Experiment No: 8

Aim: To study and implementation of RSA Cipher.

Introduction:

The RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works with two different keys which means Public Key and Private Key. As the name implies the public key is given to everyone and the private key is kept private.

An examples of asymmetric cryptography are given below:

1. A client sends its public key to a server with requests for specific information.

2. The server encrypts the data using the client's public key and sends encrypted data.

3. The client receives this data and deletes the recording.

RSA is key pair generator. Process of calculating the public key and private key: -

1. Choose two different random random numbers p and q

2. Count n = p q

n is a module for public key and private keys

3. Count ϕ (n) = (p - 1) (q - 1)

4. Select the whole number k so that 1 <k <ϕ (n) and k are cohesive to ϕ (n): k and ϕ (n) with no elements other than 1; gcd (k, ϕ (n)) = 1.

5. k is issued as a public key advertisement

6. Calculate d to satisfy d k ≡ 1 (mod ϕ (n)) eg .: D k = 1 + x ϕ (n) by a certain number x

7. d is maintained as a private key provider

The public key consists of n and k.

The private key consists of p, q, and the private exponent d.

Encryption: -

- Encrypted text will be pow(message,k) mod n

Decryption: -

- Decrypted text will be pow(cipher_text,d) mod n

Program (Source Code):

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

int encry(long int n, long int e, long int msg)
{
	long int earr[e+1];

	earr[e] = 1;

	for(long int i=0; i<e; i++)
		earr[i] = msg;

	long int m=1;
	while(m<e)
	{
		long int k=0;
		if(fmod((e/m), 2) == 0)
		{
			long int i=0;
			for(i=0; i<=(e+1)/m - 2; i+=2)
				earr[k++] = (earr[i] * earr[i+1]) % n;

			earr[k] = earr[i];
		}
```

```
		else
		{
			for(long int i=0; i<=(e+1)/m - 1; i+=2)
				earr[k++] = (earr[i] * earr[i+1]) % n;
		}

		m = m*2;
	}

	return earr[0];
}

int decry(long int n, long int d, long int msg)
{
	long int darr[d+1];

	darr[d] = 1;

	for(long int i=0; i<d; i++)
		darr[i] = msg;

	long int m=1;
	while(m<d)
	{
		long int k=0;
		if(fmod((d/m), 2) == 0)
		{
			long int i=0;
			for(i=0; i<=(d+1)/m - 2; i+=2)
				darr[k++] = (darr[i] * darr[i+1]) % n;
```

```
                darr[k] = darr[i];
            }
            else
            {
                for(long int i=0; i<=(d+1)/m - 1; i+=2)
                    darr[k++] = (darr[i] * darr[i+1]) % n;
            }

            m = m*2;
        }

        return darr[0];
}

int main()
{
        long int p,q,n,euler,e,d;

        cout<<"enter prime value for p:"<<endl;
        cin>>p;
        cout<<"enter prime value for q:"<<endl;
        cin>>q;

        for(int i=2; i<p/2+1; i++)
        {
            if(p % i == 0)
            {
                cout<<"either p or q is not prime"<<endl;
                return 0;
```

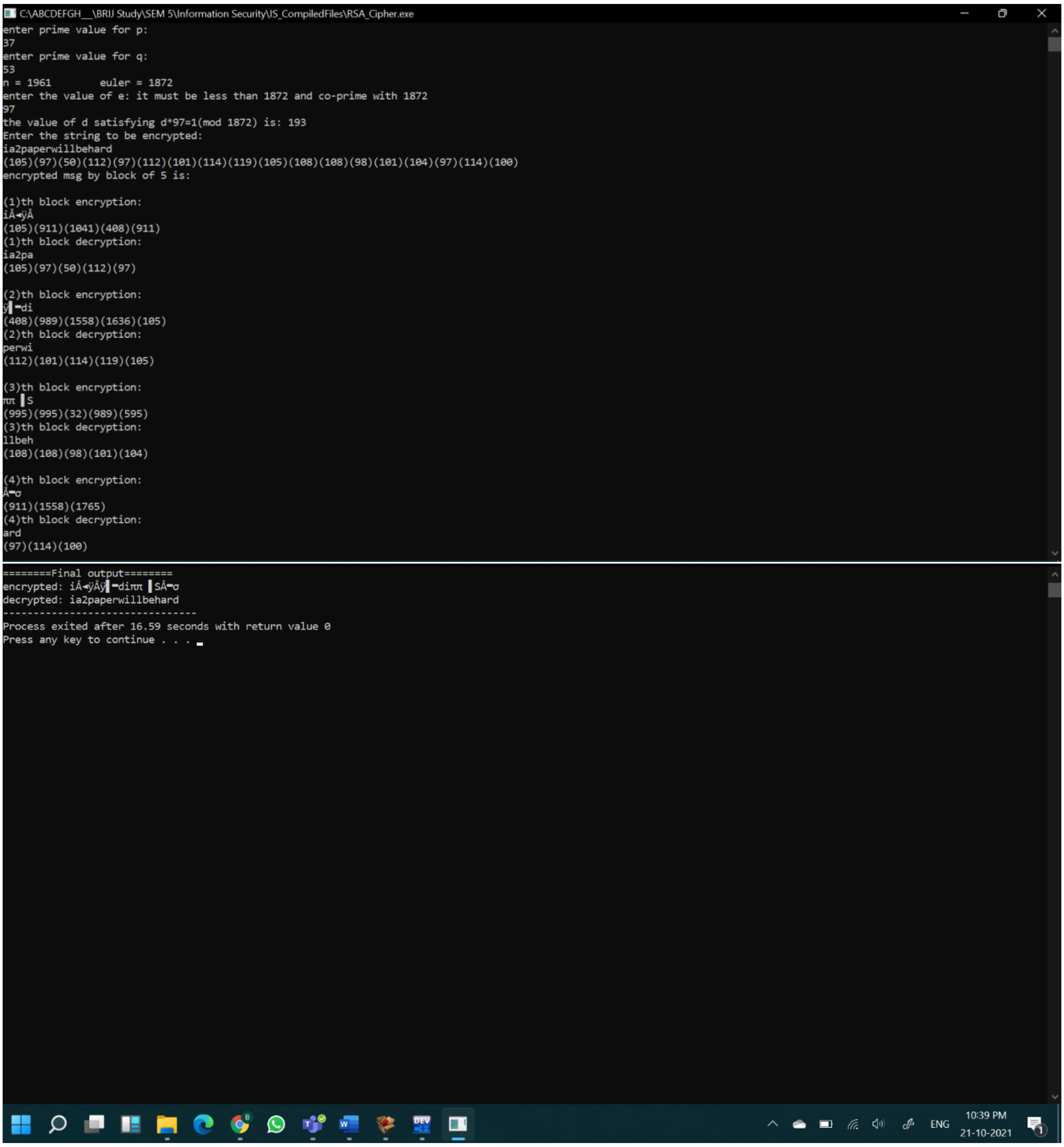
```
		}
	}

	for(int i=2; i<q/2+1; i++)
	{
		if(q % i == 0)
		{
			cout<<"either p or q is not prime"<<endl;
			return 0;
		}
	}

	n = p*q;
	euler = (p-1)*(q-1);

	cout<<"n = "<<n<<"\teuler = "<<euler<<endl;

	cout<<"enter the value of e: it must be less than "<<euler<<" and co-prime with
"<<euler<<endl;
	cin>>e;

	for(int i=1; i>0; i++)
	{
		if(((euler*i)+1) % e == 0)
		{
			d = ((euler*i)+1)/e;
			i=-1;
		}
	}

	cout<<"the value of d satisfying d*"<<e<<"=1(mod "<<euler<<") is: "<<d<<endl;
```

```
string in,eout="",dout="";
cout<<"Enter the string to be encrypted: "<<endl;
getline(cin>>ws, in);

for(int i=0; i<in.length(); i++)
        cout<<"("<<(int)in[i]<<")";

cout<<endl;

long int msg[5];

cout<<"encrypted msg by block of 5 is: "<<endl;

int j=0,x=1,y;

if(in.length() % 5 == 0)
        y = in.length()/5;
else
        y = in.length()/5 + 1;

for(int a=0; a<y; a++)
{
        int z = 5;
        if(a == y-1 && in.length() % 5 != 0)
        {
                z = in.length() % 5;
        }

        cout<<endl<<"("<<x<<")"<<"th block encryption:"<<endl;
```

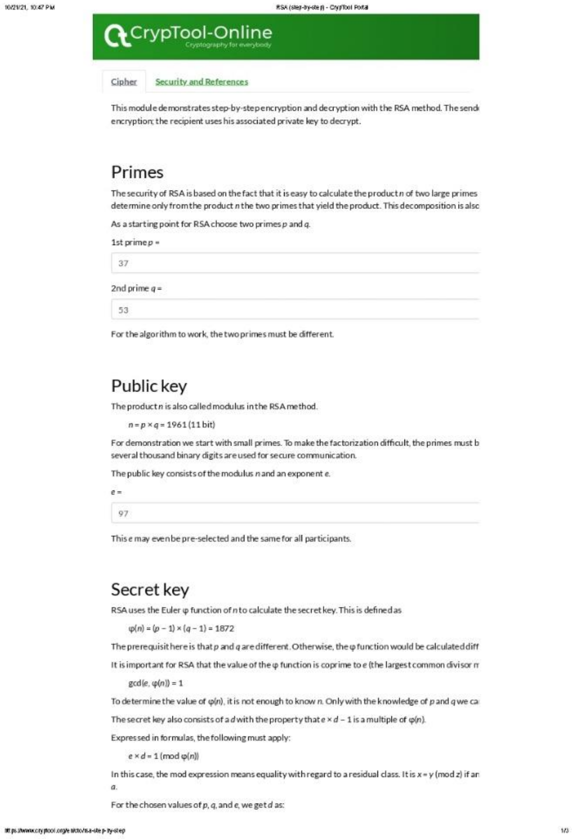
```
for(int i=0; i<z; i++)
{
	msg[i] = (int)in[j++];
	msg[i] = encry(n, e, msg[i]);
	cout<<(char)msg[i];
	eout += (char)msg[i];
}

cout<<endl;

for(int i=0; i<z; i++)
	cout<<"("<<msg[i]<<")";

cout<<endl;

cout<<"("<<(x++)<<")"<<"th block decryption:"<<endl;

for(int i=0; i<z; i++)
{
	msg[i] = decry(n, d, msg[i]);
	cout<<(char)msg[i];
	dout += (char)msg[i];
}

cout<<endl;

for(int i=0; i<z; i++)
	cout<<"("<<msg[i]<<")";

cout<<endl;
```

```
	}

	cout<<"========Final output========"<<endl<<"encrypted: "<<eout<<endl<<"decrypted: "<<dout;

	return 0;
}
```

Output (Program):



Output (Cryptool):



10/21/21, 10:47 PM RSA (step-by-step) - CrypTool Portal

CrypTool-Online
Cryptography for everybody

Cipher Security and References

This module demonstrates step-by-step encryption and decryption with the RSA method. The send
encryption; the recipient uses his associated private key to decrypt.

Primes

The security of RSA is based on the fact that it is easy to calculate the product n of two large primes
determine only from the product n the two primes that yield the product. This decomposition is also

As a starting point for RSA choose two primes p and q.

1st prime p =

37

2nd prime q =

53

For the algorithm to work, the two primes must be different.

Public key

The product n is also called modulus in the RSA method.

$n = p \times q = 1961$ (11 bit)

For demonstration we start with small primes. To make the factorization difficult, the primes must b
several thousand binary digits are used for secure communication.

The public key consists of the modulus n and an exponent e.

e =

97

This e may even be pre-selected and the same for all participants.

Secret key

RSA uses the Euler φ function of n to calculate the secret key. This is defined as

$\varphi(n) = (p - 1) \times (q - 1) = 1872$

The prerequisite here is that p and q are different. Otherwise, the φ function would be calculated diff

It is important for RSA that the value of the φ function is coprime to e (the largest common divisor m

$\gcd(e, \varphi(n)) = 1$

To determine the value of $\varphi(n)$, it is not enough to know n. Only with the knowledge of p and q we ca

The secret key also consists of a d with the property that $e \times d - 1$ is a multiple of $\varphi(n)$.

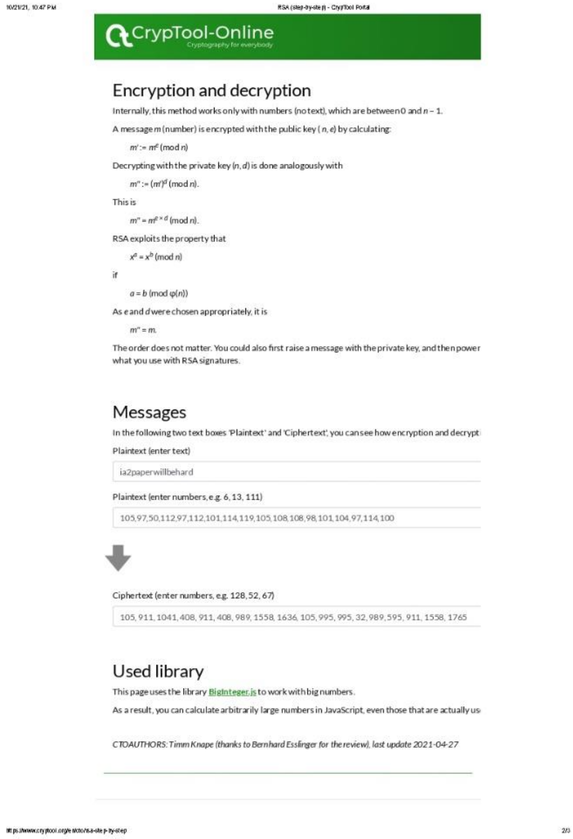
Expressed in formulas, the following must apply:

$e \times d = 1 \pmod{\varphi(n)}$

In this case, the mod expression means equality with regard to a residual class. It is $x = y \pmod z$ if an
a.

For the chosen values of p, q, and e, we get d as:

https://www.cryptool.org/en/cto/rsa-step-by-step 1/3



10/21/21, 10:47 PM RSA (step-by-step) - CrypTool Portal

CrypTool-Online
Cryptography for everybody

Encryption and decryption

Internally, this method works only with numbers (no text), which are between 0 and $n - 1$.

A message m (number) is encrypted with the public key (n, e) by calculating:

$m' := m^e \pmod n$

Decrypting with the private key (n, d) is done analogously with

$m'' := (m')^d \pmod n$.

This is

$m'' = m^{e \times d} \pmod n$.

RSA exploits the property that

$x^a = x^b \pmod n$

if

$a = b \pmod{\varphi(n)}$

As e and d were chosen appropriately, it is

$m'' = m$.

The order does not matter. You could also first raise a message with the private key, and then power
what you use with RSA signatures.

Messages

In the following two text boxes 'Plaintext' and 'Ciphertext', you can see how encryption and decrypt

Plaintext (enter text)

ia2paperwillbehard

Plaintext (enter numbers, e.g. 6, 13, 111)

105,97,50,112,97,112,101,114,119,105,108,108,98,101,104,97,114,100

Ciphertext (enter numbers, e.g. 128, 52, 67)

105, 911, 1041, 408, 911, 408, 989, 1558, 1636, 105, 995, 995, 32, 989, 595, 911, 1558, 1765

Used library

This page uses the library BigInteger.js to work with big numbers.

As a result, you can calculate arbitrarily large numbers in JavaScript, even those that are actually us

CTOAUTHORS: Timm Knape (thanks to Bernhard Esslinger for the review), last update 2021-04-27

https://www.cryptool.org/en/cto/rsa-step-by-step 2/3

Cryptanalysis:

We will create a state-of-the-art research platform that introduces as many ways as possible to attack the RSA algorithm some of which are very new. We will also show real computer demos with simple tools. The goal is NOT to define all the background figures.

Attacks:
1) Small factors
2) Fermat factorization
3) Batch GCD
4) Elliptic Curve Method (ECM)
5) Weak entropy
6) Smooth p-1 or p+1
7) Fault injection
8) Small private exponent
9) Known partial bits
10) p/q near a small fraction
11) Shared bits

12) Weaknesses in signatures
13) Side channel attacks
14) Number Field Sieve (NFS)
15) Shor quantum algorithm

Applications:

- Banking: - The RSA algorithm is widely used by banks to protect their personal information, such as customer information and transaction records. Other cases are credit cards and office computers.
- Telecommunications :- RSA algorithm helps encrypt telephone data such as concerns about privacy issues.
- Ecommerce :- The RSA algorithm helps to protect transaction user identity.

References:

1. https://www.youtube.com/watch?v=rlJTMUBXhKE
2. https://www.youtube.com/watch?v=14tGYQNibGI
3. https://www.geeksforgeeks.org/modulus-two-float-double-numbers/