NAME - RAM CHANDRA JANGIR
ROLL NO. - CS21M517
Subject - CS6530 - Assignment - 3.

RSA Algorithm:-

(i) Introduction

(ii) Setting up RSA public key and private key

(iii) Encryption using RSA

(iv) Decryption using RSA

(v) Decryption using RSA with CRT (Chinese Remainder Theorem)

(vi) Comparison of decryption using RSA and RSA with CRT

~~(vii)~~ from our program ~~output~~ Execution

(vii) Program running with some input values.

(i) Introduction:

The RSA algorithm is an asymmetric cryptography algorithm this means that it uses a public key and a private key (i.e. two different, mathematically linked keys). As their name suggest, a public key is shared publically, while private key is secret and must not be shared with anyone.

(ii) setting up RSA public key and private key :-

There are four basic steps involved in setting up and using an RSA public and private key pair.

step-1.

Choose two primes

Select two large prime numbers, p and q.

step-2.

Compute the RSA modulus

This is an easy step that involves multiplying p and q together. The resulting number $N = p*q$ is part of the public key. This number N will be the modulus that we use when we compute RSA encryptions and decryptions using modular arithmetic.

~~step-3~~

Calculate the totient function

$$phi(N) = \phi(N) = (p-1)*(q-1)$$

step -3 Public key (N, e)

The encryption key 'e' is chosen such that

$$1 < e < \phi(N)$$

and e is coprime to $\phi(N)$

$$\gcd(e, \phi(N)) = 1$$

step-4 Compute the private key

calculate the decryption key 'd' such that

$$e \cdot d = 1 \bmod \phi(N)$$

d canbe found using the extended euclidean algorithm. The pair (n, d) makes up the private key.

(iii) Encryption using RSA:

To encrypt a plaintext M using a RSA public key we simply represent the plaintext as a number between '0' and N-1 and then compute ciphertext C as:

$$C = (M)^e . \bmod N$$

(iv) Decryption using RSA:

To decrypt a ciphertext 'C' using private key (N,d) the plaintext can be found using:

$$M = C^d . \bmod N$$

(V) Decryption using RSA with CRT (Chinese Remainder Theorem):-

The complexity of RSA decryption $M = C^d \bmod N$ depends directly on the size of d and N. This will need more computational time. CRT (Chinese Remainder Theorem) helps to use the following optimization for decryption. The following values are precomputed and stored as part of the private key (N,d).

- p and q : are the prime numbers
- $dp = d \bmod (p-1)$
- $dq = d \bmod (q-1)$
- $qInv = q^{-1} \bmod (p)$

These values allow to compute the exponentiation $m = c^d \bmod pq$ more efficiently computed as follows:

- $m_p = C^{dp} \cdot \mathrm{mod}\ (p)$.
- $m_q = C^{dq} \cdot \mathrm{mod}\ (q)$
- $h = qInv * (m_p - m_q) \ \mathrm{mod}\ (p)$

& if $m_p < m_q$ then we can calculate h as following

$$h = qInv * (m_p + p - m_q) \ \mathrm{mod}\ (p)$$

- $m = m_q + h * q$

This is more efficient than computing $m = C^d \cdot \mathrm{mod}\ (N)$ even though two modular exponentiations have to be computed. The reason is that these two modular exponentiations both use a smaller exponent and a smaller modulus.

(vi) <u>Comparison of decryption using RSA and RSA with CRT from our program Execution:-</u>

To compare decryption with RSA and RSA with CRT, we have ~~calculated~~ calculated the time taken (execution time) of both the functions in milliseconds.

vi) Program Execution Time with RSA Decryption and RSA Decryption using CRT (Chinese Remainder Theorem):

```
rjangir@rjangir-linux:/local/mnt/workspace/rjangir/WORKSPACE/rsa$ ./rsa

Enter first prime number (p): 137
Enter another prime number (q): 131
Enter a message to encrypt (Plain text) : 513

Compute RSA modulus  n = p * q = 137 * 131 = 17947
Compute  φ(n) = totient = (p – 1) * (q – 1) = 136 * 130 = 17680
Possible values of (public key, private key) <--> (e, d) are :
(3, 11787)
(7, 10103)
(11, 11251)
(19, 15819)
(23, 7687)
(29, 1829)
(31, 1711)
(37, 14813)
(41, 3881)
(43, 2467)

The computed parameters for CRT(Chinese Raminder Theorem):  dP=91, dQ=87, qInv=114

----------------------------------------------------------------------------------------

RSA Encryption:

        Plaintext '513'
        Public key (e, n) : (3, 17947)
        The encrypted message (ciphertext) is : 8363


        Encryption time is 0.064000 ms

----------------------------------------------------------------------------------------

RSA Decryption:

        Ciphertext 8363
        Private key (d , n) <--> (11787 , 17947)
        The decrypted message (plaintext) is : 513


        Deccryption time is 0.842000 ms

----------------------------------------------------------------------------------------

Decryption usig CRT(Chine Remainder Theorem) :

        Ciphertext 8363
        Private Key (d, n):(11787, 17947)
        The decrypted message ( plaintext ) with CRT is : 513


        Decryption time with CRT(Chinese Remainder Theorem) is 0.065000 ms

        -------------------------------------------------------------------

         Comparison between RSA Decryption Time and RSA Decryption Time using CRT(Chinese Remainder Theorem)

         RSA Decryption Time using CRT = (RSA Decryption Time/CRT Decryption Time)/100= 7.719715 times fast
-----------------------------------------------------------------------------
rjangir@rjangir-linux:/local/mnt/workspace/rjangir/WORKSPACE/rsa$
```

Conclusion:

From above program output, we find that execution time of RSA decryption using CRT is **0.065ms** whereas normal RSA Decryption takes **0.842ms**. Hence RSA decryption using CRT (Chinese Remainder Theorem) is around **7.71** times faster than normal RSA decryption.

The RSA Decryption with CRT used here requires very low computational power. The decryption used here is RSA with CRT which enhances the speed of decryption compared to basic RSA algorithm.