Assignment No 2

Group B (Network Security)

Aim: Implement a client and a server on different computers using python. Perform the authentication of sender between these two entities by using RSA digital signature cryptosystem.

Objectives:

1. To learn various client/server environments to use application layer protocols.
2. To understand the network security by using public key cryptography algorithms.

Theory:

RSA Digital Signature Scheme

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and the Private key is kept private.

An example of asymmetric cryptography :

- A client (for example browser) sends its public key to the server and requests for some data.
- The server encrypts the data using the client's public key and sends the encrypted data.
- Client receives this data and decrypts it.

Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of browser.

Digital signatures are used to verify the authenticity of the message sent electronically. A digital signature algorithm uses a public key system. The intended transmitter signs his/her message with his/her private key and the intended receiver verifies it with the transmitter's public key. A digital signature can provide message authentication, message integrity and non-repudiation services.

Algorithm

RSA Key Generation:

- Choose two large prime numbers p and q
- Calculate n=p*q
- Select public key e such that it is not a factor of (p-1)*(q-1)
- Select private key d such that the following equation is true (d*e)mod(p-1)(q-1)=1 or d is inverse of E in modulo (p-1)*(q-1)

RSA Digital Signature Scheme:

In RSA, d is private; e and n are public.

- Alice creates her digital signature using S=M^d mod n where M is the message
- Alice sends Message M and Signature S to Bob
- Bob computes M1=S^e mod n
- If M1=M then Bob accepts the data sent by Alice.

Basic Implementation:

```
# Function to find gcd
# of two numbers
def euclid(m, n):

    if n == 0:
        return m
    else:
        r = m % n
        return euclid(n, r)


# Program to find
# Multiplicative inverse
def exteuclid(a, b):

    r1 = a
    r2 = b
    s1 = int(1)
    s2 = int(0)
    t1 = int(0)
    t2 = int(1)

    while r2 > 0:
```

```
            q = r1//r2
            r = r1-q * r2
            r1 = r2
            r2 = r
            s = s1-q * s2
            s1 = s2
            s2 = s
            t = t1-q * t2
            t1 = t2
            t2 = t

    if t1 < 0:
            t1 = t1 % a

    return (r1, t1)

# Enter two large prime
# numbers p and q
p = 823
q = 953
n = p * q
Pn = (p-1)*(q-1)

# Generate encryption key
# in range 1<e<Pn
key = []

for i in range(2, Pn):

    gcd = euclid(Pn, i)
```

```
        if gcd == 1:
                key.append(i)


# Select an encryption key
# from the above list
e = int(313)

# Obtain inverse of
# encryption key in Z_Pn
r, d = exteuclid(Pn, e)
if r == 1:
        d = int(d)
        print("decryption key is: ", d)

else:
        print("Multiplicative inverse for\
        the given encryption key does not \
        exist. Choose a different encryption key ")


# Enter the message to be sent
M = 19070

# Signature is created by Alice
S = (M**d) % n

# Alice sends M and S both to Bob
# Bob generates message M1 using the
# signature S, Alice's public key e
```

```
# and product n.
M1 = (S**e) % n

# If M = M1 only then Bob accepts
# the message sent by Alice.

if M == M1:
        print("As M = M1, Accept the\
        message sent by Alice")
else:
        print("As M not equal to M1,\
        Do not accept the message\
        sent by Alice ")
```