Vivekanand Education Society's Institute of Technology

Department of AI & DS Engineering

Subject: Cryptography and System Security

Class: D11AD

Roll No: 26	**Name: Dyotak Kachare**
Practical No: 2	**Title: Analysis of RSA cryptosystem and Digital signature scheme**
DOP: 16/1/24	**DOS: 23/1/24**
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:

1. Steps of the RSA cryptosystem:

Step 1: Generating Keys: To start, we need to generate two large prime numbers, p and q. These primes should be of roughly equal length and their product should be much larger than the message we want to encrypt. Once we have the two primes, we can compute their product n = p*q, which will be the modulus for our RSA system. Next, we need to choose an integer e such that 1 < e < phi(n) and gcd(e, phi(n)) = 1, where phi(n) = (p-1)*(q-1) is Euler's totient function. This value of e will be the public key exponent.

To compute the private key exponent d, we need to find an integer d such that d*e = 1 (mod phi(n)). This can be done using the extended Euclidean algorithm. Our public key is (n, e) and our private key is (n, d).

Step 2: Encryption: To encrypt a message m, we need to convert it to an integer between 0 and n-1. This can be done using a reversible encoding scheme, such as ASCII or UTF-8. Once we have the integer representation of the message, we compute the ciphertext c as c = m^e (mod n).

Step 3: Decryption: To decrypt the ciphertext c, we compute the plaintext m as m = c^d (mod n). Again, we can use modular exponentiation algorithms to do this efficiently.

2. Use Public & Private Keys

The Public key is used for encryption, and the Private Key is used for decryption. Decryption cannot be done using a public key. The two keys are linked, but the private key cannot be derived from the public key. The public key is well known, but the private key is secret and it is known only to the user who owns the key. It means that everybody can send a message to the user using the user's public key. But only the user can decrypt the message using his private key.

1. The data to be sent is encrypted by sender A using the public key of the intended receiver
2. B decrypts the received ciphertext using its private key, which is known only to B. B replies to A encrypting its message using A's public key.

3. A decrypts the received ciphertext using its private key, which is known only to him.

3. Digital signature scheme using RSA/El Gamal

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.

RSA Digital Signature Scheme: In RSA, d is private; e and n are public.

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

Program:

Implementation of RSA:

```
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)
```

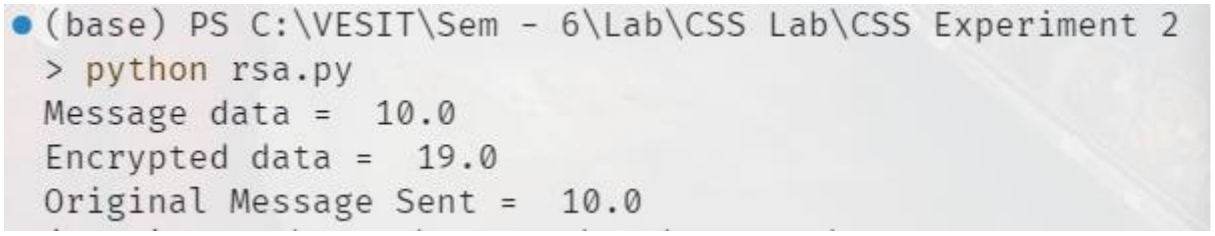
<u>Implementation of Digital signature using RSA:</u>

```
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 = (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 ")
```

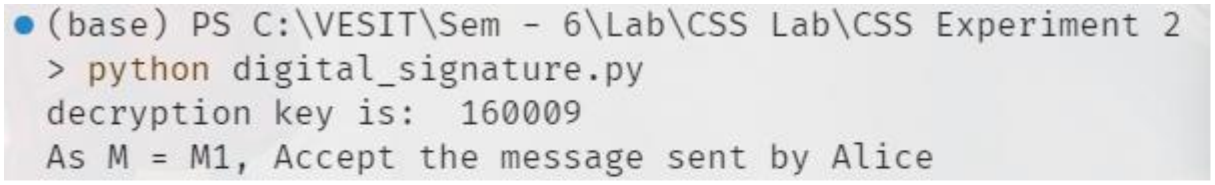
Output:

Implementation of RSA:



```
(base) PS C:\VESIT\Sem - 6\Lab\CSS Lab\CSS Experiment 2
> python rsa.py
Message data =  10.0
Encrypted data =  19.0
Original Message Sent =  10.0
```

Implementation of Digital signature using RSA:



```
(base) PS C:\VESIT\Sem - 6\Lab\CSS Lab\CSS Experiment 2
> python digital_signature.py
decryption key is:  160009
As M = M1, Accept the message sent by Alice
```

Conclusion:

Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA is done successfully.