EXPERIMENT NO 2

Aim : Implementation of Diffie Hellman algorithm

Theory:

The question of key exchange was one of the first problems addressed by a cryptographic protocol. This was prior to the invention of public key cryptography. The Diffie-Hellman key agreement protocol (1976) was the first practical method for establishing a shared secret over an unsecured communication channel. The point is to agree on a key that two parties can use for a symmetric encryption, in such a way that an eavesdropper cannot obtain the key.

What is Diffie–Hellman?

Diffie–Hellman key exchange (**DH**) is a method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as originally conceptualized by Ralph Merkle and named after Whitfield Diffie and Martin Hellman. DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography.

Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical channel, such as paper key lists transported by a trusted courier. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

Diffie-Hellman is a way of generating a shared secret between two people in such a way that the secret can't be seen by observing the communication. That's an important distinction: You're not sharing information during the key exchange, you're creating a key together.

Why do we care about DH ?

We care about Diffie-Hellman in light of the fact that it is a standout amongst the most widely recognized protocols utilized as a part of networking today. This is genuine, despite the fact that most by far of the time the user has no clue it is working. DH is usually utilized when you encrypt data on the Web utilizing either SSL (Secure Socket Layer) or TLS (Transport Layer Security). The Secure Shell (SSH) protocol also uses DH. Obviously, in light of the fact that DH is a piece of the key exchange mechanism for IPSec, any VPN based on that technology uses DH too. Truly a VPN or SSL system could be being used for a considerable length of time without the system head knowing anything about Diffie-Hellman. In any case, I think that an understanding of underlying protocols and processes helps a great deal when trouble-shooting a system.

The following diagram Figure 1 shows the general thought of the key exchange by utilizing colors rather than an large number. The procedure starts by having the two gatherings, Alice and Bob, agree on an arbitrary starting color that does not should be kept secret; in this example the color is yellow. Each of them chooses a secret color - red and aqua respectively - that they keep to themselves. The vital piece of the procedure is that Alice and Bob now combine their secret color with their commonly shared color, bringing about orange and blue mixtures respectively, then freely exchange the two mixed colors. At last, each of the two combine the color they got from the partner with their own particular private color. The outcome is a last color mixture (brown) that is identical with the partner's color mixture. On the off chance that another gathering had been listening in on the exchange, it is computationally troublesome for that individual to decide the common secret color and in fact, when utilizing large numbers rather than colors, this activity is outlandish for cutting edge supercomputers to do in a sensible measure of time

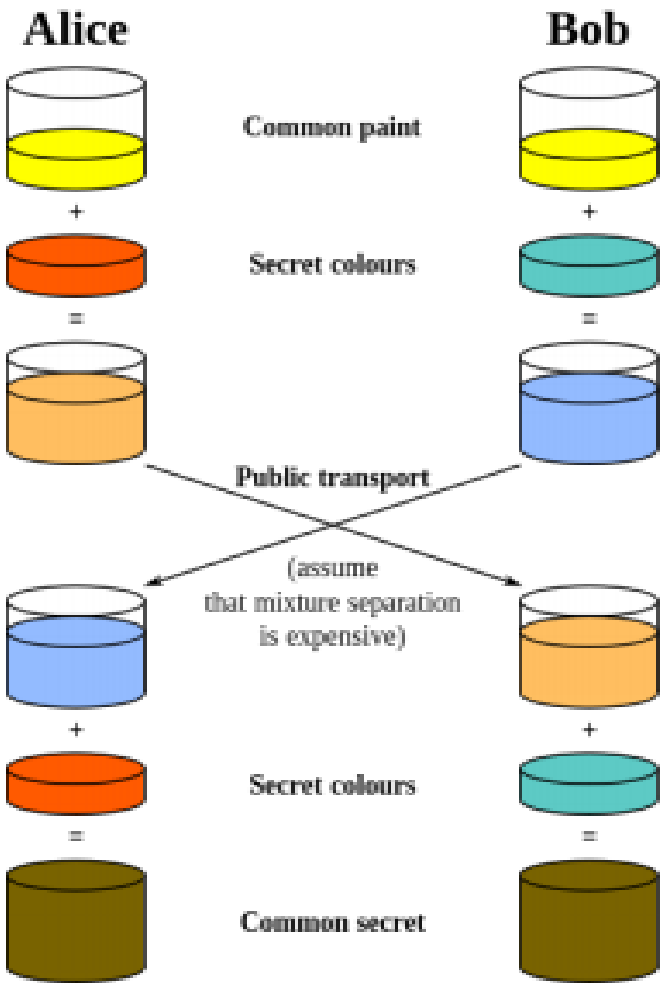

Figure 1 : Illustration with colors

Advantages and disadvantages of Diffie–Hellman Algorithm

Advantages

- The sender and receiver have no prior knowledge of each other.
- Communication can take place through an insecure channel.
- Sharing of secret key is safe.

Disadvantages

- Cannot be used for asymmetric key exchange.
- Cannot be used for signing digital signatures.
- The nature of the Diffie-Hellman key exchange does make it susceptible to man-in-the-middle attacks since it doesn't authenticate either party involved in the exchange.

Algorithm:

Step 1 : All users [U0,U1….U(N-1)] agree on global parameters : where N=total number of users

a) large prime integer or polynomial q
b) a being a primitive root mod q

Step 2 : For Each user from [U0,U1….U(N-1)] generate their keys

a) Choose a secret key (number): $x_i < q$ where i is from { 0,1…N-1}
   where x_i is the secret key
b) And compute public key: $y_i = (a)^{xi} \bmod q$
c) each user make public that key y_i

Step 3: To compute shared session key for users Ui & Uj where [i!=j] which is K_{ij} do the following

a) $K_{ij} = a^{xi.xj} \bmod q$

   $= y_i^{xj} \bmod q$ (which Uj can compute)

   $= y_j^{xi} \bmod q$ (which Ui can compute)

b) K_{ij} is used as session key in private-key encryption scheme between Ui and Uj.
c) If Ui and Uj subsequently communicate, they will have the same key as before, unless they choose new public-keys

Example:

Step 1:

- Let no of users be 2 [U0,U1] hence n=2
- All users agree on prime q=353
- Now 353 has 160 primitive roots, and they are [3, 5, 12, 13, 14, 20, 24, 26, 27, 28, 31, 33, 37, 40, 45, 48, 51, 52, 53, 54, 55, 56, 57, 62, 63, 66, 69, 71, 74, 75, 77, 79, 80, 85, 87, 89, 90, 95, 96, 102, 103, 104, 105, 107, 108, 110, 112, 114, 115, 117, 118, 119, 123, 124, 125, 126, 129, 132, 133, 134, 138, 139, 141, 142, 143, 145, 147, 148, 149, 150, 151, 154, 158, 160, 161, 163, 170, 173, 174, 175, 178, 179, 180, 183, 190, 192, 193, 195, 199, 202, 203, 204, 205, 206, 208, 210, 211, 212, 214, 215, 219, 220, 221, 224, 227, 228, 229, 230, 234, 235, 236, 238, 239, 241, 243, 245, 246, 248, 249, 250, 251, 257, 258, 263, 264, 266, 268, 273, 274, 276, 278, 279, 282, 284, 287, 290, 291, 296, 297, 298, 299, 300, 301, 302, 305, 308, 313, 316, 320, 322, 325, 326, 327, 329, 333, 339, 340, 341, 348,350]
- All users agree on a=3

Step 2:

- All Users select random secret keys :
  - U0 chooses x_0=97,
  - U1 chooses x_1=233
- Compute respective public keys for all users:
  - $y_0=3^{97} \mod 353 = 40$ (U0)
  - $y_1=3^{233} \mod 353 = 248$ (U1)

Step 3:

- Compute shared session key for any two different user as:
- Take user U0 and U1
  - $K_{01}= y_1^{x0} \mod 353 = 248^{97} = 160$ (U0)
  - $K_{01}= y_0^{x1} \mod 353 = 40^{233} = 160$ (U1)

Code:

```
import random

def isPrime(n):          #prime number checker

   if n==1 :
```

```
        return False
    elif n==2 :
        return True
    else:
        for x in range(2,n):
            if(n % x==0):
                return False
        return True

def primitive_root(n):   # calculation of primitive root
    list_of_primitive_roots=[]
    i=2
    while i < n :
        j=0
        list_values=[]
        while j < n-1 :
            temp=(i**j)%n
            if temp in list_values:
                break
            else :
                list_values.append(temp)
            j+=1
        if j==n-1 :
            list_of_primitive_roots.append(i)
        i+=1
```

```
    return list_of_primitive_roots



no_of_users=int(input("Enter the number of users : "))
primes = [i for i in range(227,811) if isPrime(i)]
p = random.choice(primes)
        # prime number
# calculate primitive root of p
print("The prime number is : ",p)
alpha_list=primitive_root(p)
print(alpha_list)
alpha=alpha_list[random.randint(1,len(alpha_list))]
print("The value of alpha is : ",alpha,end="\n")
user_secret_key=[]
user_public_key=[]
user_shared_key=[]
i=0
for i in range(no_of_users):
    user_secret_key.append(random.randint(1,p))
    print("User "+str(i+1)+" secret key is :"+str(user_secret_key[i]))
    user_public_key.append((alpha**user_secret_key[i])%p)
    i+=1
print("\n")
i=0
```

```
for i in range(no_of_users):
    print("User "+str(i+1)+" public key is :"+str(user_public_key[i]))
    i+=1


i=0
while True :
    print("There are total ",no_of_users,"users. Enter any pair ")
    user1=int(input("First User id  : "))
    user2=int(input("Second User id : "))
    if user1 > no_of_users :
        print("First User id is not valid ")
        continue
    elif user2 > no_of_users :
        print("Second User id is not valid ")
        continue
    elif user1==user2 :
        print("Same user invalid ")
        continue


    print("User "+str(user1)+" shared key is :"+str((user_public_key[user2-
1]**user_secret_key[user1-1])%p))
    print("User "+str(user2)+" shared key is :"+str((user_public_key[user1-
1]**user_secret_key[user2-1])%p))
    descion=input("Enter no to exit : ")
    if descion=="no" :
        break
```

```
    else :
        continue
```

Output:

```
# root@coder:/coder/mnt/Rstar/CSS_CODES_PY# python3 exp2.py

# Enter the number of users : 3

# The prime number is :  353

# [3, 5, 12, 13, 14, 20, 24, 26, 27, 28, 31, 33, 37, 40, 45, 48, 51, 52, 53, 54, 55, 56, 57, 62, 63, 66,
69, 71, 74, 75, 77, 79, 80, 85, 87,89, 90, 95, 96, 102, 103, 104, 105, 107, 108, 110, 112, 114, 115,
117, 118, 119, 123, 124, 125, 126, 129, 132, 133, 134, 138, 139, 141, 142, 143, 145, 147, 148,
149, 150, 151, 154, 158, 160, 161, 163, 170, 173, 174, 175, 178, 179, 180, 183, 190, 192, 193,
195, 199, 202, 203, 204, 205, 206, 208, 210, 211, 212, 214, 215, 219, 220, 221, 224, 227, 228,
229, 230, 234, 235, 236, 238, 239, 241, 243, 245, 246, 248, 249, 250, 251, 257, 258, 263, 264,
266, 268, 273, 274, 276, 278, 279, 282, 284, 287, 290, 291, 296, 297, 298, 299, 300, 301, 302,
305, 308, 313,316, 320, 322, 325, 326, 327, 329, 333, 339, 340, 341, 348, 350]

# The value of alpha is :  28

# User 1 secret key is :102

# User 2 secret key is :73

# User 3 secret key is :169



# User 1 public key is :186

# User 2 public key is :211

# User 3 public key is :27

# There are total  3 users. Enter any pair

# First User id  : 1

# Second User id : 2

# User 1 shared key is :344
```

User 2 shared key is :344

Enter no to exit : yes

There are total 3 users. Enter any pair

First User id : 2

Second User id : 3

User 2 shared key is :141

User 3 shared key is :141

Enter no to exit : yes

There are total 3 users. Enter any pair

First User id : 1

Second User id : 3

User 1 shared key is :144

User 3 shared key is :144

Enter no to exit : yes

There are total 3 users. Enter any pair

First User id : 1

Second User id : 1

Same user invalid

There are total 3 users. Enter any pair

First User id : 5

Second User id : 1

First User id is not valid

There are total 3 users. Enter any pair

First User id : 1

Second User id : 5

Second User id is not valid

There are total 3 users. Enter any pair

First User id : 3

Second User id : 1

User 3 shared key is :144

User 1 shared key is :144

Enter no to exit : no

Conclusion: Diffie-Hellman algorithm is used in making various secure protocol creation. It is also used in transport layer security ,SSH etc. We can use this concept in many of our networking applications. The nature of the Diffie-Hellman key exchange does make it susceptible to man-in-the-middle attacks since it doesn't authenticate either party involved in the exchange. This is why Diffie-Hellman is used in combination with an additional authentication method.