Experiment No: 10

Aim: To study and implement RSA Digital Signature

Introduction: RSA is a asymmetric type cipher algorithm means it contains both the public and private key. As the name suggest the public key is know to all and the private key is private. Since this is asymmetric, nobody else except browser can decrypt the data even if a third party has public key of browser.

RSA Digital Signature Scheme: In RSA, d is private; e and 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 then Bob accepts the data sent by Alice.

- **Generating Public Key :**

Select two prime no's. Suppose P = 53 and Q = 59.

Now First part of the Public key : n = P*Q = 3127.

We also need a small exponent say e :
But e Must be An integer.

Not be a factor of n.
$1 < e < \Phi(n)$

Let us now consider it to be equal to 3.

Our Public Key is made of n and e

- **Generating Private Key :**

We need to calculate $\Phi(n)$:

Such that $\Phi(n) = (P-1)(Q-1)$

so, $\Phi(n) = 3016$

Now calculate Private Key, d : d =
$(k*\Phi(n) + 1) / e$ for some integer k For
k = 2, value of d is 2011.

Now we are ready with our – Public Key (n = 3127 and e = 3) and Private Key(d = 2011)

Now we will encrypt “HI” :

Convert letters to numbers : H = 8 and I = 9

Thus Encrypted Data c = 89e mod n.

Thus our Encrypted Data comes out to be 1394

Now we will decrypt 1394 :

Decrypted Data = cd mod n.

Thus our Encrypted Data comes out to be 89 8

= H and I = 9 i.e. "HI".Program

Program (Source Code):

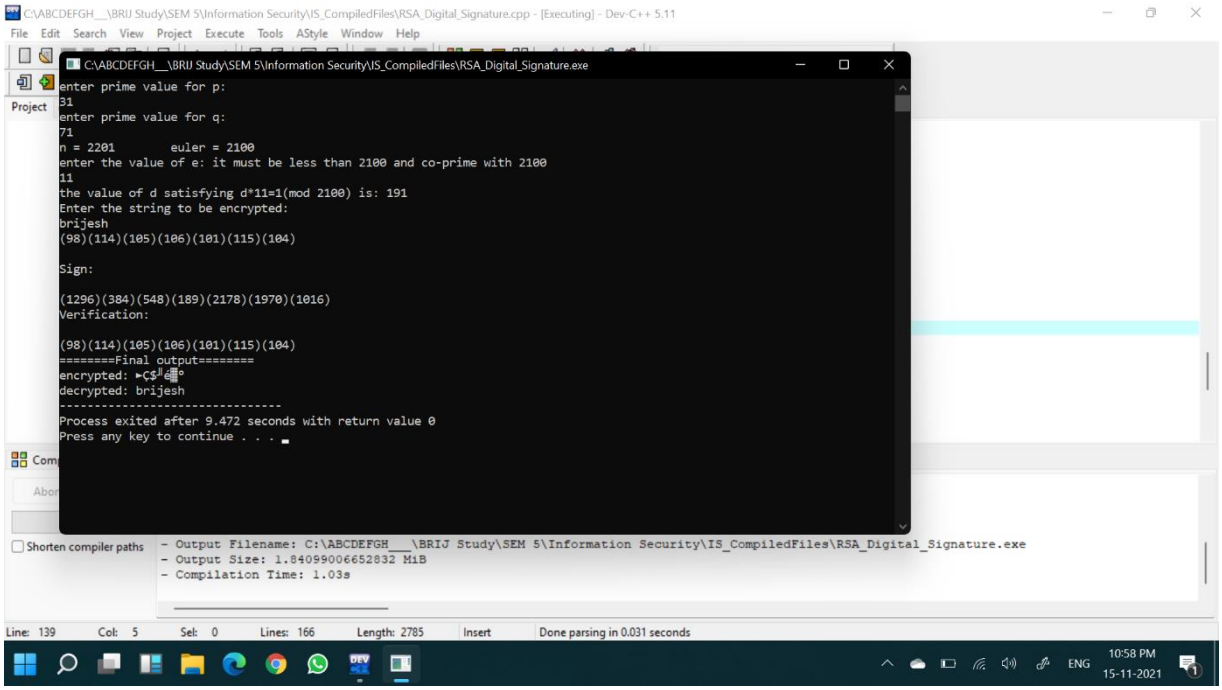
```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

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[in.length()];
```

```
        int j=0,x=1,y;

                cout<<endl<<"Sign:"<<endl;
                for(int i=0; i<in.length(); i++)
                {
                        msg[i] = (int)in[j++];
                        msg[i] = encry(n, d, msg[i]);
                        eout += (char)msg[i];
                }

                cout<<endl;

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

                cout<<endl;

                cout<<"Verification:"<<endl;

                for(int i=0; i<in.length(); i++)
                {
                        msg[i] = decry(n, e, msg[i]);
                        dout += (char)msg[i];
                }

                cout<<endl;

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

                cout<<endl;


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

        return 0;
}
```

Output (Program):



Cryptanalysis:

Due to the principle, a quantum computer with a sufficient number of entangled quantum bits (qubits) can quickly perform a factorization because it can simultaneously test every possible factor simultaneously. So far, however, there is no known quantum computer, which has just an approximately large computing capacity. Thus, effective quantum computers are currently a myth that will probably not be ready for production in the next few years. However, factoring may be over in 20 years and RSA loses its security.The larger the prime factors are, the longer actual algorithms will take and the more qubits will be needed in future quantum computers. At the moment, the product (modulus) should consist of at least 4096 binary digits to be secure. Prime numbers may not be reused! If you have two products each consisting of two primes and you know that one of the primes used is the same, then this shared prime can be determined quickly with the Euclidean algorithm. And by dividing the products by this shared prime, one obtains the other prime number.

Early implementations of RSA made this mistake to reduce the time it takes to find a prime number. Also on resource-constrained devices it came in recent times due to lack of entropy. Current implementations should not commit this error anymore.

Applications:

A client (for example browser) sends its public key to the server and requests for some data.

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

Client receives this data and decrypts it. So this is used in HTTPS for getting the information by the browser. And also in social computing it is used for digital certifications and to encrypt the text in chat and various file also and then it is verified at the receiver end.

References:

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