Vivekanand Education Society's Institute of Technology

Department of AI & DS Engineering

Subject: Cryptography and System Security

Class: D11AD

Roll No: 46	**Name: Ashish Patil**
Practical No: 2	**Title: Analysis of RSA cryptosystem and Digital signature scheme**
DOP: 16/1/24	**DOS: 28/1/24**
Grades:	**LOs Mapped: LO2**
Signature:	

Title: Analysis of RSA cryptosystem and Digital signature scheme

DOP: 16/1/24

DOS: 23/1/24

Aim: Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA/El Gamal.

Theory:

RSA Cryptosystem Steps:

1. Key Generation:
   - Start by generating two large prime numbers, denoted as p and q. These primes, of roughly equal length, should have a product much larger than the intended message for encryption.
   - Compute n = p * q, the modulus for the RSA system.
   - Choose an integer e such that $1 < e < \varphi(n)$, where $\varphi(n) = (p-1)*(q-1)$ is Euler's totient function, and $\gcd(e, \varphi(n)) = 1$. This e becomes the public key exponent.
   - Use the extended Euclidean algorithm to find an integer d such that $d * e \equiv 1 \pmod{\varphi(n)}$. The public key is (n, e), and the private key is (n, d).

2. Encryption:
   - Convert the message, denoted as m, to an integer between 0 and n-1 using a reversible encoding scheme (e.g., ASCII or UTF-8).
   - Compute the ciphertext c as c ≡ m^e (mod n).

3. Decryption:
   - To decrypt the ciphertext c, compute the plaintext m as m ≡ c^d (mod n) using efficient modular exponentiation algorithms.

4. Use of Public & Private Keys:
   - The public key (n, e) is used for encryption, and the private key (n, d) is used for decryption.
   - Decryption is exclusive to the private key and cannot be achieved with the public key.
   - The public key is known to everyone, while the private key is kept secret by the user. Messages can be sent to the user using their public key, and only the user can decrypt the messages using their private key.

Digital Signature Scheme using RSA:
- Digital signatures verify the authenticity of electronically sent messages using public-key cryptography.
- The sender signs the message with their private key, and the recipient verifies it with the sender's public key.
- RSA Digital Signature Scheme:
  - In RSA, d is private, and e, 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, Bob accepts the data sent by Alice, ensuring message authentication, integrity, and non-repudiation services.

Program:

<u>Implementation of RSA:</u>

```
import math

def gcd(a, h):

  temp = 0
  while(1):
    temp = a % h if
    (temp == 0):
      return h
    a = h
    h = temp

p = 3
q = 7
n =
p*q e
```

```
= 2
phi = (p-1)*(q-1)
```

```
while (e < phi):
  if(gcd(e, phi) == 1):
    break
  else:
    e = e+1

k = 2
d = (1 + (k*phi))/e

# Message to be encrypted
msg = 10.0
print("Message data = ", msg)

# Encryption c = (msg ^ e) % n
c = pow(msg, e)
c = math.fmod(c, n)
print("Encrypted data = ", c)

# Decryption m = (c ^ d) % n
m = pow(c, d)
m = math.fmod(m, n)
print("Original Message Sent = ", m)
```

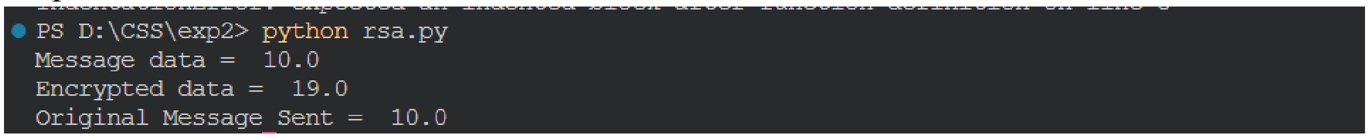
Implementation of Digital signature using RSA:

```
def euclid(m, n):

  if n == 0:

    return m
  else:
    r = m % n
    return    euclid(n,

r) 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)

p = 823
q = 953
n = p * q
Pn = (p-1)*(q-1)
key = []

for i in range(2, Pn):
  gcd = euclid(Pn, i)
  if gcd == 1:
    key.append(i)

e = int(313)

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 ")


# Message to be sent
M = 19070

# Signature is created by Alice
S = (Md) % 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 = (Se) % n
```

```python
# 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 ")
```

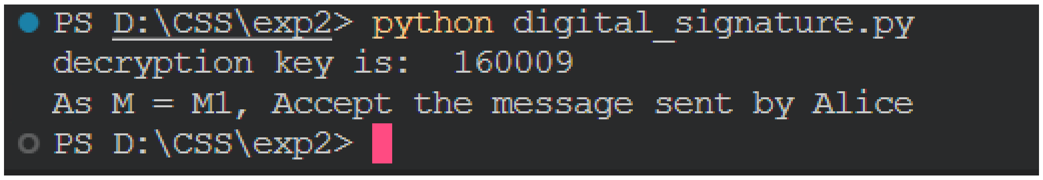
Output:

Implementation of RSA:



```
PS D:\CSS\exp2> python rsa.py
Message data =  10.0
Encrypted data =  19.0
Original Message Sent =  10.0
```

Implementation of Digital signature using RSA:



```
PS D:\CSS\exp2> python digital_signature.py
decryption key is:  160009
As M = M1, Accept the message sent by Alice
PS D:\CSS\exp2>
```

Conclusion:

The RSA cryptosystem and digital signature scheme, both implemented and analyzed, have been executed successfully.