# Siddaganga Institute of Technology, Tumkur-572103

Department of Computer Science and Engineering

**CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Student Name: ANVS Anudeep | | | USN: 1SI19CS017 | Batch No: A1 | | Date: 23/01/2023 | |
| **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. | Dr AS Poornima | | | | | |  |
| 2. | Ravi V | | | | | |  |
| **Question No: 11**  Implement RSA algorithm to process blocks of plaintext (refer Figure 9.7 of the text book), where plaintext is a string of characters and let the block size be two characters. (Note: assign a unique code to each plain text character i.e., a=00, A=26). The program should support the following.   1. Accept string of characters as plaintext. 2. Encryption takes plaintext and produces ciphertext characters 3. Decryption takes ciphertext characters obtained in step ii and produces corresponding plaintext characters. 4. Display the result after each step | | | | | | | |
| **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 1.     Fig 1 Fig 2.   1. 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. | | | | | | | |

**Client**

# include <bits/stdc++.h>  
# include <arpa/inet.h>  
using namespace std;  
  
int connectToServer(const char\* ip, int port)  
{  
    int 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;  
    }  
    return sock;  
}  
  
int randInRange(int low, int high)  
{  
    return rand()%(high-(low+1)) + (low+1) ;  
}  
  
int gcd(int a, int b)  
{  
    return b==0 ? a : gcd(b, a%b);  
}  
  
int powermod(int a, int b, int n)  
{  
    int res = 1;  
    for(int i=0; i<b; i++)  
    {  
        res = (res\*a) % n;  
    }  
    return res;  
}  
  
int decrypt(int C, int PR[2])  
{  
    return powermod(C, PR[0], PR[1]);  
}  
  
char toChar(int n)  
{  
    return (n >= 26) ? (n+'A'-26) : (n+'a');  
}

int main()

{  
    char ip[50];  
    int port;  
    cout << "Enter Server's IP address: "; cin >> ip;  
    cout << "Enter port : "; cin >> port;  
    int sock = connectToServer(ip, port);  
  
    int p,q;  
    cout << "\nEnter two large prime numbers(>100) : "; cin >> p >> q;  
    int n = p \* q ;  
    int phi = (p-1) \* (q-1);  
  
    srand(time(NULL));  
    int e, d;  
    do{ e = randInRange(1, phi); } while(gcd(e,phi) != 1);  
     
    for(d=1; d<phi; d++)  
    {  
        if((d\*e)%phi == 1) break;  
    }  
     
    int PU[2] = {e, n};  
    int PR[2] = {d, n};  
    cout << "\nPublic key , PU = {" << e << ", " << n << "}" << endl;  
    cout << "Private key, PR = {" << d << ", " << n << "}" << endl;  
  
    send(sock, &PU, sizeof(PU), 0);  
    cout << "\nSent Public key to server." << endl;  
  
    string msg = "";  
    while (true)  
    {  
        int C;  
        recv(sock, &C, sizeof(C), 0);  
        if(C == -1)   break;  
        cout << "\nCiphertext received from server : " << C << endl;  
  
        int M = decrypt(C,PR);  
        cout << "Decrypted Text : " << M << endl;  
        msg += toChar(M/100);  
        msg += toChar(M%100);  
    }  
    cout << "\nDecrypted message : " << msg << endl << endl;  
}

**Server**

# include <bits/stdc++.h>  
# include <arpa/inet.h>  
using namespace std;  
  
int createServer(int port)  
{  
    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);  
    int sock = accept(sersock, NULL, NULL);  
    cout << "Connection Established." << endl;  
  
    return sock;  
}  
  
int powermod(int a, int b, int n)  
{  
    int res = 1;  
    for(int i=0; i<b; i++)  
    {  
        res = (res\*a) % n;  
    }  
    return res;  
}  
  
  
int encrypt(int M, int PU[2])  
{  
    return powermod(M, PU[0], PU[1]);  
}  
  
int toInt(char c)  
{  
    return (c < 'a') ? (c-'A'+26) : (c-'a');  
}  
  
int main()  
{  
    int port;  
    cout << "Enter port : "; cin >> port;  
    int sock = createServer(port);  
  
    int PU[2];  
    recv(sock, &PU, sizeof(PU), 0);  
    cout << "\nPublic key received from client : {" << PU[0] << ", " << PU[1] << "}" << endl;

string msg;  
    cout << "\nEnter message to encrypt : "; cin >> msg;  
     
    if(msg.length()% 2 != 0) msg+="x";  
  
    for(int i=0; i<msg.length(); i+=2)  
    {  
        int M = toInt(msg[i])\*100 + toInt(msg[i+1]);  
        cout << "\nPlaintext block : " << M << endl;  
  
        int C = encrypt(M, PU);  
        cout << "Encrypted text  : " << C << endl;  
        send(sock, &C, sizeof(C), 0);  
    }  
    int stop = -1;  
    send(sock, &stop, sizeof(stop), 0);  
    cout << "\nSent ciphertext to client." << endl << endl;  
}



