Diffie Helman Key Exchange

Prashanth.S 19MID0020

Importing the Necessary Libraries

In [1]:
```
import numpy as np
import random
```

In [2]:
```
'''
Xa = 3 ## private key of Agent-X
Ya = 7 ## private key of Agent-Y
A = ## public key of Agent-X (shared to Agent-Y)
B = ## public key of Agent-Y (shared to Agent-X)
S = ## shared key
'''
```

Out[2]:
```
'\nXa = 3 ## private key of Agent-X\nYa = 7 ## private key of Agent-Y\nA = ## public key of Agent-X (shared to Agent-Y)\nB = ## public key of Agent-Y (shared to Agent-X)\nS = ## shared k
ey \n'
```

Primitive Roots Creation

In [3]:
```
def primitive_roots_table_creation(m):
    list1 = []
    b = 1
    for i in range(1,m-1):
        temp = []
        b+=1
        for j in range(1,m):
            temp.append(np.power(b,j) % (m))
        list1.append(temp)
    return list1
```

In [4]:
```
def primitive_roots_value(primitive_roots_table, m):
    # np.unique() --> returns the number bo unique values
    m = 13
    primitive_roots = []
    for i in primitive_roots_table:
        if (len(np.unique(i)) == m-1):
            primitive_roots.append(i[0])
    return primitive_roots
```

In [5]:
```
def public_key_generation(generator, private_key, premitive_root):
    return ((generator**private_key) % premitive_root)
def sharing(pub1, pub2):
    return (pub2, pub1)
def share_secret_key(shared_key, private_key, premitive_root):
    return ((shared_key**private_key) % premitive_root)
```

In [6]:
```
def isPrime(num):
    cnt = 0
    for i in range(2, np.int(np.sqrt(num))):
        if ((num%i) == 0):
            cnt = 1
            return False ## composite number
        if (cnt==0):
            return True ## prime number
```

In [7]:
```
def start():
    ## Alice portion
    Xa = 3
    Ya = public_key_generation(generator, Xa, premitive_root)
    print("Alice's public key : ",Ya)

    ## Bob portion
    Xb = 7
    Yb = public_key_generation(generator, Xb, premitive_root)

    Ya, Yb = sharing(Yb, Ya)
    print("\nAfter sharing")
    print("Alice's public key : ",Ya)
    print("Bob's public key : ",Yb)

    alice_K = share_secret_key(Yb, Xa, premitive_root)
    print("\nAlice's shared key : ",alice_K)

    bob_K = share_secret_key(Ya, Xb, premitive_root)
    print("Bob's shared key : ",bob_K)


    if (alice_K == bob_K):
        return alice_K
    else:
        return 0
```

In [8]:
```
def shift_characters(str1, n):
    return ''.join(chr((ord(char) - 97 - n) % 26 + 97) for char in str1)
```

In [9]:
```
primitive_roots_table = primitive_roots_table_creation(13)
primitive_roots = primitive_roots_value(primitive_roots_table,13)
primitive_roots
```

Out[9]: [2, 6, 7, 11]

In [10]:
```
premitive_root = 13
if isPrime(premitive_root):
    generator = random.choice(primitive_roots)
if (generator < premitive_root):
    print("Continue")
    key_match = start()
else:
    print("Disontinue")
```

```
Continue
Alice's public key :  5

After sharing
Alice's public key :  5
Bob's public key :  2

Alice's shared key :  8
Bob's shared key :  8

/var/folders/gq/nsqxf83n1813yysq2l8vvtxc0000gn/T/ipykernel_2148/2137807101.py:3: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `in
t` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your cur
rent use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdocs/release/1.20.0-notes.html#deprec
ations)
  for i in range(2, np.int(np.sqrt(num))):
```

Encryption

In [11]:
```
def encryption(plain_text):
    n = key_match
    return shift_characters(plain_text, n)
```

Decryption

In [12]:
```
def decryption(cipher_text):
    n = -key_match
    return shift_characters(cipher_text, n)
```

In [13]:
```
plain_text = "prashanth"
cipher_text = encryption(plain_text)
print(cipher_text)
```

```
hjskzsflz
```

In [14]:
```
decrypt_text = decryption(cipher_text)
print(decrypt_text)
```

```
prashanth
```