

# 1\_monoalphabetic\_cipher.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: MonoAlphabetic cipher
3  Author: Gangadhara Shetty P J
4
5  Algorithm
6  -----
7  A mono-alphabetic cipher is a type of simple substitution cipher. In this cipher
8  technique each letter of
9  the plaintext is replaced by another letter in the cipher-text. An example of a mono-
10 alphabetic cipher key follows:
11
12 Plain Text   >>>  a b c d e f g h i j k l m n o p q r s t u v w
13 x y z
14 Cipher Text >>>  z w x y o p r q a h c b e s u t v f g j l k m n
15 d i
16
17 This key means that any 'a' in the plaintext will be replaced by a 'z' in the cipher-
18 text, any 'z' in the plaintext will be replaced by a 'i' in the cipher-text, and so on.
19 The following program shows the simple implementation of mono-alphabetic cipher
20 technique in c language for encrypting and decrypting file
21
22 */
23 #include<bits/stdc++.h>
24 using namespace std;
25
26 int frequency_count[256];
27 vector<string> keys;
28
29 // To encrypt the plain_text with the key
30 string encrypt(string plain_text, string unique_characters, string key)
31 {
32     for(int i=0; i<plain_text.length(); i++)
33         plain_text[i] = key[ unique_characters.find( plain_text[i] ) ];
34     return plain_text;
35 }
36
37 void generateKeysByPermutation(string text)
38 {
39     sort( text.begin(), text.end());
40     // next_permutation is a readily available function to generate permutation of the
41     string
42     while(next_permutation(text.begin(), text.end()))
43     {
44         // cout<<text<<endl;
45         keys.push_back(text);
46     }
47 }
48
49 string readFile(char* file_name)
50 {
51     string text;
52     ifstream fin(file_name);
53     fin>>text;
54     fin.close();
55     return text;
56 }
57
58 void storeFile(char* file_name, string text)
59 {
60     ofstream fout(file_name);
```

```

55     fout<<text;
56     fout.close();
57 }
58
59 string getUniqueCharacters(string plain_text)
60 {
61     int flag[255]={0};
62     string unique_characters="";
63     // To get the list of unique characters in the above string
64     // Also get the number of occurrence of the characters to find the frequency
    later
65     for(int i=0;i<plain_text.length();i++){
66         if( !frequency_count[plain_text[i]] )
67             unique_characters+=plain_text[i];
68             frequency_count[plain_text[i]]++;
69     }
70     return unique_characters;
71 }
72
73 void calculateFrequency(string unique_characters, string key, string text)
74 {
75     float length = text.length();
76     cout<<"Frequency\tUnique Characters\tChosen Key"<<endl;
77     cout<<"-----"<<endl;
78     // Frequency / Length * => percentage of occurrence of the character in the
    entire string
79     for(int i=0; i<unique_characters.length();i++)
80         cout << frequency_count[ unique_characters[i] ] / length * 100 <<
"\t\t\t" << unique_characters[i] << "\t\t" << key[i] << endl;
81 }
82
83 int main() {
84     string unique_characters, key, plain_text, cipher_text;
85
86     srand(time(0));
87
88     plain_text = readFile("plain_text.txt");
89
90     cout<<"plain_text:\t"<<plain_text<<endl;
91
92     unique_characters = getUniqueCharacters( plain_text );
93     cout<<"unique character:\t"<<unique_characters<<endl;
94
95     generateKeysByPermutation( unique_characters );
96     key = keys[rand()%keys.size()];
97     cout<<"Chosen Key:\t"<<key<<endl;
98
99     cipher_text = encrypt( plain_text, unique_characters, key );
100    cout<<"cipher_text:\t"<<cipher_text<<endl;
101
102    storeFile("cipher_text.txt", cipher_text);
103
104    calculateFrequency(unique_characters, key, plain_text);
105
106    return 0;
107 }
108
109 /*
110 -----
111 Sample Output
112 -----
113
114 plain_text:      gauggsdgssgsgsasuuasudduugsugs
115 unique character:      gaussd
116 Chosen Key:      sgaud

```

117 cipher\_text: sgassudsuusuguaaguaddaasduasu

118

119 Frequency Unique Characters Chosen Key

120

121 25.8065 g s

122 9.67742 a g

123 22.5806 u a

124 29.0323 s u

125 12.9032 d d

126 \*/

## 2\_play\_fair\_cipher.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: Playfair cipher
3  Author: Gangadhara Shetty P J
4
5  Algorithm
6  -----
7      The 'key' for a playfair cipher is generally a word, for the sake of example we will
      choose 'monarchy'.
8      This is then used to generate a 'key square'. Any sequence of 25 letters can be
      used as a key,
9      so long as all letters are in it and there are no repeats. Note that there is no
      'j', it is combined with 'i'.
10     We now apply the encryption rules to encrypt the plaintext.
11
12     1. Remove any punctuation or characters that are not present in the key square (this
      may mean spelling out numbers, punctuation etc.).
13     2. Identify any double letters in the plaintext and replace the second occurrence
      with an 'x' e.g. 'hammer' -> 'hamxmerx'.
14     3. If the plaintext has an odd number of characters, append an 'x' to the end to
      make it even.
15     4. Break the plaintext into pairs of letters, e.g. 'hamxer' -> 'ha mx er'
16     5. The algorithm now works on each of the letter pairs.
17     6. Locate the letters in the key square, (the examples given are using the key
      square above)
18
19     a. If the letters are in different rows and columns, replace the pair with the
      letters on the same row respectively
20         but at the other pair of corners of the rectangle defined by the
      original pair.
21         The order is important - the first encrypted letter of the pair is
      the one that lies on the same row as the first plaintext letter.
22
23     b. If the letters appear on the same row of the table, replace them with the
      letters to their immediate right respectively
24         (wrapping around to the left side of the row if a letter in the
      original pair was on the right side of the row).
25
26     c. If the letters appear on the same column of the table, replace them with the
      letters immediately below respectively
27         (wrapping around to the top side of the column if a letter in the
      original pair was on the bottom side of the column).
28
29 */
30
31 #include<iostream>
32 #include<cmath>
33
34 using namespace std;
35
36 char matrix[5][5];
37
38 // Generate a matrix from the given key.
39 // Assuming all 'j' as 'i', since they can be used interchangeably.
40 // Remove spaces in a key.
41 void GenerateMatrix(string key)
42 {
43     int flag[26]={0};
44     int x_ind=0, y_ind=0;
45     // Add all characters present in a given key
46     for(int i=0;i<key.length();i++)
47     {
```

```

48     if(key[i]==' ') continue;
49     if(key[i]=='j') key[i]='i';
50
51     if(!flag[key[i]-97])
52         matrix[x_ind][y_ind++] = key[i], flag[key[i]-97]=1;
53
54     if(y_ind==5) x_ind++, y_ind=0;
55 }
56
57 // Add all other characters
58 for(int i=0;i<26;i++)
59 {
60     if(i==9) continue;
61     if(key[i]=='j') key[i]='i';
62     if(!flag[i]) matrix[x_ind][y_ind++] = i+97, flag[i]=1 ;
63
64     if(y_ind==5) x_ind++, y_ind=0;
65 }
66 }
67
68 string FormatMessage(string message)
69 {
70     // STEP 1 in algorithm
71     for(int i=0;i<message.length();i++)
72     {
73         if(message[i] == ' ') // add conditions here for any other special characters
74             message = message.replace(i, 1, "");
75         if(message[i] == 'j')
76             message = message.replace(i, 1, "i");
77     }
78
79     // STEP 2 in algorithm
80     for(int i=1;i<message.length();i+=2)
81         if(message[i-1] == message[i])
82             message = message.insert(i, "x");
83     // STEP 3 in algorithm
84     if(message.length()%2)
85         message += "x";
86     return message;
87 }
88
89 // Returns the row position of the given character in a matrix
90 int GetRow(char c)
91 {
92     for(int i=0;i<5;i++)
93         for(int j=0;j<5;j++)
94             if(c==matrix[i][j])
95                 return i;
96 }
97
98 //Returns the column position of the given character in a matrix
99 int GetColumn(char c)
100 {
101     for(int i=0;i<5;i++)
102         for(int j=0;j<5;j++)
103             if(c==matrix[i][j])
104                 return j;
105 }
106
107
108 string Encrypt(string message)
109 {
110     string enc_msg;
111     for(int i=0;i<message.length();i+=2) // i is incremented by 2 inorder to group by
two two characters

```

```

112 {
113     int xind1 = GetRow(message[i]);
114     int xind2 = GetRow(message[i+1]);
115     int yind1 = GetColumn(message[i]);
116     int yind2 = GetColumn(message[i+1]);
117
118     // STEP 6.a in algoritm
119     if( xind1 == xind2 )
120     {
121         enc_msg.append(1, matrix[xind1][(yind1+1)%5]);
122         enc_msg.append(1, matrix[xind2][(yind2+1)%5]);
123     }
124     //STEP 6.b in algorithm
125     else if( yind1 == yind2 )
126     {
127         enc_msg.append(1, matrix[(xind1+1)%5][yind1]);
128         enc_msg.append(1, matrix[(xind2+1)%5][yind2]);
129     }
130     // STEP 6.c in algorithm
131     else
132     {
133         enc_msg.append(1, matrix[ xind1 ][ yind2 ]);
134         enc_msg.append(1, matrix[ xind2 ][ yind1 ]);
135     }
136 }
137 return enc_msg;
138 }
139
140
141 string Decrypt(string message)
142 {
143     string msg;
144     for(int i=0;i<message.length();i+=2)
145     {
146         int xind1 = GetRow(message[i]);
147         int xind2 = GetRow(message[i+1]);
148         int yind1 = GetColumn(message[i]);
149         int yind2 = GetColumn(message[i+1]);
150
151         // STEP 6.a in algoritm (reveres)
152         if( xind1 == xind2 )
153         {
154             msg.append(1, matrix[xind1][ --yind1<0 ? 4: yind1 ]); // to handle
negative modulus if(negative) 4 else num-1
155             msg.append(1, matrix[xind2][ --yind2<0 ? 4: yind2 ]);
156         }
157         // STEP 6.b in algoritm (reverse)
158         else if( yind1 == yind2 )
159         {
160             msg.append(1, matrix[ --xind1<0 ? 4: xind1 ][yind1]);
161             msg.append(1, matrix[ --xind2<0 ? 4: xind2 ][yind2]);
162         }
163         // STEP 6.c in algoritm (reverse)
164         else
165         {
166             msg.append(1, matrix[ xind1 ][ yind2 ]);
167             msg.append(1, matrix[ xind2 ][ yind1 ]);
168         }
169     }
170     return msg;
171 }
172
173 int main()
174 {
175     string message, keys[100];

```

```

176     int num_of_keys;
177
178     cout<<"Enter the number of keys:";
179     cin>>num_of_keys;
180     cin.get();
181
182     cout<<"Enter the keys:";
183     for(int i=0;i<num_of_keys;i++)
184         getline(cin, keys[i]);
185
186     cout<<"Enter a message to be encrypted: ";
187     getline(cin, message);
188
189     for(int i=0;i<num_of_keys;i++)
190     {
191         GenerateMatrix(keys[i]);
192         cout<<endl<<endl;
193         cout<<"-----"<<endl;
194         cout<<"Using key"<<i<<": "<<keys[i]<<endl;
195         cout<<"-----"<<endl;
196         cout<<"Key Matrix: "<<endl;
197         for(int k=0;k<5;k++)
198         {
199             for(int j=0;j<5;j++)
200                 cout<<matrix[k][j]<<" ";
201             cout<<endl;
202         }
203         cout<<"Actual Message: "<<message<<endl;
204         string for_msg = FormatMessage(message);
205         cout<<"Formatted Message: "<<for_msg<<endl;
206         string enc_msg = Encrypt(for_msg);
207         cout<<"Encrypted Message: "<<enc_msg<<endl;
208         string dec_msg = Decrypt(enc_msg);
209         cout<<"Decrypted Message: "<<dec_msg<<endl;
210     }
211     return 0;
212 }
213
214 /*
215 -----
216 Sample Output
217 -----
218
219 Enter the number of keys:1
220 Enter the keys:playfair example
221 Enter a message to be encrypted: hide the gold
222
223 -----
224 Using key0: playfair example
225 -----
226 Key Matrix:
227 p l a y f
228 i r e x m
229 b c d g h
230 k n o q s
231 t u v w z
232 Actual Message: hide the gold
233 Formatted Message: hidethegoldx
234 Encrypted Message: bmodzbxndage
235 Decrypted Message: hidethegoldx
236 */

```

### 3\_hill\_cipher.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: Hill cipher
3  Author: Gangadhara Shetty P J
4  B
5  Algorithm
6  -----
7      Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter
      is represented by a number modulo 26.
8      Often the simple scheme A = 0, B = 1, ..., Z = 25 is used, but this is not an essential
      feature of the cipher.
9      To encrypt a message, each block of n letters (considered as an n-component vector)
      is multiplied by an invertible n × n matrix, against modulus 26.
10     To decrypt the message, each block is multiplied by the inverse of the matrix used
      for encryption.
11
12     The matrix used for encryption is the cipher key, and it should be chosen
      randomly from the set of invertible n × n matrices (modulo 26).
13
14  */
15
16  #include<bits/stdc++.h>
17
18  using namespace std;
19
20  int keyMatrix[100][100], inverseMatrix[100][100];
21  int order;
22
23  string FormatMessage(string message)
24  {
25      for(int i=0;i<message.length();i++)
26          if(message[i] == ' ') // add conditions here for any other special characters
27              message = message.replace(i, 1, "");
28
29
30      for(int i=1;i<message.length();i++)
31          if(message[i-1] == message[i])
32              message = message.insert(i, "x"), i++;
33
34      if(message.length()%order)
35          message += "x";
36      return message;
37  }
38
39  int GetInverseDeterminant(int R , int D = 26){ //R is the remainder or determinant
40      int i =0 ;
41      int p0= 0 , p1 =1 ;
42      int q = 1 ;
43      int q0 , q1 ;
44      while(R!=0){
45          q = D/R ;
46          int tempD = D ;
47          D = R ;
48          R = tempD%R ;
49
50          if(i==0) { p0 = 0 ; q0 = q ; }
51          else if(i==1){ p1==1 ;q1 = q ; }
52          else{
53              int temp = p1 ;
54              p1 = (p0-p1*(q0))%26 ;
55              if(p1<0)p1 = 26-(abs(p1)%26) ;
56              p0 = temp ;
```



```

57
58         q0 = q1;
59         q1 = q ;
60     }
61     //         cout<<"p0 , p1 = " << p0 <<" " <<p1<< endl;
62
63         i++ ;
64     }
65     p1 = (p0-p1*(q0))%26 ;
66     return p1 ;
67 }
68
69 int GetDeterminant()
70 {
71     int determinant = 0;
72     if(order==2)
73         determinant = keyMatrix[0][0] * keyMatrix[1][1] - keyMatrix[0][1] * keyMatrix[1]
74 [0];
75     else
76         for(int i = 0; i < 3; i++)
77             determinant = determinant + (keyMatrix[0][i] * (keyMatrix[1][(i+1)%3] *
78 keyMatrix[2][(i+2)%3] - keyMatrix[1][(i+2)%3] * keyMatrix[2][(i+1)%3]));
79
80     if(determinant<0)
81         determinant = 26 - (int(-determinant)%26);
82     else
83         determinant = int(determinant)%26;
84     determinant = GetInverseDeterminant(determinant, 26);
85
86     return determinant;
87 }
88
89 string Multiply(string msg_group, int matrix[][100])
90 {
91     string result;
92     for(int i=0; i<order; i++)
93     {
94         float val = 0 ;
95         for(int j=0; j<order; j++)
96             val = val + matrix[j][i] * (msg_group[j] - 'a');
97
98         if(val>=0)
99             val = int(val)%26 + 'a';
100         else
101             val = 26 - (int(-val)%26) + 'a';
102
103         result += int(val);
104     }
105     return result;
106 }
107
108 void FindInverse(int determinant)
109 {
110     if(order==2)
111     {
112         inverseMatrix[0][0] = keyMatrix[1][1]*determinant;
113         inverseMatrix[1][1] = keyMatrix[0][0]*determinant;
114         inverseMatrix[0][1] = -keyMatrix[0][1]*determinant;
115         inverseMatrix[1][0] = -keyMatrix[1][0]*determinant;
116     }
117     else
118         for(int j=0; j<order; j++)
119             for(int i=0; i<order; i++)
120                 inverseMatrix[i][j] = ((keyMatrix[(j+1)%3][(i+1)%3] * keyMatrix[(j+2)%3]
121 [(i+2)%3]) - (keyMatrix[(j+1)%3][(i+2)%3] * keyMatrix[(j+2)%3][(i+1)%3]))*determinant;

```

```

119         for(int j=0; j<order; j++)
120             for(int i=0; i<order; i++)
121                 if(inverseMatrix[i][j] < 0 )         inverseMatrix[i][j] = 26 - int(-
inverseMatrix[i][j])%26;
122             else inverseMatrix[i][j] = int(inverseMatrix[i][j]) %26;
123     }
124
125     string Encrypt(string message)
126     {
127         string enc_msg;
128         for(int i=0; i<message.length(); i+=order)
129         {
130             string msg_group = message.substr(i, order);
131             msg_group = Multiply(msg_group, keyMatrix);
132             enc_msg = enc_msg.append(msg_group);
133         }
134         return enc_msg;
135     }
136
137     string Decrypt(string message)
138     {
139         string msg;
140         FindInverse(GetDeterminant());
141         for(int i=0; i<message.length(); i+=order)
142         {
143             string msg_group = message.substr(i, order);
144             msg_group = Multiply(msg_group, inverseMatrix);
145             msg = msg.append(msg_group);
146         }
147         return msg;
148     }
149
150     int main()
151     {
152         string message;
153         cout << "Enter the key matrix order: ";
154         cin >> order;
155
156         cout << "Enter the keys:";
157         for(int i = 0; i < order; i++)
158             for(int j=0; j<order; j++)
159                 cin >> keyMatrix[i][j];
160         cin.get();
161
162         cout<<"Enter a message to be encrypted: ";
163         getline(cin, message);
164
165         FormatMessage(message);
166         string enc_msg = Encrypt(message);
167         string dec_msg = Decrypt(enc_msg);
168         cout<<"Message: "<<message<<endl;
169         cout<<"Encrypted Message: "<<enc_msg<<endl;
170         cout<<"Decrypted Message: "<<dec_msg<<endl;
171
172         return 0;
173     }
174
175     /*
176     -----
177     Sample Output
178     -----
179     Enter the key matrix order: 3
180     Enter the keys:17 17 5
181     21 18 21
182     2 2 19

```

183 Enter a message to be encrypted: paymoremoney

184 Message: paymoremoney

185 Encrypted Message: rrlmwbkaspdh

186 Decrypted Message: paymoremoney

187

188 \*/

## 4\_transposition\_cipher.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: Columnar Transposition Cipher
3  Author: Gangadhara Shetty P J
4  Algorithm
5  -----
6  In a transposition cipher, the order of the alphabets is re-arranged to obtain the
   cipher-text.
7      1. The message is written out in rows of a fixed length, and then read out again
   column by column,
8          and the columns are chosen in some scrambled order.
9      2. Width of the rows and the permutation of the columns are usually defined by a
   keyword.
10     3. Any spare spaces are filled with nulls or left blank or placed by a character
   to
11         construct the rectangular matrix (Example: $).
12     4. Finally, the message is read off in columns, in the order specified by the
   keyword.
13  */
14
15  #include<bits/stdc++.h>
16
17  using namespace std;
18
19
20  string Encrypt(string message, string keys)
21  {
22      string enc_msg;
23      int key[keys.length()], i;
24
25      for(int k=0;k<keys.length();k++)
26          key[keys[k]-'0'-1] = k;
27
28      for(int k=0;k<keys.length();k++){
29          for(i=key[k]; i<message.length(); i+=keys.length())
30              enc_msg+=message[i];
31          if(i%keys.length() >= message.length()%keys.length())
32              enc_msg+="$";
33      }
34      return enc_msg;
35  }
36
37  string Decrypt(string message, string keys)
38  {
39      string msg;
40      int key[keys.length()], i, len = message.length()/ keys.length();
41
42      for(int k=0;k<len;k++)
43          for(i=0;i<keys.length();i++)
44              msg+=message[(keys[i]-'0'-1)*len + k];
45      return msg;
46  }
47
48  int main()
49  {
50      string message, keys;
51
52      cout<<"Enter a message to be encrypted: ";
53      getline(cin, message);
54
55      cout << "Enter the keys as a string:";
56      getline(cin, keys);
```

```

57
58     string enc_msg = Encrypt(message, keys);
59     string dec_msg = Decrypt(enc_msg, keys);
60     cout<<"Encrypted Message: "<<enc_msg<<endl;
61     cout<<"Decrypted Message: "<<dec_msg<<endl;
62
63     return 0;
64 }
65
66 /*
67 -----
68 Output
69 -----
70 Enter a message to be encrypted: hidethegold
71 Enter the keys as a string:14253
72 Encrypted Message: hhddg$tl$ie$eo$
73 Decrypted Message: hidethegold$$$
74 */

```

## keygen.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  lab 5
3  Program: DES Key generator
4  Author: Gangadhara Shetty P J
5  */
6  #include <bits/stdc++.h>
7  using namespace std;
8
9  int permChoiceOne[] = {
10     57, 49, 41, 33, 25, 17, 9 ,
11     1 , 58, 50, 42, 34, 26, 18,
12     10, 2 , 59, 51, 43, 35, 27,
13     19, 11, 3 , 60, 52, 44, 36,
14     63, 55, 47, 39, 31, 23, 15,
15     7 , 62, 54, 46, 38, 30, 22,
16     14, 6 , 61, 53, 45, 37, 29,
17     21, 13, 5 , 28, 20, 12, 4
18 };
19
20 int permChoiceTwo[] = {
21     14, 17, 11, 24, 1 , 5 , 3 , 28,
22     15, 6 , 21, 10, 23, 19, 12, 4 ,
23     26, 8 , 16, 7 , 27, 20, 13, 2 ,
24     41, 52, 31, 37, 47, 55, 30, 40,
25     51, 45, 33, 48, 44, 49, 39, 56,
26     34, 53, 46, 42, 50, 36, 29, 32
27 };
28
29 int leftShiftTable[] = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};
30 string rotateSubKey(string s , int rot)
31 {
32     return s.substr(rot, s.length()-rot) + s.substr(0, rot) ;
33 }
34 string firstPermute(string input)
35 {
36     string res = "" ;
37     for(int i=0 ; i<56 ; i++)
38         res += input[permChoiceOne[i]-1];
39     return res ;
40 }
41 string secondPermute(string input)
42 {
43     string res = "" ;
44     for(int i=0 ; i<48 ; i++)
45         res += input[permChoiceTwo[i]-1];
46     return res ;
47 }
48 void genKeys(string left, string right)
49 {
50     ofstream fout ;
51     fout.open("keygen.txt");
52     for (int i=0; i<16; i++)
53     {
54         left = rotateSubKey(left , leftShiftTable[i]);
55         right = rotateSubKey(right, leftShiftTable[i]);
56         string key = secondPermute(left+right);
57         cout << "key " << i+1 << " \t: " << key << endl;
58         fout << key << endl;
59     }
60 }
61
```

```

62 int main()
63 {
64     unsigned long long hexkey;
65     cout << "\nEnter key in hexadecimal : " ;
66     cin >> hex >> hexkey; // to read hex input cin >> hex >> input
67     string key = bitset<64>(hexkey).to_string(); // to convert hex to binary string
68     cout << "Binary key (k) \t: " << key << endl;
69     key = firstPermute(key) ;
70     cout << "PC-1 key (k+) \t: " << key << endl;
71     cout << "\nSubKeys: " << endl;
72     genKeys(key.substr(0,28) , key.substr(28,28));
73     cout<<endl<<endl ;
74 }
75
76 /*
77 Enter key in hexadecimal : 1FE22472901BB2A3
78 Binary key (k)   : 0001111111100010001001000111001010010000000110111011001010100011
79 PC-1 key (k+)    : 11010010000010101100111001111110101100000101001000011001
80
81 SubKeys:
82 key 1   : 110010110110001010101100110001101000000110100011
83 key 2   : 001101011010010010101111001001011110000110011010
84 key 3   : 111100110000110010000010011001010001010001000011
85 key 4   : 011110001010101010110100110011101000000001101110
86 key 5   : 100101001011010000011110000001001101111111001100
87 key 6   : 011001100000011001110110000110001001010011110001
88 key 7   : 111011101101100000100100110010111100110000100001
89 key 8   : 100010101010001101111010000010100110111100011000
90 key 9   : 111000111100001011010111100101000001110000110011
91 key 10  : 001111011101001110000010100011110000101001110100
92 key 11  : 001100100001000111111011000100011110101111010000
93 key 12  : 101111010100000001010101001100011000010000010101
94 key 13  : 000001110100101110011100110010110010010010010000110
95 key 14  : 000111100011000110110101001011000110001110001101
96 key 15  : 100111110000110001101001001100100101000011000111
97 key 16  : 010010011010100110011011011010100010100101010110
98
99 */

```

## sbox input.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  lab 6
3  Program: SBOX input generator
4  Author: Gangadhara Shetty P J
5
6  */
7  #include<bits/stdc++.h>
8  using namespace std;
9
10 string key;
11 int permute[]={
12     32,1,2,3,4,5,
13     4,5,6,7,8,9,
14     8,9,10,11,12,13,
15     12,13,14,15,16,17,
16     16,17,18,19,20,21,
17     20,21,22,23,24,25,
18     24,25,26,27,28,29,
19     28,29,30,31,32,1
20 };
21 void findPermutation()
22 {
23     string key1;
24     for(int i=0;i<48;i++)
25         key1 += key[permute[i]-1+32];
26     key=key1;
27 }
28 int main()
29 {
30
31     unsigned long long hexKey;
32     string inputkey;
33     cout<<"Enter a 64 bit key in hex: ";
34     cin>>hex>>hexKey;
35     key=bitset<64>(hexKey).to_string();
36     findPermutation();
37     cout<<"Enter a 48 bit input key in hex: ";
38     cin>>hex>>hexKey;
39
40     inputkey=bitset<48>(hexKey).to_string();
41
42     for(int i=0;i<48;i++)
43         if(key[i]==inputkey[i]) key[i]='0';
44         else key[i]='1';
45
46     cout<<"S-BOX INPUT: "<<hex<< bitset<48>(key).to_ulong()<<endl;
47
48 }
49 /*
50 OUTPUT
51 -----
52 Enter a 64 bit key in hex: aaaaaaaaaaf0aaf0aa
53 Enter a 48 bit input key in hex: 1b02effc7072
54 S-BOX INPUT: 6117ba866527
55
56 */
```



## sboxoutput.cpp in gangadharashettyj/labs (master)

```
1  /*
2  lab 7
3  Program: SBOX output key generator
4  Author: Gangadhara Shetty P J
5  */
6  #include<bits/stdc++.h>
7  using namespace std;
8
9  string key, previous;
10 int permute[]={
11     16,7,20,21,29,12,28,17,
12     1,15,23,26,5,18,31,10,
13     2,8,24,14,32,27,3,9,
14     19,13,30,6,22,11,4,25
15 };
16 int sbox[][4][16]={
17     {
18         {14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7},
19         {0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8},
20         {4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0},
21         {15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13}
22     },
23     {
24         {15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10},
25         {3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5},
26         {0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15},
27         {13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9}
28     },
29     {
30         {10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8},
31         {13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1},
32         {13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7},
33         {1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12}
34     },
35     {
36         {7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15},
37         {13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9},
38         {10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4},
39         {3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14}
40     },
41     {
42         {2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9},
43         {14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6},
44         {4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14},
45         {11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3}
46     },
47     {
48         {12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11},
49         {10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8},
50         {9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6},
51         {4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13}
52     },
53     {
54         {4,11,5,14,15,0,8,13,3,12,9,7,5,10,6,1},
55         {13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6},
56         {1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2},
57         {6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12}
58     },
59     {
60         {13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7},
61         {1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2},
```

```

62         {7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8},
63         {2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11}
64     }
65 };
66 void findPermutation()
67 {
68     string key1="";
69     for(int i=0;i<32;i++)
70         key1 += key[permute[i]-1];
71     key=key1;
72 }
73 int main()
74 {
75     unsigned long long hexkey;
76     string key1, str = "";
77     cout<<"Enter a 48 bit input key in hex: ";
78     cin>>hex>>hexkey;
79
80     key=bitset<48>(hexkey).to_string();
81
82     cout<<"Enter a 64 bit key in hex: ";
83     cin>>hex>>hexkey;
84     previous=bitset<64>(hexkey).to_string();
85
86     for(int i=0, sb=0;i<48;i+=6, sb++){
87         string row = "", col = "";
88         row= row+key[i]+key[i+5];
89         col= col+key[i+1]+key[i+2]+key[i+3]+key[i+4];
90         int rowval = (int)bitset<2>(row).to_ulong();
91         int colval = (int)bitset<4>(col).to_ulong();
92         string tempKey = bitset<4>(sbox [sb] [rowval] [colval] ).to_string();
93         cout<<"SBOX " << sb + 1 <<" OUTPUT: " <<tempKey<<endl;
94         key1+=tempKey;
95     }
96
97     key=key1;
98     findPermutation();
99
100    for(int i=0;i<32;i++)
101        if(key[i]==previous[i]) key[i]='0';
102        else key[i]='1';
103    cout<<"S-BOX OUTPUT: " <<hex<< (int)bitset<32>(key).to_ulong()<<endl;
104 }
105
106 /*
107 OUTPUT
108 -----
109 Enter a 48 bit input key in hex: 6117ba866527
110 Enter a 64 bit key in hex: cc00ccfff0aaf0aa
111 SBOX 1 OUTPUT: 0101
112 SBOX 2 OUTPUT: 1100
113 SBOX 3 OUTPUT: 1000
114 SBOX 4 OUTPUT: 0010
115 SBOX 5 OUTPUT: 1011
116 SBOX 6 OUTPUT: 0101
117 SBOX 7 OUTPUT: 1001
118 SBOX 8 OUTPUT: 0111
119 S-BOX OUTPUT: ef4a6544
120
121 */

```

## rc4.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: RC4 Algorithm
3  Author: Gangadhara Shetty P J
4  */
5  #include<iostream>
6  using namespace std;
7  int main()
8  {
9      int s[256], t[256], k[256], p[256], c[256], j=0;
10     string plain, cipher, key;
11     cout<<"Enter plain text nad key:";
12     cin>>plain>>key;
13
14     cout<<"plaintext in bytes : " ;
15     for(int i=0 ;i < plain.length() ; i++) cout<< (int)plain[i]<<" " ;
16     cout<<endl;
17
18     for(int i=0;i<255;i++)
19         s[i] = i, t[i] = (int)key[i % key.length()];
20
21     for(int i=0, j=0; i<256; i++){
22         j = (j+s[i]+t[i])%256;
23         int t = s[i], s[i] = s[j], s[j] = t;
24     }
25
26     int i=0;
27     j=0;
28     for(int l=0;l<plain.length();l++){
29         i = (i+1)%256;
30         j=(j+s[i])%256;
31
32         int t = s[i], s[i] = s[j], s[j] = t;
33
34         t= (s[i]+s[j])%256;
35         k[l] = s[t];
36     }
37     cout<<"keystream in bytes : " ;
38     for(int i =0 ;i < plain.length() ; i++) cout<< k[i]<<" " ;
39     cout<<endl;
40
41     cout<<"cipher text in bytes : " ;
42     for(int i=0;i<plain.length();i++)
43         c[i] = (plain[i] ^ k[i]), cout<< (int)(plain[i] ^ k[i])<<" " ;
44     cout<<endl;
45
46     cout<<"plain text in bytes after decryption : " ;
47     for(int i=0;i<plain.length();i++)
48         cout<< (int)(c[i] ^ k[i])<<" " ;
49     cout<<endl;
50 }
51 /*
52 output:
53 Enter plain text:CNSLAB
54 Enter key:BALSNC
55 plaintext in bytes : 67 78 83 76 65 66
56 keystream in bytes : 170 24 147 247 205 32
57 cipher text in bytes : 233 86 192 187 140 98
58 plain text in bytes after decryption : 67 78 83 76 65 66
59 */
```

## BBS and Rabin Millier.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: BBS Algorithm
3  Author: Gangadhara Shetty P J
4  */
5  #include<bits/stdc++.h>
6  #define BIT_SIZE 16
7
8  using namespace std;
9
10 int powModN(int num,int p,int n)
11 {
12     int res=1;
13     for(int i=0; i<p; i++)
14         res = (res * num) % n;
15     return res;
16 }
17 bool rabinMiller(int n)
18 {
19     int k, q=n-1;
20     for(k=0; q%2==0; k++, q/=2);
21
22     for(int i=0; i<4; i++)
23     {
24         int a = rand()%(n-1)+1;
25         if(powModN(a,q,n) == 1)
26             return true;
27         for(int j=0; j<=k-1; j++)
28             if(powModN(a, pow(2,j)*q, n) == n-1)
29                 return true;
30     }
31     return false;
32 }
33 int main()
34 {
35     long long int s, p, q, n;
36     string bits="";
37     srand(time(NULL));
38     cout<<"Enter P, Q and Seed value: ";
39     cin>>p>>q>>s;
40     n = p*q;
41     s=(s*s)%n;
42     cout<<"bits generated: ";
43     for(int i=0;i<BIT_SIZE;i++)
44         s=(s*s)%n, bits+=(s%2+'0'), cout<<s%2<<" ";
45     int num = bitset<BIT_SIZE>(bits).to_ulong();
46     cout<<endl<<"Random Number: "<<num<<endl;
47     cout<<"Rabin Miller test: ";
48     if(rabinMiller(num))
49         cout<<"Composite"<<endl;
50     else
51         cout<<"InConclusive"<<endl;
52 }
53 */
54
55 output:
56 Enter P, Q and Seed value: 7 11 7
57 bits generated: 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
58 Random Number: 4369
59 Rabin Miller test: InConclusive
60 */
```

## client.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: RSA client
3  Author: Gangadhara Shetty P J
4  */
5  # include <bits/stdc++.h>
6  # include <arpa/inet.h>
7  using namespace std;
8  char buffer[100];
9  int connectToServer(const char* ip, int port)
10 {
11     int sock = socket(AF_INET, SOCK_STREAM, 0);
12     struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
13     connect(sock, (struct sockaddr *) &addr, sizeof(addr));
14     return sock;
15 }
16 int powModN(int num, int p, int n)
17 {
18     int res=1;
19     for(int i=0; i<p; i++)
20         res = (res * num) % n;
21     return res;
22 }
23 int gcd(int p, int q)
24 {
25     if(q==0) return p;
26     gcd(q, p%q);
27 }
28 void itoc(int n1, int n2)
29 {
30     string s = to_string(n1)+"|" + to_string(n2);
31     strcpy(buffer, s.c_str());
32 }
33 int GetInverseDeterminant(int e, int fi){
34     for(int i=1; i<fi; i++)
35         if((i*e)%fi==1) return i;
36     return -1;
37 }
38 void generateKey(int p, int q, int &e, int &d, int &n)
39 {
40     n = p*q;
41     int fi=(p-1)*(q-1);
42     for(int i=2; i<fi; i++)
43         if(gcd(i, fi) ==1)
44             {e=i; break;}
45     d = GetInverseDeterminant(e, fi);
46     cout<<"Public key of server: ("<<e<<"|"<<n<<")"<<endl;
47     cout<<"Private key of server: ("<<d<<"|"<<n<<")"<<endl;
48 }
49 int main()
50 {
51     char ip[50]="127.0.0.1";
52     int port=1234, p, q, e, d, n, fi, C;
53     int sock = connectToServer(ip, port);
54
55     cout << "\nEnter two prime numbers : ";
56     cin >> p >> q;
57     generateKey(p, q, e, d, n);
58
59     itoc(e,n);
60     send(sock, &buffer, sizeof(buffer), 0);
61     cout << "\nSent Public key to server." << endl;
```

```
62
63     recv(sock, &C, sizeof(C), 0);
64     cout << "\nCiphertext received from server : " << C << endl;
65
66     int M = powModN(C, d, n);
67     cout << "\nDecrypted Text : " << M << endl << endl;
68 }
69
70 /*
71 Enter two prime numbers : 11 7
72 d= 43
73 Sent Public key to server.
74 Ciphertext received from server : 64
75 Decrypted Text : 36
76 */
```

## server.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: RSA server
3  Author: Gangadhara Shetty P J
4  */
5  # include <bits/stdc++.h>
6  # include <arpa/inet.h>
7  using namespace std;
8  char buffer[100];
9  int createServer(int port)
10 {
11     int sersock = socket(AF_INET, SOCK_STREAM, 0);
12     struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
13     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
14     listen(sersock, 5);
15     int sock = accept(sersock, NULL, NULL);
16     return sock;
17 }
18 void ctoi(char buf[100], int &n1, int &n2)
19 {
20     int i=0;
21     n1=0;n2=0;
22     while(buf[i]!='|')
23         n1*=10, n1+=(buf[i++]-'0');
24     while(buf[++i])
25         n2*=10, n2+=(buf[i]-'0');
26 }
27 int powModN(int num,int p,int n)
28 {
29     int res=1;
30     for(int i=0; i<p; i++)
31         res = (res * num) % n;
32     return res;
33 }
34 int main()
35 {
36     int port=1234, e,n, M;
37     int sock = createServer(port);
38
39     recv(sock, &buffer, sizeof(buffer), 0);
40     ctoi(buffer, e,n);
41     cout << "\nPublic key received from client : (" << e << ", " << n << ")" << endl;
42
43     cout << "\nEnter message(M<" << n << ") to encrypt : ";
44     cin >> M;
45
46     int C = powModN(M, e, n);
47     cout << "\nEncrypted Text : " << C << endl;
48     send(sock, &C, sizeof(C), 0);
49     cout << "\nSent ciphertext to client." << endl << endl;
50 }
51 /*
52 Public key received from client : {7, 77}
53 Enter message(M<77) to encrypt : 36
54 Encrypted Text : 64
55 Sent ciphertext to client.
56 */
```

## client.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: RSA Block client
3  Author: Gangadhara Shetty P J
4  */
5  # include <bits/stdc++.h>
6  # include <arpa/inet.h>
7  using namespace std;
8  char buffer[100];
9  int connectToServer(const char* ip, int port)
10 {
11     int sock = socket(AF_INET, SOCK_STREAM, 0);
12     struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
13     connect(sock, (struct sockaddr *) &addr, sizeof(addr));
14     return sock;
15 }
16
17 int powModN(int num, int p, int n)
18 {
19     int res=1;
20     for(int i=0; i<p; i++)
21         res = (res * num) % n;
22     return res;
23 }
24 int gcd(int p, int q)
25 {
26     if(q==0) return p;
27     gcd(q, p%q);
28 }
29 void itoc(int n1, int n2)
30 {
31     string s = to_string(n1)+"|" + to_string(n2);
32     strcpy(buffer, s.c_str());
33 }
34 int GetInverseDeterminant(int e, int fi){
35     for(int i=1; i<fi; i++)
36         if((i*e)%fi==1) return i;
37     return -1;
38 }
39 void generateKey(int p, int q, int &e, int &d, int &n)
40 {
41     n = p*q;
42     int fi=(p-1)*(q-1);
43     for(int i=2; i<fi; i++)
44         if(gcd(i, fi) ==1)
45             {e=i; break;}
46     d = GetInverseDeterminant(e, fi);
47     cout<<"Public key of server: ("<<e<<"|"<<n<<)"<<endl;
48     cout<<"Private key of server: ("<<d<<"|"<<n<<)"<<endl;
49 }
50 int main()
51 {
52     char ip[50]="127.0.0.1";
53     int port=1234;
54     int p, q, e, d, n, fi, C, m;
55     int sock = connectToServer(ip, port);
56     string message;
57
58     cout << "\nEnter two prime numbers : ";
59     cin >> p >> q;
60
61     generateKey(p, q, e, d, n);
```



```

62
63     itoc(e,n);
64     send(sock, &buffer, sizeof(buffer), 0);
65     cout << "\nSent Public key to server." << endl;
66
67     recv(sock, &C, sizeof(C), 0);
68     cout<<"Received Encrypted Message: ";
69     for(int i=0;i<C;i++)
70     {
71         recv(sock, &m, sizeof(m), 0);
72         message = message + (char)powModN(m, d, n);
73         cout<<m<<" ";
74     }
75     cout<<endl<<"Message is: "<<message<<endl;
76 }
77
78 /*
79 Enter two prime numbers : 101 131
80 Public key of server: (3|13231)
81 Private key of server: (8667|13231)
82 Sent Public key to server.
83 Received Encrypted Message:  4436  7900  12541  2767  12965  1791
84 Message is: cnslab
85 */

```

## server.cpp in gangadharashettyj/labs (master)

```
1  /*
2  Program: RSA Block server
3  Author: Gangadhara Shetty P J
4  */
5  # include <bits/stdc++.h>
6  # include <arpa/inet.h>
7  using namespace std;
8  char buffer[100];
9  int createServer(int port)
10 {
11     int sersock = socket(AF_INET, SOCK_STREAM, 0);
12     struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
13     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
14
15     listen(sersock, 5);
16     int sock = accept(sersock, NULL, NULL);
17
18     return sock;
19 }
20 void ctoi(char buf[100], int &n1, int &n2)
21 {
22     int i=0;
23     n1=0;n2=0;
24     while(buf[i]!='\0')
25         n1*=10, n1+=(buf[i++]-'0');
26     while(buf[++i])
27         n2*=10, n2+=(buf[i]-'0');
28 }
29 int powModN(int num,int p,int n)
30 {
31     int res=1;
32     for(int i=0; i<p; i++)
33         res = (res * num) % n;
34     return res;
35 }
36 int main()
37 {
38     int port=1234, e,n, len;
39     int sock = createServer(port);
40     string message;
41
42     recv(sock, &buffer, sizeof(buffer), 0);
43     ctoi(buffer, e,n);
44     cout << "\nPublic key received from client : (" << e << ", " << n << ")" << endl;
45
46     cout << "\nEnter message to be send : ";
47     cin >> message;
48
49     len = message.length();
50     send(sock, &len, sizeof(len), 0);
51
52
53     cout << "\nEncrypted Message : ";
54     for(int i=0; message[i]; i++)
55     {
56         int C = powModN(message[i], e, n);
57         send(sock, &C, sizeof(C), 0);
58         cout<<C<<" ";
59     }
60     cout<<endl<<"Message Sent"<<endl;
61 }
```

```
62 /*
63 Public key received from client : {3, 13231}
64 Enter message to be send : cnslab
65 Encrypted Message : 4436 7900 12541 2767 12965 1791
66 Message Sent
67 */
```

## client.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: RSA Key Exchange client
3  Author: Gangadhara Shetty P J
4  */
5  # include <bits/stdc++.h>
6  # include <arpa/inet.h>
7  using namespace std;
8  char buffer[100];
9  int connectToServer(const char* ip, int port)
10 {
11     int sock = socket(AF_INET, SOCK_STREAM, 0);
12     struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
13     connect(sock, (struct sockaddr *) &addr, sizeof(addr));
14     return sock;
15 }
16 int GetInverseDeterminant(int e, int fi){
17     for(int i=1; i<fi; i++)
18         if((i*e)%fi==1) return i;
19     return -1;
20 }
21 int powModN(int num, int p, int n)
22 {
23     int res=1;
24     for(int i=0; i<p; i++)
25         res = (res * num) % n;
26     return res;
27 }
28 void ctoi(char buf[100], int &n1, int &n2)
29 {
30     int i=0;
31     n1=0; n2=0;
32     while(buf[i]!='|')
33         n1*=10, n1+=(buf[i++]-'0');
34     while(buf[++i])
35         n2*=10, n2+=(buf[i]-'0');
36 }
37 void itoc(int n1, int n2)
38 {
39     string s = to_string(n1)+"|" + to_string(n2);
40     strcpy(buffer, s.c_str());
41 }
42 int gcd(int p, int q)
43 {
44     if(q==0) return p;
45     gcd(q, p%q);
46 }
47 void generateKey(int p, int q, int &e, int &d, int &n)
48 {
49     n = p*q;
50     int fi=(p-1)*(q-1);
51     for(int i=2; i<fi; i++)
52         if(gcd(i, fi) ==1)
53             {e=i; break;}
54     d = GetInverseDeterminant(e, fi);
55     cout<<"Public key of client: ("<<e<<"|"<<d<<")"<<endl;
56     cout<<"Private key of client: ("<<d<<"|"<<d<<")"<<endl;
57 }
58 int main()
59 {
60     char ip[50]="127.0.0.1";
61     int port=1234, pue, pre, pus, sid, nonces, noncec, cid, noncec1, ns, nc, p, q;
```

```

62     int sock = connectToServer(ip, port);
63     srand(time(NULL));
64
65     cout << "\n1. Enter two prime numbers : ";
66     cin >> p >> q;
67
68     generateKey(p, q, pue, pre, nc);
69     itoc(pue, nc);
70     cout<<"Sending pue\n " <<buffer<<endl;
71     send(sock, &buffer, sizeof(buffer), 0);
72
73     recv(sock, &buffer, sizeof(buffer), 0);
74     ctoi(buffer, pus, ns);
75     cout<<"received server pus\n " <<buffer<<endl;
76
77
78     recv(sock, &buffer, sizeof(buffer), 0);
79     ctoi(buffer, sid, nonces);
80     nonces = powModN(nonces, pre, nc);
81     sid=powModN(sid, pre, nc);
82     cout<<"received encrypted sid|nonces " <<buffer<<endl;
83     cout<<"received decrypted sid|nonces " <<sid<<"|" <<nonces<<endl;
84
85     noncec = rand()%100;
86     itoc(powModN(nonces, pus, ns), powModN(noncec, pus, ns));
87     send(sock, &buffer, sizeof(buffer), 0);
88     cout<<"Sending plain nonces|noncec " << nonces<<"|" <<noncec<<endl;
89     cout<<"Sending encrypted nonces|noncec " << buffer<<endl;
90
91     recv(sock, &buffer, sizeof(buffer), 0);
92     noncec1 = atoi(buffer);
93     noncec1 = powModN(noncec1, pre, nc);
94     cout<<"received encrypted noncec " <<buffer<<endl;
95     cout<<"received decrypted noncec " <<noncec1<<endl;
96     if(noncec!=noncec1)
97     {
98         cout<<"Nonce din't match"<<endl;
99         exit(0);
100     }
101     else
102         cout<<"Server Authenticated"<<endl;
103
104     recv(sock, &buffer, sizeof(buffer), 0);
105     cout<<"received encrypted key " <<buffer<<endl;
106     int key = powModN(atoi(buffer), pre, nc);
107     key = powModN(key, pus, ns);
108     cout<<"received decrypted key " <<key<<endl;
109     return 0;
110 }
111
112 /*
113 1. enter client (e|n): 7477|18281
114 Sending pue\n 7477|18281
115 2. enter client (d): 14413
116 received server pus\n 4551|13231
117 received encrypted sid|nonces 4168|11880
118 received decrypted sid|nonces 29|28
119 Sending plain nonces|noncec 28|28
120 Sending encrypted nonces|noncec 6840|6840
121 received encrypted noncec 6726
122 received decrypted noncec 14381
123 received encrypted key 5502
124 received decrypted key 454
125 */

```

## server.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: RSA Key Exchange server
3  Author: Gangadhara Shetty P J
4  */
5  # include <bits/stdc++.h>
6  # include <arpa/inet.h>
7  using namespace std;
8  char buffer[100];
9  int createServer(int port)
10 {
11     int sersock = socket(AF_INET, SOCK_STREAM, 0);
12     struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
13     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
14     listen(sersock, 5);
15     int sock = accept(sersock, NULL, NULL);
16     return sock;
17 }
18 int powModN(int num,int p,int n)
19 {
20     int res=1;
21     for(int i=0; i<p; i++)
22         res = (res * num) % n;
23     return res;
24 }
25 void ctoi(char buf[100], int &n1, int &n2)
26 {
27     int i=0;
28     n1=0;n2=0;
29     while(buf[i]!='|')
30         n1*=10, n1+=(buf[i++]-'0');
31     while(buf[++i])
32         n2*=10, n2+=(buf[i]-'0');
33 }
34 void itoc(int n1, int n2)
35 {
36     string s = to_string(n1)+"|" +to_string(n2);
37     strcpy(buffer, s.c_str());
38 }
39 void itoc(int n1)
40 {
41     string s = to_string(n1);
42     strcpy(buffer, s.c_str());
43 }
44 int GetInverseDeterminant(int e ,int fi){
45     for(int i=1;i<fi;i++)
46         if((i*e)%fi==1) return i;
47     return -1;
48 }
49 int gcd(int p, int q)
50 {
51     if(q==0) return p;
52     gcd(q, p%q);
53 }
54 void generateKey(int p, int q, int &e, int &d, int &n)
55 {
56     n = p*q;
57     int fi=(p-1)*(q-1);
58     for(int i=2;i<fi; i++)
59         if(gcd(i, fi) ==1)
60             {e=i; break;}
61     d = GetInverseDeterminant(e, fi);
```

```

62     cout<<"Public key of server: ("<<e<<"|"<<d<<")"<<endl;
63     cout<<"Private key of server: ("<<d<<"|"<<d<<")"<<endl;
64 }
65
66 int main()
67 {
68     int port=1234, sid, sid1, cid, nonces,nonces1, noncec, pue, ns, nc, pus, prs, key,
p, q;
69     int sock = createServer(port);
70     srand(time(NULL));
71
72     recv(sock, &buffer, sizeof(buffer), 0);
73     ctoi(buffer, pue,nc);
74     cout<<"received pue|n "<<buffer<<endl;
75
76     cout << "\n2. Enter two prime numbers : ";
77     cin >> p >> q;
78
79     generateKey(p, q, pus, prs, ns);
80     itoc(pus, ns);
81     cout<<"Sending pue|n "<<buffer<<endl;
82     send(sock, &buffer, sizeof(buffer), 0);
83
84     cout<<"3. Enter server ID: ";
85     cin>>sid;
86     nonces = rand()%100;
87     itoc(powModN(sid, pue,nc), powModN(nonces, pue,nc));
88     send(sock, &buffer, sizeof(buffer), 0);
89     cout<<"sending plain sid|nonces "<<sid<<"|"<<nonces<<endl;
90     cout<<"sending encrypted sid|nonces "<<buffer<<endl;
91
92     recv(sock, &buffer, sizeof(buffer), 0);
93     ctoi(buffer, nonces1, noncec);
94     nonces1 = powModN(nonces1,prs,ns);
95     noncec= powModN(noncec, prs,ns);
96     cout<<"received encrypted nonces|noncec from client "<<buffer<<endl;
97     cout<<"received decrypted nonces|noncec from client "<<nonces1<<"|"  
<<noncec<<endl;
98     if(nonces!=nonces1)
99     {         cout<<"Nonce din't match"<<endl;         exit(0);         }
100     else     cout<<"Client Authenticated"<<endl;
101
102     itoc(powModN(noncec, pue, nc));
103     send(sock, &buffer, sizeof(buffer), 0);
104     cout<<"Sending plain noncec "<<noncec<<endl;
105     cout<<"Sending encrypted noncec "<<buffer<<endl;
106
107     cout<<"4. Enter the key: ";
108     cin>>key;
109     cout<<"Sending plain key "<<key<<endl;
110     key = powModN(key, prs, ns);
111     key = powModN(key, pue, nc);
112     itoc(key);
113     send(sock, &buffer, sizeof(buffer), 0);
114     cout<<"Sending encrypted key "<<buffer<<endl;
115
116 }
117
118 /*
119 received pue|n 7477|18281
120 3. Enter server (e|n): 4551|13231
121 4. Enter server (d): 1951
122 5. Enter server ID: 29
123 sending plain sid|nonces 29|28
124 sending encrypted sid|nonces 4168|11880

```

```
125 received encrypted nonces|noncec 6840|6840
126 received decrypted nonces|noncec 28|28
127 Sending plain noncec 28
128 Sending encrypted noncec 28
129 6. Enter the key: 454
130 Sending plain key 454
131 Sending encrypted key 5502
132 */
```



## client.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: Diffie Hellman client
3  Author: Gangadhara Shetty P J
4  */
5  # include <bits/stdc++.h>
6  # include <arpa/inet.h>
7  using namespace std;
8  char buffer[100];
9  int connectToServer(const char* ip, int port)
10 {
11     int sock = socket(AF_INET, SOCK_STREAM, 0);
12     struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
13     connect(sock, (struct sockaddr *) &addr, sizeof(addr));
14     return sock;
15 }
16 int powModN(int num, int p, int n)
17 {
18     int res=1;
19     for(int i=0; i<p; i++)
20         res = (res * num) % n;
21     return res;
22 }
23 void itoa(int x)
24 {
25     string s= to_string(x);
26     strcpy(buffer, s.c_str());
27 }
28 int main()
29 {
30     char ip[50]="127.0.0.1";
31     int port=1234;
32     int sock = connectToServer(ip, port);
33     int q, alpha, xa, ya, yb, cipher, key, message;
34
35     cout<<"1. Enter prime number and primitive root: ";
36     cin>>q>>alpha;
37
38     cout<<"2. Enter private key of client (<<q<<) : ";
39     cin>>xa;
40
41     ya =powModN(alpha, xa, q);
42
43     recv(sock, &buffer, sizeof(buffer), 0);
44     yb = atoi(buffer);
45
46     itoa(ya);
47     send(sock, &buffer, sizeof(buffer), 0);
48
49     cout<<"public key of client = "<<ya<<endl;
50     cout<<"received public key of server is : "<<yb<<endl;
51
52     key = powModN(yb, xa,q);
53     cout <<"secret key of client = "<< key<<endl;
54
55     cout<<"Enter a message: ";
56     cin>>message;
57     cipher = message ^ key;
58     cout<<"Encrypted message : "<<cipher<<endl;
59     itoa(cipher);
60     send(sock, &buffer, sizeof(buffer), 0);
61 }
```

```

62     recv(sock, &buffer, sizeof(buffer), 0);
63     cipher = atoi(buffer);
64     message = cipher ^ key;
65     cout<<"received encrypted message is: "<<cipher<<endl;
66     cout<<"received decrypted message is: "<<(cipher^key)<<endl;
67
68     return 0;
69 }
70
71 /*
72 1. Enter prime number and primitive root: 761 6
73 2. Enter private key of client (<761) : 100
74 public key of client = 399
75 received public key of server is : 152
76 secret key of client = 357
77 Enter a message: 76
78 Encrypted message :297
79 received encrypted message is: 325
80 received decrypted message is: 32
81 */

```

## server.cpp in gangadharashetty/pj/labs (master)

```
1  /*
2  Program: Diffie Hellman server
3  Author: Gangadhara Shetty P J
4  */
5  # include <bits/stdc++.h>
6  # include <arpa/inet.h>
7  using namespace std;
8  char buffer[100];
9  int createServer(int port)
10 {
11     int sersock = socket(AF_INET, SOCK_STREAM, 0);
12     struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
13     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
14     listen(sersock, 5);
15     int sock = accept(sersock, NULL, NULL);
16     return sock;
17 }
18 int powModN(int num,int p,int n)
19 {
20     int res=1;
21     for(int i=0; i<p; i++)
22         res = (res * num) % n;
23     return res;
24 }
25 void itoa(int x)
26 {
27     string s= to_string(x);
28     strcpy(buffer, s.c_str());
29 }
30 int main()
31 {
32     int port=1234, q, alpha, xb, ya, yb, cipher, key, message;
33     int sock = createServer(port);
34
35     cout<<"3. Enter prime number and primitive root: ";
36     cin>>q>>alpha;
37
38     cout<<"4. Enter private key server : ";
39     cin>>xb;
40
41     yb = powModN(alpha, xb,q);
42     itoa(yb);
43     send(sock, &buffer, sizeof(buffer), 0);
44
45     recv(sock, &buffer, sizeof(buffer), 0);
46     ya = atoi(buffer);
47
48     cout<<"public key of client = "<<ya<<endl;
49     cout<<"public key of server is : "<<yb<<endl;
50
51     key = powModN(ya, xb,q);
52     cout <<"secret key of server = "<< key<<endl;
53
54     recv(sock, &buffer, sizeof(buffer), 0);
55     message =atoi(buffer);
56     cipher = message^key;
57     cout<<"received encrypted message is: "<<message<<endl;
58     cout<<"received decrypted message is: "<<cipher<<endl;
59
60     cout<<"Enter a message: ";
61     cin>>message;
```

```
62     cipher = message ^ key;
63     cout<<"Encrypted message :"<<cipher<<endl;
64     itoa(cipher);
65     send(sock, &buffer, sizeof(buffer), 0);
66     return 0;
67 }
68
69 /*
70 3. Enter prime number and primitive root: 761 6
71 4. Enter private key server : 200
72 public key of client = 399
73 public key of server is : 152
74 secret key of server = 357
75 received encrypted message is: 297
76 received decrypted message is: 76
77 Enter a message: 32
78 Encrypted message :325
79 */
```