A comparison between elliptic curve cryptography and other methods to determine which is the most secure.

How does elliptic curve cryptography compare to other methods of encryption?

Mathematics

Word Count = 2869

Contents

1 Introduction **1**

2 Background Math for Asymmetric Encryption **1**

2.1 Modular Arithmetic 1

2.2 Group Theory 3

2.2.1 Euler's Phi Function 5

2.2.2 **Fermat's Theorem and Euler's Theorem** 6

2.2.3 **Euclidean Algorithm** 6

2.2.4 **Extended Euclidean Algorithm** 7

3 RSA Encryption **9**

4 Diffie-Hellman Key Exchange **10**

5 Elliptic Curve Cryptography **12**

6 Highest amount of Security **15**

7 Conclusion **16**

1 Introduction

From Bitcoin to iMessage texts, all data sent over the public space of the internet needs to be encrypted to prevent malicious parties from accessing the data. The two main methods of encryption over a public channel are the Rivest, Shamir, and Adleman (otherwise known as RSA) encryption and encryption methods based on the discrete logarithm problem (DLP), such as elliptic curve (or ECC) encryption and ElGamal encryption. These are asymmetric encryption methods, which establishes a secure method of communication over a public channel as opposed to symmetric encryption which needs a private channel and requires more keys to be generated. With elliptic curve cryptography being the newest method, **how does elliptic curve cryptography compare to other methods of encryption**? Each of these encryption methods might have different ways of generating the keys, but the foundational math behind each one is relatively similar.

2 Background Math for Asymmetric Encryption

2.1 Modular Arithmetic

Modern cryptography is preformed in finite sets. The English alphabet is an example of a finite set with 26 elements. If I want to know what is the letter from $A + 10$ letters, I take A's position in the set, which is 0 and add 10 giving me 10. The 10th letter is K, but what if I want to know what letter is 10 from X. X is the 23rd letter, and there is no 33rd letter in the set, so how do I compute this? Well there are 26 elements in the set, so I would do $(23+10)/26 = 1$ remainder 7, and we use the remainder of the division because the quotient is how many times the divisor fit into the set, and whats left over is the amount of units we need to move from the beginning of the set, which in this case the 7th letter is H. A simpler way of doing this arithmetic is using the modulo operator. Modular arithmetic is a way of preforming an operation in a finite set, specifically we use a modulus operator to create this finite set and cycling back pattern where $a, r, m \in \mathbb{Z}$ and $m > 0$.

$$a \equiv r \mod m$$

(Paar & Pelzl, 2009, p. 14) This is the specific notation used. The 3 lines mean equivalent and r is the remainder after a divides m. For example, if I want to calculate the letter 20 after Y(the 44th letter) I would do $(24+20) \equiv 18 \mod 26$, and the 18th letter is S. In order to find the remainder given a and m, we can use

$$a = q \cdot m + r$$

where q is the quotient. The remainder is found after a/m is computed(aka the quotient). For example, $80 = 3 \cdot 26 + 2$ where $(80 - 2) = 3 \cdot 26$, however it should be noted that the remainder is not unique since $80 = 4 \cdot 26 + -24$ where $(80 - -24) = 4 \cdot 26$. These set of remainders are part of the same class modulo 26 or equivalent when a certain a value is used, hence the $\equiv$ symbol. For modulo 26 and a as the constant 80, the equivalence class is

$$\{... - 50, -24, 2, 28, 54, 80, 106...\}$$

What about the numbers missing from this class of elements modulo 26? Well we can generate more classes depending on the a value, for example $26 \equiv 0 \mod 26$ because it doesn't have a remainder and $52 \equiv 26 \mod 26$ as well. So we have the equivalence classes,

$$\{... - 50, -24, 2, 28, 54, 80, 106...\}$$

and

$$\{... - 26, 0, 26, ...\}$$

What about an equivalence class with 1 or 3 as an element?

$$\{... - 25, 1, 27...\}$$

$$\{..., -49, -23, 3, 29, 55, ...\}$$

This happens for all elements up to $m-1$

$$\vdots$$

$$\{...,-27,-1,25,51,...\}$$

You can generate all $\mathbb{Z}$ using equivalence classes. However, we only need the elements 0-25 to preform any arithmetic because the other elements are just repeats, so we only consider 0-25 to be part of the set modulo 26. For example, $(55 \cdot 106) - -50 \equiv ? \mod 26$, if we do modular arithmetic, we can substitute the smallest positive integer or 0 to give us the result modulo 26. So $(3 \cdot 2) - 0 \equiv 6 \mod 26$ would be the simplified form. Modular arithmetic helps to simplify the computation required to preform the arithmetic by reducing the set of elements to however big the modulus is! These sets form what are called cyclic groups.

2.2 Group Theory

Group Theory is a foundational aspect of asymmetric encryption. According to Christof Paar and Jan Pelzl in *Understanding Cryptography: A Textbook for Students*, a group's definition is:

A group is a set of elements G together with an operation $\circ$ which combines two elements of G. A group has the following properties:

1. The group operation $\circ$ is closed. That is, for all $a, b, \in G$, it holds that $a \circ b = c \in G$.
2. The group operation is associative. That is, $a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in G$.
3. There is an element $e \in G$, called the neutral element (or identity element), such that $a \circ e = e \circ a = a$ for all $a \in G$. This identity element can be 0 or 1 depending on the operation.

4. For each $a \in G$ there exists an element $a^{-1} \in G$, called the inverse of a , such that $a \circ a^{-1} = a^{-1} \circ a = e$.

5. A group G is abelian (or commutative) if, furthermore, $a \circ b = b \circ a$ for all $a, b \in G$.

(Paar & Pelzl, 2009, p. 92) For encryption based on the DLP, cyclic groups are used with a finite set G or $\mathbb{Z}_p^*$(this * means we exclude zero from multiplicative groups because they have no inverse) number of elements, in fact, the number of elements is referred to as the "order" or "cardinality" of G and is a prime. The operator is either addition for elliptic curves or multiplication for ElGamal encryption and other methods. The reason we need a prime as the modulus is because groups with an composite modulus will not have an inverse for some elements in there set. For example, $2 \cdot 5 \equiv 1 \mod 9$ because $(10/9)$ gives a remainder of 1, which is the neutral element for multiplication. However, $6 \cdot a^{-1} \equiv 1 \mod 9$ doesn't exist because the two share a common divisor(aka $gdc(6,9) = 3$). A prime is used because it won't have any factors with any of the other elements ensuring there is a inverse for every element. Even with a prime modulus, we would want to find an element in the set that can generate the rest of the elements $a^k = \underbrace{a \circ a \circ a \circ ... \circ a \circ a}_{k} = 1$. For instance,

$$\mathbb{Z}_{13}^{\cdot} = \{1,2,3,4,5,6,7,8,9,10,11,12\}$$

has an order of 12(the symbol used to denote the order is ϕ with $\phi(12)$). If a is 2 we can generate

$$\begin{aligned}
a^1 &= 2 \mod 13 = 2\\
a^2 = a \cdot a &= 4 \mod 13 = 4\\
a^3 = a^2 \cdot a &= 8 \mod 13 = 8\\
a^4 &= 16 \mod 13 = 3\\
a^5 = a^4 \cdot a = 16 \cdot 2 &= 3 \cdot 2 \mod 13 = 6\\
a^6 = a^5 \cdot a &= 6 \cdot 2 \mod 13 = 12\\
a^7 = a^6 \cdot a &= 12 \cdot 2 \mod 13 = 11\\
a^8 = a^7 \cdot a &= 11 \cdot 2 \mod 13 = 9\\
a^9 = a^8 \cdot a &= 18 \mod 13 = 5\\
a^{10} = a^9 \cdot a = 5 \cdot 2 &= 10 \mod 13 = 10\\
a^{11} = a^{10} \cdot a = 10 \cdot 2 &= 20 \mod 13 = 7\\
a^{12} = a^1 1 \cdot a = 7 \cdot 2 &= 14 \mod 13 = 1
\end{aligned} \tag{1}$$

We have now found a generator $a^{\phi 12}$ is able to compute all the elements since $a^{12} = 1$ the neutral element. In order to easily compute the number of elements that are generators in an order we need to introduce some theorems.

2.2.1 Euler's Phi Function

Christof Paar and Jan Pelzl in *Understanding Cryptography: A Textbook for Students* define the function as "The number of integers in $\mathbb{Z}_m$ relatively prime to m is denoted by $\phi(m)$". (2009, p. 165) This will help us compute all elements in a set that are relatively prime to the order, and if an element is, it can be classified as a generator. The quickest way to compute the order or $\phi(m)$ given we know the prime factorization of m is by using:

$$\phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1})$$

. Not knowing the prime factorization would make computing this computationally impossible for sufficiently large values of $n(2^{2000})$ being used since the prime numbers are quite large.

2.2.2 Fermat's Theorem and Euler's Theorem

Two more theorems that are less important but still used frequently in cryptography are Fermat's and Euler's theorems, because they provide useful rules to help with operations in the group. Fermat's Little Theorem is defined as:

Let a be an integer and p be a prime, then:

$$a^p = a \mod p$$

(Paar & Pelzl, 2009, p.167). Essentially this is saying that if you have the element a from the prime set p, a^p will be in the same equivalence class as a. Euler's Theorem is defined as:

Let a and m be integers with $\gcd(a,m) = 1$, then:

$$a^{\phi(m)} \equiv 1 \mod m$$

(Paar & Pelzl, 2009, p.167) which is what I showed in the example above with $a = 2$ and $m = 13$.

2.2.3 Euclidean Algorithm

The Euclidean algorithm provides a way to find the greatest common divisor (gcd) of 2 positive integer numbers. Usually, to find the gcd of two numbers, we take their prime factorization and use the greatest common factor. However, prime factorization will take too long to compute for the large numbers used in encryption. The notation commonly used for the gcd and two numbers is $\gcd(r_0,r_1)$ where $r_0 > r_1$ and we can reduce r_0 by using modulo r_1 to get

$$\gcd(r_0,r_1) = \gcd(r_0 \mod r_1, r_1)$$

or

$$\gcd(r_0, r_1) = \gcd(r_1, r_0 \mod r_1)$$

(Paar & Pelzl, 2009, p.158) since $(r_0 \mod r_1) < r_1$. We can verify this for $r_0 = 42$ and $r_1 = 27$ where

$$\begin{aligned} \gcd(42 \mod 27, 27) = \\ \gcd(27, 15) = \gcd(27 \mod 15, 15) \\ \gcd(15, 12) = \gcd(15 \mod 12, 12) \\ \gcd(12 \mod 3, 3) = 3 \end{aligned} \tag{2}$$

We calculated the remainder throughout the operation using $a = q \cdot m + r$. When the remainder = 0, we are done with the algorithm.

2.2.4 Extended Euclidean Algorithm

We can then take the Euclidean Algorithm and write the gcd as a linear integer combination in the form:

$$\gcd(r_0, r_1) = (s \cdot r_0) + (t \cdot r_1)$$

(Paar & Pelzl, 2009, p.160) This extended version is primarily used to find the inverses of elements in the set. In order to find s and t, we first compute the normal Euclidean Algorithm where

$$r_0 = q_1 + r_2$$

where

$$r_2 = s \cdot r_0 + t_1$$

This continues for all iterations till we reach the last remainder, which will give us the values of s and t by substituting the previous remainders from the iteration in terms of $(s \cdot r_0 + t \cdot r_1)$. For example:

$$r_1 = q_2 \cdot r_2 + r_3 = q_2(s_0 + t_1) + r_3$$

and we can rearrange to solve for r_3. Christof Paar and Jan Pelzl in *Understanding Cryptography: A Textbook for Students* run through an example of how the Extended Euclidean Algorithm(EEA) is computed.

$$\begin{aligned}
\gcd(973, 301) = s \cdot 973 + t \cdot 301 \\
973 = 3 \cdot 301 + 70 \\
r_2 = 70 = s \cdot 973 + t \cdot 301 = 1 \cdot 973 + -3 \cdot 301 \\
301 = 4 \cdot 70 + 21 \\
r_3 = 21 = 301 - (4 \cdot 70) = 301 - 4((1 \cdot 973) + (-3 \cdot 301)) = -4 \cdot 973 + 13 \cdot 301 \\
70 = 3 \cdot 21 = 7 \\
r_4 = 7 = 70 - 3 \cdot 21 = ((1 \cdot 973) + (-3 \cdot 301) - 3((-4 \cdot 973) + (13 \cdot 301)) = \\
13 \cdot 973 + -42 \cdot 301 \\
\text{where s = 13 and t = -42}
\end{aligned} \tag{3}$$

(2009, p.161) We can use this to find the inverse as Christof Paar and Jan Pelzl in *Understanding Cryptography: A Textbook for Students* show because

$$a^{-1} \cdot a \equiv 1 \mod n$$

where

$$\gcd(n, a) = 1 = s \cdot n + t \cdot a \equiv 1 \mod n$$

$s \cdot n \mod n = 0$ but $0 + t \mod n = 1$ showing that t the is the inverse of a.

3 RSA Encryption

Asymmetric encryption has 2 types of keys: a public key(K_{pub}) available to everyone and a private key(K_{priv}) generated. These two keys are used to create a connection through computation in the RSA algorithm. Alice and Bob are the names of the people who want to establish a connection and share a message with one another, and Oscar is the attacker who is trying to intercept and read the data they are trying to exchange.

Commonly Used Notation:

p,q = large primes(around 2^{512} or 2^{1024} depending on level of security wanted)

$n = p \cdot q$

e = public key or encryption function

d = private key or decryption function

In order to generate the public key, we choose a element e from the set from $\{1,2,...,\phi(n-1)\}$ where e is relatively prime to $\phi(n)$. Since e and $\phi(n)$ are relatively prime the inverse must exist. K_{priv} or d will be that inverse to e. We can use the Extended Euclidean Algorithm to find d since we know the prime factorization of n. If Alice wants to send Bob a message, she requests his public key containing n and e, but Bob never shares the primes p and q protecting him from Oscar breaking the key using Euler's Phi function and subsequently the Extended Euclidean algorithm to find the inverse, aka the private key. To encrypt,

given $Kpub = (n,e), x \in \mathbb{Z}_n = \{1,2,...,n-1\}$ where x is the plain-text we want to encrypt

$$y = e_{K_{pub}}(x) \equiv x^e \mod (n)$$

y is the cipher-text and to decrypt given $K_{priv} = d, y \in \mathbb{Z}_n$

$$x = d_{K_{priv}}(y) \equiv y^d \mod (n)$$

(Paar & Pelzl, 2009, p.178-179) since $y = x^e$ and $d = e^{-1}$ then $y^d = (x^e)^{-e} = x^{e-e} = x$

An example of this encryption in the smaller scale can be:

$$p = 5, q = 13$$

$$n = 5 \cdot 13 = 65$$

$$\phi(n) = (5-1)(13-1) = 48$$

$$e = 7$$

$$\gcd(7, 48) = 1$$

Bob will then send Alice 65 and 7. Alice will encrypt using these numbers and her plain-text:

$$x = 2$$

$$2^7 = 128 \mod (65) = 63 = y$$

Alice sends Bob the encrypted data y, and Bob does the decryption:

$$d \cdot e \equiv 1 \mod (48) = d \cdot 7 \equiv 1 \mod (48) = 7 \mod (48)$$

$$x = 63^7 \mod (63) = 2 \tag{4}$$

The time it takes to do this exponentiation of x^e for large values of e presents a problem. However, we can use the "square and multiply algorithm" (Paar & Pelzl, 2009, p.181) to quickly find this out. For example, $x^{26} = x^{11010}$ in binary. For every place value we square x to produce a zero and then if we want to fill that place value with a 1 we multiply by x. This makes the computation of the RSA Algorithm much faster.

4 Diffie-Hellman Key Exchange

The main method of establishing a connection between 2 parties (commonly referred to Alice and Bob) is using the Diffie-Hellman key exchange to have the parties generate each a private key and public key to encrypt and decrypt the messages between the 2 parties. There are two public parameters, p and α, where p is a large prime and α is a generator of the cyclic group created by mod p.

Common Notation:

$$\begin{aligned}
a &= K_{prA} \in \{2,3,...,p-2\} \\
A &= \alpha^a \mod (p) \equiv K_{pubA} \\
b &= K_{prB} \in \{2,3,...,p-2\} \\
B &= \alpha^b \mod (p) \equiv K_{pubB}
\end{aligned} \tag{5}$$

Alice and Bob share their public keys with one another. Then compute $A^b \mod (p)$ and $B^a \mod (p)$ to produce their shared key K_{AB} that they will use for encryption and decryption for future communication. If Alice wants to send a message to Bob, she can use:

$$y \equiv x \cdot K_{AB} \mod (p)$$

and to decrypt Bob can use:

$$x \equiv y \cdot K_{AB}^{-1} \mod (p)$$

where the inverse of the shared public key is calculated using the extended euclidean algorithm. The only way Oscar can break the encryption would be from finding out what a or b is, but this is computationally impossible because of the Discrete Logarithm Problem where $a = \log_{\alpha} A \mod (p)$ because of the size of the prime p. If you were to brute force the number of solutions it would take the order of p to find a,for instance if the prime is 2^{128} bits that would be 1 billion years to crack the encryption. There are other methods of attack Oscar can take such as the square root attack, which reduces the number of bits needed to compute by $\sqrt{n}$, so 2^{128} bits would become 2^{64} bits of security. This method of attack is the best known attack for Elliptic Curve Cryptography, but RSA and ElGamal (which is a slightly modified version of the Diffie-Hellman protocol to run faster) have the square root attack and another form of attack called index calculus attacks that can reduce the bits of security even further. The index calculus attacks work on RSA and ElGamal because they have multiplicative groups, while Elliptic Curve Cryptography uses additive groups, as I will explain in the next section.

5 Elliptic Curve Cryptography

There were two operations mentioned that cyclic groups can have: addition or multiplication. Well, elliptic curves use a form of point addition to do their modular arithmetic. An elliptic curve in the use of cryptography is defined as:

$$y^2 \equiv x^3 + ax + b \mod (p)$$

(Paar & Pelzl, 2009, p.241) The elliptic curve over $\mathbb{Z}_p, p > 3$ is the set of all pairs $(x, y) \in \mathbb{Z}_p$. Together with an imaginary point at infinity Θ, where $a, b \in \mathbb{Z}_p$ and $4a^3 + 27b^2! \equiv 0 \mod (p)$. The abstract point at infinite is used as the identity element of the group to meet the group laws. Also, the last equation ensures we only use elliptic curves, which are accepted as the standard for cryptography. The way we perform point addition for elliptic curves is by picking two points that are part of our set, and drawing a line through them. The next point the line intersects is then mirrored to give us $P + Q = R$. If a point is being added to itself, then we take the tangent line to the point to find a second point and mirror it to find the addition of P+P. The definition allows elliptic curves to work as cyclic groups because they have a finite set with the modulus operator, a neutral element defined at infinity, and the inverse of an element. Elliptic curves are y^2 equations, meaning every y value has a corresponding negative y value. These two points are opposite one another, which makes them the inverse. An example of point addition and point doubling is shown in figures 1 and 2 respectively. The way we can compute the point addition analytically is through finding the $y = mx + b$ equation of the straight line going through our point and substitute the $mx + b$ for y^2 in the equation $y^2 \equiv x^3 + ax + b \mod (p)$. Since the equation has a degree of 3, we know there are 3 roots and we know the first two since they are our first two points, and the last root is the x-value of the addition of our two points. The equations are shown in figure 3. In order to form our public and private keys we need still the generator and a large p value for the set. However, instead of the private key being raise to the generator(α^a), it is instead multiplied by the generator($\alpha \cdot a$) because it is the number of times that the generator is added to it's self. The result

Figure 1: This is an example of point addition represented graphically. (Buchanan, Adding points in elliptic curve cryptography 2019)

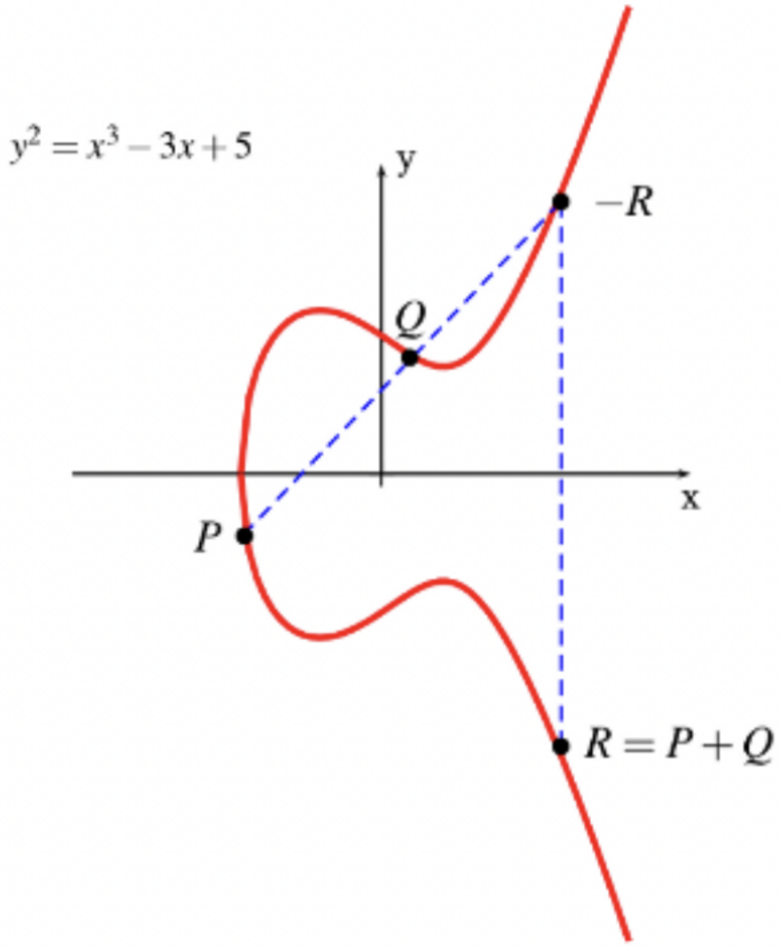


Figure 2: This is an example of point doubling by taking the tangent line. (Panchbhai & Ghodeswar, 1970)

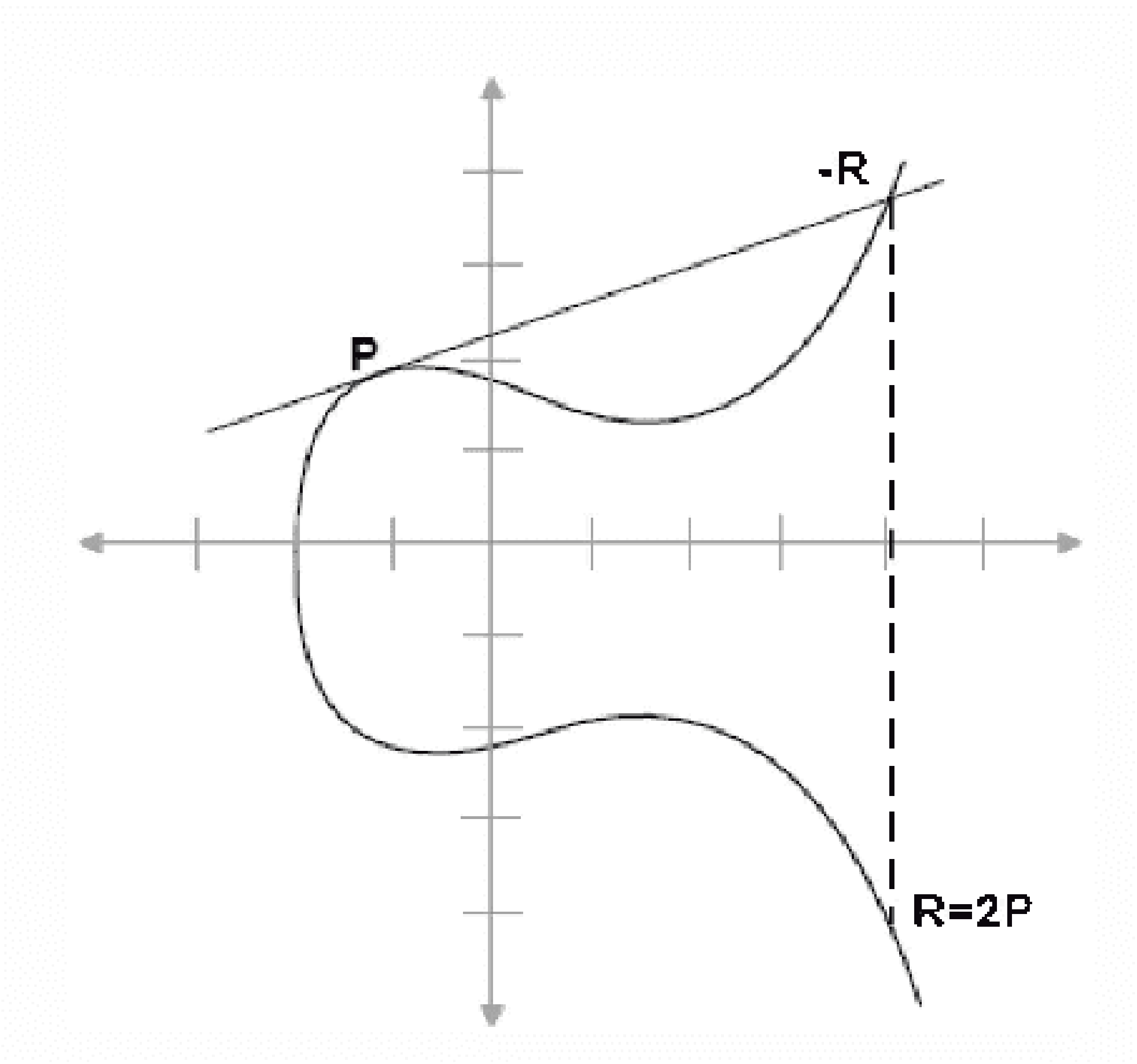
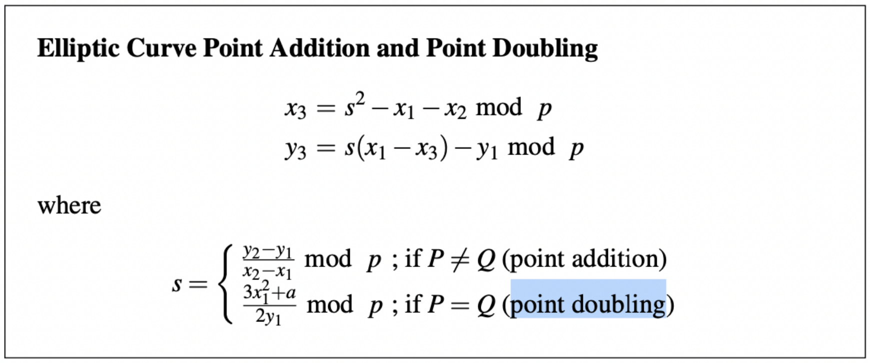


Figure 3: The equations used to find the third point analytically. (Paar & Pelzl, 2009, p.244)



Elliptic Curve Point Addition and Point Doubling

$$x_3 = s^2 - x_1 - x_2 \bmod p$$
$$y_3 = s(x_1 - x_3) - y_1 \bmod p$$

where

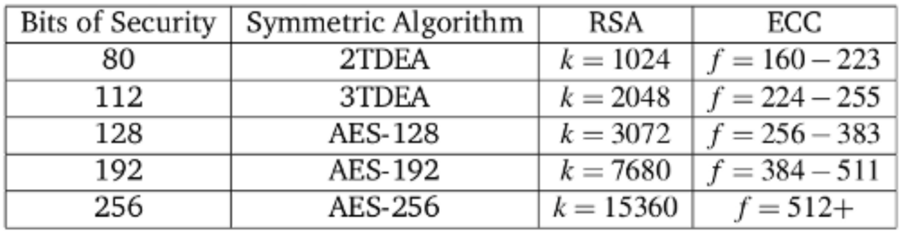
$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \bmod p \text{ ; if } P \neq Q \text{ (point addition)} \\ \frac{3x_1^2 + a}{2y_1} \bmod p \text{ ; if } P = Q \text{ (point doubling)} \end{cases}$$

of $\alpha \cdot a \mod (p) \equiv K_{pubA}$ which is then shared with Bob who multiplies his private key to produce the shared key K_{AB}, which can now be used to encrypt and decrypt communication between the two parties.

6 Highest amount of Security

When taking into consideration what type of algorithm to use for encryption and decryption, we should take into account the number of computational bits required to achieve a certain number of security bits. Also, the longer key length with more bits required to achieve a certain standard of security results in longer computation times because the computer takes longer to compute the larger values. The different methods of attack discussed in the paper, from the square root attack to the index calculus attack, create situations where cryptographers need to compute larger key sizes for the same amount of security, resulting in slower times to establish a connection. The elliptic curve cryptography method of encryption is only affected by the square root attack, while the other encryption methods need to produce much larger key sizes for the same amount of security due to attacks like the index calculus attack. This vulnerability makes these other algorithms computationally inferior to elliptic curve cryptography. If you double the key sizes for elliptic curve cryptography, you achieve the same amount of bits of security before-hand, while RSA needs up to needs 60 times the amount of bits to achieve the same strength as seen in figure 4.

Figure 4: This table shows the differences in the key sizes with respect to a certain level of bit security(Dams, An introduction to elliptic curve cryptography 2022)



Bits of Security	Symmetric Algorithm	RSA	ECC
80	2TDEA	$k = 1024$	$f = 160 - 223$
112	3TDEA	$k = 2048$	$f = 224 - 255$
128	AES-128	$k = 3072$	$f = 256 - 383$
192	AES-192	$k = 7680$	$f = 384 - 511$
256	AES-256	$k = 15360$	$f = 512+$

7 Conclusion

With advancements in computing, such as quantum computers, we don't know exactly how these algorithms will hold up, but we should use the most efficient and secure way to protect our data and information. It has been clearly shown that elliptic curve cryptography is the fastest and most secure way to achieve this. While RSA might have been the standard for over 20 years, the development of different attack methods and increase in computational power has made it obsolete. It is still the most widely implemented form of asymmetric encryption in areas like web browsing and banking, but RSA has become outdated. Transitioning over to elliptic curve cryptography has already become standard for new applications like Bitcoin, but the legacy code using RSA on the internet should be updated to the more secure method of encryption: Elliptic Curve Cryptography.

References

Buchanan, B. (2019, May 12). Adding points in elliptic curve cryptography. Retrieved August 23, 2022, from https://medium.com/asecuritysite-when-bob-met-alice/adding-points-in-elliptic-curve-cryptography- a1f0a1bce638

Dams, J. (2022, February 02). An introduction to elliptic curve cryptography. Retrieved August 23, 2022, from https://www.embedded.com/an-introduction-to- elliptic-curve-cryptography/

Paar, C., &amp; Pelzl, J. (2009). Understanding cryptography: A Textbook for Students and Practitioners. Springer Science &amp; Business Media. Panchbhai, M., & Ghodeswar, U.

(1970, January 01). Figure 3 from implementation of point addition & point doubling for elliptic curve: Semantic scholar. Retrieved August 23, 2022, from https://www.semanticscholar.org/paper/Implementation-of- point-addition -%26-point-doubling-Panchbhai- Ghodeswar/8571543aba5d9c34809709580d536dc57e2a48b6/figur e/2