Experiment No. 4

Title: RSA Cipher

Batch: A3 **Roll No.: 16010421059** **Experiment No.:04**

Aim: To implement RSA cipher

Resources needed: Windows/Linux

Theory: Pre Lab/ Prior Concepts:

RSA is a public key algorithm named after its inventers Rivest, Shamir and Adleman. The characteristics of public key cryptography (which is also called as Asymmetric Cryptography) are as below:

- It has two keys. One key is called as private key and the other one is called as public key. Everyone who uses this cryptography has to have two keys each.
- Keys used for encryption and decryption should be different. If one is used for encryption, then other must be used for decryption. Any key can be used for encryption and then remaining key can be used for decryption.
- Public Key Cryptography is based on solid foundation of mathematics.
- It has large computational overheads. Hence ciphertext generated is of much larger size than the plaintext. Hence it is normally used for encrypting small size of data blocks. For example, passwords, symmetric keys, etc. It is not preferred to encrypt large files.
- RSA algorithm gets its security from the fact that it is extremely difficult to factorize large prime number.
- Security of the algorithm depends on the size of the key. Greater the size of the key, larger is the security. The key length is variable. The most commonly used key length is 512 bits.

Procedure / Approach /Algorithm / Activity Diagram:

A. Key generation Algorithm:

1. Choose large prime numbers p and q.
2. Calculate product n= pq. The value of n can be revealed publicly, but n is large enough that even supercomputers cannot factor it in a reasonable amount of time (years or even centuries).
3. Calculate phi = (p-1)(q-1)
4. Select e < phi such that it is relatively prime to phi. The public key is (e, n).
5. Determine d such that ed = 1 mod phi. The private key is (d, n). d, p ,q and phi are kept secret, only (e,n) is made public.

B. Encryption:

6. Suppose M is the message.
7. Encrypt this message using public key of the receiver as follows
   C = M^e mod n.

C. Decryption:

8. Decrypt the C generated in step 7 using private key of the receiver as follows M = C^d mod n.

Results: (Program printout as per the format)

Input:

```
#include <iostream>
#include <vector>
#include <cstdlib>

using namespace std;
int euclid(int m, int n) {
 if (n == 0) {
 return m;
 } else {
 int r = m % n;
 return euclid(n, r);
 }
}

pair<int, int> exteuclid(int a, int b)
{  int r1 = a;
 int r2 = b;
 int s1 = 1;
 int s2 = 0;
 int t1 = 0;
```

```
int t2 = 1;
while (r2 > 0) {

int q = r1 / r2;

int r = r1 - q * r2;

r1 = r2;

r2 = r;

int s = s1 - q * s2;

s1 = s2;

s2 = s;

int t = t1 - q * t2;

t1 = t2;

t2 = t;

}


if (t1 < 0) {

t1 = t1 % a;

}


return make_pair(r1, t1); }


bool isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i <= sqrt(num); ++i)

{  if (num % i == 0) {

return false;
}

}


return true;
```

```
}

int main() {

 int p, q;

 cout << "Enter value of p:

";  cin >> p;

 cout << "Enter the value of q:

";  cin >> q;


 isPrime(p);

 isprime_number(q);


 int n = p * q;

 int Pn = (p - 1) * (q - 1);


 vector<int> key;

 for (int i = 2; i < Pn; ++i)

{  int gcd = euclid(Pn, i);  if

(gcd == 1) {

 key.push_back(i);

 }

 }


 int e = key[rand() % key.size()];
 cout << "Encryption key: " << e <<endl;


 pair<int, int> result = exteuclid(Pn, e);

 int r = result.first;

 int d = result.second;


 if (r == 1) {
```

```
d = d < 0 ? d + Pn : d;

cout << "Decryption key is: " << d << std::endl;

} else {

cout << "Multiplicative inverse for the given encryption key does not exist. Choose a
different  encryption key." << endl;

}



int M;

cout << "Enter the message to be sent: ";

cin >> M;



int S = 1;

for (int i = 0; i < d; ++i) {

S = (S * M) % n;

}

cout << "Signature created by Sender: " << S << endl;



int M1 = 1;

for (int i = 0; i < e; ++i) {

M1 = (M1 * S) % n;

}
cout << "Message generated at reciever's end: " << M1 << endl;



if (M == M1) {

cout << "As M = M1, Accept the message sent by Sender" <<endl;

} else {

cout << "As M not equal to M1, Do not accept the message sent by Sender." <<

endl;  }



return 0;

}
```

Output:

```
Output
/tmp/IDMqblfFLS.o
Enter value of p: 23
Enter the value of q: 43
Encryption key: 893
Decryption key is: 149
Enter the message to be sent: 474456
Signature created by Sender: 308
Message generated at reciever's end: 725
As M not equal to M1, Do not accept the message sent by Sender.
```

Questions:

1. Generating RSA Key Pair, Encrypting, and Decrypting the Message:

First, let's generate the RSA key pair for the receiver, Alice. In RSA, we choose two large prime numbers, p and q, and compute the public and private keys based on those primes.

Given values:
- Message: "Hello World"
- Message represented as numbers: 07 04 11 11 14 26 22 14 17 11 03
- p = 11
- q = 3

Now, let's calculate the receiver's (Alice's) RSA key pair:

1. Compute n:
   n = p * q = 11 * 3 = 33

2. Compute φ(n) (Euler's totient function):
   φ(n) = (p - 1) * (q - 1) = (11 - 1) * (3 - 1) = 10 * 2 = 20

3. Choose a public exponent, e:
   Typically, e is chosen to be a small prime number greater than 1 and less than φ(n). Let's choose e = 7.

4. Compute the private exponent, d:
   To find d, we need to calculate the modular multiplicative inverse of e modulo φ(n). In this case, d is the multiplicative inverse of 7 mod 20. You can use the Extended Euclidean Algorithm to find d. In this case, d = 3.

So, the public key is (n, e) = (33, 7), and the private key is (n, d) = (33, 3).

Now, let's encrypt the message "Hello World" using Alice's public key:

- Convert each number to its corresponding ciphertext using the public key: C = M^e (mod n)
- M is the plaintext number, e is Alice's public exponent, and n is the modulus.

Encryption:
07^7 (mod 33) = 20
04^7 (mod 33) = 30
11^7 (mod 33) = 22
11^7 (mod 33) = 22
14^7 (mod 33) = 27
26^7 (mod 33) = 4
22^7 (mod 33) = 3
14^7 (mod 33) = 27
17^7 (mod 33) = 19
11^7 (mod 33) = 22
03^7 (mod 33) = 15

The encrypted message is: 20 30 22 22 27 4 3 27 19 22 15.

To decrypt the message, Bob (the receiver) will use his private key:
- Convert each ciphertext number back to the original plaintext using the private key: M = C^d (mod n)
- C is the ciphertext number, d is Bob's private exponent, and n is the modulus.

Decryption:
20^3 (mod 33) = 07
30^3 (mod 33) = 04
22^3 (mod 33) = 11
22^3 (mod 33) = 11
27^3 (mod 33) = 14
4^3 (mod 33) = 26
3^3 (mod 33) = 22
27^3 (mod 33) = 14
19^3 (mod 33) = 17
22^3 (mod 33) = 11
15^3 (mod 33) = 03

The decrypted message is: "Hello World."

2. Attacks on RSA:

RSA, like any cryptographic system, is susceptible to various attacks, including:

- Brute Force Attack: An attacker tries every possible private key until the correct one is found. RSA keys with sufficient length are considered secure against brute force attacks.

- Factorization Attack: If an attacker can factorize the modulus n into its prime factors (p and q), they can compute the private key. This is the main weakness of RSA, and its security depends on the difficulty of factoring large numbers.

- Chosen-Plaintext Attack: An attacker with knowledge of the public key can encrypt chosen plaintexts to derive information about the private key. This attack emphasizes the importance of using a secure padding scheme.

- Timing Attack: Attackers can use information about the time it takes to perform private key operations to deduce information about the key. This is particularly relevant in implementations where the time to compute the private key differs based on the correctness of specific bits.

- Side-Channel Attacks: Attackers may exploit various side channels such as power consumption, electromagnetic radiation, or acoustic emanations to infer private key information.

To defend against these attacks, it's crucial to use secure key lengths, employ strong padding schemes like RSA-OAEP, and implement RSA securely in practice. Additionally, regularly updating keys and monitoring for any unusual activities are essential to maintain security.

Outcomes:

CO 2 : Illustrate different cryptographic algorithms for security

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

Implemented RSA cipher successfully.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References: Books/ Journals/ Websites:

1. Behrouz A. Forouzan, “Cryptography and Network Security”, Tata McGraw Hill
2. William Stalling, “Cryptography and Network Security”, Prentice Hall