A Brief Introduction to Elliptic Curve Cryptography

1- Cryptographic Background:

a) Hash Functions:

Hash functions take some string or binary input and return some fixed length integer value. The return value is pseudo-random and **irreversible by design** (There is no smart way to return the input value, brute-force methods take too much time). There are several important parameters of the function must be satisfied to be cryptographic hash function. In theory there could be collision, however in the modern cryptographic hash functions such as SHA-3, there is no known collusion.

Deterministic: the same input message should always result in the same hash value.
Quick: it should be fast to compute the hash value for any given message.
Hard to analyze: a small change to the input message should totally change the output hash value.
Irreversible: generating a valid input message from its hash value should be infeasible. This means that there should be no significantly better way than brute force (try all possible input messages).
No collisions: it should be extremely hard (or practically impossible) to find two different messages with the same hash.
Avalanche Effect: Small changes in the input causes significant difference in output, i.e algorithm cannot be tracked by changing input slightly. Results seem pseudo-random. (this part retrieved from)

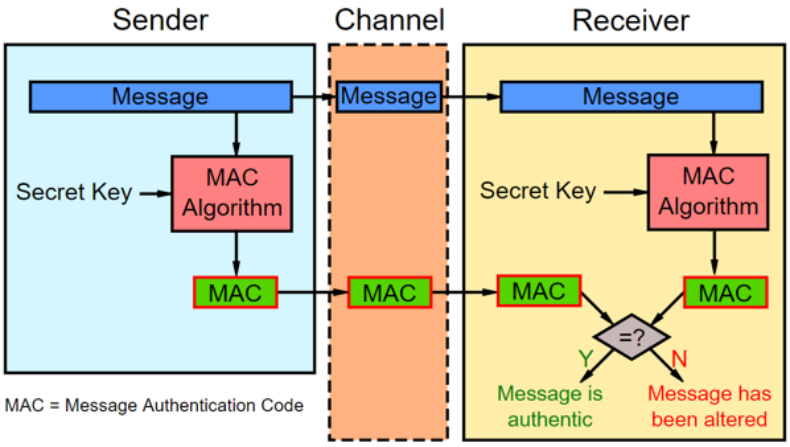
Some applications:

i) **Password Storing:** Rather than storing the password itself by its own string. It is safer to keep its hash value.
ii) **Data integrity:** It is possible to check whether there is a data loss or not by checking hash values of the message. It is also possible to check data manipulations over the message.
iii) **Generating ID, Password, Random numbers**

b) MAC Functions & Key Derivation Functions:

MAC functions are very similar to the hash functions. Only difference is that there is an additional **key** inserted the function and resulted mac value is decided by he combination of these two (auth_code = MAC(key, message)).

MAC Algorithms could be used to decide whether a message is changed or not during the transfer. Firstly, there should be a secret key decided between the sender and receiver. The message is encrypted by MAC algorithm by combining this secret key. Message and MAC code sent through a channel. Since the receiver has the same key, he also can combine the received message by the key and can get a MAC code by using the same algorithm. If the resulted code and received code are the same, then the message is not changed and sent by the sender. (Unless the key is not stolen)



To create a secret key, Key derivation functions are used. These functions take some input and create a key value. Some hash functions could be used for key derivation functions. For example SHA256 could be used directly to generate key but this causes some security weakness against the **dictionary attacks.** To avoid that an additional random value inserted to the key production process which is the **salt.** Salt is stored with the key to regenerate the key when it is necessary.

Authenticated Encryption:

1- User inputs a password and by using this password a key derived.
2- The message is encrypted according to this key. Ciphertext stored in the output.
3- Finally by using a MAC algorithm (key and original message given as input) to get an authentication code. Resulting code appended to the output.

Decryption:

1- Derive a key from the entered password (Could be true or not, later decided)
2- $Decrypt\ the\ message\ by\ the\ resulted\ key.$
3- Check the MAC code of the deciphered text and retrieved key. If the MAC value is identical to received key. Password is correct and message encrypted correctly.

c) Diffie-Hellman Key Exchange

DHKE is used for creating a "common secret" in a public channel. The purpose is to generate a common key. The process works as follows.

1- 2 users (Let's say X and Y) decide some arbitrary large prime numbers g and n. These numbers can be transferred publicly or could be a built-in part of the system.
2- User X and Y decides arbitrary large number (which will serve as private key), let's say a and b accordingly.
3- User X generates a number (A) by using a, g and n as follows.

$$\boldsymbol{A = g^a \ mod\ n}$$

A is the public key of the user X. User Y also repeats the same procedure and gets his own public key B.

$$\boldsymbol{B = g^b\ mod\ n}$$

4- Now X and Y shares their public keys over the channel. X applies the received public key the following procedure to get some K value.

$$\boldsymbol{K = B^a\ mod\ n}$$

When the Y applies the same procedure by using X's public key he gets:

$$\boldsymbol{K = A^b\ mod\ n}$$

Resulting value must be equal for both of them by rules of modular arithmetic. That's how X and Y decides a common key to be used in symmetric cryptography.

Unless there is a cyclic group and a generator for this group. It is possible to design the process over this group and group operators. That is why same process could be designed by using **elliptic curves.**

d) Encryption Basics

Encryption is used to transform data (plaintext) into an unrecognizable form for security purpose. Algorithms generally takes a key and message as input and gives an unrecognizable output (ciphertext) by using these two inputs. Ciphertext should be back transformed to its original version by using the same algorithm and key.

Encryption process said to be **symmetric** if encryption and decryption is done by the same key. This type of encryption is faster but harder since the key must be exist with both sender and receiver. Share of the key is hard especially when the channel is not secure.

Encryption process said to be **asymmetric** if there is a pair of keys which are the public and private key. Public key is used to encrypt the data. Encrypted data can be decrypt by the private keys. This is why only owners of private key can understand the message.

(AES and RSA could be examined as example of each type)

2- Mathematical Background

a) What is an elliptic Curve?

An elliptic curve is a mathematical equation which has the following form:

$$y^2 = x^3 + ax + b \; where \; a, b \; \in K$$

and the following equation for a and b are holds:

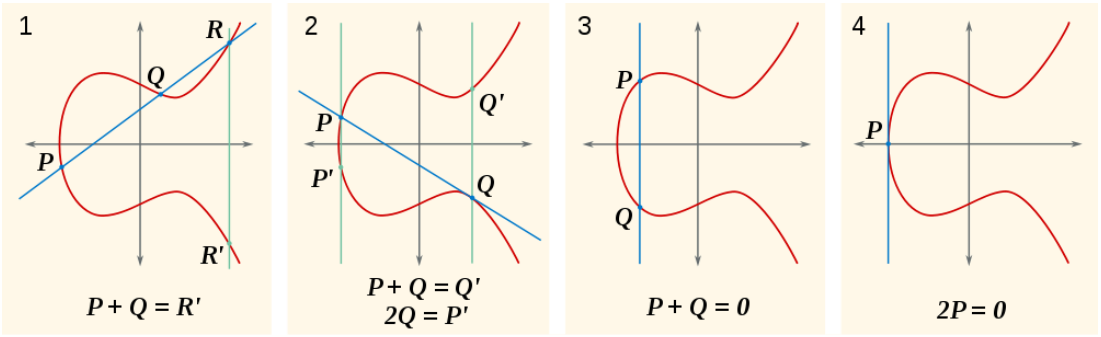
$$\mathbf{4a^3 + 27b^2 \neq 0}$$

Here K could be any field. Such as real numbers, complex numbers or any mod. Additionally, there is an element “O”.

Some important properties of elliptic curves:

1- A line could intersect with an elliptic curve at most at 3 points.
2- Elliptic curves are always symmetric according to the x-axis.
3- Addition over the elliptic curve defined as follows:

a) Let's say P and Q are 2 points over the curve, if they are not the same point and not symmetric respect to the x axis, first draw the line segment passing through both of the points, and then detect the third point intersecting with the curve. Symmetric point of this point according to the x axis is the intersection point.
b) If P and Q are symmetric according to the x-axis, connecting line will be vertical so the result will be point at infinity (O)
c) If P = Q, tangential line over the point is drawn and intersection with the curve is the result.



Algebraic way of calculating a point is as follows:

S is the slope of the line segment,

If P + Q = R, then:

$$X_R = s^2 - (X_P + X_Q)$$

$$Y_R = s(X_P - X_R) - Y_P$$

$$Here\ s\ is\ defines\ as\ follows:$$

$$s = \frac{Y_P - Y_Q}{X_P - X_Q}$$

$$(if P\ and\ Q\ are\ two\ distinct\ point\ and\ not\ symmetric)$$

$$\frac{3X_P^2 + a}{2Y_P}$$

$$\boldsymbol{if\ P\ =\ Q}$$

What is a finite field?

A finite field is a mathematical structure consists of finitely many elements and 2 binary operations (generally addition and multiplication). The operations must satisfies some axiomatic properties such as being associative, commutative and distributive.

Order of the field is the number of the elements in the finite field.

This type of fields could be built by modulus operation. For example F(3) (a.k.a Z/3Z) is a finite field with 3 elements: {0,1,2}. It is possible to do addition and multiplication among them in (mod 3).

Groups & Cyclic Groups

Group is a mathematical structure with a set of elements and 1 binary operation satisfies several properties:

1- Every element must has some inverse according to the operator. For example if g is the element and operator is the addition, then -g must be element of the group.
2- Operator must be associative, i.e. (Let's assume operator is the addition) a + (b + c) = (a+b) + c
3- Operator must be closed over the elements of the set.
4- There must be some element e (identity element), holds the equation a + e = a for all a

For example whole numbers (Z) constitutes a group under addition.

A group is said to be **cyclic group** if every element of the group can be **generated** by a **generator element (g).** Genereation means the implementation of group operation finitely many times over the g or its inverse respect to the operator. From this perspective, Z is also a cyclic group by generator <1>. That means every element in the z could be obtained by either applying addition over 1 or its inverse (-1) finitely many times.

Trap door function and discrete logarithm problem in elliptic curves

A function is said to be a trap door function if it is very easy to calculate in one direction but there is no easy way of inverting it. For example, if p and q are 2 prime numbers, it is very easy to calculate n = p*q. However, opposite direction is not that much obvious, i.e. when n is given, it is not easy to decide which two numbers multiplication it is.

In elliptic curves there is a similar scenario. Let's say we have an elliptic curve over a finite field by order n. G is the generator point of the field. It is very easy to obtain any point in the field just by adding G to itself finitely many times. That means for any point P:

$$P = kG \; and \; 0 \leq k \leq n - 1$$

Mathematically it is very easy to travel among the points by addition rules mentioned above. However, it is not easy to decide number k for a given point P. This property defines a **trap door function in elliptic curves**. This also define a problem called **discrete logarithm problem for elliptic curves.**

Diffie Hellman Key Exchange by Using Elliptic Curves:

Above we mentioned that for any cyclic groups it is possible to re-design Diffie Hellman key exchange accordingly. ECDH process works as follow.

1- Firstly, Alice and Bob agree on a curve on some finite field and generator point g in the first handshake.
2- Alice decides her own private key, let's say a, and computes the point A = aG. Bob does the same thing, b for his private key, and calculates another point B = bG. Here A and B are the 2 public keys derived from the generator point by elliptic curve addition according to the rules mentioned above. (aG means addition of G to itself for a times)
3- Now Alice adds B to itself for a times and Bob do the same for A for b times. Resulting point Q will be equal for both and that will constitute the common secret. This common secret can used as a key for a symmetric encryption protocol.

$$A * b = B * a = Q$$

Hybrid Cryptography

Sometimes asymmetric cryptography is not applicable due to the large amount of data to transmit. However, implementation of symmetric cryptograph is also not easy since there is no decided common key. In this kind of scenarios hybrid cryptography could be used as follows:

1- First, initiator of the communication encrypt the common key by an asymmetric cryptography algorithm.
2- Send the encrypted key to the receiver.
3- Receiver decrypt the key by his own private key.
4- A verification message sent to inform key obtained successfully
5- Original message encrypted according to the shared key by using some symmetric key. Since both sides have the same key.

Data can be encrypt and decrypt successfully by a symmetric algorithm.

Elliptic curve cryptography can be used as a part of some hybrid schematics. Elliptic curves provides asymmetric cryptography option in these schematics.