Practical No. 1

Aim: To demostrate classical cipher encrypt the given text message with the help of following algorithm.

1) Replace each alphabet with its equivalent 7-bit ASCII code
2) Swap first four bits with last four bits for each alphabets.
3) Write a hexadecimal equivalent for every four bit.

Theory: Cryptography is a method of protecting information & communication through the use of codes so that only those for when the information is intended can read & process it.

Cryptography refers to secure informatn & communication tech derived from mathematices concept & a set of rule-based calculatn called algorithm to transform message in ways that they are hard to deapher

Plain text: Information that can be directly read by human or a machine.

Cipher text: ~~Information that can be~~ the encrypted text

Encryption: The process of converting the plain text to cipher text.

Decryption: The process of reversing back ciphertext to plaintext.

Program

```
num = input (" Enter the string")
 ASCII values = [ord (characters) for character in num]
print (" Equivalent ascii value of string :"
                                ascii_values )
str1 = [] ; f = [] ; c = [], sw = [], s = []
  def decimal to binary (n);
   return bin(n). replace ("b", " ")
for i in num ;
   Num :- Ord (i)
    ch = decimal to binary (Num)
   str1. append (ch)
print (" Before swapping : "str1 )
for j in range (len (str1) : # → for reversing
                        can bin of character.
for i in str1 [i]
    e. insert (0,1)
    f. append (e)
    e = []
# print (" rev : *f)
     e = []
for i in range (len(f)): # swapping first 4 bit
  with last 4 bits of each character
   for j in f [i] ;
     if len(e) < 4 ;
      e. insert (0, j)
else ;
     e. insert (4, j)
     sw append (e)
      e = []
print ("After swapping :", sw)
  for i, in range (len (sw))
```

```
s. append c" join (sw [i],j)
# print (s)
    e= []
for i in range (len (s)):
  e. append (s[i] [4])
print (e)
def bin To Hexa (n)
      num - int (n,2)
    hex_num = hex (num)
    return (hex-num)
print ("Hexadecimal Equivalent :")
  for i in range (lence))
print (bin To Hexa (e [i]), end = " ")
```

Result :

The program to demonstrate classical cipher executed successfully.
Replacing each alphabet with its equivalent 7-bit ASCII code
Swapping first four bit with last four bit for each alphabet.
Obtaining a hexadecimal equivalent for every four bit is implemented successfully.

Practical No. 2

Aim: To demostrate substitution cipher

Theary: Ceasar Cipher

It is one of the simplest encryption technique in which each character in plain text is replaced by a character some fixed number of position down to it.

For ex. If key is 3 then we have a replace character by another character that is 3 position down to it like A will be replaced by D will be replaced by F & so on.

Cipher is refferred to as encryption & decryption algo. So, this can also be used to determine various cryptography algorithms It is always not necessary that the sender & receiver have their unique cipher for secured communication. As a result millions of communication can be served by a single cipher.

Process: In order to encrypt a plaintext letter, the sender position the sliding ruler underneath the first set of plaintext letters & slides it to RIGHT by the no. of positions of the secreat shift.

The plaintext letter is then encrypted to the ciphertexts letter on sliding ruler underneath the reseult of this process is depicated in the following illustration for an agreed shift of three position. In this case the PT is encrypted to the CT 'wxwruldo' Here is the ciphertext alphabet for a shift 3

Program :-

```
#include <stdio.h>
#include <conio.h>
int main()
{
  char message[100], ch;
  int i, key;
  printf("Enter a message to encrypt...");
  gets(message);
  printf("Enter key:");
  scanf("%d", &key);
  for(i=0; message[i] != '\0'; ++i)
    {
     ch = message[i];
    if(ch >= 'a' && ch <= 'z')
      {
       ch = ch + key;
        if(ch > 'z')
        ch = ch - 'z' + 'A' - 1;
      }
    message[i] = ch;
  }
  else if(ch >= 'A' && ch <= 'z') {
        ch = ch + key
          ch = ch - 'z' + 'A' - 1;
    }
      message[i] = ch;
  }
  }
printf("Inecrypted message: %s\n", message);
}
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	

X	Y	Z
A	B	C

On receiving the ciphertext, the receiver who also knows the secreat shift, position his sliding ruler underneath the ciphertext & slides it to "Right" by the agreed shift no. 3, in this case.

It is mono-alphabetic cipher where in each letter of the plaintext is substituted by another letter to form the ciphertext. It is simplest form of substitution cipher scheme.

It is kind of replacement (substituted) cipher where all letter of plain-text is replaced by another letter.

Result :

Thus, we demonstrate substitution cipher, successfully.

Practical No. 3

Aim: To demostrate transposition cipher write a program to implement columnar transposition cipher.

Theary: Columnar transposition cipher.

It is encryption method that swaps the columns of table containing the plain message to obtain encrypted message.

The columnar transposition cipher. It is transposition cipher that follows a simple rule for mixing up the character in the plaintext to form ciphertext.

although work on its own. It can be combined with other cipher, such as substitution cipher, the combination of which can be more difficult to break than cipher on its own.

Cryptoanalysis:

For a guide on how to automatically break columnar transposition cipher. The columnar transposition cipher is not the easiest of transposition cipher to break, but there are statistical cipher properties of language that can be exploited to recover, the key to greatly increase the security a substitution cipher caused the employed as well as the transposition.

Program :

```
#include <iostream>
using namespace std;
#include <map>
#include <string.h>
string const key: "HACK"
Map <int, int> keymap;
void setpermutation order()
{
  for (int i= 0; i<key : length(); i++)
{
  keymap [key [i]] = i;
}
} String encryptmessage (string msg)
{int row, col, j ;
   String cipher = " ";
   col = key length ();
   row = msg . length () /col;
    if  = msg length() % col
    row + = 1;
    char matrix [row] [col];
  for (int i =0 ; k=0 ; i< row ; i++)
{ for (int j=0 ; j < col ; )
{ if (msg [k] = = '\0')
{ matrix [i][j] = "-' ;
  j ++ }
  if (is alpha (msg[k] ||msg [k]==' ')
{
   matrix [i][j] = msg [k];
    j++ ; }
    k ++ ; }
} for (map <int, int> :: iterates ii = key map . begin()
;; ; = keymap. end (); ++ii )
   { j = ii -> second
```

```
for(int i=0 ; i<row ; i++)
{ if (isalpha(matrix[i][j]) (matrix[i][j]==' ' ||
    matrix[i][j] == '_')
  cipher+ = matrix[i][j];
}
} return cipher;
} string decryptmessage (string cipher)
{ int col = key.length();
  int row = cipher.length()/col;
for(int j=0 ; k=0; j<col ; j++)
 for (int i=0 ; i<row ; i++)
  ciphermat[i][j] = cipher[k++];
  int index = 0
for (map<int, int>. iterater ii = keymap. begain()
 ii != keymap.end(); ++ii)
 ii → second = index++;
 char decipher[row][col];
 map <int, int>::iterater :, =keymap.begin();
  int k=0;
for (int l=0, j; ~~key~~ key[l] = '\0' ; k++)
   { j = keymap[key[l++]];
   for (int i=0 ; i<row; i++)
 { decipher[i][k] = ciphertext[i][j];
 }
  string msg = " ";
for (int i=0; i<row; i++)
{
  for (int j=0, j<col ; j++)
  {
   if (decipher[i][j] : = '_')
    msg+ = decipher[i][j];
 }
 } return msg;
}
```

```
int main(void)
{ string msg = "          ";
setpermutation order();
string cipher = encryptmessage(msg);
Count << "\n encrypted message:" << cipher << ends;
Count << "Decrypted message:" <<
        decryp - messag(cipher << ends;
  return 0;
}
```

Result: Hence, the columnar transposition cipher executed successfully.

Practical No. 4

Aim: To demonstrate symmetric secret key cipher in lab. W.A.P to implement DES (Data Encryption Standard).

Theary:

Data Encryption standard (DES) is a block cipher algorithm that takes plain text in blocks of 64 bits & converts them to cipher text using key of 48 bits. It is symmetric key algorithm, which means that the same key is used for ~~every~~ encrypting & decryption data.

DES is actually based on two fundamental concepts of encryptography substitution & transposition. It consist of 16 steps called as a round. Each round is responsible for performing the steps of substitution and transposition.

Step 1 -

The 64 bit PLAINTEXT message is given to the initial permutation (IP)

Step 2 -

Initial permutation (IP) divides 64-bit PLAINTEXT into two holoes; say left PLAINTEXT (LPT) & Right- R PLAINTEXT (RPT)

Step 3 -

Now each LPT & RPT go through 16 round of encryption processes.

Step 4 -

In the end, LPT & RPT are re-joined and a final permutation (FP) is performed on the combined 64 bit block

Step 5 -

The result of the process produces 64 bit CIPHERTEXT.

Program 11:

* The code for Encryption is Mentioned Below.

```
import base23hex
import hashlib
for Cryto.cipher import DES.
password = "Password"
salt = '\x28\xAB\xBC\xCD\xDE\xEF\x00\x33'
key = password + salt
m = hashlib.md5(key)
key = m.digest()
(dk, iv) = (key[:8], key[8:])
crypter = DES.new(dk, DES.MODE_CBC, iv)

plain_text = "I see you"

print("The plain text is:", plain_text)
plain_text += '\x00' * (8 - len(plain_text) % 8)
ciphertext = crypter.encrypt(plain_text)
encode_string = base32hex.b32encode(ciphertext)
print("The encoded string is:", encode_string).
```

* The code for Decryption is Mentioned Below.

```
import base32hex
import hashlib
for Crypto.cipher import DES
password = "Password"
salt = '\x28\xAB\xBC\xCD\xDE\xEF\x00\x33'
key = password + salt
m = hashlib.md5(key)
key = m.digest()
(dk, iv) = (key[:8], key[8:])
crypter = DES.new(dk, DES.MODE_CBC, iv).
```

```
encrypted_string = 'UH562EGM8RCHHTOUCSCTRS59OG==='
print (" The ecrypted string is : ", encrypted_string)
encrypted_string = base32hex.b32decode (encrypted_string)
decrypted_string = crypter.decrypt(encrypted_string)
print(" The decrypted string is : ", decrypted_string).
```

Practical No. 5

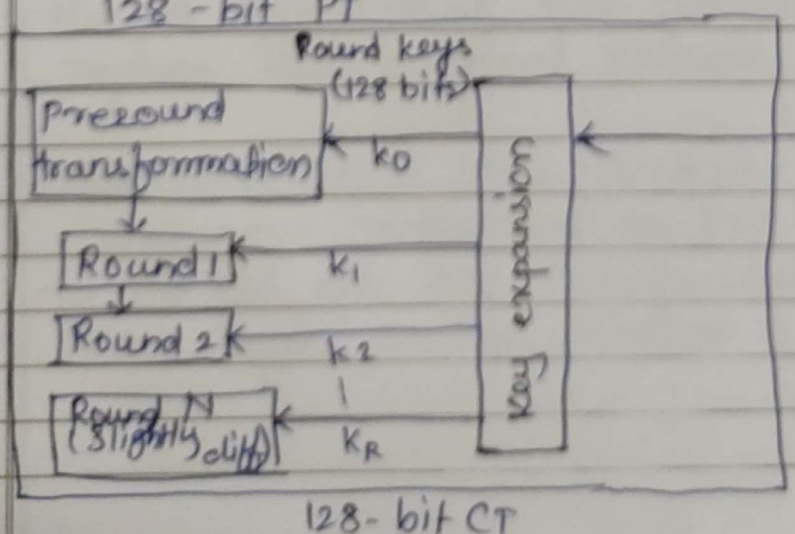
Aim: To demostrate block cipher technique (Writ a program to implement AES)

Theary: AES is advance encryption standard the key size is 128/192/256 bits

- Encrypts data in blocks of 128 bits each.
- That means it takes 128 bits as input and output 128 bits encrypted CT as output.
- AES relies on substitution permutation network principle which means it is performed using a service of linked operation which involves replacing & shuffling of the input data.
- AES performed all its computations on bytes rather that bits Hence, AES treats the 128 bit of a PT block as 16 bytes these 16 bytes are arranged in four columns & four rows for processing as a matrix.

The round in AES is variable & depends on the length of the keys, 12 rounds for 192-bit keys & 14-rounds uses a different 128-bits round key. which is calculated from the original AES key.

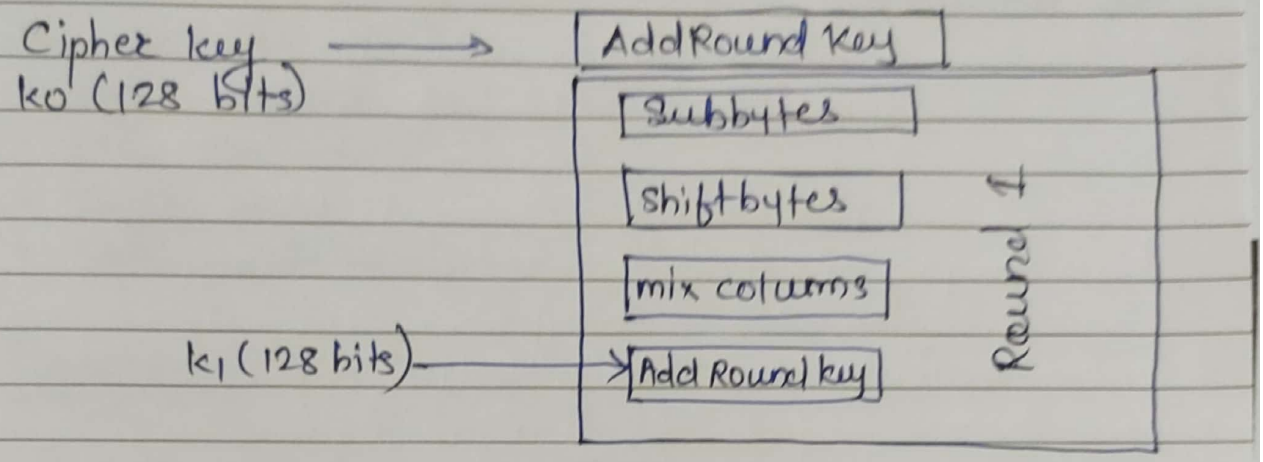
AES Structure :



R	keysize
10	128
12	192
14	256

Relationship betn no. of round & key size.

each round comprises of four sub-process the first round process is depicted below-



1) Byte substitution : The 16 input bytes are substitution by looking up a fixed table (s-box) given in design. the result is in a matrix of four rows & four column.

2) Shiftrows : Each of the four rows of the matrix is shifted to the left any entries that 'fall off' are re-inserted on the right side of row shift is carried out as follows-

3) Mixcolumns : Each column of four bytes is now transformed. Using a special mathematical function. This function takes as input the 4 bytes that are completly new & they replace the original column.

4) AddRoundkey : The 16-bytes of the matrix are now considered as 128 bits & are XORed to the 128 bits are interpreted as 16 bytes & we begin another similar round.

Decryption Process : It is same as the encryptn process but in reverse order each round consists of the four processes conducted in the reverse order.

- Add round key
- Mix columns
- Shift rows
- Byte substitution.

Program :

```
import
java.nio.charset.StandardCharsets;
import java.security.spec.KeySpec;
import java.util.crypto.Cipher;
import javax.util.Base64;
import javax.crypto.SecretKey;
import
javax.crypto.SecretKeyFactory;
import
javax.crypto.spec.IvParameterSpec;
import
javax.crypto.spec.PBEKeySpec;
import
javax.crypto.spec.SecretKeySpec;
class AFS {
SECRET-KEY = "MY-super-secret_key_ho_ho_ho";
Private static final String SALT = "ssshhhhhn!!!.:";

public static string encrypt (string strTOEncrypt)
{
try {
byte[] iv = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
```

```
IvParameterSpec ivspec = new IvParameterSpec(iv);
SecretKeyFactory factory = SecretKeyFactory.
  getInstance("PBKDF2withHmacSHA256");
  KeySpec spec = new PBEKeySpec(
      SECRET_key.toCharArray(),
      SALT.getBytes()
       65536, 256);
    SecretKey temp =
      factory.generateSecret(spec);
      SecretKeySpec secretKey = new
      SecretKeySpec(
        tmp.getEncoded(), "AES");

Cipher cipher = Cipher.getInstance(
"AES/CBC/PKCS5Padding");
cipher.init(Cipher.ENCREPT_MODE, secretKey,
                                 ivspec());
return Base64.getEncoder().encodeTo
  String(
cipher.doFinal(strToEncrypt.getBytes
  (
    StandardCharsets.UTF_8)));
  }
    catch(Exception e) {
System.out.println("Error while encrypting:"
  + e.toString()};
}
return null;
}
Public static String decrypt(String strToDecrypt)
{
  try {
```

```
byte [] iv = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
IvParameterSpec ivspec = new IvParameterSpec
                                (iv);
SecretKeyFactory factory = SecretKeyFactory.
        getInstance ("PBKDF2 withHMac{
        HA256");
KeySpec spec = new PBEKeySpec (
        SECRET-KEY.toCharArray();
        SALT.getBytes(),
        65536, 256);
    SecretKey tmp = factory.generateSecret
                        (spec);
    SecretKeySpec SecretKey = new secretKeySpec (
    tmp.getEncoded(), "AES");
Cipher cipher = Cipher.getInstance (
    "AES/CBC/PKCS5PADDING");
  cipher.intt (cipher.DECRYPT-MODE,
      SecretKey, ivspec);
return new String (cipher.doFinal (Base64.
      getDecoder().decode (strToDecrypt)));
}
  catch (Exception e) {
  system.out.println ("Error while decrypting:
      " + e.toString());
}
  return null;
}}
  public class main {
  public static void main (String [] args)
  {
  String originalString = "Pooja";
  String encryptedString = AES.encrypt (original
  System.out.println ("plain Text:");          String)"
  System.out.println (originalString);
```

```
System.out.println("Encrypted Text");
System.out.println(encryptedstring);
System.out.println("Decrypted Text:");
System.out.println(decryptedstring);
}
}
```

Result: Thus, the AES program is implemented successfully.