Examples Diffie Hellman Key Exchange Protocol

February 16, 2018

Name: SONGA MUGABE Fabrice
Click HERE to go to the Link of the Code

1 Diffie Algorithm

```
def public_Key(Alpha, element, q):
    Y_key = (Alpha ** element) % q
    return Y_key

def key_generation(Alpha, element, q):
    K = (Alpha ** element) % q
    return K

def check_success(element, elements):
    if element == elements:
        decision = print("Public KEYS are same, The Exchange is SUCCESSFUL")
    else:
        decision = print("Public KEYS are different")
    return decision

def encrypt_another(plaintext, key):
    ciphertext = ""
    for c in plaintext.upper():
        ciphertext += I2L[(L2I[c] + key) % 36]
    return ciphertext

def decrypt_another(ciphertext, key):
    plaintext = ""
    for c in ciphertext.upper():
        plaintext += I2L[(L2I[c] - key) % 36]
    return plaintext
```

2 The Main Function for the Program

```
if __name__ == '__main__':
    try:
```

```
    input = raw_input
except NameError:
    pass
try:
    chr = unichr
except NameError:
    pass

q = int(input("\nPlease Enter Common Prime value of q: "))
a = int(input("Please Enter Primitive root Alpha value : "))
# q = 23
# a = 5

A = int(input("\nPlease Enter Private Key of Party A : "))
B = int(input("Please Enter Private Key of Party B : "))
# A = 6
# B = 15

L2I = dict(zip("ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789", range(1, 37)))
I2L = dict(zip(range(1, 37), "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"))
print(L2I)
print(I2L)

print ("The Entered Common Prime value: \n q = " + str(q) + " , \n and
Primitive root of \n Alpha value = " + str(a) + "\n")
print ("The Entered Private Key of Party: \n A = " + str(A) + ", \n and
Private Key of Party: \n B = " + str(B) + "\n")

Y_A = public_Key(a, A, q)
Y_B = public_Key(a, B, q)
print("The Calculation Key of User: \n A = " + str(Y_A) + "\n")
print("The Calculation Key of User: \n B = " + str(Y_B) + "\n")

K_A = key_generation(Y_B, A, q)
K_B = key_generation(Y_A, B, q)
print("The Generated Public Key of User: \n A = " + str(K_A) + "\n")
print("The Generated Public Key of User: \n B = " + str(K_B) + "\n")

check_success(K_A, K_B)

plaintext = "helloIamcommingon23offebruary"
print("\nThe PlainText is: \n " + plaintext + "\n"
+ "\n The PlainText in UPPERCASE is: \n " + plaintext.upper() + "\n")

ciphertext = encrypt_another(plaintext, K_A)
print("\nThe Generated CipherText is: \n ", ciphertext)
plain = decrypt_another(ciphertext, K_A)
print("\nThe Decrypted CipherText is: \n ", plain)
```

3 The OUTPUT of the Program

```
/usr/local/Cellar/python3/3.6.4_2/Frameworks/Python.framework/Versions/3.6/bin
/python3.6 "/Users/admin/Dropbox/Practical/Data_sec/Lab 05/SONGA.py"

Please Enter Common Prime value of q: 23
Please Enter Primitive root Alpha value : 5

Please Enter Private Key of Party A : 6
Please Enter Private Key of Party B : 15
{'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7, 'H': 8,
'I': 9, 'J': 10, 'K': 11, 'L': 12, 'M': 13, 'N': 14, 'O': 15, 'P': 16,
'Q': 17, 'R': 18, 'S': 19, 'T': 20, 'U': 21, 'V': 22, 'W': 23, 'X': 24,
'Y': 25, 'Z': 26, '0': 27, '1': 28, '2': 29, '3': 30, '4': 31, '5': 32,
'6': 33, '7': 34, '8': 35, '9': 36}

{1: 'A', 2: 'B', 3: 'C', 4: 'D', 5: 'E', 6: 'F', 7: 'G', 8: 'H', 9: 'I',
10: 'J', 11: 'K', 12: 'L', 13: 'M', 14: 'N', 15: 'O', 16: 'P', 17: 'Q',
18: 'R', 19: 'S', 20: 'T', 21: 'U', 22: 'V', 23: 'W', 24: 'X', 25: 'Y',
26: 'Z', 27: '0', 28: '1', 29: '2', 30: '3', 31: '4', 32: '5', 33: '6',
34: '7', 35: '8', 36: '9'}
The Entered Common Prime value:
 q = 23 ,
 and Primitive root of
 Alpha value = 5

The Entered Private Key of Party:
 A = 6,
 and Private Key of Party:
 B = 15

The Calculation Key of User:
 A = 8

The Calculation Key of User:
 B = 19

The Generated Public Key of User:
 A = 2

The Generated Public Key of User:
 B = 2

Public KEYS are same, The Exchange is SUCCESSFUL

The PlainText is:
 helloIamcommingon23offebruary

 The PlainText in UPPERCASE is:
 HELLOIAMCOMMINGON23OFFEBRUARY

The Generated CipherText is:
  JGNNQKCOEQOOKPIQP45QHHGDTWCTO
```

```
The Decrypted CipherText is:
  HELLOIAMCOMMINGON23OFFEBRUARY

Process finished with exit code 0
```

4 The CipherText generated is:

JGNNQKCOEQOOKPIQP45QHHGDTWCT0