

16 Dec 2019

Cryptography and Network Security Lab

SIT, Tumkur

Author:



AKSHAT AGARWAL

1SI16CS010

<https://github.com/git-akshat/CNS-Lab>

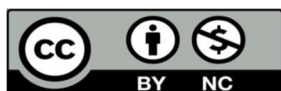
Contributor:



NATESH M BHAT

1SI16CS066

<https://github.com/nateshmbhat>



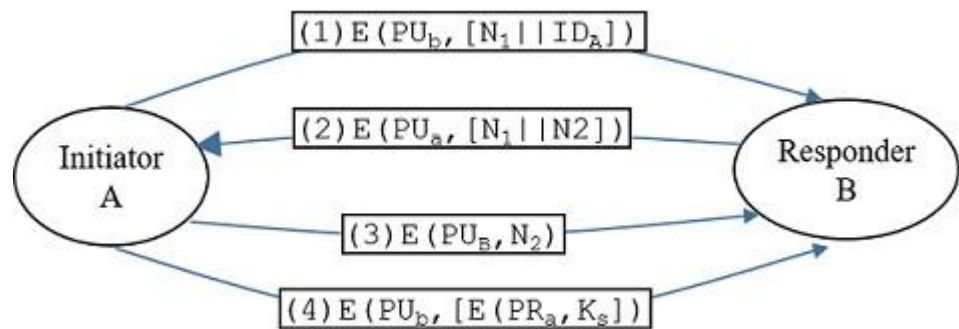
Cryptography and Network Security Lab

This repository contains programs implemented in Cryptography and network security Lab in my 7th semester of SIT(VTU).

1. Perform encryption and decryption using mono-alphabetic cipher. The program should support the following :
 - Construct an input file named plaintext.txt (consisting of 1000 alphabets, without any space or special characters)
 - Encrypt the characters of plaintext.txt and store the corresponding ciphertext characters in ciphertext.txt
 - Compute the frequency of occurrence of each alphabet in both plaintext.txt and ciphertext.txt and tabulate the results
2. Write a program to perform the following using Playfair cipher technique
 - Encrypt a given message M with different keys $\{k_1, k_2, \dots, k_n\}$. Print key and cipher text pair
 - Decrypt the cipher texts obtained in (i) to get back M
3. Write a program to perform the following using Hill cipher:
 - Encrypt a message M with a given key matrix of size 2X2 and 3X3
 - Decrypt the cipher text obtained in (i) by computing inverse of the respective key matrix.
4. Write a program to perform encryption and decryption using transposition technique with column permutation given as key.
5. Generate and print 48-bit keys for all sixteen rounds of DES algorithm, given a 64-bit initial key.
6. Given 64-bit output of (i-1)th round of DES, 48-bit ith round key K_i and E table, find the 48-bit input for S-box.
7. Given 48-bit input to S-box and permutation table P, find the 32-bit output R_i of ith round of DES algorithm.
8. Implement the following with respect to RC4:
 - Print first n key bytes generated by key generation process.
 - Illustrate encryption/decryption by accepting one byte data as input on the above generated keys.
9. Write a program to generate large random number using BBS random number generator algorithm and check whether the generated number is prime or not using RABIN-MILLER primality testing algorithm.
10. Implement RSA algorithm using client-server concept. The program should support the following :
 - Client generates $\{PU, PR\}$ and distributes PU to Server.
 - Sever encrypts message M using client's public key $\{PU\}$.
 - Client decrypts the message sent by server using its private key $\{PR\}$.

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.
 - Accept string of characters as plaintext.
 - Encryption takes plaintext and produces ciphertext characters.
 - Decryption takes ciphertext characters obtained in step ii and produces corresponding plaintext characters
 - Display the result after each step

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 :
 - Both client and server generate {PU, PR} and distribute PU to each other.
 - Establish a secret key K between client and server by exchanging the messages as shown in



below figure.

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.)

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).

```

1  /* Author : Natesh */
2  /*****
3  1. Perform encryption and decryption using mono-alphabetic cipher.The program
4     should support the following:
5  i.  Construct an input file named plaintext.txt (consisting of 1000 alphabets,
6     without any space or special characters)
7  ii. Compute key space (Permutation of set of all letters appeared in plaintext.txt:
8     there are n! permutations of a set of n elements)
9  iii. Encrypt the characters of plaintext.txt using any one key from (ii)
10     and store the corresponding ciphertext characters in ciphertext.txt
11  iv. Compute the frequency of occurrence of each alphabet in both plaintext.txt
12     and ciphertext.txt and tabulate the results as follows
13 *****/
14
15 #include<bits/stdc++.h>
16 using namespace std;
17
18 char uniqtext[26]; // Global variable
19
20 /* read plain text from plaintext.txt file */
21 string readPlainText()
22 {
23     ifstream fin;
24     string ptext;
25
26     fin.open("plaintext.txt");
27     fin >> ptext;
28     fin.close();
29
30     return ptext;
31 }
32
33 /* write cipher text to ciphertext.txt file */
34 void writecipherText(string ctext)
35 {
36     ofstream fout;
37     fout.open("ciphertext.txt");
38     fout << ctext;
39     fout.close();
40 }
41
42 /* function to find all possible permutation */
43 void permute(char a[], int l, int r, vector<string>& keyspace) // keyspace is passed
44 by reference
45 {
46     if(l == r)
47     {
48         keyspace.push_back(a);
49     }
50     else
51     {
52         for(int i = l; i <= r; i++)
53         {
54             swap(a[l], a[i]); //inbuilt swap function
55             permute(a, l+1, r, keyspace);
56             swap(a[l], a[i]);
57         }
58     }
59 }

```

```

60 vector<string> genKeySpace(string plaintext)
61 {
62     set<char> uniqSet;
63     vector<string> keyspace; // contains all possible permutation of letters in
plaintext
64     int count = 0;
65
66     /* store all the unique letters of plain text in uniqSet */
67     for(int i=0; i < plaintext.length(); i++)
68     {
69         uniqSet.insert(plaintext[i]);
70     }
71
72     /* copy uniqSet to uniqtext char array */
73     for(set<char>::iterator it = uniqSet.begin(); it != uniqSet.end(); it++)
74     {
75         uniqtext[count++] = (*it);
76     }
77
78     permute(uniqtext, 0, strlen(uniqtext)-1, keyspace);
79     return keyspace;
80 }
81
82 /* create cipher text using key */
83 string encrypt(string unique, string key)
84 {
85     string plaintext = readPlainText();
86     string ciphertext = "";
87
88     for(int i=0; i < plaintext.length(); i++)
89     {
90         int idx = unique.find(plaintext[i]);
91         ciphertext += key[idx];
92     }
93     return ciphertext;
94 }
95
96 /* frequency = (no of occurrence of a character / length of text) */
97 /* show frequency of all characters of plain text and cipher text */
98 void showFrequency(string pt , string ct)
99 {
100     map<char , int > mPlain ;
101     map<char , int > mCipher ;
102
103     for(int i =0 ;i < pt.length() ; i++)
104     {
105         mPlain[pt[i]]++ ;
106         mCipher[ct[i]]++ ;
107     }
108     cout<<"\nFrequency\t\tPlaintext Character\t\tCipher text character" <<endl;
109     cout<<"=====\t\t=====\t\t=====" <<endl;
110     for(int i=0 ; i<pt.length() ; i++)
111     {
112         cout<< (float)mPlain[pt[i]]/pt.length() << "\t\t\t\t" << pt[i] << "\t\t\t\t" <<
ct[i] << endl ;
113     }
114 }
115
116 int main()
117 {
118     srand(time(NULL)) ;

```

```

119
120 string plaintext = readPlainText() ;
121 cout<<"Plain text = \t" << plaintext << endl;
122
123 vector<string> keySpace = genKeySpace(plaintext);
124 string key = keySpace[rand()%keySpace.size()];
125
126 cout<<"Unique chars = \t" << uniqtext <<endl;
127 cout<<"Chosen key = \t" << key <<endl;
128
129 string ciphertext = encrypt(uniqtext , key) ;
130 writecipherText(ciphertext) ;
131 showFrequency(plaintext , ciphertext) ;
132 }
133
134
135 /***** OUTPUT-1 *****/
136 Plain text =      HelloWorld
137 Unique chars =   HWdelor
138 Chosen key =     WHoedr1
139
140 Frequency          Plaintext Character          Ciphertext character
141 =====
142 0.1                 H                          W
143 0.1                 e                          e
144 0.3                 l                          d
145 0.3                 l                          d
146 0.2                 o                          r
147 0.1                 W                          H
148 0.2                 o                          r
149 0.1                 r                          l
150 0.3                 l                          d
151 0.1                 d                          o
152
153 /***** OUTPUT-2 *****/
154 Plain text =      workisworship
155 Unique chars =   hikoprsw
156 Chosen key =     rphskowi
157
158 Frequency          Plaintext Character          Ciphertext character
159 =====
160 0.153846           w                          i
161 0.153846           o                          s
162 0.153846           r                          o
163 0.0769231         k                          h
164 0.153846           i                          p
165 0.153846           s                          w
166 0.153846           w                          i
167 0.153846           o                          s
168 0.153846           r                          o
169 0.153846           s                          w
170 0.0769231         h                          r
171 0.153846           i                          p
172 0.0769231         p                          s
173 *****/

```

```

1  /* Author : Gangadhar, Akshat */
2  /*****
3  2. Write a program to perform the following using Playfair cipher technique
4     - Encrypt a given message M with different keys {k1,k2,...,kn}.
5       Print key and cipher text pair.
6     - Decrypt the cipher texts obtained in (i) to get back M
7  *****/
8
9  #include <bits/stdc++.h>
10 using namespace std;
11
12 typedef struct{
13     int row;
14     int col;
15 }position;
16
17 char mat[5][5]; // Global Variable
18
19 void generateMatrix(string key)
20 {
21     /* flag keeps track of letters that are filled in matrix */
22     /* flag = 0 -> letter not already present in matrix */
23     /* flag = 1 -> letter already present in matrix */
24     int flag[26] = {0};
25     int x = 0, y = 0;
26
27     /* Add all characters present in the key */
28     for(int i=0; i<key.length(); i++)
29     {
30         if(key[i] == 'j') key[i] = 'i'; // replace j with i
31
32         if(flag[key[i]-'a'] == 0)
33         {
34             mat[x][y++] = key[i];
35             flag[key[i]-'a'] = 1;
36         }
37         if(y==5) x++, y=0;
38     }
39
40     /* Add remaining characters */
41     for(char ch = 'a'; ch <= 'z'; ch++)
42     {
43         if(ch == 'j') continue; // don't fill j since j was replaced by i
44
45         if(flag[ch - 'a'] == 0)
46         {
47             mat[x][y++] = ch;
48             flag[ch - 'a'] = 1 ;
49         }
50         if(y==5) x++, y=0;
51     }
52 }
53
54 /* function to add filler letter('x') */
55 string formatMessage(string msg)
56 {
57     for(int i=0; i<msg.length(); i++)
58     {
59         if(msg[i] == 'j') msg[i] = 'i';
60     }

```

```

61
62     for(int i=1; i<msg.length(); i+=2) //pairing two characters
63     {
64         if(msg[i-1] == msg[i]) msg.insert(i, "x");
65     }
66
67     if(msg.length()%2 != 0) msg += "x";
68     return msg;
69 }
70
71 /* Returns the position of the character */
72 position getPosition(char c)
73 {
74     for(int i=0; i<5; i++)
75         for(int j=0; j<5; j++)
76             if(c == mat[i][j])
77             {
78                 position p = {i, j};
79                 return p;
80             }
81 }
82
83 string encrypt(string message)
84 {
85     string ctext;
86     for(int i=0; i<message.length(); i+=2) // i is incremented by 2 inorder to
check for pair values
87     {
88         position p1 = getPosition(message[i]);
89         position p2 = getPosition(message[i+1]);
90         int x1 = p1.row; int y1 = p1.col;
91         int x2 = p2.row; int y2 = p2.col;
92
93         if( x1 == x2 ) // same row
94         {
95             ctext.append(1, mat[x1][(y1+1)%5]);
96             ctext.append(1, mat[x2][(y2+1)%5]);
97         }
98         else if( y1 == y2 ) // same column
99         {
100             ctext.append(1, mat[ (x1+1)%5 ][ y1 ]);
101             ctext.append(1, mat[ (x2+1)%5 ][ y2 ]);
102         }
103         else
104         {
105             ctext.append(1, mat[ x1 ][ y2 ]);
106             ctext.append(1, mat[ x2 ][ y1 ]);
107         }
108     }
109     return ctext;
110 }
111
112
113 string Decrypt(string message)
114 {
115     string ptext;
116     for(int i=0; i<message.length(); i+=2) // i is incremented by 2 inorder to check
for pair values
117     {
118         position p1 = getPosition(message[i]);
119         position p2 = getPosition(message[i+1]);

```



```

120     int x1 = p1.row; int y1 = p1.col;
121     int x2 = p2.row; int y2 = p2.col;
122
123     if( x1 == x2 ) // same row
124     {
125         ptext.append(1, mat[x1][ --y1<0 ? 4: y1 ]);
126         ptext.append(1, mat[x2][ --y2<0 ? 4: y2 ]);
127     }
128     else if( y1 == y2 ) // same column
129     {
130         ptext.append(1, mat[ --x1<0 ? 4: x1 ][y1]);
131         ptext.append(1, mat[ --x2<0 ? 4: x2 ][y2]);
132     }
133     else
134     {
135         ptext.append(1, mat[ x1 ][ y2 ]);
136         ptext.append(1, mat[ x2 ][ y1 ]);
137     }
138 }
139 return ptext;
140 }
141
142 int main()
143 {
144     string plaintext;
145     cout << "Enter message : "; cin >> plaintext;
146
147     int n; // number of keys
148     cout << "Enter number of keys : "; cin >> n;
149
150     string key[n];
151     for(int i=0; i<n; i++)
152     {
153         cout<< "\nEnter key " << i+1 << " : " << key[i];
154         cin >> key[i];
155
156         generateMatrix(key[i]);
157
158         cout << "Key " << i+1 << " Matrix:" << endl;
159         for(int k=0;k<5;k++)
160         {
161             for(int j=0;j<5;j++)
162             {
163                 cout << mat[k][j] << " ";
164             }
165             cout << endl;
166         }
167
168         cout << "Actual Message \t\t: " << plaintext << endl;
169
170         string fmsg = formatMessage(plaintext);
171         cout << "Formatted Message \t: " << fmsg << endl;
172
173         string ciphertext = encrypt(fmsg);
174         cout << "Encrypted Message \t: " << ciphertext << endl;
175
176         string decryptmsg = Decrypt(ciphertext);
177         cout<< "Decrypted Message \t: " << decryptmsg << endl;
178     }
179 }
180

```

```
181 /***** OUTPUT *****/
182
183 Enter message : balloon
184 Enter number of keys : 2
185
186 Enter key 1 : monarchy
187 Key 1 Matrix:
188 m o n a r
189 c h y b d
190 e f g i k
191 l p q s t
192 u v w x z
193 Actual Message      : balloon
194 Formatted Message   : balxloon
195 Encrypted Message   : ibsupmna
196 Decrypted Message   : balxloon
197
198 Enter key 2 : playfair
199 Key 2 Matrix:
200 p l a y f
201 i r b c d
202 e g h k m
203 n o q s t
204 u v w x z
205 Actual Message      : balloon
206 Formatted Message   : balxloon
207 Encrypted Message   : hbyrvvqo
208 Decrypted Message   : balxloon
209
210 *****/
211
```

```

1  /* Author : Natesh
2
3  3. Write a program to perform the following using Hill cipher:
4      - Encrypt a message M with a given key matrix of size 2X2 and 3X3
5      - Decrypt the cipher text obtained in (i) by computing inverse of
6      the respective key matrix.
7  */
8
9  #include<bits/stdc++.h>
10 using namespace std ;
11
12 int key[3][3] ; // Global
13
14 int mod26(int x)
15 {
16     return x >= 0 ? (x%26) : 26-(abs(x)%26) ;
17 }
18
19 /* findDet(matrix , order_of_matrix) */
20 int findDet(int m[3][3] , int n)
21 {
22     int det;
23     if(n == 2)
24     {
25         det = m[0][0] * m[1][1] - m[0][1]*m[1][0] ;
26     }
27     else if (n == 3)
28     {
29         det = m[0][0]*(m[1][1]*m[2][2] - m[1][2]*m[2][1]) - m[0][1]*(m[1][0]*m[2][2] -
m[2][0]*m[1][2] ) + m[0][2]*(m[1][0]*m[2][1] - m[1][1]*m[2][0]);
30     }
31     else det = 0 ; // invalid input
32     return mod26(det);
33 }
34
35 int findDetInverse(int R , int D = 26) // R is the remainder or determinant
36 {
37     int i = 0 ;
38     int p[100] = {0,1};
39     int q[100] = {0} ; // quotient
40
41     while(R!=0)
42     {
43         q[i] = D/R ;
44         int oldD = D ;
45         D = R ;
46         R = oldD%R ;
47         if(i>1)
48         {
49             p[i] = mod26(p[i-2] - p[i-1]*q[i-2]) ;
50         }
51         i++ ;
52     }
53     if (i == 1) return 1;
54     else return p[i] = mod26(p[i-2] - p[i-1]*q[i-2]) ;
55 }
56
57 void multiplyMatrices(int a[1000][3] , int a_rows , int a_cols , int b[1000][3] ,
int b_rows , int b_cols , int res[1000][3])
58 {

```

```

59 for(int i=0 ; i < a_rows ; i++)
60 {
61     for(int j=0 ; j < b_cols ; j++)
62     {
63         for(int k=0 ; k < b_rows ; k++)
64         {
65             res[i][j] += a[i][k]*b[k][j] ;
66         }
67         res[i][j] = mod26(res[i][j]) ;
68     }
69 }
70 }
71
72 /* Inverse = (matrix * detInverse) mod 26 */
73 /* findInverse(matrix , order_of_matrix , result_matrix) */
74 void findInverse(int m[3][3] , int n , int m_inverse[3][3] )
75 {
76     int adj[3][3] = {0};
77
78     int det = findDet(m , n); // findDet(matrix , order_of_matrix)
79     int detInverse = findDetInverse(det);
80
81     if(n==2)
82     {
83         adj[0][0] = m[1][1];
84         adj[1][1] = m[0][0];
85         adj[0][1] = -m[0][1];
86         adj[1][0] = -m[1][0];
87     }
88     else if(n==3)
89     {
90         int temp[5][5] = {0} ;
91         // fill the 5x5 matrix
92         for(int i=0; i<5; i++)
93         {
94             for(int j=0; j<5; j++)
95             {
96                 temp[i][j] = m[i%3][j%3] ;
97             }
98         }
99         /* except first row and first column, multiply elements along rows and place
them along columns */
100         for(int i=1; i<=3 ; i++)
101         {
102             for(int j=1; j<=3 ; j++)
103             {
104                 adj[j-1][i-1] = temp[i][j]*temp[i+1][j+1] - temp[i][j+1]*temp[i+1][j];
105             }
106         }
107     }
108
109     for(int i=0; i<n ; i++)
110     {
111         for(int j=0; j<n ; j++)
112         {
113             m_inverse[i][j] = mod26(adj[i][j] * detInverse) ;
114         }
115     }
116 }
117
118 // C = PK

```

```

119 string encrypt(string pt, int n)
120 {
121     int P[1000][3] = {0} ; // plaintext
122     int C[1000][3] = {0} ; // cipher text
123     int ptIter = 0 ;
124
125     while(pt.length()%n != 0)
126     {
127         pt += "x" ; // pad extra x
128     }
129     int row = (pt.length())/n; // number of rows in P
130
131     for(int i=0; i<row ; i++)
132     {
133         for(int j=0; j<n; j++)
134         {
135             P[i][j] = pt[ptIter++]-'a' ;
136         }
137     }
138
139     // multiplyMatrices(mat_a , row_a , col_a , mat_b, row_b, col_b , mat_result)
140     multiplyMatrices(P, row , n , key , n , n , C) ;
141
142     string ct = "" ;
143     for(int i=0 ; i<row ; i++)
144     {
145         for(int j=0 ; j<n ;j++)
146         {
147             ct += (C[i][j] + 'a');
148         }
149     }
150     return ct ;
151 }
152
153 // P = C*(k_inverse)
154 string decrypt(string ct, int n)
155 {
156     int P[1000][3] = {0} ; // plaintext
157     int C[1000][3] = {0} ; // cipher text
158     int ctIter = 0 ;
159
160     int row = ct.length()/n; // number of rows in C
161
162     for(int i=0; i<row ; i++)
163     {
164         for(int j=0; j<n; j++)
165         {
166             C[i][j] = ct[ctIter++]-'a' ;
167         }
168     }
169
170     int k_inverse[3][3] = {0};
171     /* findInverse(matrix , order_of_matrix , result_matrix) */
172     findInverse(key, n , k_inverse);
173
174     /* multiplyMatrices(mat_a , row_a , col_a , mat_b, row_b, col_b , mat_result) */
175     multiplyMatrices(C, row , n , k_inverse , n , n , P) ;
176
177     string pt = "" ;
178     for(int i = 0 ; i<row ; i++)
179     {

```

```

180     for(int j=0 ; j<n ; j++)
181     {
182         pt += (P[i][j] + 'a');
183     }
184 }
185 return pt ;
186 }
187
188 int main(void)
189 {
190     string pt ;
191     int n ;
192
193     cout << "Enter the text to be encrypted    : " ;
194     cin >> pt;
195
196     cout << "Enter order of key matrix : ";
197     cin >> n ;
198
199     cout<<"Enter key matrix: " <<endl;
200     for(int i=0; i<n; i++)
201     {
202         for(int j=0; j<n; j++)
203         {
204             cin >> key[i][j];
205         }
206     }
207
208     cout << "\nOriginal text : " << pt << endl;
209
210     string ct = encrypt(pt, n) ;
211     cout << "Encrypted text : " << ct << endl;
212
213     string dt = decrypt(ct, n);
214     cout << "Decrypted text : " << dt << endl;
215 }
216
217 /***** OUTPUT-1 *****/
218 Enter the text to be encrypted    : meetmenow
219 Enter order of key matrix : 2
220 Enter key matrix:
221 9   4
222 5   7
223
224 Original text : meetmenow
225 Encrypted text : yybtyyfubp
226 Decrypted text : meetmenowx
227 *****/
228
229 /***** OUTPUT-2 *****/
230 Enter the text to be encrypted    : paymoremoney
231 Enter order of key matrix : 3
232 Enter key matrix:
233 17  17  5
234 21  18  21
235 2   2   19
236
237 Original text : paymoremoney
238 Encrypted text : rrlmwbkaspdh
239 Decrypted text : paymoremoney
240 *****/

```

```
241
242 /***** OUTPUT-3 *****/
243 Enter the text to be encrypted : attackistonight
244 Enter order of key matrix : 3
245 Enter key matrix:
246 3 10 20
247 20 9 17
248 9 4 17
249
250 Original text : attackistonight
251 Encrypted text : fnwagwjgjdknrrq
252 Decrypted text : attackistonight
253 *****/
254
255 /***** OUTPUT-4 *****/
256 Enter the text to be encrypted : hillciphertechnique
257 Enter order of key matrix : 2
258 Enter key matrix:
259 3 3
260 2 5
261
262 Original text : hillciphertechnique
263 Encrypted text : ljdkwuhcutnzupdbksgx
264 Decrypted text : hillciphertechniquex
265 *****/
```

```

1  /* Author : AKSHAT AGARWAL
2
3  4. Write a program to perform encryption and decryption using
4     transposition technique with column permutation given as key.  */
5
6  #include<bits/stdc++.h>
7  using namespace std ;
8
9  string encrypt(string pt , string key)
10 {
11     string ct = ""; // ciphertext
12     int k = 0;      // plaintext iterator
13
14     int num_row = ceil((float) pt.length()/key.length());
15     int num_col = key.length();
16     char mat[num_row][num_col];
17
18     cout << "\nEncryption Matrix :" << endl;
19     cout << "-----" << endl;
20     for(int i=0; i<num_row ; i++)
21     {
22         for(int j=0; j<num_col; j++)
23         {
24             if(k < pt.length())
25             {
26                 cout << (mat[i][j] = pt[k++]) << " ";
27             }
28             else
29             {
30                 cout << (mat[i][j] = 'x') << " ";
31             }
32         }
33         cout << endl;
34     }
35     for(int i=0; i<num_col; i++)
36     {
37         for(int j=0; j<num_row; j++)
38         {
39             ct += mat[j][key.find(i+'1')];
40         }
41     }
42     return ct;
43 }
44
45 string decrypt(string ct , string key)
46 {
47     string pt = ""; // plaintext
48     int k = 0; // ciphertext iterator
49
50     int num_row = ceil((float)ct.length() / key.length());
51     int num_col = key.length();
52     char mat[num_row][num_col];
53
54     for(int i=0; i<num_col; i++)
55     {
56         for(int j=0; j<num_row; j++)
57         {
58             mat[j][key.find(i+'1')] = ct[k++];
59         }
60     }

```



```

61
62     cout << "\nDecryption Matrix :" << endl;
63     cout << "-----" << endl;
64     for(int i=0; i<num_row ; i++)
65     {
66         for(int j=0; j<num_col; j++)
67         {
68             cout << mat[i][j] << " ";
69             pt += mat[i][j];
70         }
71         cout << endl;
72     }
73     return pt;
74 }
75
76 int main()
77 {
78     string plaintext , key , ciphertext , decrypttext;
79
80     cout << "Enter text : ";
81     cin >> plaintext;
82
83     cout << "Enter key  : ";
84     cin >> key;
85
86     ciphertext = encrypt(plaintext , key);
87     cout << "\nEncrypted text \t: " << ciphertext << endl;
88
89     decrypttext = decrypt(ciphertext , key);
90     cout << "\nDecrypted text \t: " << decrypttext << endl;
91 }
92
93 /*
94 Enter text : transpositioncipher
95 Enter key  : 4231
96
97 Encryption Matrix :
98 -----
99 t r a n
100 s p o s
101 i t i o
102 n c i p
103 h e r x
104
105 Encrypted text  : nsopxrptceaoiirtsinh
106
107 Decryption Matrix :
108 -----
109 t r a n
110 s p o s
111 i t i o
112 n c i p
113 h e r x
114
115 Decrypted text  : transpositioncipherx
116 */

```

```

1  /* Author : Natesh
2
3  5. Generate and print 48-bit keys for all sixteen rounds of DES algorithm, given a
4  64-bit initial key. */
5  #include <bits/stdc++.h>
6  using namespace std;
7
8  int permChoiceOne[] = {
9      57, 49, 41, 33, 25, 17, 9 ,
10     1 , 58, 50, 42, 34, 26, 18,
11     10, 2 , 59, 51, 43, 35, 27,
12     19, 11, 3 , 60, 52, 44, 36,
13     63, 55, 47, 39, 31, 23, 15,
14     7 , 62, 54, 46, 38, 30, 22,
15     14, 6 , 61, 53, 45, 37, 29,
16     21, 13, 5 , 28, 20, 12, 4 };
17
18  int permChoiceTwo[] = {
19     14, 17, 11, 24, 1 , 5 , 3 , 28,
20     15, 6 , 21, 10, 23, 19, 12, 4 ,
21     26, 8 , 16, 7 , 27, 20, 13, 2 ,
22     41, 52, 31, 37, 47, 55, 30, 40,
23     51, 45, 33, 48, 44, 49, 39, 56,
24     34, 53, 46, 42, 50, 36, 29, 32 };
25
26  int leftShiftTable[] = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};
27
28  string rotateSubKey(string s , int rot) // rot is the number of left shift rotation
29  {
30      return s.substr(rot, s.length()-rot) + s.substr(0, rot) ;
31  }
32
33  string firstPermute(string input)
34  {
35      string res = "" ;
36      for(int i=0 ; i<56 ; i++)
37      {
38          res += input[permChoiceOne[i]-1];
39      }
40      return res ;
41  }
42
43  string secondPermute(string input)
44  {
45      string res = "" ;
46      for(int i=0 ; i<48 ; i++)
47      {
48          res += input[permChoiceTwo[i]-1];
49      }
50      return res ;
51  }
52
53  void genKeys(string left, string right)
54  {
55      ofstream fout ;
56      fout.open("keygen.txt"); //saving output to keygen.txt file
57
58      for (int i=0; i<16; i++)
59      {

```

```

60     left = rotateSubKey(left , leftShiftTable[i]);
61     right = rotateSubKey(right, leftShiftTable[i]);
62
63     string key = secondPermute(left+right);
64
65     cout << "key " << i+1 << " \t: " << key << endl; // display
66     fout << key << endl; // save to file
67 }
68 }
69
70 int main()
71 {
72     unsigned long long hexkey;
73     cout << "\nEnter 64-bit key in hexadecimal(16-digits) : " ;
74     cin >> hex >> hexkey; // to read hex input cin >> hex >> input
75
76     string key = bitset<64>(hexkey).to_string(); // to convert hex to binary string
77     cout << "Binary key (k) \t: " << key << endl;
78
79     key = firstPermute(key) ;
80     cout << "PC-1 key (k+) \t: " << key << endl;
81
82     cout << "\nSubKeys: " << endl;
83     genKeys(key.substr(0,28) , key.substr(28,28));
84
85     cout<<endl<<endl ;
86 }
87
88
89 /*****OUTPUT-1 *****/
90 Enter 64-bit key in hexadecimal(16-digits) : 1FE22472901BB2A3
91 Binary key (k)   : 0001111111000100010001000111001010010000000110111011001010100011
92 PC-1 key (k+)   : 1101001000001010110011100111110101100000101001000011001
93
94 SubKeys:
95 key 1   : 110010110110001010101100110001101000000110100011
96 key 2   : 001101011010010010101111001001011110000110011010
97 key 3   : 111100110000110010000010011001010001010001000011
98 key 4   : 011110001010101010110100110011101000000001101110
99 key 5   : 100101001011010000011110000001001101111111001100
100 key 6   : 011001100000011001110110000110001001010011110001
101 key 7   : 111011101101100000100100110010111100110000100001
102 key 8   : 100010101010001101111010000010100110111100011000
103 key 9   : 111000111100001011010111100101000001110000110011
104 key 10  : 001111011101001110000010100011110000101001110100
105 key 11  : 001100100001000111111011000100011110101111010000
106 key 12  : 101111010100000001010101001100011000010000010101
107 key 13  : 000001110100101110011100110010110010010010000110
108 key 14  : 000111100011000110110101001011000110001110001101
109 key 15  : 100111110000110001101001001100100101000011000111
110 key 16  : 010010011010100110011011011010100010100101010110
111 *****/
112
113
114 /***** OUTPUT-2 *****/
115 Enter 64-bit key in hexadecimal(16-digits) : 133457799BBCDF1
116 Binary key (k)   : 0001001100110100010101110111100110011011100110111111110001
117 PC-1 key (k+)   : 111100001100110010101010111101010101100110011110001111
118
119 SubKeys:
120 key 1   : 00011011000000101110111111111000111000001110010

```

```
121 key 2 : 011110011010111011011001110110111100100111100101
122 key 3 : 01010101111110010001010010000101100111110011001
123 key 4 : 011100101010110111010110110110011010100011101
124 key 5 : 011111001110110000000111111010110101001110101000
125 key 6 : 011000111010010100111110010100000111101100101111
126 key 7 : 111011001000010010110111111101100001100010111100
127 key 8 : 11110111100010100011101011000001001110111111011
128 key 9 : 11100000110110111110101111101101111001111000001
129 key 10 : 101100011111001101000111101110100100011001001111
130 key 11 : 001000010101111111010011110111101101001110000110
131 key 12 : 011101010111000111110101100101000110011111101001
132 key 13 : 100101111100010111010001111110101011101001000001
133 key 14 : 01011111010000111011011111100101110011100111010
134 key 15 : 101111111001000110001101001111010011111100001010
135 key 16 : 110010110011110110001011000011100001011111110101
136 *****
*/
```

```

1  /* Author : AKSHAT AGARWAL
2
3  6. Given 64-bit output of (i-1)th round of DES, 48-bit ith round key Ki and E table,
   find the 48-bit input for S-box. */
4
5  #include <bits/stdc++.h>
6  using namespace std;
7
8  int expPermute[] = {
9      32, 1 , 2 , 3 , 4 , 5 ,
10     4 , 5 , 6 , 7 , 8 , 9 ,
11     8 , 9 , 10, 11, 12, 13,
12     12, 13, 14, 15, 16, 17,
13     16, 17, 18, 19, 20, 21,
14     20, 21, 22, 23, 24, 25,
15     24, 25, 26, 27, 28, 29,
16     28, 29, 30, 31, 32, 1 };
17
18 string expansionPermute(string input)
19 {
20     string res = "";
21     for(int i=0; i<48; i++)
22     {
23         res += input[expPermute[i]-1];
24     }
25     return res;
26 }
27
28 string XOR(string input1, string input2)
29 {
30     string res = "";
31     for(int i=0; i<input1.length(); i++)
32     {
33         res += (input1[i] == input2[i]) ? "0" : "1";
34     }
35     return res;
36 }
37
38 int main()
39 {
40     int i; // round i
41     unsigned long long hexInput;
42     string Ki; // ith round key
43     ifstream fin;
44
45     cout << "\nEnter Round number (i) : ";
46     cin >> i;
47
48     cout << "Enter 64-bit (i-1)th round output in hex: " ;
49     cin >> hex >> hexInput;
50     string input = bitset<64>(hexInput).to_string();
51
52     fin.open("keygen.txt");
53     for(int j=1; j<=i; j++)
54     {
55         fin >> Ki;
56     }
57
58     // ---- To insert key manually uncomment below lines ---
59     // unsigned long long hexKey;

```

```

60 // cout << "Enter 48 bit key for ith round: " ;
61 // cin >> hex >> hexKey;
62 // Ki = bitset<48>(hexKey).to_string();
63
64 if(Ki.length() == 0)
65 {
66     cout << "\nkeygen.txt not found !!! \n" << endl;
67     exit(1);
68 }
69
70 cout << "\n64-bit Binary Input = " << input << endl ;
71 cout << "key for ith round (Ki) = " << Ki << endl ;
72
73 string Ri_1 = input.substr(32,32); // 32 bit Right half of input R[i-1]
74 cout << "\nRight half of 64-bit input, Ri_1 = " << Ri_1 << endl;
75
76 string R48 = expansionPermute(Ri_1);
77 cout << "Ri_1 after expansion permutation = " << R48 << endl;
78
79 string sBoxInput = XOR(R48, Ki);
80 cout << "\nInput to s-box : " << sBoxInput << endl << endl;
81 }
82
83 /***** Output-1 *****/
84 Enter Round number (i) : 1
85 Enter 64-bit (i-1)th round output in hex: cc00ccfff0aaf0aa
86
87 64-bit Binary Input =
88 110011000000000110011001111111111110000101010101111000010101010
89 key for ith round (Ki) = 00011011000000101110111111111000111000001110010
90 Right half of 64-bit input, Ri_1 = 11110000101010101111000010101010
91 Ri_1 after expansion permutation = 011110100001010101010101011110100001010101010101
92
93 Input to s-box : 011000010001011110111010100001100110010100100111
94 *****/

```

```

1  /* Author : AKSHAT AGARWAL
2
3  7. Given 48-bit input to S-box and permutation table P,
4     find the 32-bit output Ri of ith round of DES algorithm.    */
5
6  #include <bits/stdc++.h>
7  using namespace std;
8
9  unsigned int sBoxes[8][64] = {
10     {14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,
11     0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,
12     4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,
13     15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13},
14
15     {15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,
16     3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,
17     0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,
18     13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9},
19
20     {10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,
21     13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,
22     13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,
23     1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12},
24
25     {7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,
26     13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,
27     10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,
28     3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14},
29
30     {2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,
31     14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,
32     4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,
33     11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3},
34
35     {12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,
36     10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8,
37     9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,
38     4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13},
39
40     {4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,
41     13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,
42     1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,
43     6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12},
44
45     {13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,
46     1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,
47     7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,
48     2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11}
49 };
50
51 int permTable[] = {
52     16, 7 , 20, 21, 29, 12, 28, 17,
53     1 , 15, 23, 26, 5 , 18, 31, 10,
54     2 , 8 , 24, 14, 32, 27, 3 , 9 ,
55     19, 13, 30, 6 , 22, 11, 4 , 25 };
56
57 string substitution(string input)
58 {
59     string res = ""; // to store final s-box output
60     for(int i=0; i<8; i++)

```

```

61     {
62         string sInput = input.substr(6*i, 6) ;
63         int row = bitset<2>( sInput.substr(0,1) + sInput.substr(5,1) ).to_ulong() ;
64         int col = bitset<4>( sInput.substr(1,4) ).to_ulong() ;
65         res += bitset<4>(sBoxes[i][row*16+col]).to_string() ;
66
67         // To display individual s-box input and output un-comment this block
68         // string value = "";
69         // value = bitset<4>(sBoxes[i][row*16+col]).to_string() ;
70         // cout << "sbox-" << i+1 << " : \t" << sInput << "\t\t" << value << endl;
71     }
72     return res;
73 }
74
75 string permute(string input)
76 {
77     string res = "";
78     for(int i=0; i<32; i++)
79     {
80         res += input[permTable[i]-1];
81     }
82     return res;
83 }
84
85 string XOR(string input1, string input2)
86 {
87     string res = "";
88     for(int i=0; i<input1.length(); i++)
89     {
90         res += (input1[i] == input2[i]) ? "0" : "1";
91     }
92     return res;
93 }
94
95 int main()
96 {
97     unsigned long long hexSBoxInput, hexInput;
98
99     cout << "\nEnter 48-bit input for S-Box in hex(12-digits)      : " ;
100    cin >> hex >> hexSBoxInput;
101
102    cout << "Enter 64-bit (i-1)th round output in hex(16-digits) : " ;
103    cin >> hex >> hexInput;
104
105    string sBoxinput = bitset<48>(hexSBoxInput).to_string();
106    cout << "\nS-Box Input      : " << sBoxinput << endl;
107
108    string input = bitset<64>(hexInput).to_string();
109    cout << " Round(i-1) output : " << input << endl;
110
111    string Li_1 = input.substr(0,32);
112    cout << "\nLi_1          : " << Li_1 << endl;
113
114    string sBoxOutput = substitution(sBoxinput);
115    cout << "\nS-Box output    = " << sBoxOutput << endl;
116
117    string P = permute(sBoxOutput);
118    cout << "Permuted output = " << P << endl;
119
120    string Ri = XOR(P, Li_1); // P is permuted string and Li_1 is left half output
of (i-1)th round

```



```

121     cout << "\nOutput of ith round (Ri) = " << Ri << endl << endl;
122 }
123
124 /***** Output-1 *****/
125 Enter 48 bit input for S-Box in hex(12-digits)      : 6117ba866527
126 Enter 64-bit (i-1)th round output in hex(16-digits) : cc00ccfff0aaf0aa
127
128 S-Box Input      : 011000010001011110111010100001100110010100100111
129 Round(i-1) output : 11001100000000001100110011111111110000101010101111000010101010
130 Li_1             : 11001100000000001100110011111111
131
132 S-Box output     = 01011100100000101011010110010111
133 Permuted output = 0010001101001010101010100110111011
134
135 Output of ith round (Ri) = 11101111010010100110010101000100
136 *****/

```

```

1  /* Author : AKSHAT AGARWAL
2
3  8. Implement the following with respect to RC4:
4     - Print first n key bytes generated by key generation process.
5     - Illustrate encryption/decryption by accepting one byte data
6       as input on the above generated keys. */
7
8  #include <bits/stdc++.h>
9  using namespace std;
10
11 int main()
12 {
13     int S[256], T[256], keyStream[256];
14     string ptString, keyString, dtString = "";
15     int pt[256], ct[256], dt[256];
16
17     cout << "Enter message : "; cin >> ptString;
18     cout << "Enter key      : "; cin >> keyString;
19     int n = ptString.length();
20
21     cout << "\nPlain text \t: " ;
22     for(int i=0; i<n; i++)
23     {
24         pt[i] = ptString[i]; // converting char to their ASCII value
25         cout << pt[i] << " ";
26     }
27
28     // Initialization
29     for(int i=0; i<256; i++)
30     {
31         S[i] = i;
32         T[i] = (int)keyString[i%keyString.length()]; // fill T with ASCII value of
33                                                       // key T[256]=[keykeykeykey...]
34     }
35
36     // Initial permutation
37     int j=0;
38     for(int i=0; i<256; i++)
39     {
40         j = (j + S[i] + T[i]) % 256;
41         swap(S[i], S[j]);
42     }
43
44     // Stream Generation
45     cout << "\nKey Stream \t: ";
46     j=0;
47     for(int i=0; i<n; i++) // generate keystream of same length as plaintext
48     {
49         j = (j + S[i]) % 256;
50         swap(S[i], S[j]);
51         int t = (S[i] + S[j]) % 256;
52         keyStream[i] = S[t];
53         cout << keyStream[i] << " ";
54     }
55
56     // Encryption
57     cout << "\nCipher Text \t: ";
58     for(int i=0; i<n; i++)
59     {
60         ct[i] = pt[i] ^ keyStream[i]; // xor

```

```

61     cout << ct[i] << " ";
62 }
63
64 // Decryption
65 cout << "\nDecrypted text \t: " ;
66 for(int i=0; i<n; i++)
67 {
68     dt[i] = ct[i] ^ keyStream[i];
69     cout << dt[i] << " ";
70     dtString += (char)dt[i]; // change ASCII value to char
71 }
72 cout << "\nDecrypted text \t: " << dtString << endl;
73 }
74
75 /***** output-1 *****/
76 Enter message : Siddaganga
77 Enter key      : Institute
78
79 Plain text      : 83 105 100 100 97 103 97 110 103 97
80 Key Stream      : 236 34 53 234 27 158 64 219 222 102
81 Cipher Text     : 191 75 81 142 122 249 33 181 185 7
82 Decrypted text  : 83 105 100 100 97 103 97 110 103 97
83 Decrypted text  : Siddaganga
84 *****/
85
86 /***** output-2 *****/
87 Enter message : Washington
88 Enter key      : DC
89
90 Plain text      : 87 97 115 104 105 110 103 116 111 110
91 Key Stream      : 116 117 178 145 231 124 255 200 240 115
92 Cipher Text     : 35 20 193 249 142 18 152 188 159 29
93 Decrypted text  : 87 97 115 104 105 110 103 116 111 110
94 Decrypted text  : Washington
95 *****/

```

```

1  /* Author : AKSHAT AGARWAL
2
3  9. Write a program to generate large random number using BBS random number generator
   algorithm and check whether the generated number is prime or not using RABIN-MILLER
   primality testing algorithm.  */
4
5  #include <bits/stdc++.h>
6  using namespace std;
7
8  int randInRange(int low, int high) // excluding high and low
9  {
10     return rand()%(high-(low+1)) + (low+1) ;
11 }
12
13 int genPrime3mod4()
14 {
15     while(true)
16     {
17         int num = randInRange(10000,100000); // to generate large random number
18         if(num%4 != 3) continue;
19
20         bool prime = true;
21         for(int i=2; i<=sqrt(num); i++)
22         {
23             if(num % i == 0)
24             {
25                 prime = false;
26                 break;
27             }
28         }
29         if(prime) return num;
30     }
31 }
32
33 int bbs(int p, int q)
34 {
35     long long n = (long long)p*q ;
36
37     long long s; // non-zero and relatively prime to n
38     do{ s = rand(); } while(s%p==0 || s%q==0 || s==0);
39
40     int B = 0;
41     long long x = (s*s) % n;
42     for(int i=0; i<10; i++) // to generate 10 bit output
43     {
44         x = (x*x) % n;
45         B = B<<1 | (x & 1); // x%2 = x&1
46     }
47
48     cout<<"Blum Blum Shub"<<endl<<"-----"<<endl;
49     cout<<"p = " << p <<"\nq = " << q <<"\nn = " << n <<"\ns = " << s <<endl;
50     return B;
51 }
52
53 // (a pow b) % n
54 int powModN(int a, int b, int n)
55 {
56     int res=1;
57     for(int i=0; i<b; i++)
58     {

```

```

59     res = (res * a) % n;
60 }
61 return res;
62 }
63
64 string rabinMiller(int n)
65 {
66     int k = 0;           // power of 2
67     int q = n-1;        // q -> odd
68     while(q % 2 == 0) // until q becomes odd
69     {
70         q = q/2 ;
71         k++ ;
72     }
73
74     int a = randInRange(1, n-1);
75
76     cout << "\nRabin Miller(" << n << ")\n-----" << endl;
77     cout << n-1 << " = 2^" << k << " * " << q << endl;
78     cout << "k = " << k << "\nq = " << q << "\na = " << a << endl;
79
80     // if (a pow q)%n == 1
81     if(powModN(a,q,n) == 1) return "inconclusive";
82
83     for(int j=0; j<k ; j++)
84     {
85         if(powModN(a, pow(2,j)*q, n) == n-1) return "inconclusive";
86     }
87     return "composite";
88 }
89
90 int main()
91 {
92     srand(time(NULL));
93     int p = genPrime3mod4(); // large prime number (p%4=3)
94     int q = genPrime3mod4(); // large prime number (q%4=3)
95     int randNum = bbs(p, q);
96     cout << "Random number generated by BBS = " << randNum << endl;
97
98     // To check for multiple values of 'a' un-comment below line
99     // for(int i=0; i<4; i++)
100    cout << rabinMiller(randNum) << endl ;
101 }
102
103 /***** output-1 *****/
104 Blum Blum Shub
105 -----
106 p = 10223
107 q = 34543
108 n = 353133089
109 s = 22252
110 Random number generated by BBS = 443
111
112 Rabin Miller(443)
113 -----
114 442 = 2^1 * 221
115 k = 1
116 q = 221
117 a = 81
118 Inconclusive
119 *****/

```

```

1  /* Author : AKSHAT AGARWAL
2
3  10. Implement RSA algorithm using client-server concept. The program should support
   the following :
4     - Client generates {PU, PR} and distributes PU to Server.
5     - Sever encrypts message M using client's public key {PU}.
6     - Client decrypts the message sent by server using its private key {PR} */
7
8  /* Server Program */
9
10 # include <bits/stdc++.h>
11 # include <arpa/inet.h>
12 using namespace std;
13
14 int createServer(int port) // TCP connection
15 {
16     int sersock = socket(AF_INET, SOCK_STREAM, 0);
17     struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
18
19     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
20     cout << "\nServer Online. Waiting for client...." << endl;
21
22     listen(sersock, 5);
23     int sock = accept(sersock, NULL, NULL);
24     cout << "Connection Established." << endl;
25
26     return sock;
27 }
28
29 // C = M^e mod n
30 int encrypt(int M, int PU[2]) // PU = {e, n}
31 {
32     int C=1;
33     for(int i=1; i<=PU[0]; i++)
34     {
35         C = (C * M) % PU[1];
36     }
37     return C;
38 }
39
40 int main()
41 {
42     int port;
43     cout << "\nEnter port : "; cin >> port;
44     int sock = createServer(port);
45
46     int PU[2];
47     recv(sock, &PU, sizeof(PU), 0); // receive public key from client
48     cout << "\nPublic key received from client : {" << PU[0] << ", " << PU[1] << "}"
   << endl;
49
50     int M; // plaintext message (M < n)
51     cout << "\nEnter message(M< " << PU[1] << ") to encrypt : "; cin >> M;
52
53     int C = encrypt(M, PU);
54     cout << "\nEncrypted Text : " << C << endl;
55     send(sock, &C, sizeof(C), 0); // send ciphertext to client
56     cout << "\nSent ciphertext to client." << endl << endl;
57 }
58

```

```
59 /*
60 Enter port : 4444
61
62 Server Online. Waiting for client....
63 Connection Established.
64
65 Public key received from client : {3, 391}
66
67 Enter message(M<391) to encrypt : 231
68
69 Encrypted Text : 116
70
71 Sent ciphertext to client.
72 */
```

```

1  /* Author : AKSHAT AGARWAL
2
3  10. Implement RSA algorithm using client-server concept. The program should support
   the following :
4     - Client generates {PU, PR} and distributes PU to Server.
5     - Sever encrypts message M using client's public key {PU}.
6     - Client decrypts the message sent by server using its private key {PR} */
7
8  /* Client Program */
9
10 # include <bits/stdc++.h>
11 # include <arpa/inet.h>
12 using namespace std;
13
14 int connectToServer(const char* ip, int port)
15 {
16     int sock = socket(AF_INET, SOCK_STREAM, 0);
17     struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
18
19     if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 ){
20         cout << "\nRun server program first." << endl; exit(0);
21     }else{
22         cout << "\nClient is connected to Server." << endl;
23     }
24     return sock;
25 }
26
27 int gcd(int a, int b)
28 {
29     return b==0 ? a : gcd(b, a%b);
30 }
31
32 // M = C^d mod n
33 int decrypt(int C, int PR[2]) // PR = {d, n}
34 {
35     int M = 1;
36     for(int i=1; i<=PR[0]; i++)
37     {
38         M = (M*C) % PR[1];
39     }
40     return M;
41 }
42
43 int main()
44 {
45     char ip[50];
46     int port;
47     cout << "\nEnter server's IP address: "; cin >> ip;
48     cout << "Enter port : "; cin >> port;
49     int sock = connectToServer(ip, port);
50
51     int p,q;
52     cout << "\nEnter two prime numbers : "; cin >> p >> q;
53     int n = p * q ;
54     int phi = (p-1) * (q-1);
55
56     srand(time(NULL));
57     int e, d;
58     do{ e = rand()%(phi-2)+2; } while(gcd(e,phi) != 1);
59     for(d=1; d<phi; d++)

```



```

60 {
61     if((d*e)%phi == 1) break;
62 }
63
64 int PU[2] = {e, n}; // public key
65 int PR[2] = {d, n}; // private key
66 cout << "\nPublic key , PU = {" << e << ", " << n << "}" << endl;
67 cout << "Private key, PR = {" << d << ", " << n << "}" << endl;
68
69 send(sock, &PU, sizeof(PU), 0); // send public key to server
70 cout << "\nSent Public key to server." << endl;
71
72 int C; // ciphertext
73 recv(sock, &C, sizeof(C), 0); // receive ciphertext from server
74 cout << "\nCiphertext received from server : " << C << endl;
75
76 int M = decrypt(C, PR); // decrypted text
77 cout << "\nDecrypted Text : " << M << endl << endl;
78 }
79
80 /*
81 Enter server's IP address: 192.168.224.74
82 Enter port : 4444
83
84 Client is connected to Server.
85
86 Enter two prime numbers : 23 17
87
88 Public key , PU = {3, 391}
89 Private key, PR = {235, 391}
90
91 Sent Public key to server.
92
93 Ciphertext received from server : 116
94
95 Decrypted Text : 231
96 */

```

```

1  /* Author: Akshat Agarwal
2
3  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.
4     - Accept string of characters as plaintext.
5     - Encryption takes plaintext and produces ciphertext characters.
6     - Decryption takes ciphertext characters obtained in step ii and produces
   corresponding plaintext characters
7     - Display the result after each step */
8
9  /* Server Program */
10
11 #include <bits/stdc++.h>
12 #include <arpa/inet.h>
13 using namespace std;
14
15 int createServer(int port) // TCP connection
16 {
17     int sersock = socket(AF_INET, SOCK_STREAM, 0);
18     struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
19
20     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
21     cout << "\nServer Online. Waiting for client...." << endl;
22
23     listen(sersock, 5);
24     int sock = accept(sersock, NULL, NULL);
25     cout << "Connection Established." << endl;
26
27     return sock;
28 }
29
30 // C = M^e mod n
31 int encrypt(int M, int PU[2])
32 {
33     int C=1;
34     for(int i=1; i<=PU[0]; i++)
35     {
36         C = (C * M) % PU[1];
37     }
38     return C;
39 }
40
41 // a=00, b=01, ... A=26, B=27...
42 int toInt(char c)
43 {
44     return (c < 'a') ? (c-'A'+26) : (c-'a');
45 }
46
47 int main()
48 {
49     int port;
50     cout << "Enter port : "; cin >> port;
51     int sock = createServer(port);
52
53     int PU[2];
54     recv(sock, &PU, sizeof(PU), 0); // receive public key from client
55     cout << "\nPublic key received from client : {" << PU[0] << ", " << PU[1] << "}"
   << endl;

```

```

56
57 string msg; // plaintext message
58 cout << "\nEnter message to encrypt : "; cin >> msg;
59
60 if(msg.length()% 2 != 0) msg+="x";
61
62 for(int i=0; i<msg.length(); i+=2) // increment 2 for block
63 {
64     int M = toInt(msg[i])*100 + toInt(msg[i+1]); // block consist of two msg
character
65     cout << "\nPlaintext block : " << M << endl;
66
67     int C = encrypt(M, PU);
68     cout << "Encrypted text : " << C << endl;
69     send(sock, &C, sizeof(C), 0); // send ciphertext to client
70 }
71 int stop = -1; // at end send -1 to tell client to stop
72 send(sock, &stop, sizeof(stop), 0); //at end send -1 to client
73 cout << "\nSent ciphertext to client." << endl << endl;
74 }
75
76 /*
77 Enter port : 1234
78
79 Server Online. Waiting for client....
80 Connection Established.
81
82 Public key received from client : {1007, 13231}
83
84 Enter message to encrypt : cryptographylab
85
86 Plaintext block : 217
87 Encrypted text : 9189
88
89 Plaintext block : 2415
90 Encrypted text : 4027
91
92 Plaintext block : 1914
93 Encrypted text : 10957
94
95 Plaintext block : 617
96 Encrypted text : 534
97
98 Plaintext block : 15
99 Encrypted text : 2422
100
101 Plaintext block : 724
102 Encrypted text : 7387
103
104 Plaintext block : 1100
105 Encrypted text : 8051
106
107 Plaintext block : 123
108 Encrypted text : 9570
109
110 Sent ciphertext to client.
111 */

```

```

1  /* Author: Akshat Agarwal
2
3  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.
4     - Accept string of characters as plaintext.
5     - Encryption takes plaintext and produces ciphertext characters.
6     - Decryption takes ciphertext characters obtained in step ii and produces
   corresponding plaintext characters
7     - Display the result after each step */
8
9  /* Client Program */
10
11 #include <bits/stdc++.h>
12 #include <arpa/inet.h>
13 using namespace std;
14
15
16 int connectToServer(const char* ip, int port)
17 {
18     int sock = socket(AF_INET, SOCK_STREAM, 0);
19     struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
20
21     if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 ){
22         cout << "\nRun server program first." << endl; exit(0);
23     }else{
24         cout << "\nClient is connected to Server." << endl;
25     }
26     return sock;
27 }
28
29 int gcd(int a, int b)
30 {
31     return b==0 ? a : gcd(b, a%b);
32 }
33
34 // M = C^d mod n
35 int decrypt(int C, int PR[2])
36 {
37     int M = 1;
38     for(int i=1; i<=PR[0]; i++)
39     {
40         M = (M*C) % PR[1];
41     }
42     return M;
43 }
44
45 // a=00, b=01, ... A=26, B=27...
46 char toChar(int n)
47 {
48     return (n >= 26) ? (n+'A'-26) : (n+'a');
49 }
50
51 int main()
52 {
53     char ip[50];
54     int port;
55     cout << "Enter Server's IP address: "; cin >> ip;
56     cout << "Enter port : "; cin >> port;

```

```

57 int sock = connectToServer(ip, port);
58
59 int p,q;
60 cout << "\nEnter two large prime numbers(>100) : "; cin >> p >> q;
61 int n = p * q ; // should be greater than 5151 (since ZZ=5151)
62 int phi = (p-1) * (q-1);
63
64 srand(time(NULL));
65 int e, d;
66 do{ e = rand()%(phi-2)+2; } while(gcd(e,phi) != 1);
67
68 for(d=1; d<phi; d++)
69 {
70     if((d*e)%phi == 1) break;
71 }
72
73 int PU[2] = {e, n}; // public key
74 int PR[2] = {d, n}; // private key
75 cout << "\nPublic key , PU = {" << e << ", " << n << "}" << endl;
76 cout << "Private key, PR = {" << d << ", " << n << "}" << endl;
77
78 send(sock, &PU, sizeof(PU), 0); // send public key to server
79 cout << "\nSent Public key to server." << endl;
80
81 string msg = "";
82 while (true)
83 {
84     int C; // ciphertext
85     recv(sock, &C, sizeof(C), 0);
86     if(C == -1) break; // at the end -1 will be received
87     cout << "\nCiphertext received from server : " << C << endl;
88
89     int M = decrypt(C,PR);
90     cout << "Decrypted Text : " << M << endl;
91     msg += toChar(M/100); // first char in block
92     msg += toChar(M%100); // second char in block
93 }
94 cout << "\nDecrypted message : " << msg << endl << endl;
95 }
96
97 /*
98 Enter Server's IP address: 192.168.224.74
99 Enter port : 1234
100
101 Client is connected to Server.
102
103 Enter two large prime numbers : 101 131
104
105 Public key , PU = {1007, 13231}
106 Private key, PR = {2143, 13231}
107
108 Sent Public key to server.
109
110 Ciphertext received from server : 9189
111 Decrypted Text : 217
112
113 Ciphertext received from server : 4027
114 Decrypted Text : 2415
115
116 Ciphertext received from server : 10957
117 Decrypted Text : 1914

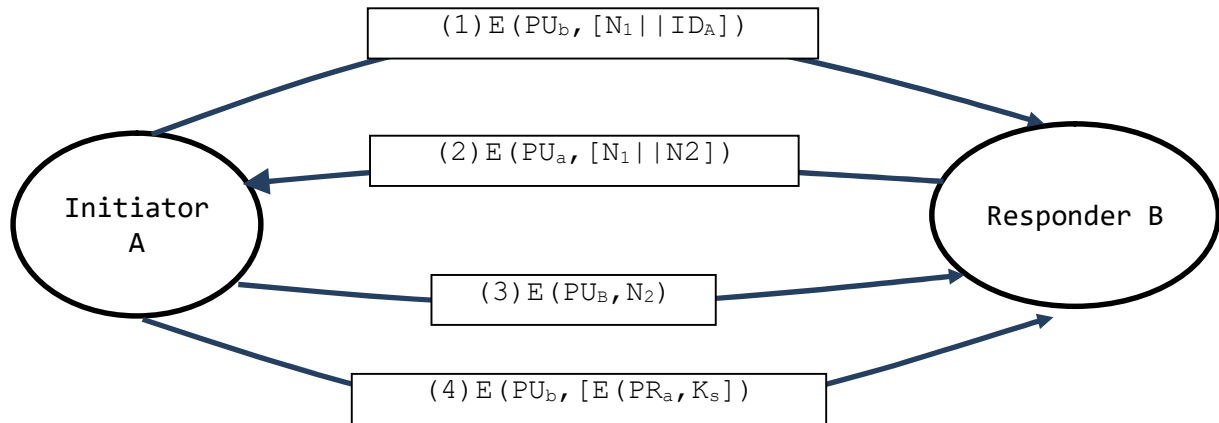
```

```
118
119 Ciphertext received from server : 534
120 Decrypted Text : 617
121
122 Ciphertext received from server : 2422
123 Decrypted Text : 15
124
125 Ciphertext received from server : 7387
126 Decrypted Text : 724
127
128 Ciphertext received from server : 8051
129 Decrypted Text : 1100
130
131 Ciphertext received from server : 9570
132 Decrypted Text : 123
133
134 Decrypted message : cryptographylabx
135 */
136
137 /* Note: give p and q values such that p*q > 5151 (since ZZ=5151) */
```

```

1 /* Author : AKSHAT AGARWAL
2
3 12. Implement RSA algorithm using client-server concept. Using this illustrate
4 secret key distribution scenario with confidentiality and authentication. The
5 program should support the following:
6 - Both client and server generate {PU, PR} and distribute PU to each other.
7 - Establish a secret key K between client and server by exchanging the messages
8 as shown in figure. */

```



```

7
8 /* Server Program */
9
10 # include <bits/stdc++.h>
11 # include <arpa/inet.h>
12 using namespace std;
13
14 int p, q, e, d, n, phi; // global variables
15 int PUs[2], PRs[2];    // server's keys
16 int PUC[2];           // client's public key
17 int sock;
18
19 void createServer(int port) // TCP connection
20 {
21     int sersock = socket(AF_INET, SOCK_STREAM, 0);
22     struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
23
24     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
25     cout << "\nServer Online. Waiting for client..." << endl;
26
27     listen(sersock, 5);
28     sock = accept(sersock, NULL, NULL);
29     cout << "Connection Established." << endl;
30 }
31
32 int gcd(int a, int b)
33 {
34     return b==0 ? a : gcd(b, a%b);
35 }
36
37 void genKey()
38 {
39     cout << "\nEnter two prime numbers (>100): "; cin >> p >> q;
40     n = p * q;
41     phi = (p-1) * (q-1);
42

```

```

43  srand(time(NULL));
44  do{ e = rand()%(phi-2)+2; } while(gcd(e,phi) != 1);
45  for(d=1; d<phi; d++)
46  {
47      if((d*e)%phi == 1) break;
48  }
49
50  PUs[0] = e; PUs[1] = n; // public key
51  PRs[0] = d; PRs[1] = n; // private key
52  cout << "\nPublic key , PUs = {" << e << " , " << n << "}" << endl;
53  cout << "Private key, PRs = {" << d << " , " << n << "}" << endl;
54 }
55
56 void shareKey() // first send then receive
57 {
58     send(sock, &PUs, sizeof(PUs), 0); // send Server's public key to client
59     recv(sock, &PUc, sizeof(PUc), 0); // receive public key from client
60     cout << "Sent Server's Public key to client." << endl;
61     cout << "\nPublic key received from client : {" << PUc[0] << " , " << PUc[1] <<
62     "}" << endl;
63 }
64 // C = M^e mod n
65 int encrypt(int M, int P[2]) // P = {e or d, n}
66 {
67     int C=1;
68     for(int i=1; i<=P[0]; i++)
69     {
70         C = (C * M) % P[1];
71     }
72     return C;
73 }
74
75 int decrypt(int C, int P[2])
76 {
77     return encrypt(C,P);
78 }
79
80 int main()
81 {
82     int port; cout<<"\nEnter port : "; cin>>port;
83     srand(time(NULL));
84
85     createServer(port);
86     genKey();
87     shareKey(); // share public keys
88
89     int ID; cout<<"\nEnter Server's ID number (<100): "; cin>>ID;
90     int N1 = rand()%100; // nonce
91     cout << "Nonce generated, N1 = " << N1 << endl;
92
93     // step-1: send En(PUc, [N1||ID]) to client
94     int msg = N1*100 + ID; // append ID to nonce
95     int cipher = encrypt(msg, PUc);
96     send(sock, &cipher, sizeof(cipher), 0);
97     cout << "Sent encrypted (N1||ID) to client : " << cipher << endl;
98
99     // step-2: recv cipher from client and Dec(PRs, (N1||N2))
100    recv(sock, &cipher, sizeof(cipher), 0);
101    cout << "\nReceived encrypted (N1||N2) from client : " << cipher << endl;
102    msg = decrypt(cipher, PRs);

```



```

103 int N1c = msg/100; // N1 received from client
104 int N2 = msg%100;
105 cout << "Decrypted Server's Nonce, N1 = " << N1c << endl;
106 cout << "Decrypted Client's Nonce, N2 = " << N2 << endl;
107 if(N1 != N1c) {cout << "\nNonce didn't match!\n"; exit(-1);}
108 else {cout << "----- Client Authenticated -----" << endl;}
109
110 // step-3: send En(PUc, N2) to client
111 cipher = encrypt(N2, PUc);
112 send(sock, &cipher, sizeof(cipher), 0);
113 cout << "\nSent encrypted (N2) to client : " << cipher << endl;
114
115 // step-4: send C = En(PUc, En(PRs, k))
116 int k; // secret key
117 cout << "\nEnter secret key (integer) to send : "; cin >> k;
118 cipher = encrypt(encrypt(k,PRs), PUc);
119 send(sock, &cipher, sizeof(cipher), 0);
120 cout << "Sent encrypted secret key to client : " << cipher << endl << endl;
121 }
122
123 /*
124 Enter port : 8888
125
126 Server Online. Waiting for client....
127 Connection Established.
128
129 Enter two prime numbers (>100): 101 131
130
131 Public key , PUs = {4551, 13231}
132 Private key, PRs = {1951, 13231}
133 Sent Server's Public key to client.
134
135 Public key received from client : {7477, 18281}
136
137 Enter Server's ID number (<100): 29
138 Nonce generated, N1 = 50
139 Sent encrypted (N1||ID) to client : 9531
140
141 Received encrypted (N1||N2) from client : 7905
142 Decrypted Server's Nonce, N1 = 50
143 Decrypted Client's Nonce, N2 = 23
144 ----Authenticated client----
145
146 Sent encrypted (N2) to client : 15328
147
148 Enter secret key (integer) to send : 454
149 Sent encrypted secret key to client : 15001
150 */

```

```

1  /* Author : AKSHAT AGARWAL
2
3  /* 12. Client Program */
4
5  # include <bits/stdc++.h>
6  # include <arpa/inet.h>
7  using namespace std;
8
9  int p, q, e, d, n, phi; // global variables
10 int PUC[2], PRC[2];    // client's keys
11 int PUs[2];           // server's public key
12 int sock;
13
14 void connectToServer(const char* ip, int port)
15 {
16     sock = socket(AF_INET, SOCK_STREAM, 0);
17     struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
18
19     if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 ){
20         cout << "\nRun server program first." << endl; exit(0);
21     }else{
22         cout << "\nClient is connected to Server." << endl;
23     }
24 }
25
26 int gcd(int a, int b)
27 {
28     return b==0 ? a : gcd(b, a%b);
29 }
30
31 void genKey()
32 {
33     cout << "\nEnter two prime numbers (>100) : "; cin >> p >> q;
34     n = p * q ;
35     phi = (p-1) * (q-1);
36
37     srand(time(NULL));
38     do{ e = rand()%(phi-2)+2; } while(gcd(e,phi) != 1);
39     for(d=1; d<phi; d++)
40     {
41         if((d*e)%phi == 1) break;
42     }
43
44     PUC[0] = e; PUC[1] = n; // public key
45     PRC[0] = d; PRC[1] = n; // private key
46     cout << "\nPublic key , PUC = {" << e << ", " << n << "}" << endl;
47     cout << "Private key, PRC = {" << d << ", " << n << "}" << endl;
48 }
49
50 void shareKey() // first receive then send
51 {
52     recv(sock, &PUs, sizeof(PUs), 0); // receive public key from server
53     send(sock, &PUC, sizeof(PUC), 0); // send client's public key to server
54     cout << "Public key received from server, PUs = {" << PUs[0] << ", " << PUs[1]
55 << "}" << endl;
56     cout << "\nSent client's Public key to server." << endl;
57 }
58 // C = M^e mod n
59 int encrypt(int M, int P[2]) // P = {e or d, n}

```

```

60 {
61     int C=1;
62     for(int i=1; i<=P[0]; i++)
63     {
64         C = (C * M) % P[1];
65     }
66     return C;
67 }
68
69 int decrypt(int C, int P[2])
70 {
71     return encrypt(C,P);
72 }
73
74 int main()
75 {
76     char ip[50]; cout<<"\nEnter server's IP address: "; cin>>ip;
77     int port;    cout<<"Enter port : "; cin>>port;
78     srand(time(NULL));
79
80     connectToServer(ip, port);
81     genKey();
82     shareKey(); // share public keys
83
84     // step-1: recv cipher from server and Dec(Prc, [N1||ID])
85     int cipher;
86     recv(sock, &cipher, sizeof(cipher), 0);
87     cout << "\nReceived encrypted (N1||ID) from server : " << cipher << endl;
88     int msg = decrypt(cipher, PRc);
89     int N1 = msg/100;
90     int ID = msg%100;
91     cout << "Decrypted Server's ID,   IDs = " << ID << endl;
92     cout << "Decrypted Server's nonce, N1 = " << N1 << endl;
93
94     // step-2: send En(PUs, (N1||N2)) to server
95     int N2 = rand() % 100; // nonce
96     cout << "\nNonce generated, N2 = " << N2 << endl;
97     msg = N1*100 + N2;
98     cipher = encrypt(msg, PUs);
99     send(sock, &cipher, sizeof(cipher), 0);
100    cout << "Sent encrypted (N1||N2) to server : " << cipher << endl;
101
102    // step-3: recv enc(N2) from client and Dec(Prc, N2)
103    recv(sock, &cipher, sizeof(cipher), 0);
104    cout << "\nReceived encrypted (N2) from server : " << cipher << endl;
105    int N2s = decrypt(cipher, PRc);
106    cout << "Decrypted Client's Nonce, N2 = " << N2s << endl;
107    if(N2s != N2) {cout << "\nNonce didn't match!\n"; exit(-1);}
108    else {cout << "----- Server Authenticated -----" << endl;}
109
110    // step-4: recv cipher and perform k = Dec(PUs, Dec(Prc, C))
111    int k;
112    recv(sock, &cipher, sizeof(cipher), 0);
113    cout << "\nReceived cipher from Server : " << cipher << endl;
114    k = decrypt(decrypt(cipher, PRc), PUs);
115    cout << "Decrypted Secret Key : " << k << endl << endl;
116 }
117
118 /*
119 Enter server's IP address: 127.0.0.1
120 Enter port : 8888

```

```
121
122 Client is connected to Server.
123
124 Enter two prime numbers (>100) : 101 181
125
126 Public key , PUC = {7477, 18281}
127 Private key, PRc = {14413, 18281}
128 Public key received from server, PUs = {4551, 13231}
129
130 Sent client's Public key to server.
131
132 Received encrypted (N1||ID) from server : 9531
133 Decrypted Server's ID, IDs = 29
134 Decrypted Server's nonce, N1 = 50
135
136 Nonce generated, N2 = 23
137 Sent encrypted (N1||N2) to server : 7905
138
139 Received encrypted (N2) from server : 15328
140 Decrypted Client's Nonce, N2 = 23
141 -----Authenticated Server-----
142
143 Received cipher from Server : 15001
144 Decrypted Secret Key : 454
145 */
```

```

1  /* Author : AKSHAT AGARWAL
2
3  13. Compute common secret key between client and server using Diffie-Hellman key
4  exchange technique. Perform encryption and decryption of message using the shared
5  secret key (Use simple XOR operation to encrypt and decrypt the message.) */
6
7  /* Server Program */
8
9  # include <bits/stdc++.h>
10 # include <arpa/inet.h>
11 using namespace std;
12
13 int createServer(int port) // TCP connection
14 {
15     int sersock = socket(AF_INET, SOCK_STREAM, 0);
16     struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
17
18     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
19     cout << "\nServer Online. Waiting for client...." << endl;
20
21     listen(sersock, 5);
22     int sock = accept(sersock, NULL, NULL);
23     cout << "Connection Established." << endl;
24     return sock;
25 }
26
27 long powermod(long a, long b, long q)
28 {
29     long res=1;
30     for(long i=0; i<b; i++)
31     {
32         res=(res*a)%q;
33     }
34     return res;
35 }
36
37 int main()
38 {
39     int port; cout << "\nEnter port : "; cin >> port;
40     int sock = createServer(port);
41
42     long q, alpha;
43     cout<<"\nEnter a prime number, q : "; cin >> q;
44     cout<<"Enter primitive root of q, alpha : "; cin >> alpha;
45
46     long Yc;
47     recv(sock, &Yc, sizeof(Yc), 0); // recv client's public key
48     cout<< "\nClient's public key, Yc = " << Yc <<endl;
49
50     srand(time(NULL));
51     long Xs = rand()%(q-2)+2; // server's private key (1<Xs<q)
52     cout<< "\nServer's private key, Xs = " << Xs <<endl;
53
54     long Ys = powermod(alpha, Xs, q); // server's public key
55     send(sock, &Ys, sizeof(Ys), 0); // send server's public key
56     cout<< "Server's public key, Ys = " << Ys <<endl;
57
58     long k = powermod(Yc,Xs,q);
59     cout<<"\nSecret Key, k = "<<k<<endl;
60 }

```

```

59 long msg;
60 cout<<"\nEnter a message(number) to send : "; cin>>msg;
61
62 long cipher=msg^k;
63 send(sock, &cipher, sizeof(cipher), 0);
64 cout << "Encrypted msg sent to client: " << cipher << endl << endl;
65 }
66
67 /*
68 some values for q and alpha
69 q=11, alpha=2
70 q=71, alpha=7
71 */
72
73 /*
74 Enter port : 4444
75
76 Server Online. Waiting for client....
77 Connection Established.
78
79 Enter a prime number, q : 11
80 Enter primitive root of q, alpha : 2
81
82 Client's public key, Yc = 5
83
84 Server's private key, Xs = 10
85 Server's public key, Ys = 1
86
87 Secret Key, k = 1
88
89 Enter a message(number) to send : 453
90 Encrypted msg sent to client: 452
91 */

```

```

1  /* Author : AKSHAT AGARWAL
2
3  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.) */
4
5  /* Client Program */
6
7  # include <bits/stdc++.h>
8  # include <arpa/inet.h>
9  using namespace std;
10
11 int connectToServer(const char* ip, int port)
12 {
13     int sock = socket(AF_INET, SOCK_STREAM, 0);
14     struct sockaddr_in addr = {AF_INET, htons(port),inet_addr(ip)};
15
16     if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 ){
17         cout << "\nRun server program first." << endl; exit(0);
18     }else{
19         cout << "\nClient is connected to Server." << endl;
20     }
21     return sock;
22 }
23
24 long powermod(long a, long b, long q)
25 {
26     long res=1;
27     for(long i=0;i<b;i++)
28     {
29         res=(res*a)%q;
30     }
31     return res;
32 }
33
34 int main()
35 {
36     char ip[50]; cout << "\nEnter server's IP address: "; cin >> ip;
37     int port;    cout << "Enter port : "; cin >> port;
38     int sock = connectToServer(ip, port);
39
40     long q, alpha;
41     cout<<"\nEnter a prime number, q : "; cin >> q;
42     cout<<"Enter primitive root of q, alpha : "; cin >> alpha;
43
44     srand(time(NULL));
45     long Xc = rand()%(q-2)+2; // client's private key (1<Xc<q)
46     cout<< "\nClient's private key, Xc = " << Xc <<endl;
47
48     long Yc = powermod(alpha, Xc, q); // client's public key
49     send(sock, &Yc, sizeof(Yc), 0); // send client's public key
50     cout<< "Client's public key, Yc = " << Yc <<endl;
51
52     long Ys;
53     recv(sock, &Ys, sizeof(Ys), 0); // recv server's public key
54     cout<< "\nServer's public key, Ys = " << Ys <<endl;
55
56     long k = powermod(Ys,Xc,q);
57     cout<<"\nSecret Key, k = "<<k<<endl;
58

```

```

59 long cipher;
60 recv(sock, &cipher, sizeof(cipher), 0);
61 cout<<"\nMessage received from Server : " << cipher << endl;
62
63 long decipher = cipher ^ k;
64 cout << "Decrypted message : " << decipher << endl << endl;
65 }
66
67 /*
68 some values for q and alpha
69 q=11, alpha=2
70 q=71, alpha=7
71 Note: input q and alpha value same as server's
72 */
73
74 /*
75 Enter server's IP address: 127.0.0.1
76 Enter port : 4444
77
78 Client is connected to Server.
79
80 Enter a prime number, q : 11
81 Enter primitive root of q, alpha : 2
82
83 Client's private key, Xc = 4
84 Client's public key, Yc = 5
85
86 Server's public key, Ys = 1
87
88 Secret Key, k = 1
89
90 Message received from Server : 452
91 Decrypted message : 453
92 */

```



```

1  /* Author : AKSHAT AGARWAL
2
3  14. Implement DSS algorithm for signing and verification of messages between two
4  parties (obtain H(M) using simple XOR method of hash computation on M). */
5  /* Server Program */
6
7  # include <bits/stdc++.h>
8  # include <arpa/inet.h>
9  using namespace std;
10
11 int createServer(long port) // TCP connection
12 {
13     int sersock = socket(AF_INET, SOCK_STREAM, 0);
14     struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
15
16     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
17     cout << "\nServer Online. Waiting for client..." << endl;
18
19     listen(sersock, 5);
20     int sock = accept(sersock, NULL, NULL);
21     cout << "Connection Established." << endl;
22     return sock;
23 }
24
25 long randInRange(long min, long max) // excluding min and max value
26 {
27     return rand()%(max-min-1) + (min+1);
28 }
29
30 long mod(long a, long b)
31 {
32     return a >= 0 ? (a%b) : b-(abs(a)%b) ;
33 }
34
35 long powermod(long a, long b, long c)
36 {
37     long res=1;
38     for(int i=0; i<b; i++)
39     {
40         res = (res * a) % c;
41     }
42     return res;
43 }
44
45 long findInverse(long R , long D)
46 {
47     int i = 0;
48     long N = D; // copy D to N for taking mod
49     long p[100] = {0,1};
50     long q[100] = {0} ;
51
52     while(R!=0)
53     {
54         q[i] = D/R ;
55         long oldD = D ;
56         D = R ;
57         R = oldD%R ;
58         if(i>1)
59         {

```

```

60         p[i] = mod(p[i-2] - p[i-1]*q[i-2], N) ;
61     }
62     i++ ;
63 }
64 if (i == 1) return 1;
65 else     return p[i] = mod(p[i-2] - p[i-1]*q[i-2], N) ;
66 }
67
68 long H(long M)
69 {
70     return (M ^ 1234); //hash key = 1234
71 }
72
73 int main()
74 {
75     int port;  cout << "\nEnter port : "; cin >> port;
76     int sock = createServer(port);
77
78     long p, q; // prime numbers
79     long r, s; // signature
80     long k, x, y, g; // keys
81     long M, hashval; // Message and Hash
82     srand(time(NULL));
83
84     cout << "\nEnter a large prime number, p (>4) :";   cin >> p;
85     cout << "Enter a prime number, q (p-1 divisible by q & q>2) : "; cin >> q;
86     if( (p-1)%q != 0 || q <=2) { cout << "\nInvalid input\n"; exit(-1); }
87
88     cout<<"Enter message, M = "; cin >> M;
89
90     hashval = H(M);
91     cout << "\nH(M) = " << hashval << endl;
92
93     long h;
94     do{
95         h = randInRange(1, p-1);           // 1 < h < p-1
96         g = powermod(h, (p-1)/q, p);       //g > 1
97     } while(g<=1);
98     cout << "g = " << g;
99
100    x = randInRange(1, q);  cout << "\nServer's Private key, x = " << x;
101    y = powermod(g, x, p);  cout << "\nServer's Public key, y = " << y;
102    k = randInRange(1, q);  cout << "\nSecret key, k = " << k << endl;
103
104    //Signing
105    r = powermod(g, k, p) % q;
106    s = (findInverse(k,q) * (hashval + x*r )) % q;
107    cout << "\nServer's Signature {r,s} = {" << r << ", " << s << "}" << endl;
108
109    send(sock, &p, sizeof(p), 0);
110    send(sock, &q, sizeof(q), 0);
111    send(sock, &g, sizeof(g), 0);
112    send(sock, &y, sizeof(y), 0);
113    send(sock, &M , sizeof(M), 0);
114    send(sock, &r, sizeof(r), 0);
115    send(sock, &s, sizeof(s), 0);
116
117    cout << "\nSent p, q, g, and public key to client.";
118    cout << "\nSent message along with signature to client." << endl << endl;
119 }
120

```

```
121 /*
122 p=71, q=7
123 p=13, q=3
124 p=11, q=5
125 p=569, q=71
126 */
127
128 /*
129 Enter port : 3333
130
131 Server Online. Waiting for client....
132 Connection Established.
133
134 Enter a prime number, p (>4) : 13
135 Enter a prime number, q (p-1 divisible by q & q>2) : 3
136 Enter message, M = 243
137
138 H(M) = 1057
139 g = 3
140 Server's Private key, x = 1
141 Server's Public key, y = 3
142 Secret key, k = 1
143
144 Server's Signature {r,s} = {0, 1}
145
146 Sent p, q, g, and public key to client.
147 Sent message along with signature to client.
148 */
149
```

```

1  /* Author : AKSHAT AGARWAL
2
3  14. Implement DSS algorithm for signing and verification of messages between two
4  parties (obtain H(M) using simple XOR method of hash computation on M). */
5  /* Client Program */
6
7  # include <bits/stdc++.h>
8  # include <arpa/inet.h>
9  using namespace std;
10
11 int connectToServer(const char* ip, long port)
12 {
13     int sock = socket(AF_INET, SOCK_STREAM, 0);
14     struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
15
16     if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 ){
17         cout << "\nRun server program first." << endl; exit(0);
18     }else{
19         cout << "\nClient is connected to Server." << endl;
20     }
21     return sock;
22 }
23
24 long mod(long a, long b)
25 {
26     return a >= 0 ? (a%b) : b-(abs(a)%b) ;
27 }
28
29 long powermod(long a, long b, long c)
30 {
31     long res=1;
32     for(int i=0; i<b; i++)
33     {
34         res = (res * a) % c;
35     }
36     return res;
37 }
38
39 long findInverse(long R , long D)
40 {
41     int i = 0;
42     long N = D; // copy D to N for taking mod
43     long p[100] = {0,1};
44     long q[100] = {0} ;
45
46     while(R!=0)
47     {
48         q[i] = D/R ;
49         long oldD = D ;
50         D = R ;
51         R = oldD%R ;
52         if(i>1)
53         {
54             p[i] = mod(p[i-2] - p[i-1]*q[i-2], N) ;
55         }
56         i++ ;
57     }
58     if (i == 1) return 1;
59     else return p[i] = mod(p[i-2] - p[i-1]*q[i-2], N) ;

```

```

60 }
61
62 long H(long M)
63 {
64     return (M ^ 1234); //hash key = 1234
65 }
66
67 int main()
68 {
69     char ip[50]; cout << "\nEnter server's IP address: "; cin >> ip;
70     int port;    cout << "Enter port : "; cin >> port;
71     int sock = connectToServer(ip, port);
72
73     long p, q; // prime numbers
74     long r, s; // signature
75     long g, y; // keys
76     long M, hashval; // Message and Hash
77     long w, v; // verify
78     srand(time(NULL));
79
80     recv(sock, &p, sizeof(p), 0);
81     recv(sock, &q, sizeof(q), 0);
82     recv(sock, &g, sizeof(g), 0);
83     recv(sock, &y, sizeof(y), 0);
84     recv(sock, &M, sizeof(M), 0);
85     recv(sock, &r, sizeof(r), 0);
86     recv(sock, &s, sizeof(s), 0);
87
88     cout << "Received p = " << p << endl;
89     cout << "Received q = " << q << endl;
90     cout << "Received g = " << g << endl;
91     cout << "Received y = " << y << endl;
92     cout << "Received M' = " << M << endl;
93     cout << "Received r' = " << r << endl;
94     cout << "Received s' = " << s << endl;
95
96     hashval = H(M) ;
97     cout << "\nH(M') = " << hashval << endl;
98
99     //Verifying
100    w = findInverse(s,q) % q; cout << "w = " << w << endl;
101    long u1 = (hashval * w) % q;
102    long u2 = (r * w) % q;
103    v = ((powermod(g,u1,p)*powermod(y,u2,p)) %p) %q; cout<<"v = "<<v<<endl;
104    if(v == r) cout<<"\nDigital Signature Verified. " << endl << endl;
105    else      cout<<"\nDigital Signature is invalid !!!" << endl << endl;
106 }
107
108 /*
109 Enter server's IP address: 127.0.0.1
110 Enter port : 3333
111
112 Client is connected to Server.
113 Received p = 13
114 Received q = 3
115 Received g = 3
116 Received y = 3
117 Received M' = 1057
118 Received r' = 0
119 Received s' = 1
120

```

```
121 H(M') = 243
122 w = 1
123 v = 0
124
125 Digital Signature Verified.
126 */
127
```