**Tutorial 2**

**Name**:Bhavin Patil

**Roll No:** 66

**TY-CS-D**

# Diffe- Hellman Algorithm

**Background**

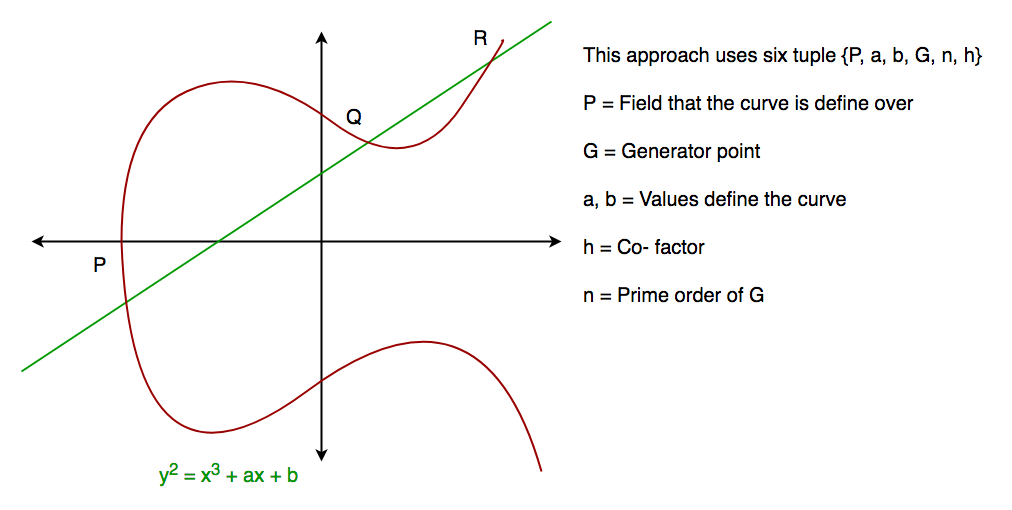
**Elliptic Curve Cryptography (ECC)** is an approach to public-key cryptography, based on the algebraic structure of elliptic curves over finite fields. ECC requires a smaller key as compared to non-ECC cryptography to provide equivalent security (a 256-bit ECC security has equivalent security attained by 3072-bit RSA cryptography).

For a better understanding of Elliptic Curve Cryptography, it is very important to understand the basics of the Elliptic Curve. An elliptic curve is a planar algebraic curve defined by an equation of the form

Where ‘a’ is the co-efficient of x and ‘b’ is the constant of the equation

The curve is non-singular; that is, its graph has no cusps or self-intersections (when the characteristic of the Co-efficient field is equal to 2 or 3).

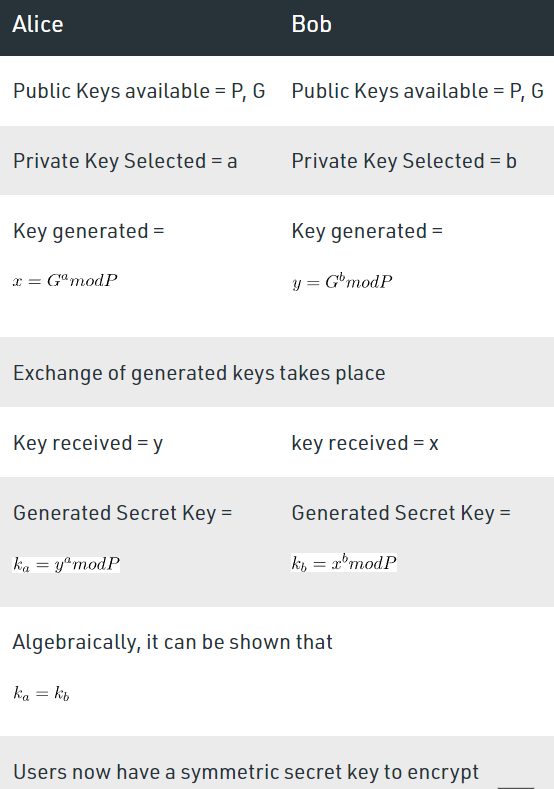
In general, an elliptic curve looks like as shown below. Elliptic curves can intersect almost 3 points when a straight line is drawn intersecting the curve. As we can see, the elliptic curve is symmetric about the x-axis. This property plays a key role in the algorithm.



**Diffie-Hellman algorithm**

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

* For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
* P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.



**Example:**

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and

Bob selected a private key b = 3

Step 3: Alice and Bob compute public values

Alice: x =(9^4 mod 23) = (6561 mod 23) = 6

Bob: y = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and

Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys

Alice: ka = y^a mod p = 65536 mod 23 = 9

Bob: kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.

**Implementation in Python**

from random import randint

if \_\_name\_\_ == '\_\_main\_\_':

# Both the persons will be agreed upon the

# public keys G and P

# A prime number P is taken

P = 23

# A primitive root for P, G is taken

G = 9

print('The Value of P is :%d'%(P))

print('The Value of G is :%d'%(G))

# Alice will choose the private key a

a = 4

print('The Private Key a for Alice is :%d'%(a))

# gets the generated key

x = int(pow(G,a,P))

# Bob will choose the private key b

b = 3

print('The Private Key b for Bob is :%d'%(b))

# gets the generated key

y = int(pow(G,b,P))

# Secret key for Alice

ka = int(pow(y,a,P))

# Secret key for Bob

kb = int(pow(x,b,P))

print('Secret key for the Alice is : %d'%(ka))

print('Secret Key for the Bob is : %d'%(kb))

