Elliptic Curve Cryptography

Professors: Manish K.Gupta , Manoj Raut

Group Members and their Contribution

1. **Dev Changela(202101069)**
   Contribution: Research[RSA] & Youtube Video

2. **Jay Sabva(202101224)**
   Contribution: Latex(.TeX) File & Research[ECC] & Content

3. **Swet Lakhani(202101218)**
   Contribution: Bibliography & Research[DHKA]

4. **Soham Viradiya(202101472)**
   Contribution: WikiPage Content & Web Implementation

5. **Vandit Bhalani(202101478)**
   Contribution: Research[ECC] & Youtube Video

6. **Shrut Kalathiya(202101479)**
   Contribution: Research[ECC-DHKA] & Youtube Video

7. **Vasu Golakiya(202101487)**
   Contribution: WikiPage Design & Web Implementation

   Visit Our : WikiPage Youtube Video GitHub

Contents

1 Introduction and History of Cryptography

The two eras in the history of cryptography are the classical era and the modern era.The introduction of the RSA algorithm and the Diffie-Hellman key exchange algorithm in 1977 established the separation of the two.These new algorithms were earth-shattering because they were the first effective cryptographic algorithms whose security was based on the theory of numbers;they were also the first to make it possible for two parties to communicate securely without sharing a secret.When it comes to communication between any two parties,cryptography has advanced from being used to safely carry codebooks around the globe to allowing for provably secure key exchanges between two parties.

The basis of modern cryptography is the concept that the key used to encrypt data may be made public but the key required to decode it can be kept secret. These systems are referred to be public key cryptography systems as a result. The first and currently most popular of these systems is called RSA, after the names of the three individuals who initially revealed the algorithm to the public: Ron Rivest, Adi Shamir, and Leonard Adleman. A group of algorithms that are simple to process in one way but difficult to reverse is necessary for a public key cryptography system to function. The simple procedure for RSA multiplies two prime values. Factoring the product of the multiplication into its two component primes is the tough pair algorithm, if multiplication is the simple method. Trap door Functions are a set of algorithms that are fast in one direction and difficult in the other. Creating a safe public key cryptography system requires finding a decent trapdoor function.Simply said, a cryptographic system based on a Trapdoor Function will be more secure the greater the difference between the complexity of moving in one way and going in the other.[1]

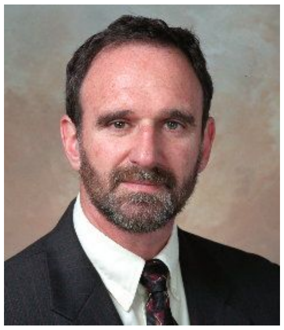


Figure 1: Martin Hellman

Figure 2: Rivet-Shamir-Adleman

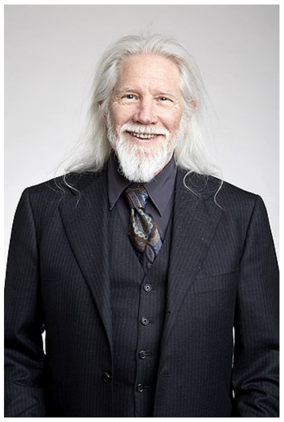


Figure 3: Whitefield Diffle

2 RSA algorithm

A widely used public-key cryptosystem for secure data transfer is RSA (Rivet-Shamir-Adleman). The names Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly revealed the algorithm in 1977, are the source of the abbreviation "RSA." The encryption key of a public-key cryptosystem is distinct from the decryption key, which is kept confidential (Private). A user of RSA generates and distributes a public key based on two significant prime numbers and an auxiliary value. It is a hidden what the prime numbers are. Anyone can encrypt messages using the public key, but only someone who is familiar with the prime numbers can decode them. The "factoring issue," which is the practical challenge of factoring the product of two huge prime numbers, is the foundation for RSA security. The RSA problem is the process of deciphering RSA encryption.. It is arguable if it is pretty hard as the factoring problem. If a sufficiently enough key is shared, the system cannot be attacked using any known techniques. The RSA algorithm operates quite slowly.

Let's use an illustration to help make this more clear. If we combine the prime numbers 13 and 7, we get our maximum value of 91. Let's say that the number 5 is our public encryption key. The Extended Euclidean Algorithm is then used to determine that the private key is the number 29, given the knowledge that 7 and 13 are factors of 91.

$$\boxed{public\ key\ (e)\ : 5 \quad private\ key\ (d)\ : 29 \quad maximum\ key\ (n)\ : 91}$$

Let's use these values to encrypt the message "$CLOUD$" by encrypting the values. We must convert the characters into numbers in order to represent a message numerically.UTF-8 is a popular way to represent the Latin alphabet.A number is assigned to each character.

A	B	C	D	E	F	G	H	I	J	K	L	M
65	66	67	68	69	70	71	72	73	74	75	76	77
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
78	79	80	81	82	83	84	85	86	87	88	89	90

$CLOUD$ is 67,76,79,85,68 in this encoding. These numbers may all be encrypted separately because they are all less than our limit of 91. Let's Start with the initial letter, To obtain the encrypted value for C=67, we must multiply it by itself five times (using the public key).

$$\boxed{Encrypted\ Data\ E\ = A^e * (modn)} \qquad \boxed{Decrypted\ Data\ D\ = L^d * (modn)}$$

$$EncryptedDataC\ = 67^5 mod 91 = 58 \qquad DecryptedDataC\ = 58^{29} mod 91 = 67$$
$$EncryptedDataL\ = 76^5 mod 91 = 20 \qquad DecryptedDataL\ = 20^{29} mod 91 = 76$$
$$EncryptedDataO\ = 79^5 mod 91 = 53 \qquad DecryptedDataO\ = 53^{29} mod 91 = 79$$

$$\begin{array}{llll} EncryptedDataU & = 85^5 mod 91 = 50 & DecryptedDataU & = 50^{29} mod 91 = 85 \\ EncryptedDataD & = 68^5 mod 91 = 87 & DecryptedDataD & = 87^{29} mod 91 = 68 \end{array}$$

Therefore, it is clear that the keys are not symmetric, makes decryption more difficult.

2.1 Encryption

Let's say Alice wants to tell her buddy Bob something private.A single letter "C" is the message. Assuming that her sworn enemy Eve may read this letter while it is in transit, how can she deliver it in confidence?

Example: The public key, or "Lock" (5,14) Alice (the sender) uses this public pair of numbers to encrypt messages. The private key is the "Key" (17,14) This pair's first number is private, meaning that only Bob is aware of it (the receiver). To decrypt messages, this pair is used.

1. The message "C" is encrypted by Alice using (5,14).

   a. "C" is first converted into an integer."C" corresponds to the number 67 in ASCII. To keep things simple, let's assume it maps to 3 instead.

   b.3 is then encrypted $3^5 mod(14) = 5$ receives the result, 5, from Alice. Even if Eve peeks and notices this message, it says nothing to them.

2. Bob uses the resulting decryption to (17,14).

   a. $5^{17} mod(14) = 3$

   b.The result is 3. To obtain the original message, Bob maps 3 back to a character, which is "C".

2.1.1 Lock & Key Generation

Bob generates these two points of number in our illustration. So that only he can decrypt messages, he makes the first pair of numbers (5,14) publicly available to everyone while keeping the second pair of numbers (17,14) secret.

1. Bob Pick p and q, two prime numbers. They should be large in real-world scenarios (for security), but we'll pick modest numbers to make things simpler.

$$\boxed{p = 2 \quad q = 7}$$

2. Compute

$$\boxed{n = p * q}$$

3. Compute e(for encryption) such that:

$$\boxed{1 < e < (p-1)(q-1) \qquad and \qquad GCD(e, (p-1)(q-1) = 1}$$

4. Compute d (for decryption) such that:

$$\boxed{d = e^{-1} mod((p-1)(q-1))}$$

$$\boxed{d * 5 = 1 mod 6}$$

This means d can be 5,or 11,or 17,and so on.We"ll choose d=17. That's it,we're done

$$\boxed{The\,"Lock" \; : (e, n) \; = \; (5, 14)}$$

$$\boxed{The\,"Key" \; : (d, n) \; = \; (17, 14)}$$

As clearly explained, messages may be encrypted by senders using the first pair of numbers, and decrypted by receivers using the second set of numbers.

The fundamental idea behind RSA is that it is possible to obtain positive numbers e, d, and n such that $(m^e \, mod n)^d \, mod n = m$ and such that even if e and n are made public, it would be very difficult to figure out what d is

The key point is that you may take a number, multiply it by itself a certain number of times to get a totally arbitrary number, and then multiply that number by itself a different, secret number of times to return to the original number. As the size of the numbers being factored increases, these algorithms become more effective. As the number (the key's bit length) grows more, the difference in difficulty between multiplying large numbers and factoring large numbers is less. The size of the keys must expand even more quickly as the resources available to decipher numbers grow. For portable, low-power devices with limited processing capability, this is an unsustainable position. Factoring and multiplication are separated by an impossible distance.[2] [3]

2.2 Issues with RSA[Motivation]

The largest number factored as of 2005 is 663 bits long and was done so using distributed methods that are state-of-the-art. Although argued, experts believe that 1024-bit keys may eventually become breakable. Using a home computer, 256-bit length keys may be cracked in a short amount of time. The 2048-bit key length is currently proposed [wik]. Although most personal computers are not affected by this length, it renders low processing power portable devices like smartcards ineffective. In these devices, there are limitations on processor word length, available memory, and clock speeds.It became necessary to develop a system that provided the same level of cryptographic security with shorter key lengths as the requirement for portable and secure identification progressively becomes a need and as RSA key sizes will increase in proportion to the processing power available. One such method is ECC, which is discussed in the section below.

All of this simply indicates that RSA is not the best solution for the cryptography of the future.

In a perfect trapdoor function, the difficulty of the easy and difficult paths increases according to the numbers under consideration.[4]

Key Comparison Between RSA and ECC

Symmetric Key Size(bits)	RSA-Diffie-Hellman Key Size(bits)	Elliptic Curve Key Size(bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

[5]

3 Diffie-Hellman key exchange

3.1 Introduction

In order to solve the issue of key agreement and exchange, Whitefield and Martin Hellman developed the Diffie-Hellman key exchange algorithms in 1976. After Whitefield Diffe and Martin Hellman, it was one of the earliest public-key exchange protocols developed by Ralph Merkle. It makes it possible for the two parties to agree on a symmetric key and communicate with one another (a key that can only be used for encrypting and decrypting). This algorithm cannot be used for encryption or decryption; it can only be used for key exchange. Mathematical concepts are the foundation of the algorithm.

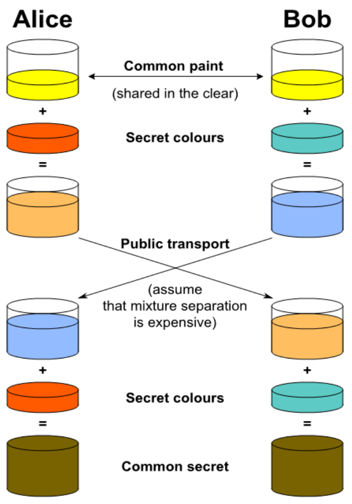
3.2 Reason to use Algorithm

A key can be attacked between the instant it is being sent and received (i.e. Insecurity may occurs). It enables the establishment of a shared secret key across an unsecure channel by two people with no prior knowledge.

3.3 General Overview of an algorithm

Publicly agreeing on a random starting colour that is not private, Alice and Bob (but should be different every time). Yellow is used as an example. Additionally, each participant chooses a secret colour that they keep to themselves; in this example, the colours are red and cyan. The key step in the procedure is when Alice and Bob mix their respective secret colours with the colour they both share to produce combinations of orange-tan and light blue, respectively, before publicly exchanging the two colours. Finally, they each combine their personal colour with the color they received from their buddy. The outcome is a final colour mixture (in this example, yellow-brown) that is the same as their partner's final colour mixture. It would be challenging for a third party to determine the final secret colour from the exchange if it just knew the common colour (yellow) and the first mixed colours (orange-tan and light blue) (yellow-brown).

Figure 4: Illustration of Diffie-Hellman key exchange



3.4 Mathematical Explanation

The multiplicative group of integers modulo p, where p is prime, and g is a primitive root modulo p are used in the protocol's initial and most basic implementation. In assure that the resulting shared secret can have any value between 1 and $p-1$, these two values were selected in this manner. Here is an illustration of the protocol,with non-secret values: blue,and secret values in red

1. Here Alice and Bob wants to share there secret but Eve is keeping watch on Them So Alice and Bob publicly agree to use a modulus p=23 and base $g = 5$(primitive root of modulo 23).
2. Alice choose a secret integer $a = 4$,then sends Bob $A = g^a \bmod p$
   - $A = 5^4 \bmod 23$=4
3. Bob choose a secret integer $b = 3$,then sends Alice $B = g^b \bmod p$
   - $B = 5^3 \bmod 23$=10
4. Alice computes s= $B^a \bmod p$
   - s= $10^4 \bmod 23$=18
5. Bob computes s= $A^b \bmod p$
   - s= $4^3 \bmod 23$=18
6. Alice and Bob now share a secret

Both Alice and Bob arrived at the same values because under mod p,

$$\boxed{A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p}$$

More specifically

$$\boxed{(g^a \bmod p)^b \, mod\, p \; = \; (g^b \bmod p)^a \, mod\, p}$$

Only a and b are kept secret.All the other values – p, g, g^a mod p, and g^b mod p – are sent in the clear. The strength of the scheme comes from the fact that g^{ab} mod p = g^{ba} mod p take extremely long times to compute by any known algorithm just from the knowledge of p, g, g^a mod p, and g^b mod p. Once Alice and Bob compute the shared secret they can use it as an encryption key, known only to them, for sending messages across the same open communications channel.

Of course, much larger values of a, b, and p would be needed to make this example secure, since there are only 23 possible results of n mod 23. However, if p is a prime of at least 600 digits, then even the fastest modern computers using the fastest known algorithm cannot find a given only g, p and g^a mod p. Such a problem is called the **discrete logarithm problem**.The computation of g^a mod p is known as modular exponentiation and can be done efficiently even for large numbers. Note that g need not be large at all, and in practice is usually a small integer (like 2, 3, ...)[6]

3.4.1 Secrecy Table

Alice	
Known	Unknown
$p = 10$	
$g = 18$	
$a = 4$	b
$x = 18^4 (mod 10) = 6$	
$K_a = 8^4 (mod 10) = 6$	

Bob	
Known	Unknown
$p = 10$	
$g = 18$	
$b = 5$	a
$y = 18^5 (mod 10) = 8$	
$K_b = 6^5 (mod 10) = 6$	

Eve	
Known	Unknown
$p = 23$	
$g = 9$	
	a, b
$x = 6, y = 8$	
	K_A, K_B

After the development of RSA and Diffie-Hellman, researchers have explored further mathematically based cryptographic techniques in search of more algorithms than factoring that would provide ideal trapdoor functions. On the basis of an unusual area of mathematics known as elliptic curves, cryptographic algorithms were first proposed in 1985.

4 Elliptic curve Cryptography

A method of public key cryptography based on the algebraic stricture of elliptic curves over finite fields is known as elliptic-curve cryptography. ECC enables similar security to be achieved with fewer keys as compared to non-EC encryption (based on simple Galois Fields). Neal Koblitz and Victor Miller each made a separate suggestion for the method in 1985. The Elliptic Curve Discrete Logarithm Problem, an NP-Hard problem, is the core of the ECC. In 2004 to 2005, elliptic curve encryption methods became widely used. The equation describes an elliptic curve.

$$\boxed{y^2 = x^3 + ax + b}$$

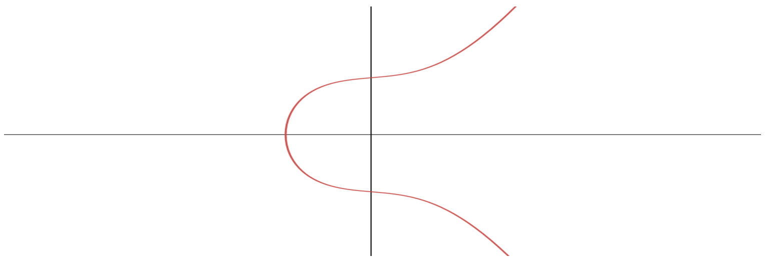


Figure 5: Elliptic Curve $y^2 = x^3 + ax + b$

Elliptic curves cryptography uses elliptic curves over the fintie field Z_p(where p is prime and $p > 3$) or Z_{2m} (where the fields size p=2m).This indicates that the curve's point can only have integer coordinates within the field, which is a square matrix of size $(p * p)$. Each algebric operation performed on the field (such as point addition and multiplication) produces a new point. **The Elliptic curve equation over the finite field** Zp

$$\boxed{y^2 = x^3 + ax + b(modp)}$$

The Bitcoin curve(secp256k1) equation over the finite field Zp

$$\boxed{y^2 = x^3 + 7(modp)}$$

The ECC uses the points (x, y) within the Galois field Zp (where x and y are integers in the range [0...p-1]), in comparison to RSA, which uses the integers in the range [0...p-1] for its key space. [7]

4.1 Mathematics

Given a curve,E,defined along some equation in a finite field(such as E:$y^2 = x^3 + ax + b$,point multiplicatioj is defined as the repeated addition of a point along that curve.denote as $nP = P + P + P... + P$ for some scalar n and a point $P = (x, y)$ that lies on the curve,E.

4.1.1 Point Operations

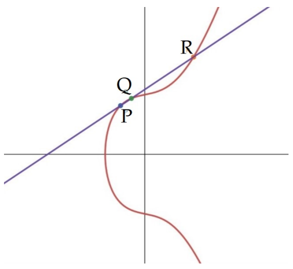


Figure 6: $P+Q+R=0$

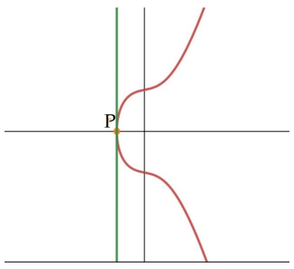


Figure 7: $P+P+0=0$

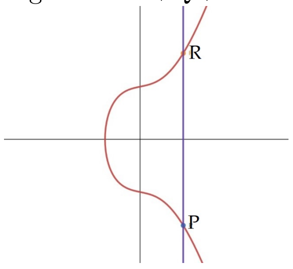


Figure 8: $P+R+0=0$

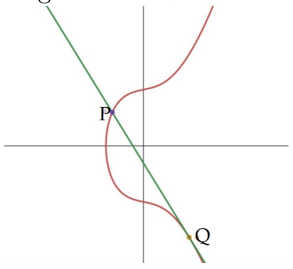


Figure 9: $P+Q+Q=0$

Point at Infinity

Point at infinity $\mathcal{O}$ is the identity element of elliptic curve arithmetic.Adding it to ant point result in that other point,including adding point at infinity to itself.That is

$$\boxed{\mathcal{O}+\mathcal{O}=\mathcal{O}}$$

$$\boxed{\mathcal{O}+\mathcal{P}=\mathcal{P}}$$

Point negation

Point negation is finding such a point,that adding it to itself will result in point at infinity($\mathcal{O}$).

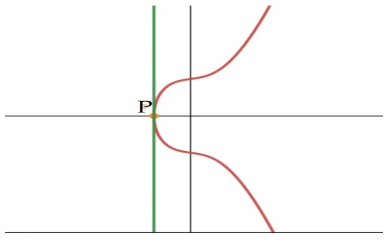


Figure 10: Point negation $P+P+0=0$

$$\boxed{\mathcal{P} + (-\mathcal{P}) = \mathcal{O}}$$

For elliptic curves that is a point with same x coordinate but negated y coordinate:

$\boxed{(x,y) + (-(x,y)) = \mathcal{O}}$ $\boxed{(x,y) + (x,-y)) = \mathcal{O}}$ $\boxed{(x,-y) = -(x,y)}$

Point addition
Given Two point in the set $E = (x,y) \,|\, y^2 + ax + b \cup \mathcal{O}$

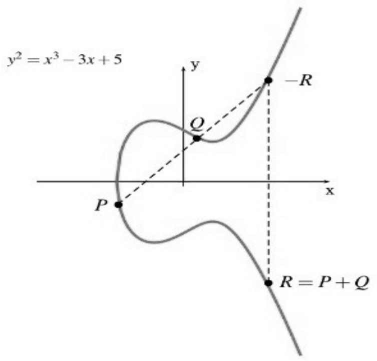


Figure 11: Point addition $P + Q = R$

With 2 distinct points,P and Q,addition is defined as the negation of the point resulting from the intersection of the curve,E,and the straight line defined by the points P and Q giving the point,R

$$\boxed{\text{P} + \text{Q} = \text{R}}$$

Assuming the elliptic curve,E, is given by $y^2 = x^3 + ax + b$, so the slope of line (PQ) λ can be calculated as:

$$\boxed{\lambda = \frac{y_q - y_p}{x_q - x_p}}$$

$$\boxed{x_r = \lambda^2 - x_p - x_q} \quad \boxed{y_r = \lambda((x_p - x_r) - y_p}$$

These equations are correct when neither point is the point at infinity,O,and if the points have diffrent x coordinates(they're not mutual inverses).This is important for the Elliptic curve digital singature algorithm(ESDSA) where the hash value could be zero.

Point doubling
Where the point P and Q are coincident(at same coordinate),addition is similar,except that there is no well-defined straight line through P,so the operation is closed using limiting case,the tangent to the curve,E,at P.

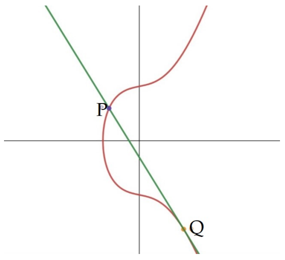


Figure 12: Point doubling $P + Q + Q = 0$

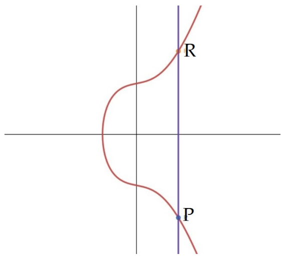


Figure 13: Point doubling $P + R + 0 = 0$

$$\boxed{\lambda = \frac{3x_p^2 + a}{2y_p}}$$

where a is from the defining equation of the curve,E,above.

$$\boxed{x_r = \lambda^2 - 2x_p} \quad \boxed{y_r = \lambda((x_p - x_r) - y_p}$$

4.1.2 Order and Cofactor of Elliptic Curve

All the points on an elliptic curve over a finite field can be combined to form a finite cyclic algebraic group. When two EC points from the same cyclic group are combined or multiplied by an integer, the result is another EC point from the same cyclic group (and on the same curve). The order of the curve is determined by the total number of EC points on the curve. The special point known as "$point\ at\ infinity$," which is produced when a point is multiplied by 0, is also included in this total number of points.

Others form a single cyclic group that contains all of the curve's EC points, while other curves generate numerous non-overlapping cyclic subgroups that each contain a portion of the curve's EC points. In the second case, the curve's points are split into h cyclic subgroups, each of order r. (each subgroup holds equal number of points). The overall order of the group is (n=h*r) (the number of subgroup). The number of subgroups h with EC points is denoted by **cofactor**. The cofactor is typically expressed by the following formula:

$$\boxed{h = \frac{n}{r}}$$

where

- The curve's order is n. (the number of all its points)
- The curve cofactor is h. (the number of non-overlapping subgroups of points,which together hold all curve points).
- The subgroups are arranged in r. (the number of points in each subgroup,including the infinity point for each subgroup).

In other words, the points across an elliptic curve remain in one or more cyclic subgroups, which are non-overlapping subsets. The term "cofactor" refers to the quantity of subgroups. The "order" of the curve, which is often represented by the number n, is the total number of points in all subgroups. If there is just one cyclic subgroup in the curve, then its cofactor h = 1. If the curve has several subgroups, its cofactor is more than 1.

4.1.3 The Generator Point in ECC

The ECC cryptosystems define a special pre-defined (constant) EC point called generator point G (base point) for elliptic curves over finite fields. By multiplying G by an integer in the range $[0...r]$, G can create any other point in its subgroup over the elliptic curve. The cyclic subgroup's "order" is represented by the integer r. (the total number of all points in the subgroup).

The order n of the curve (the total number of distinct points over the curve, including the infinity), for curves with cofactor=1, is equal to the number r. There is only one subgroup for these curves. When G and n are carefully selected, and the cofactor = 1, all possible EC points on the curve (including the special point infinity) can be generated from the generator G by multiplying it by integer in the range $[1...n]$. This integer n is known as "order of the curve".

It is important to understand that the order r of the subgroup, which might differ from the order of the curve, is what defines the total number of possible private keys for this curve: $r = n/h$. (curve order, divided by the curve cofactor). To ensure that the key space is sufficiently large for a given cryptographic strength, cryptographers carefully choose the elliptic curve domain parameters (curve equation, generating point, cofactor, etc.).

In summary, the EC points and the generator point G together generate cyclic groups (or cyclic subgroups) in the ECC cryptography. This implies that a number r exists (r > 1) such that $r * G = 0 * G$ =infinite and all points in the subgroup can be generated by multiplying G by integer in the range [1...r]. Order of the group refers to the number r. (or subgroup). Elliptic curve subgroups typically have a large number of generating points, but cryptographers carefully choose one of them to act as the entire group's (or subgroup's) generator and to allow for performance improvements in the calculations. This is the so-called "G" generator.

It is very well known that some curves generate subgroups of varying orders at certain generator points. More specifically, if the group order is n, then there exists a point Q such that $d * Q=$ infinite for any prime d dividing n. As a result, certain locations that are employed as the curve's generators will produce smaller subgroups than others. The security is weak if the group is small. Small-subgroup attacks are what these are called. This is the rationale behind why cryptographers usually select r as a prime integer for the subgroup order.

Different base points can generate numerous subgroups of EC points on elliptic curves with cofactors h > 1 on the curve. The majority of EC point operations and ECC cryptographic methods will function well if we select a certain generator point and operate on that subgroup of points on the curve. It is still advised to use only tried-and-true ECC implementations, algorithms, and software programmes in specific circumstances.

Generator Point Example:

For The EC over finite field $y^2 = x^3 + 7 mod 17$,if we take the point G=15,13 as generator,any other point from the curve can be obtained by multiplying G by some integer in the range[1...18].Thus the order of this EC is n=18 and its cofactor h=1.

Note that the curve has 17 normal EC points(shown at the above figure)+one special"point at infinity",all staying in a single subgroup,and the curve order is 18(not 17)

Note also,that if we take the point5,9 as generator,it will generate just 3 EC points;5,8,5,9 and infinity.Because the curve order is not prime number,different generators may generate subgroups of different order.This is a good example why we should not "invent" our own elliptic curves for cryptographic purposes and we should use proven curves.

4.1.4 Key Generation

In the ECC, an EC point P is generated by multiplying a fixed EC point G (the generator point) by a certain number k, which may be considered of as a private key (its corresponding public key).

Consequently,in ECC we have:

- Elliptic curve (EC) over finite field Z_p
- Generator point= G (fixed constant,a base point on the EC)
- Private Key = k
- Public Key = P

Using the well-known ECC multiplication methods in time $\mathcal{O}(log2(k))$, such as the "double and add algorithm," $P = k * G$ can be calculated relatively quickly.

It will just require a few hundred straight forward EC operations for 256-bit curves.

Calculating $k = P/G$ is extremely time-consuming and is considered to be impossible for big k.

The security strength of the ECC cryptography, also known as the ECDLP, is based on this asymmetry (quick multiplication and impractically slow opposite operation) (Discrete Logarithmatic problem).[8] [9]

5 Elliptic Curve Diffie Hellman

The generation of a shared key is illustrated by the example below. Imagine Alice and Bob wish to create a shared key, but the only channel that is accessible to them may be monitored by a third party.

First,it is necessary to come to an agreement on the domain parameters,which are (p, a, b, G, n, h) for the prime case or $(m, f(x), a, b, G, n, h)$ for the binary case.Additionally, each participant must have a key pair ideal for elliptic curve cryptography,consisting of a public key represented by a point Q(where $Q = k(\alpha\ or\ \beta) * G$,that is,the outcome of adding G to itself k times),and a private key k (a randomly chosen number in the range $[1, n-1]$.Let Bob's key pair be (β, Q_B) and Alice's key pair be (α, Q_A).Before the algorithm can be used,each party has to be aware of the other's public key.

where

- $p = field(modulo\, p)$
- $(a, b) = Curve\ Parameters$
- $G = Generator\ Point$
- $n = order(G)$
- $h = cofactor$

5.1 Algorithm

1. The private and public keys are first computed separately by Alice and Bob.

   Alice has the private key α and the public key $K_A = \alpha * G$.

   Bob has the private key β and the public key $K_B = \beta * G$.

   Both Bob and Alice are using the same domain parameters G on the same elliptic curve on the same finite field.

2. Over an insecure channel, Alice and Bob exchange public keys $K_A\ \ \&\ \ K_B$.Eve find $\alpha\ \ \&\ \ \beta$,however without solving the discrete logarithm problem, it will be impossible to determine either α or β.

3. Alice calculates $S = k_A * \beta$(using her own private key and bob's public key),and Bob calculates $S = k_B * \alpha$(S is secret shared).

$$\boxed{S = \alpha * K_B = \alpha(\beta * G) = \beta(\alpha * G) = \beta * K_A}$$

4. Eve However,only knows K_A and K_B (together with the other domain parameters) and would not be able to find out the shared secret S.This is known as the Diffie-Hellman Problem.

5.2 Mathematical Explanation

1. First we choose elliptic curve of equation $y^2 = x^3 + 2x + 3$

   $E : y^2\ mod13\ \equiv\ (x^3 + 2x + 3)\ mod13$

   $p\ =\ modulo(13)$

   $Curve\ Parameter\ (a, b)\ = (2, 3)$

   $Generator\ Point\ G\ =\ (7, 3)$

2. After that we perform Elliptic Curve Multiplication $n * G$ and we have to use above equation for operation on Elliptic Curve.

$G = (7, 3)$	$10G = (11, 2)$
$2G = (0, 4)$	$11G = (4, 6)$
$3G = (10, 3)$	$12G = (3, 6)$
$4G = (9, 10)$	$13G = (6, 6)$
$5G = (6, 7)$	$14G = (9, 3)$
$6G = (3, 7)$	$15G = (10, 10)$
$7G = (4, 7)$	$16G = (0, 9)$
$8G = (11, 11)$	$17G = (7, 10)$
$9G = (12, 0)$	$18G = \mathcal{O}$

   Here $18G$ became point at infinity.So $n = Order(G) = 18$

   $$\boxed{h = \frac{p}{n} = \frac{13}{18} \cong 1}$$

3. Now Both Alice and Bob will selecet their private key $\alpha = 5\ \&\ \beta = 7$.

4. Alice compute $K_A = \alpha * G = 5G = (6, 7)$ and Bob compute $K_B = \beta * G = 7G = (4, 7)$ after that Alice and Bob interchange $K_A\ \&\ K_B$.

5. Alice receive $K_B = (4, 7)$ and Bob receive $K_B = (6, 7)$.

6. Alice use her private key $\alpha = 5$ and compute

   $S = \alpha * K_B = 5(7G) = 35G = 17G = (7, 10)$

   Bob use his private key $\beta = 7$ and compute

   $S = \beta * K_A = 7(5G) = 35G = 17G = (7, 10)$.

7. Now They both Alice and Bob have same key $S = (7, 10)$ and it can be used to verify(Authenticate) each other and transfer data.

5.2.1 Secrecy Table

Alice	
Known	Unknown
$G = (7, 3)$	
$n = 18$	
$\alpha = 5$	β=7
$K_A = 5 * G = (6, 7)$	
$K_B = (4, 7)$	
$S = 5 * K_B = (7, 10)$	

Bob	
Known	Unknown
$G = (7, 3)$	
$n = 18$	
$\beta = 7$	α=5
$K_B = 7 * G = (4, 7)$	
$K_A = (6, 7)$	
$S = 7 * K_A = (7, 10)$	

Eve	
Known	Unknown
$G = (7, 3)$	
$n = 18$	
	$\alpha = 5$,β=7
$K_A = (6, 7)$	
$K_B = (4, 7)$	
	$S = 5 * K_B = (7, 10)$

6 Commercialization

Establishing a secure client authentication system is possible using elliptic curve cryptography. For example, if Alice is a client computer and Bob is a server, they would first agree to determine the curve's prime and generating point in order to get a larger subgroup. A key will be generated after key sharing protocols have been started. This key will be stored on the server as the client's identity, and it will be saved on the client's device as a passcode to access the server. Every time the client wants service, both parties will be able to identify one another due to this methodology.

The new platform for programming competitions can be built on cryptocurrencies. This will allow users to earn cryptocurrency by answering questions, participating in writing contests, and receiving unique badges and stickers for solving particular questions. In various online auctions, these same people will be able to trade badges and stickers for cryptocurrency. The site will be a tremendous resource for anyone who wish to learn about cryptocurrency and programming. The site will also be helpful for people trying to develop their financial literacy. The account holder's name will be used to identify the data on the server that is required for all of this. However, the server must correctly identify it using the above-mentioned method before it can be accessed.

1) We can make money for the platform by contacting advertising agencies and asking them to post advertisements on our website. In turn, this enables us to provide our users valuable content. We can keep up the platform and keep giving everyone useful content with the money we make.

2) Software development companies can get in touch with us to organise a hackathon if they wish to hire qualified software engineers. Developers compete in hackathons to provide the best software to solve an issue. The challenge is presented by the software development company, and developers compete to produce the best solution. The winner team usually receives a prize, and the software development company will observe the skill of the participating developers.

References

[1] Nick Sullivan, “A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography,” 2013.

[2] Douglas R. Stinson, Maura B. Paterson, *Cryptography Theory and Practice.* CRC PRESS, Boca Raton, London, New York, Washington, D.C., 2019.

[3] Dindayal Mahto, Dilip Kumar Yadav, “RSA and ECC: A Comparative Analysis,” updated by 2017, Department of Computer Applications, National Institute of Technology Jamshedpur, Adityapur, Saraikella-Kharsawan, Jharkhand, India.

[4] Dareel Hankerson, Alfred Menezes, Scott Vanstone, *Guide to Elliptic Curve Cryptography*, 2003.

[5] WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE, “New Directions in Cryptography,” vol. IT-22, 1976.

[6] Souvik Nandi, “Diffie-Hellman-algorithm,” updated by jan 2022.

[7] Jonathan Katz and Yehuda Lindell, *Introduction to Modern Cryptography.* CRC PRESS, Boca Raton, London, New York, Washington, D.C., 2007.

[8] Joseph H. Silverman, *An Intoduction to Theory of Elliptical Curves*, Summer School on Computational Number Theory and Applications to Cryptography University of Wyoming, 2006.

[9] Josh Koncovy, supervised under Dr. Colin Ingalls, “Applications of Elliptic Curves Over Finite Fields,” p. 8, 2014.