# 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: 14**  Implement DSS algorithm for signing and verification of messages between two parties (obtain H(M) using simple XOR method of hash computation on M). | | | | | | | |
| **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; }  long mod(long a, long b) {     return a >= 0 ? (a%b) : b-(abs(a)%b) ; }  long powermod(long a, long b, long  c) {     long res=1;     for(int i=0; i<b; i++)     {         res = (res \* a) % c;     }     return res; }  long findInverse(long R , long D) {     int i = 0;     long N = D;     long p[100] = {0,1};     long q[100] = {0} ;         while(R!=0)     {         q[i] = D/R ;         long oldD = D ;         D = R ;         R = oldD%R ;         if(i>1)         {             p[i] = mod(p[i-2] - p[i-1]\*q[i-2], N) ;         }         i++ ;     }     if (i == 1) return 1;     else        return p[i] = mod(p[i-2] - p[i-1]\*q[i-2], N) ; }  long H(long M) {     return (M ^ 1234); }  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 p, q;     long r, s;     long g, y;     long M, hashval;     long w, v;     srand(time(NULL));      recv(sock, &p, sizeof(p), 0);     recv(sock, &q, sizeof(q), 0);     recv(sock, &g, sizeof(g), 0);     recv(sock, &y, sizeof(y), 0);     recv(sock, &M , sizeof(M), 0);     recv(sock, &r, sizeof(r), 0);     recv(sock, &s, sizeof(s), 0);      cout << "Received p =  " << p << endl;     cout << "Received q =  " << q << endl;     cout << "Received g =  " << g << endl;     cout << "Received y =  " << y << endl;     cout << "Received M'=  " << M << endl;     cout << "Received r' = " << r << endl;     cout << "Received s' = " << s << endl;      hashval = H(M) ;     cout << "\nH(M') = " << hashval << endl;      w = findInverse(s,q) % q;  cout << "w = " << w << endl;     long u1 = (hashval \* w) % q;     long u2 = (r \* w) % q;     v = ((powermod(g,u1,p)\*powermod(y,u2,p)) %p) %q;  cout<<"v = "<<v<<endl;     if(v == r) cout<<"\nDigital Signature Verified. " << endl << endl;     else   cout<<"\nDigital Signature is invalid !!!" << 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; }  long randInRange(long low, long high) {     return rand()%(high-(low+1)) + (low+1) ; }  long mod(long a, long b) { return a >= 0 ? (a%b) : b-(abs(a)%b) ; }  long powermod(long a, long b, long  c) {     long res=1;     for(int i=0; i<b; i++)     {         res = (res \* a) % c;     }     return res; }  long findInverse(long R , long D) {     int i = 0;     long N = D;     long p[100] = {0,1};     long q[100] = {0} ;      while(R!=0)     {         q[i] = D/R ;         long oldD = D ;         D = R ;         R = oldD%R ;         if(i>1)         {             p[i] = mod(p[i-2] - p[i-1]\*q[i-2], N) ;         }         i++ ;     }     if (i == 1) return 1;     else        return p[i] = mod(p[i-2] - p[i-1]\*q[i-2], N) ; }  long H(long M) { return (M ^ 1234); }  int main() {     int port;  cout << "\nEnter port : "; cin >> port;     int sock = createServer(port);      long p, q;     long r, s;     long k, x, y, g;     long M, hashval;     srand(time(NULL));      cout << "\nEnter a large prime number, p : ";   cin >> p;     cout << "Enter a prime number, q (p-1 divisible by q & q>2) : "; cin >> q;     if( (p-1)%q != 0 || q <3) { cout << "\nInvalid input\n"; exit(-1); }      cout<<"Enter message, M = "; cin >> M;      hashval = H(M);     cout << "\nH(M) = " << hashval << endl;      long h;     do{         h = randInRange(1, p-1);                 g = powermod(h,(p-1)/q, p);       } while(g<=1);     cout << "g    = " << g;      x = randInRange(1, q);  cout << "\nServer's Private key, x = " << x;     y = powermod(g, x, p);  cout << "\nServer's Public  key, y = " << y;     k = randInRange(1, q);  cout << "\nSecret key, k = " << k << endl;          r = powermod(g, k, p) % q;     s = (findInverse(k,q) \* (hashval + x\*r )) % q;     cout << "\nServer's Signature {r,s} = {" << r << ", " << s << "}" << endl;      send(sock, &p, sizeof(p), 0);     send(sock, &q, sizeof(q), 0);     send(sock, &g, sizeof(g), 0);     send(sock, &y, sizeof(y), 0);     send(sock, &M , sizeof(M), 0);     send(sock, &r, sizeof(r), 0);     send(sock, &s, sizeof(s), 0);      cout << "\nSent p, q, g, and public key to client.";     cout <<"\nSent message along with signature to client." << endl << endl; } | | | | | | | |