**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: 13**  Compute common secret key between client and server using Diffie-Hellman key exchange technique. Perform encryption and decryption of message using the shared secret key **(Use simple XOR operation to encrypt and decrypt the message.)** | | | | | | |
| **Algorithm:** | | | | | | |
| **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) ;  }   long powermod(long a, long b, long  q)  {  long res=1;  for(long i=0;i<b;i++)  {  res=(res\*a)%q;  }  return res;  }   int main()  {     char ip[50]; cout << "\nEnter server's IP address: "; cin >> ip;     int port;    cout << "Enter port : "; cin >> port;     int sock = connectToServer(ip, port);      long q, alpha;  cout<<"\nEnter a prime number, q : "; cin >> q;  cout<<"Enter primitive root of q, alpha : "; cin >> alpha;   srand(time(NULL));  long Xc = randInRange(1, q);  cout<< "\nClient's private key, Xc = " << Xc <<endl;   long Yc = powermod(alpha, Xc, q);  send(sock, &Yc, sizeof(Yc), 0);  cout<< "Client's public key,  Yc = " << Yc <<endl;  long Ys;  recv(sock, &Ys, sizeof(Ys), 0);  cout<< "\nServer's public key,  Ys = " << Ys <<endl;   long k = powermod(Ys,Xc,q);  cout<<"\nSecret Key, k = "<<k<<endl;   long cipher;  recv(sock, &cipher, sizeof(cipher), 0);  cout<<"\nMessage received from Server  : " << cipher << endl;   long decipher = cipher ^ k;  cout << "Decrpyted message : " << decipher << 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 randInRange(int low, int high)  {     return rand()%(high-(low+1)) + (low+1) ;  }   long powermod(long a, long b, long  q)  {  long res=1;  for(long i=0; i<b; i++)  {  res=(res\*a)%q;  }  return res;  }   int main() {     int port; cout << "\nEnter port : "; cin >> port;     int sock = createServer(port);   long q, alpha;   cout<<"\nEnter a prime number, q : "; cin >> q;  cout<<"Enter primitive root of q, alpha : "; cin >> alpha;   long Yc;  recv(sock, &Yc, sizeof(Yc), 0);  cout<< "\nClient's public key,  Yc = " << Yc <<endl;   srand(time(NULL));  long Xs = randInRange(1, q);  cout<< "\nServer's private key, Xs = " << Xs <<endl;   long Ys = powermod(alpha, Xs, q);  send(sock, &Ys, sizeof(Ys), 0);  cout<< "Server's public key,  Ys = " << Ys <<endl;   long k = powermod(Yc,Xs,q);  cout<<"\nSecret Key, k = "<<k<<endl;   long msg;  cout<<"\nEnter a message(number) to send : "; cin>>msg;   long cipher = msg ^ k;  send(sock, &cipher, sizeof(cipher), 0);  cout << "Encrypted msg sent to client: " << cipher << endl << endl;  } | | | | | | |



