A with respect to B, and it is denoted by $\log_B A$.

4.1. *Double-and-add algorithm.* There exists an efficient algorithm for computing $A = nB$ given n and B. The interested reader may consult [4, Section 5.3] for a description of this algorithm, which is very similar to the fast modular exponentiation algorithm discussed in [8, Section 4.1].

4.2. *Hardness of ECDLP.* As with the discrete logarithm problem in $U(p)$, it is computationally infeasible to naively compute n by trial-and-error for elliptic curves $E(\mathbb{F}_p)$ where p is very large. In fact, the ECDLP is thought to be more difficult than the DLP in $U(p)$. A more detailed discussion of the computational hardness of the problem is outside the scope of this project, but more information can be found in [4, Chapter 5].

5. **Elliptic curves and cryptography.** Now that we have defined elliptic curves and shown that they form an Abelian group, we wish to develop cryptosystems using them. We will begin by talking about how we might generate public-private key pairs for use in the Elliptic Curve Diffie-Hellman Exchange and beyond.

5.1. *Elliptic curve key generation.* Let E be an elliptic curve defined over a finite field $\mathbb{F}_p$. Let P be a point in $E(\mathbb{F}_p)$, and suppose that P has prime order n. Then the cyclic subgroup of $E(\mathbb{F}_p)$ generated by P is

$$\langle P \rangle = \{\infty, P, 2P, 3P, \ldots, (n-1)P\}$$

The prime p, the equation of the elliptic curve E, and the point P and its order n, are the public domain parameters. A private key is an integer d that is selected uniformly at random from the interval $[1, n-1]$, and the corresponding public key is $Q = dP$. To reiterate, key generation follows the following steps:

(1) Select $d \in [1, n-1]$ as the private key
(2) Compute $Q = dP$ as the public key.

5.2. *Elliptic curve Diffie-Hellman.* The Diffie Hellman problem over an elliptic curve $E(\mathbb{F}_p)$ is similar to that over $U(p)$. If Alice and Bob wished to exchange a key via this method, they would do the following [4, Section 5.4]:

(1) Alice and Bob agree on a particular elliptic curve E and a point $Q \in E$.
(2) Bob chooses a secret integer b and computes $B = bQ$. Alice chooses a secret integer a and computes $A = aQ$.
(3) Alice and Bob exchange their values.

(4) Alice and Bob each compute their shared secret value $aB = a(bQ) = b(aQ) = bA$.

For Eve to discover Bob and Alice's shared secret value, she is left to solve the ECDLP to obtain a or b given $A, B, andQ$.

5.3. *Elliptic ElGamal public key cryptosystem.* A plaintext m is first represented as a point M, and then encrypted by adding it to kQ where k is a randomly selected integer, and Q is the intended recipient's public key. The sender transmits the points $C_1 = kP$ and $C_2 = M + kQ$ to the recipient who uses her private key d to compute

$$dC_1 = d(kP) = k(dP) = kQ$$

and thereafter recovers $M = C_2 - kQ$. An eavesdropper who wishes to recover M needs to compute kQ. This task of computing kQ from the domain parameters, Q, and $C_1 = kP$, is the elliptic curve analogue of the Diffie-Hellman problem.

6. **Encoding with Elliptic Curves.** Although elliptic curve cryptosystems are very robust and secure, they are not without their quirks. In particular, it is not immediately clear how to encode messages using them. In order to encode a message in string, we first need to convert them to numeric values (usually binary) by UNICODE, which can convert every language in the world to their unique numeric correspondence. This step is not interesting, but next, the challenging problem is constructing a hash function to map those numbers to points on an elliptic curve.

Encoding is an algorithm f to map a message $m \in \{0,1\}^l$ to elements in a group G. Optimally, we would like to have the bit size of the messages l to be close to the bit size of the group G. If that is the case, we call the algorithm f an *injective encoding* [3]. This is not always achievable for all elliptic curve groups, but one known example was discovered on the curve $E : y^2 = x^3 + b$ over finite field $\mathbb{F}_p$ with $p \equiv 2 \pmod 3$ by Boneh and Franklin [1]. The optimal map is defined as $f : \mathbb{F}_p \to E(\mathbb{F}_p) \setminus \{\infty\}$ given by $f(u) = ((u^2 - b)^{\frac{1}{3}}, u)$.

Next, we give a process of constructing injective encoding. Assume we have a message in binary values $m \in \{0,1\}^l$. We choose a prime $p > 5$ such that $p \geq 2^{\frac{2}{1-\epsilon}l}$ for some fixed constant $0 < \epsilon < \frac{1}{2}$. Then, we apply the following algorithm. Note that $\mathcal{F}$ is probabilistic not an actual mapping.

Input: an elliptic curve $E(\mathbb{F}_p)$,
Output: $\mathcal{F} : \{0,1\}^l \to E(\mathbb{F}_p)$
Random integer generator:

$x \leftarrow x_n x_{n-1} \ldots x_{l+1} x_l \ldots x_2 x_1 \in \{0, 1, \ldots, p-1\}$ in base 2 with $x_l \ldots x_2 x_1 = m$
If $(x \pmod p, y) \in E(\mathbb{F}_p)$ for some $y \in \mathbb{F}_p$
    then return $\mathcal{F}(m) = x$
Else
    goto Random integer generator

Let us estimate roughly the probability of generating a point on the elliptic curve. Recall that previously we estimated the number of points on an elliptic curve E over $\mathbb{F}_p$, to be around $p+1$; if we eliminate the point at infinity and the possible point with zero y coordinate which have unique x coordinate, we are left with $\frac{p-1}{2}$ choices of x coordinates on the curve. Now the generated random integer x modulo p falls between 0 and $p-1$, thus, the chance of x being an x coordinate of the curve $E(\mathbb{F}_p)$ is $\dfrac{(p-1)/2}{p+1} \sim \dfrac{1}{2}$ for large p. The following is a more accurate result on the efficiency of the algorithm:

Theorem 3.3. *If p is large enough, the expected number of iterations in $\mathcal{F}$ on any input is less than* 3.

The rigorous proof uses the Hasse's theorem 3.2, Bombieri's bound and Erdös Turán-Koksma inequality to show that the compiling time for this algorithm is bounded, and can be found in [3].

This algorithm is shown to be efficient in encoding a message in its binary form; however, it is not sure if it is genuinely injective, ie, how large a prime is needed to encode messages of certain length. Besides, we are uncertain about any methods to recover the encoded message.

7. **Elliptic Curve Cryptosystem versus other Crytosystems.** Why should elliptic curve cryptosystems be used over the more straightforward cryptosystems we discussed earlier? The main reason is that elliptic curve cryptosystems seem to offer similar security to RSA, but with a smaller key size and faster processing times [9, Chapter 10]. As a matter of a fact, a 256-bit ECC key is equivalent in security to a 3072-bit RSA key [6, Section 9.7]. This is of great benefit in limited bandwidth situations, as the increasingly-large key sizes in RSA are becoming cumbersome.

4. Conclusion

When the authors of this paper began their work, they seriously underestimated the level of sophistication in the topic of cryptography. Indeed, the goal of "writing a paper about cryptography" was perhaps a tad ambitious in scope. Nevertheless, the authors hope that the paper provides

a coherent overview of some of the most important cryptographic topics from an algebraic and number-theoretic perspective.

For the interested reader, a plethora of sources books on cryptography are available. Those interested in gaining a practical, working knowledge of cryptography may be interested in Christof Paar's book *Understanding Cryptography* [6] and its accompanying lectures posted freely online. Those with a craving for mathematical depth, however, may find more substance in Jeffrey Hoffstein's *Introduction to Mathematical Cryptography* [4]. Finally, the those readers who are focused on elliptic curve cryptography should certainly consult Lawrence Washington's *Elliptic Curves: Number Theory and Cryptography* [10].

REFERENCES

[1] Boneh, Dan, and Matt Franklin. 2001. "Identity-Based Encryption from the Weil Pairing." In Advances in Cryptology — CRYPTO 2001, edited by Joe Kilian, 213–29. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-44647-8_13.

[2] Buchmann, Johannes. 2004. Introduction to Cryptography. 2nd edition. Undergraduate Texts in Mathematics. New York, NY: Springer.

[3] Fouque, Pierre-Alain and Joux, Antoine and Tibouchi, Mehdi. 2013. Injective Encodings to Elliptic Curves, 203–218, Information Security and Privacy Lecture Notes in Computer Science.

[4] Hoffstein, Jeffrey, Jill Catherine Pipher, and Joseph H. Silverman. 2008. An Introduction to Mathematical Cryptography. Undergraduate Texts in Mathematics. New York; London: Springer.

[5] Lang Serge, 1987, Elliptic functions. New York, Verlag: Springer.

[6] Paar, Christof, and Jan Pelzl. 2010. Understanding Cryptography: A Textbook for Students and Practitioners. Heidelberg; New York: Springer.

[7] Rubinstein-Salzedo, Simon. 2018. Cryptography. New York, NY: Springer Berlin Heidelberg.

[8] Rosen, Kenneth H. 2011. Elementary Number Theory and Its Applications. 6th ed. Boston: Addison-Wesley.

[9] Stallings, William. 2017. Cryptography and Network Security: Principles and Practice. Seventh edition, Global edition. Boston: Pearson.

[10] Washington, Lawrence C. 2003. Elliptic Curves: Number Theory and Cryptography. Discrete Mathematics and Its Applications. Boca Raton: Chapman & Hall/CRC.

Email address: schau@unbc.ca

Email address: wlan@unbc.ca

Email address: shaw5@unbc.ca