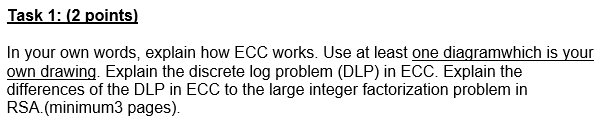
Applied Cryptography

Section: L01

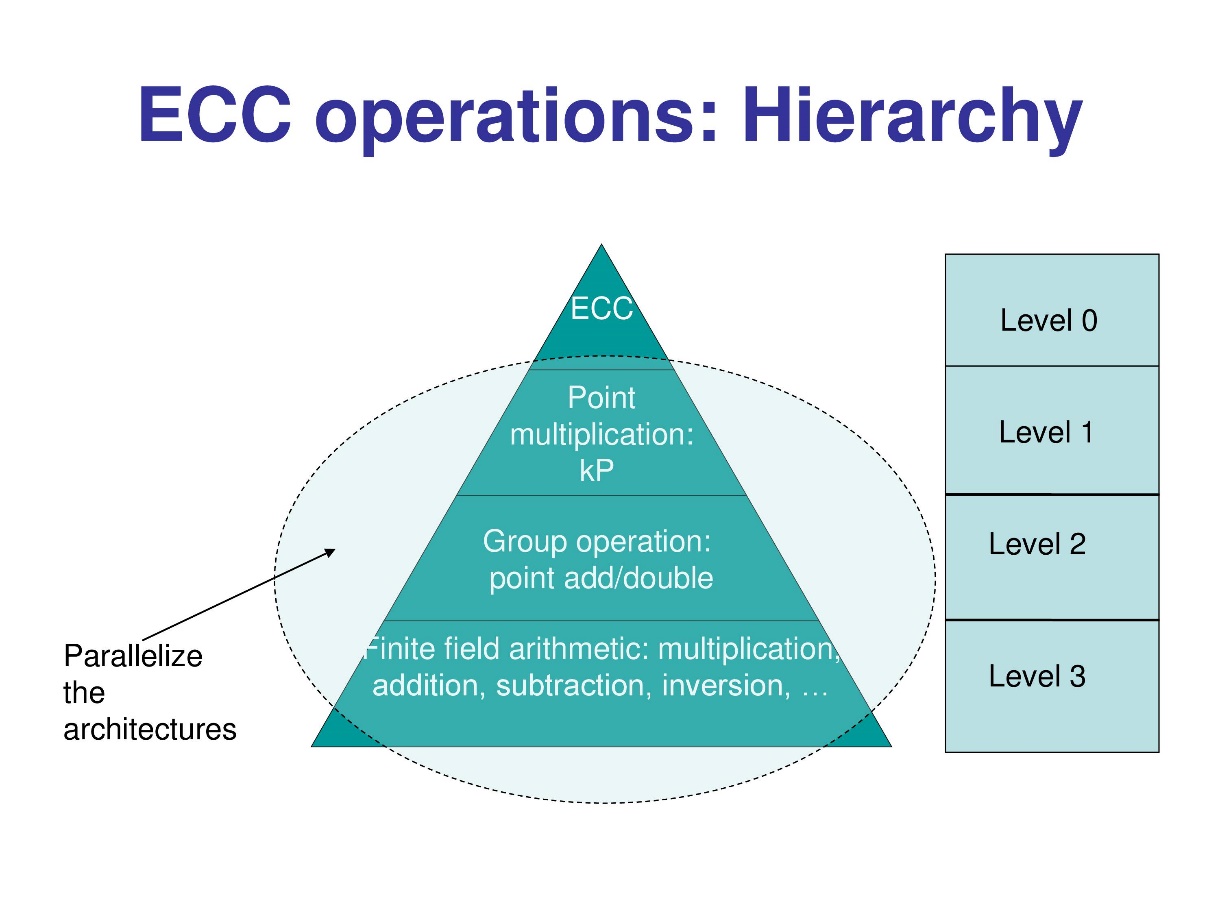
Assignment Date: 12/05/2021

Name: Talha Abdullah Punjabi

QUID: 201903446



**Working of ECC**



Elliptic curve encryption works on the principle of logarithmic problems. It is a powerful approach. The common motive of elliptic curve encryption is for key exchange and digital signatures. Two notable schemes of Elliptic curves are Elliptic Curve Diffie-Hellman (ECDH) specifically for key exchange and Elliptic Curve Digital Signature Algorithm (ECDSA) specifically for digital signatures. It works by generating key pairs for public key encryption by using elliptical curves. Elliptic curves use points on curve and these elliptic curves are defined by their equations. For instance,



This is an equation representing the elliptic curve. Point of infinity is used to provide identity to the operations for points on the curve. There is an important condition to be fulfilled for ECC to work. The curve must have non-repeating roots, which is the curve should be non-singular.

The condition in equation form is:



Many cyclic groups are required to construct the discrete logarithmic problem. There are two types of group operations that need to be performed on the line

1. Point Addition – In such a geometric representation, two curves are made which intersect on two points to form a third point between the curve and the line. A simple mathematic formula to depict the Point addition is A+B = C. This in terms of coordinates are (x(A), y(A)) + (x(B), y(B)) = (x(C), y(C))

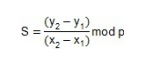
This method point addition can be performed on the elliptic curve.

The point addition equations are as follows:

Text

Description automatically generated

Which yields the result to be in this form of equation:



1. Point Doubling – In such a geometric representation, the same point is added to itself to form a curve. A tangent line is drawn through this curve to obtain the second point. This is due to the intersection of tangent line drawn and the curve. A simple mathematic formula to depict this is 2A=A+A.

Equations here result in

Text, letter

Description automatically generated

**Diagram showing representation of ECC**

Diagram

Description automatically generated

**Discrete Logarithm Problem**

The discrete logarithm problem is defined for a group G, a generator g, and any element h of the group, to find the discrete logarithm base g of h in the group G. The hardness of solving this issue depends on the complexity of the group. The discrete logarithm problem is used to find the private keys that lie in all the points inside the elliptical curve. In elliptical curve encryption, the public key is a random multiple of generator, while the private key is used to generate the multiple. Considering an elliptic curve E, with two elements K and Z. The discrete logarithmic problem is to find the integer d, where 1<=d <=E, such that

K+K+…+K=dK=Z

It is of cryptographic interest because its intractability is a security concern for elliptic curve cryptography.

**Difference between DLP in ECC and large integer factorization problem in RSA**

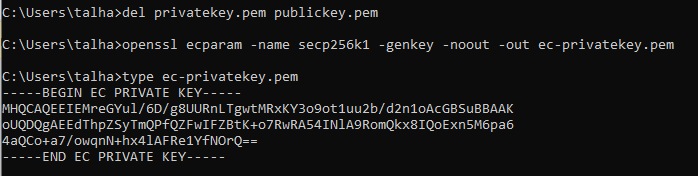
The discrete logarithm problem is defined for a group G, a generator g, and any element h of the group, to find the discrete logarithm base g of h in the group G. The hardness of solving this issue depends on the complexity of the group. While, as the product is bigger for RSA, and the numbers itself are bigger to check upon. Each check takes a lot of time. When numbers are sufficiently the large, non-quantum number, no efficient number introduces the large integer factorization problem in RSA. No efficient algorithm is proved to work to solve this issue. This difficulty is heart of widely used algorithms in cryptography. The primary difference is the encryption strength. ECC small key size has an equivalent value for big key size of RSA. The discrete algorithm would intensely complex in case of RSA key size. Shor’s algorithm can solve the large integer factorization problem of RSA, while cannot be for ECC.

A screenshot of a computer

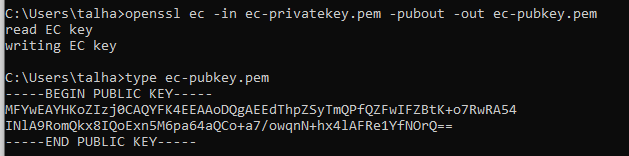
Description automatically generated with medium confidence

**a.**

Generate Private key



Generate Public key



**b.**

Text

Description automatically generated

SECP256K1 curve parameters

Field Type: prime field is a finite one with a prime number of elements

Prime: has the hexadecimal numbered prime number

Generator: has an uncompressed value of the generator number

Order: is the number of elements in a field

Cofactor: shows the cofactor value of the secp256k1 used

**c.**

Prime: 00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fe:ff:ff:fc:2f

A: 0

B: 7 (0x7)

Generator: 04:79:be:66:7e:f9:dc:bb:ac:55:a0:62:95:ce:87:0b:07:02:9b:fc:db:2d:ce:28:d9:59:f2:81:5b:16:f8:17:98:48:3a:da:77:26:a3:c4:65:5d:a4:fb:fc:0e:11:08:a8:fd:17:b4:48:a6:85:54:19:9c:47:d0:8f:fb:10:d4:b8

Order: 00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fe:ba:ae:dc:e6:af:48:a0:3b:bf:d2:5e:8c:d0:36:41:41

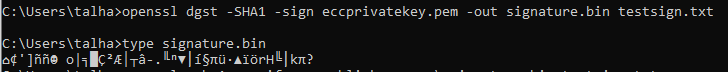
Cofactor: 1(0x1)

**d.**

Text

Description automatically generated

**Signing Message with ECC**



**Verification**



**e.**

Generating The data to be sent between the sender and receiver

Text

Description automatically generated

Generating key and encrypting the data with it

Text

Description automatically generated



Encrypting the symmetric key with private key of the sender





**f.**

Decrypting at the receiver’s end





Decrypting The data



