Student Name: Pragati Shankar	USN: 1SI19CS090	Batch No: B2	Date: 17-01-2023

Evaluation:

Write Up (10 marks)	Clarity in concepts (10 marks)	Implementation and execution of the algorithms (10 marks)	Viva (05 marks)	Total (35 marks)

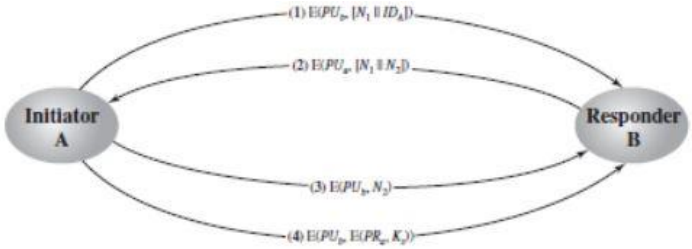
Sl.No	Name of the Faculty In-Charge	Signature
1.	H K Vedamurthy	
2.	Gururaj S P	

Question No: 12

Implement RSA algorithm using client-server concept. Using this illustrate secret key distribution scenario with confidentiality and authentication. The program should support the following:

i. Both client and server generates {PU, PR} and distributes PU to each other.

ii. ii. Establish a secret key K between client and server by exchanging the messages as shown in below figure.



Algorithm:

1. Generate *e,p,q* using random number generator.
2. Calculate n value , n=p×q.
3. Determine public and private keys *(e,n) and (d,n).*
4. Accept plain text in string format and assign numbers between 0 to 26 for characters (a to z)
5. Plain text in decimal string {p1,p2,p3….} is encrypted using public key as shown in fig 13.

$$C_1 = P_1^e \bmod n$$
$$C_2 = P_2^e \bmod n$$
$$\vdots$$

Fig 1

Recovered decimal text

$$P_1 = C_1^d \bmod n$$
$$P_2 = C_2^d \bmod n$$
$$\vdots$$

Fig 2.

6. Transmit the cipher text in decimal format to server using through sockets for decryption.

Server should decrypt the cipher text {c1,c2,c3…} shown in fig 2. and print the string in character format back to screen.

CODE:

Server side:

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

int p, q, e, d, n, phi; // global variables

// server's keys
int PUs[2], PRs[2];
//client's public key
int PUc[2];
int sock;

void createServer(int port) // TCP connection
{
	int sersock = socket(AF_INET, SOCK_STREAM, 0);
	struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
	bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
	cout << "\nServer Online. Waiting for client...." << endl;
	listen(sersock, 5);
	sock = accept(sersock, NULL, NULL);
	cout << "Connection Established." << endl;
}

int gcd(int a, int b)
{
	return b==0 ? a : gcd(b, a%b);
}

void genKey()
{
	cout << "\nEnter two prime numbers (>100): "; cin >> p >> q;
	n = p * q ;
	phi = (p-1) * (q-1);
	srand(time(NULL));
	do
	{
		e = rand()%(phi-2)+2;
	} while(gcd(e,phi) != 1);
	for(d=1; d<phi; d++)
	{
		if((d*e)%phi == 1)
			break;
	}
	PUs[0] = e; PUs[1] = n; // public key
	PRs[0] = d; PRs[1] = n; // private key
	cout << "\nPublic key , PUs = {" << e << ", " << n << "}" << endl;
	cout << "Private key, PRs = {" << d << ", " << n << "}" << endl;
```

```
}

void shareKey() // first send then receive
{
	send(sock, &PUs, sizeof(PUs), 0); // send Server's public key to client
	recv(sock, &PUc, sizeof(PUc), 0); // receive public key from client
	cout << "Sent Server's Public key to client." << endl;
	cout << "\nPublic key received from client : {" << PUc[0] << ", " << PUc[1] <<"}" << endl;
}

// C = M^e mod n
int encrypt(int M, int P[2]) // P = {e or d, n}
{

	int C=1;
	for(int i=1; i<=P[0]; i++)
	{
		C = (C * M) % P[1];
	}
	return C;
}

int decrypt(int C, int P[2])
{
	return encrypt(C,P);
}

int main()
{
	int port; cout<<"\nEnter port : "; cin>>port;
	srand(time(NULL));
	createServer(port);
	genKey();
	shareKey(); // share public keys
	int ID; cout<<"\nEnter Server's ID number (<100): "; cin>>ID;
	int N1 = rand()%100; // nonce
	cout << "Nonce generated, N1 = " << N1 << endl;

	// step-1: send En(PUc, [N1||ID]) to client
	int msg = N1*100 + ID; // append ID to nonce
	int cipher = encrypt(msg, PUc);
	send(sock, &cipher, sizeof(cipher), 0);
	cout << "Sent encrypted (N1||ID) to client : " << cipher << endl;

	// step-2: recv cipher from client and Dec(PRs, (N1||N2))
	recv(sock, &cipher, sizeof(cipher), 0);
	cout << "\nReceived encrypted (N1||N2) from client : " << cipher << endl;
	msg = decrypt(cipher, PRs);
	int N1c = msg/100; // N1 received from client
	int N2 = msg%100;
	cout << "Decrypted Server's Nonce, N1 = " << N1c << endl;
	cout << "Decrypted Client's Nonce, N2 = " << N2 << endl;
	if(N1 != N1c) {cout << "\nNonce didn't match!\n";
```

```
		exit(-1);}
	else
	{
		cout << "------ Client Authenticated ------" << endl;
	}

	// step-3: send En(PUc, N2) to client
	cipher = encrypt(N2, PUc);
	send(sock, &cipher, sizeof(cipher), 0);
	cout << "\nSent encrypted (N2) to client : " << cipher << endl;

	// step-4: send C = En(PUc, En(PRs, k))
	int k; // secret key
	cout << "\nEnter secret key (integer) to send : "; cin >> k;
	cipher = encrypt(encrypt(k,PRs), PUc);
	send(sock, &cipher, sizeof(cipher), 0);

	cout << "Sent encrypted secret key to client : " << cipher << endl << endl;
}
```

Client side:

```
# include <bits/stdc++.h>

# include <arpa/inet.h>



using namespace std;



int p, q, e, d, n, phi; // global variables

int PUc[2], PRc[2];



// client's keys

int PUs[2];

// server's public key

int sock;



void connectToServer(const char* ip, int port)

{

	sock = socket(AF_INET, SOCK_STREAM, 0);

	struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
```

```
	if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 )
	{
		cout << "\nRun server program first." << endl; exit(0);
	}
	else
	{
		cout << "\nClient is connected to Server." << endl;
	}
}

int gcd(int a, int b)
{
	return b==0 ? a : gcd(b, a%b);
}

void genKey()
{
	cout << "\nEnter two prime numbers (>100) : "; cin >> p >> q;
	n = p * q ;
	phi = (p-1) * (q-1);
	srand(time(NULL));
	do
	{
		e = rand()%(phi-2)+2;
	} while(gcd(e,phi) != 1);
	for(d=1; d<phi; d++)
	{
		if((d*e)%phi == 1)
			break;
```

```
        }

        PUc[0] = e; PUc[1] = n; // public key
        PRc[0] = d; PRc[1] = n; // private key
        cout << "\nPublic key , PUc = {" << e << ", " << n << "}" << endl;
        cout << "Private key, PRc = {" << d << ", " << n << "}" << endl;
}

void shareKey() // first receive then send
{
        recv(sock, &PUs, sizeof(PUs), 0); // receive public key from server
        send(sock, &PUc, sizeof(PUc), 0); // send client's public key to server
        cout << "Public key received from server, PUs = {" << PUs[0] << ", " << PUs[1] << "}" << endl;
        cout << "\nSent client's Public key to server." << endl;
}

// C = M^e mod n
int encrypt(int M, int P[2]) // P = {e or d, n}
{
        int C=1;
        for(int i=1; i<=P[0]; i++)
        {
                C = (C * M) % P[1];
        }
        return C;
}

int decrypt(int C, int P[2])
{
```

```
        return encrypt(C,P);
}

int main()
{
        char ip[50]; cout<<"\nEnter server's IP address: "; cin>>ip;
        int port;
        cout<<"Enter port : "; cin>>port;
        srand(time(NULL));
        connectToServer(ip, port);
        genKey();
        shareKey(); // share public keys

        // step-1: recv cipher from server and Dec(PRc, [N1||ID])
        int cipher;
        recv(sock, &cipher, sizeof(cipher), 0);
        cout << "\nReceived encrypted (N1||ID) from server : " << cipher << endl;
        int msg = decrypt(cipher, PRc);
        int N1 = msg/100;
        int ID = msg%100;
        cout << "Decrypted Server's ID, IDs = " << ID << endl;
        cout << "Decrypted Server's nonce, N1 = " << N1 << endl;

        // step-2: send En(PUs, (N1||N2)) to server
        int N2 = rand() % 100; // nonce
        cout << "\nNonce generated, N2 = " << N2 << endl;
        msg = N1*100 + N2;
        cipher = encrypt(msg, PUs);
        send(sock, &cipher, sizeof(cipher), 0);
```

```
        cout << "Sent encrypted (N1||N2) to server : " << cipher << endl;

        // step-3: recv enc(N2) from client and Dec(PRc, N2)
        recv(sock, &cipher, sizeof(cipher), 0);
        cout << "\nReceived encrypted (N2) from server : " << cipher << endl;
        int N2s = decrypt(cipher, PRc);
        cout << "Decrypted Client's Nonce, N2 = " << N2s << endl;
        if(N2s != N2)
        {
                cout << "\nNonce didn't match!\n";
                exit(-1);
        }
        else
        {
                cout << "----- Server Authenticated -----" << endl;
        }
        // step-4: recv cipher and perform k = Dec(PUs, Dec(PRc, C))
        int k;
        recv(sock, &cipher, sizeof(cipher), 0);
        cout << "\nReceived cipher from Server : " << cipher << endl;
        k = decrypt(decrypt(cipher, PRc), PUs);
        cout << "Decrypted Secret Key : " << k << endl << endl;
}
```

Output Screenshots: