Elgamal Based Digital Signature

Prashanth.S 19MID0020

Importing the Libraries

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

Getting inputs from the user

In [2]:
```
prime_number = 11
generator = 2
Message = 5
```

Checking the validity of Generator and Prime Number

In [3]:
```
if (generator < prime_number):
    if (np.gcd(prime_number, generator) == 1):
        pass
```

Private Key Generation

In [4]:
```
if (generator < prime_number):
    private_key = random.randint(2, prime_number-2)

print("Private Key : ",private_key)
```

```
Private Key :  2
```

Public Key Generation

In [5]:
```
public_key = (generator**private_key) % prime_number
print("Public Key : ",public_key)
```

```
Public Key :  4
```

Digital Signature Generation

In [6]:
```
hash_value = hash(Message)
print("Hash Value : ",hash_value)
```

```
Hash Value :  5
```

Select the secret-key (random number)

In [7]:
```
for i in range(2, prime_number):
    if np.gcd(i, prime_number - 1) == 1:
        secret_key = i
        break

print("Secret Key : ",secret_key)
```

```
Secret Key :  3
```

In [8]:
```
def inverse_value_generate(secret_key, prime_number):
    condition = True
    cnt = 0

    while condition:
        cnt+=1
        if ( (secret_key  * cnt) % prime_number == 1):
            condition = False

    return cnt
```

Computing the digital signature

In [9]:
```
 inv_secret_key = inverse_value_generate(secret_key, prime_number - 1)

y1 = (generator ** secret_key) % prime_number
y2 = ((inv_secret_key) * (hash_value - (private_key * y1))) % (prime_number - 1)

print("Digital Signature Y1 = {} and Y2 = {}".format(y1, y2))
```

```
Digital Signature Y1 = 8 and Y2 = 3
```

User-X to User-Y

In [10]:
```
print("User-X sending ---")
Message = 5
print("Message : {} and Digital Signature Y1 = {} and Y2 = {}".format(Message, y1, y2))
```

```
User-X sending ---
Message : 5 and Digital Signature Y1 = 8 and Y2 = 3
```

User-Y

In [11]:
```
hash_value = hash(Message)
print("Hash Value : ",hash_value)
```

```
Hash Value :  5
```

In [12]:
```
V1 = (generator ** hash_value) % prime_number
V2 = ( (public_key ** y1) * (y1 ** y2) ) % prime_number

print("V1 : ",V1)
print("V2 : ",V2)
```

```
V1 :  10
V2 :  10
```

In [13]:
```
if (V1 == V2):
    print("Successful")
```

```
Successful
```