**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: 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 :   1. Both client and server generate {PU, PR} and distribute PU to each other. 2. Establish a secret key K between client and server by exchanging the messages as shown in below figure.     **Algorithm:**   1. Both client and server generate {PU, PR} and distribute PU to each other.      1. Establish a secret key K between client and server by exchanging the messages as shown in below figure. | | | | | | |
|  | | | | | | |
| **Client**  # include <bits/stdc++.h> # include <arpa/inet.h> using namespace std;   int p, q, e, d, n, phi; int PUc[2], PRc[2];     int PUs[2];             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 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); }  void genKey() {     cout << "\nEnter two prime numbers (>100) : "; cin >> p >> q;     n = p \* q ;     phi = (p-1) \* (q-1);      srand(time(NULL));     do{ e = randInRange(1, phi); } while(gcd(e,phi) != 1);     for(d=1; d<phi; d++)     {         if((d\*e)%phi == 1) break;     }         PUc[0] = e; PUc[1] = n;     PRc[0] = d; PRc[1] = n;     cout << "\nPublic key , PUc = {" << e << ", " << n << "}" << endl;     cout <<   "Private key, PRc = {" << d << ", " << n << "}" << endl; }  void shareKey() {     recv(sock, &PUs, sizeof(PUs), 0);     send(sock, &PUc, sizeof(PUc), 0);     cout << "Public key received from server, PUs = {" << PUs[0] << ", " << PUs[1] << "}" << endl;     cout << "\nSent client's Public key to server." << endl; }  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 decrypt(int C, int PR[2]) {     return powermod(C, PR[0], PR[1]); }  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();      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;      int N2 = rand() % 100;     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;      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;}      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; }  **Server**  # include <bits/stdc++.h> # include <arpa/inet.h> using namespace std;  int p, q, e, d, n, phi; int PUs[2], PRs[2];     int PUc[2];           int sock;  void 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);     sock = accept(sersock, NULL, NULL);     cout << "Connection Established." << endl; }  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);  }  void genKey() {     cout << "\nEnter two prime numbers (>100): "; cin >> p >> q;     n = p \* q ;     phi = (p-1) \* (q-1);      srand(time(NULL));     do{ e = randInRange(1, phi); } while(gcd(e,phi) != 1);     for(d=1; d<phi; d++)     {         if((d\*e)%phi == 1) break;     }         PUs[0] = e; PUs[1] = n;     PRs[0] = d; PRs[1] = n;     cout << "\nPublic key , PUs = {" << e << ", " << n << "}" << endl;     cout <<   "Private key, PRs = {" << d << ", " << n << "}" << endl; }  void shareKey() {     send(sock, &PUs, sizeof(PUs), 0);     recv(sock, &PUc, sizeof(PUc), 0);     cout << "Sent Server's Public key to client." << endl;     cout << "\nPublic key received from client : {" << PUc[0] << ", " << PUc[1] << "}" << endl; }  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 decrypt(int C, int PR[2]) {     return powermod(C, PR[0], PR[1]); }  int main() {     int port;  cout<<"\nEnter port : ";  cin>>port;     srand(time(NULL));      createServer(port);     genKey();     shareKey();     int ID;  cout<<"\nEnter Server's ID number (<100): "; cin>>ID;     int N1 = rand()%100;     cout << "Nonce generated, N1 = " << N1 << endl;         int msg = N1\*100 + ID;     int cipher = encrypt(msg, PUc);     send(sock, &cipher, sizeof(cipher), 0);     cout << "Sent encrypted (N1||ID) to client : " << cipher << endl;      recv(sock, &cipher, sizeof(cipher), 0);     cout << "\nReceived encrypted (N1||N2) from client : " << cipher << endl;     msg = decrypt(cipher, PRs);     int N1c = msg/100;     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;}      cipher = encrypt(N2, PUc);     send(sock, &cipher, sizeof(cipher), 0);     cout << "\nSent encrypted (N2) to client : " << cipher << endl;      int k;     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; } | | | | | | |



