AJK COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS) AN ISO 21001 : 2018 CERTIFIED INSTITUTION

Affiliated to Bharathiar University, Coimbatore. Approved by Govt. of Tamil Nadu
Recognized by UGC, New Delhi Under Section 2(f) and 12(B)

Palakkad Main Road, Navakkarai, Coimbatore, Tamil Nadu - 641105. Ph: 0422 - 3501700

www.ajkcas.com /ajkcollege /ajkinstitutions

ACCREDITED WITH GRADE A+ NAAC

DEPARTMENT OF DIGITAL & CYBER FORENSIC SCIENCE

Association of Digital & Cyber Forensic Science

DCFS

CYBERSE

LAB MANUAL

NAME OF THE SUBJECT	**:** Cryptography and Network Security Lab
SUBJECT CODE	**:** 63P
SEMESTER	**:** VI
CLASS	**:** III B.Sc., D&CFS

Prepared By **Approved By**

Subject Name	**:**	CRYPTOGRAPHY AND NETWORK SECURITY LAB
Department	**:**	DIGITAL AND CYBER FORENSIC SCIENCE
Class	**:**	III B.SC D&CFS
Semester	**:**	VI

Note : All programs are written with procedures with respect to Linux system especially Ubuntu 22.10 / Ubuntu 22.04 LTS.

Language to be used : C/C++

EXERCISE I

1. Perform encryption, decryption using the following substitution techniques

(i) Ceaser cipher,

(ii) playfair cipher

iii) Hill Cipher

iv) Vigenere cipher

EXERCISE II

2. Perform encryption and decryption using following transposition techniques

i) Rail fence

ii) row & Column Transformation

EXERCISE III

3. Apply DES algorithm for practical applications.

EXERCISE IV

4. Apply AES algorithm for practical applications.

EXERCISE V

5. Implement RSA Algorithm using HTML and JavaScript

EXERCISE VI

6. Implement the Diffie-Hellman Key Exchange algorithm for a given problem.

Prepared by **Verified by**

EXERCISE I

AIM :

To Perform encryption, decryption using the following substitution techniques

i) Ceaser cipher,

ii) Playfair cipher,

iii) Hill Cipher,

iv) Vigenere cipher.

ALGORITHM :

Step – 1 : Start the Process.

Step – 2 : Open terminal and create files with the cipher name, like *ceasar.cpp, playfair.cpp, hill.cpp, vigenere.cpp*.

Step – 3 : Create the files with the touch command :

touch ceasar.cpp,

Step – 4 : Edit the files with nano CLI editor.

Step – 5 : Import the header file iostream.

Step – 6 : define std namespace as default usage statements.

Step – 7 : For ceasar cipher, define encryption function with a for loop and if statement for conditions of shifting, and call the string encrypt function in the main() function declared with int.

Step – 8 : For playfair cipher, include bits/stdc++.h header file.

Step – 9 : Define void function toLowerCase function to convert all letters to lower case, int removeSpaces function to remove spaces in between the input string if any, void generateKeyTable with 3 for loops, void search along with int mod5, then finally void encrypt, encryptByPlayfairCipher, int main functions to complete the code.

Step – 10 : For Hill Cipher, include iostream and use namespace to be std, start with defining void generate-KeyMatrix.

Step – 11 : Then Define void encrypt with 3 variables and 3 for loops and HillCipher function with encryption code, and int main function to give key and main plain text for ciphering.

Step – 12 : For Vigenere cipher, include bits/stdc++.h header and define the following functions – string gen-erateKey, string cipherText, string originalText and int main to give plain text, key and cipher output.

Step – 13 : Run the gcc compiler for compiling the cpp files, use the command :

gcc –o ceasar ceasar.cpp –lstdc++.

gcc –o playfair playfair.cpp –lstdc++.

gcc –o hill hill.cpp –lstdc++.

gcc –o vigenere vigenere.cpp –lstdc++.

Step – 14 : run the file with the following command : ./ceasar, ./playfair, ./hill, ./vigenere.

Step – 15 : Stop the Process.

CODING :

CEASER CIPHER IN C++

```
#include <iostream>
using namespace std;
string encrypt(string text, int s)
{
	string result = "";
	for (int i = 0; i < text.length(); i++) {
		if (isupper(text[i]))
			result += char(int(text[i] + s - 65) % 26 + 65);
		else
			result += char(int(text[i] + s - 97) % 26 + 97);
	}
	return result;
}
int main()
{
	string text = "This_is_a_sample_encryption";
	int s = 4;
	cout << "Text : " << text;
	cout << "\nShift: " << s;
	cout << "\nCipher: " << encrypt(text, s);
	cout << "\n";
	return 0;
}
```

PLAYFAIR CIPHER IN C++

```
#include <bits/stdc++.h>
using namespace std;
#define SIZE 30
void toLowerCase(char plain[], int ps)
{
	int i;
	for (i = 0; i < ps; i++) {
		if (plain[i] > 64 && plain[i] < 91)
```

```
                plain[i] += 32;
        }
}
int removeSpaces(char* plain, int ps)
{
        int i, count = 0;
        for (i = 0; i < ps; i++)
                if (plain[i] != ' ')
                        plain[count++] = plain[i];
        plain[count] = '\0';
        return count;
}
void generateKeyTable(char key[], int ks, char keyT[5][5])
{
        int i, j, k, flag = 0;
        int dicty[26] = { 0 };
        for (i = 0; i < ks; i++) {
                if (key[i] != 'j')
                        dicty[key[i] - 97] = 2;
        }
        dicty['j' - 97] = 1;
        i = 0;
        j = 0;
        for (k = 0; k < ks; k++) {
                if (dicty[key[k] - 97] == 2) {
                        dicty[key[k] - 97] -= 1;
                        keyT[i][j] = key[k];
                        j++;
                        if (j == 5) {
                                i++;
                                j = 0;
                        }
                }
        }
```

```
        for (k = 0; k < 26; k++) {
                if (dicty[k] == 0) {
                        keyT[i][j] = (char)(k + 97);
                        j++;
                        if (j == 5) {
                                i++;
                                j = 0;
                        }
                }
        }
}

void search(char keyT[5][5], char a, char b, int arr[])
{
        int i, j;
        if (a == 'j')
                a = 'i';
        else if (b == 'j')
                b = 'i';
        for (i = 0; i < 5; i++) {
                for (j = 0; j < 5; j++) {
                        if (keyT[i][j] == a) {
                                arr[0] = i;
                                arr[1] = j;
                        }
                        else if (keyT[i][j] == b) {
                                arr[2] = i;
                                arr[3] = j;
                        }
                }
        }
}
int mod5(int a) { return (a % 5); }
int prepare(char str[], int ptrs)
```

```
{
	if (ptrs % 2 != 0) {
		str[ptrs++] = 'z';
		str[ptrs] = '\0';
	}
	return ptrs;
}
void encrypt(char str[], char keyT[5][5], int ps)
{
	int i, a[4];
	for (i = 0; i < ps; i += 2) {
		search(keyT, str[i], str[i + 1], a);
		if (a[0] == a[2]) {
			str[i] = keyT[a[0]][mod5(a[1] + 1)];
			str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
		}
		else if (a[1] == a[3]) {
			str[i] = keyT[mod5(a[0] + 1)][a[1]];
			str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
		}
		else {
			str[i] = keyT[a[0]][a[3]];
			str[i + 1] = keyT[a[2]][a[1]];
		}
	}
}
void encryptByPlayfairCipher(char str[], char key[])
{
	char ps, ks, keyT[5][5];
	ks = strlen(key);
	ks = removeSpaces(key, ks);
	toLowerCase(key, ks);
	ps = strlen(str);
	toLowerCase(str, ps);
```

```
        ps = removeSpaces(str, ps);
        ps = prepare(str, ps);
        generateKeyTable(key, ks, keyT);
        encrypt(str, keyT, ps);
}
int main()
{
        char str[SIZE], key[SIZE];
        strcpy(key, "cybercops");
        cout << "Key text: " << key << "\n";
        strcpy(str, "cyberforensicscience");
        cout << "Plain text: " << str << "\n";
        encryptByPlayfairCipher(str, key);
        cout << "Cipher text: " << str << "\n";
        return 0;
}
```

HILL CIPHER IN C++

```
#include <iostream>
using namespace std;
void getKeyMatrix(string key, int keyMatrix[][3])
{
        int k = 0;
        for (int i = 0; i < 3; i++)
        {
                for (int j = 0; j < 3; j++)
                {
                        keyMatrix[i][j] = (key[k]) % 65;
                        k++;
                }
        }
}
void encrypt(int cipherMatrix[][1],
                        int keyMatrix[][3],
                        int messageVector[][1])
```

```
{
        int x, i, j;
        for (i = 0; i < 3; i++)
        {
                for (j = 0; j < 1; j++)
                {
                        cipherMatrix[i][j] = 0;
                        for (x = 0; x < 3; x++)
                        {
                                cipherMatrix[i][j] +=
                                        keyMatrix[i][x] * messageVector[x][j];
                        }
                        cipherMatrix[i][j] = cipherMatrix[i][j] % 26;
                }
        }
}
void HillCipher(string message, string key)
{
        int keyMatrix[3][3];
        getKeyMatrix(key, keyMatrix);
        int messageVector[3][1];
        for (int i = 0; i < 3; i++)
                messageVector[i][0] = (message[i]) % 65;
        int cipherMatrix[3][1];
        encrypt(cipherMatrix, keyMatrix, messageVector);
        string CipherText;
        for (int i = 0; i < 3; i++)
                CipherText += cipherMatrix[i][0] + 65;
        cout << " Ciphertext:" << CipherText;
}
int main()
{
        string message = "ACT";
        string key = "GYBNQKURP";
```

```
	HillCipher(message, key);
	return 0;
}
```

VIGENERE CIPHER IN C++

```
#include<bits/stdc++.h>
using namespace std;
string generateKey(string str, string key)
{
	int x = str.size();
	for (int i = 0; ; i++)
	{
		if (x == i)
			i = 0;
		if (key.size() == str.size())
			break;
		key.push_back(key[i]);
	}
	return key;
}
string cipherText(string str, string key)
{
	string cipher_text;
	for (int i = 0; i < str.size(); i++)
	{
		char x = (str[i] + key[i]) %26;
		x += 'A';
		cipher_text.push_back(x);
	}
	return cipher_text;
}
string originalText(string cipher_text, string key)
{
	string orig_text;
```

```
        for (int i = 0 ; i < cipher_text.size(); i++)
        {
                char x = (cipher_text[i] - key[i] + 26) %26;
                x += 'A';
                orig_text.push_back(x);
        }
        return orig_text;
}
int main()
{
        string str = "GEEKSFORGEEKS";
        string keyword = "AYUSH";
        string key = generateKey(str, keyword);
        string cipher_text = cipherText(str, key);
        cout << "Ciphertext : "
                << cipher_text << "\n";

        cout << "Original/Decrypted Text : "
                << originalText(cipher_text, key);
        return 0;
}
```

OUTPUT :

CEASER CIPHER IN C++

Text : This_is_a_sample_encryption

Shift: 4

Cipher: Xlmwcmwcecweqtpicirgvctxmsr

PLAYFAIR CIPHER IN C++

Key text: cybercops

Plain text: cyberforensicscience

Cipher text: yberckdcbqahboefbqyr

HILL CIPHER IN C++

Ciphertext:POH

VIGENERE CIPHER IN C++

Ciphertext : GCYCZFMLYLEIM

Original/Decrypted Text : HELLOFORDCFS

RESULT : Implementation of ceasar cipher, playfair cipher, hill cipher, vigenere cipher is successfully completed.

EXERCISE II

2. Perform encryption and decryption using following transposition techniques

i) Rail fence ii) row & Column Transformation

AIM :

To Perform encryption and decryption using following transposition techniques

i) Rail fence

ii) row & Column Transformation

ALGORITHM :

Step – 1 : Start the Process.

Step – 2 : Open terminal and create files with the cipher name, like railfence.py, rowtrans.py

Step – 3 : Create the files with the touch command :

touch railfence.py

touch rowtrans.py

Step – 4 : Edit the files with nano CLI editor.

Step – 5 : To open the file type the editor along with file name & extension: *nano railfence.py*

Step – 6 : define 2 functions namely encryptRailFence & decryptRailFence with 2 variables each.

Step – 7 : For encryptRailFence, define 2 variables and use 2 for loops o create the matrix.

Step – 8 : Define dir_down variable to be false and row, col to be 0, 0, and 2 for loops to assign the plain text into the matrix and perform the cipher.

Step – 9 : For decryptRailFence, define 2 variables and use 2 for loops o create decryption matrix.

Step – 10 : Define dir_down variable to be None, additionally index to be also 0 and row, col to be 0, 0, and 2 for loops to assign the cipher text into the matrix and perform the deciphering, by conditionally reordering.

Step – 11 : Finally define encryption and decryption condition inside a __name__==__main__ functional condition.

Step – 12 : For row transformation, import a math library and use Key as “HACK”.

Step – 13 : Define encryptMessage function with msg as a variable and decryptMessage function with cipher as a variable, and use appropriate variables with floating conditions and matrix functions of the math library in both the functions and return with cipher and return with msg respectively, finally complete the code by calling the functions and print the strings.

Step – 14 : Run the python compiler for compiling the py files, use the command :

Python railfence.py.

Python rowtrans.py.

Step – 15 : Stop the Process.

CODING :

Rail fence

```
def encryptRailFence(text, key):
	rail = [['\n' for i in range(len(text))]
				for j in range(key)]
	dir_down = False
	row, col = 0, 0
	for i in range(len(text)):
		if (row == 0) or (row == key - 1):
			dir_down = not dir_down
		rail[row][col] = text[i]
		col += 1
		if dir_down:
			row += 1
		else:
			row -= 1
	result = []
	for i in range(key):
		for j in range(len(text)):
			if rail[i][j] != '\n':
				result.append(rail[i][j])
	return("" . join(result))
def decryptRailFence(cipher, key):
	rail = [['\n' for i in range(len(cipher))]
				for j in range(key)]
	dir_down = None
	row, col = 0, 0
	for i in range(len(cipher)):
		if row == 0:
			dir_down = True
		if row == key - 1:
			dir_down = False
		rail[row][col] = '*'
		col += 1
```

```
            if dir_down:
                row += 1
            else:
                row -= 1
        index = 0
        for i in range(key):
            for j in range(len(cipher)):
                if ((rail[i][j] == '*') and
                (index < len(cipher))):
                    rail[i][j] = cipher[index]
                    index += 1
        result = []
        row, col = 0, 0
        for i in range(len(cipher)):
            if row == 0:
                dir_down = True
            if row == key-1:
                dir_down = False
            if (rail[row][col] != '*'):
                result.append(rail[row][col])
                col += 1
            if dir_down:
                row += 1
            else:
                row -= 1
        return("".join(result))
if __name__ == "__main__":
        print(encryptRailFence("attack at once", 2))
        print(encryptRailFence("hellodcfs ", 3))
        print(encryptRailFence("defend the east wall", 3))
        print(decryptRailFence("somerandomtext", 3))
        print(decryptRailFence("atc toctaka ne", 2))
        print(decryptRailFence("dnhaweedtees alf tl", 3))
```

Column Transformation

```
import math
key = "HACK"
def encryptMessage(msg):
        cipher = ""
        k_indx = 0
        msg_len = float(len(msg))
        msg_lst = list(msg)
        key_lst = sorted(list(key))
        col = len(key)
        row = int(math.ceil(msg_len / col))
        fill_null = int((row * col) - msg_len)
        msg_lst.extend('_' * fill_null)
        matrix = [msg_lst[i: i + col]
                        for i in range(0, len(msg_lst), col)]
        for _ in range(col):
                curr_idx = key.index(key_lst[k_indx])
                cipher += ''.join([row[curr_idx]
                                                for row in matrix])
                k_indx += 1
        return cipher
def decryptMessage(cipher):
        msg = ""
        k_indx = 0
        msg_indx = 0
        msg_len = float(len(cipher))
        msg_lst = list(cipher)
        col = len(key)
        row = int(math.ceil(msg_len / col))
        key_lst = sorted(list(key))
        dec_cipher = []
        for _ in range(row):
                dec_cipher += [[None] * col]
        for _ in range(col):
```

```
            curr_idx = key.index(key_lst[k_indx])
            for j in range(row):
                dec_cipher[j][curr_idx] = msg_lst[msg_indx]
                msg_indx += 1
            k_indx += 1
    try:
        msg = ''.join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot",
                                    "handle repeating words.")
    null_count = msg.count('_')
    if null_count > 0:
        return msg[: -null_count]
    return msg
msg = "hello students of dcfs"
cipher = encryptMessage(msg)
print("Encrypted Message: {}".
                format(cipher))
print("Decryped Message: {}".
    format(decryptMessage(cipher)))
```

OUTPUT :

Rails Fence

```
atc toctaka ne
hoseldf lc
dnhaweedtees alf  tl
sreaonxdmotmet
attack at once
delendfthe east wal
```

Column Transposition

```
Encrypted Message: e ds slse d_houtffltnoc_
Decryped Message: hello students of dcfs
```

RESULT : Implementation of railfence cipher, column transposition cipher, is successfully completed.

EXERCISE III

3. Apply DES algorithm for practical applications.

AIM :

To write DES algorithm for practical applications in python.

ALGORITHM :

Step – 1 : Start the Process.

Step – 2 : Open terminal and create files with the cipher name, like DES.py

Step – 3 : Create the files with the touch command :

touch DES.py

Step – 4 : Edit the files with nano CLI editor.

Step – 5 : To open the file type the editor along with file name & extension: *nano DES.py*

Step – 6 : Start with defining a function for hexadecimal to binary array, then a for loop for searching cycle through it, and similarly for binary to hexadecimal conversion array also, Similarly also create functions for decimal to binary and binary to decimal too with for looping statements.

Step – 7 : Define the following functions: permutation function, shiftkey function, XOR operative function.

Step – 8 : Define the following arrays: Initial permutation, Exp_d, Permut, S-Box, final permutation.

Step – 9 : Define the Encryption function with operations of

i) Dividing plain text,

ii) Adding Initial permutation,

iii) Shifting Rows,

iv) Mix Columns,

v) Adding Round Key,

vi) Continuing for 16 rounds,

vii) Finally performing final permutation.

Step – 10 : Run the python compiler for compiling the py files, use the command :

Python DES.py.

Step – 11 : Stop the Process.

CODING :

```
def hex2bin(s):
        mp = {'0': "0000",
              '1': "0001",
              '2': "0010",
              '3': "0011",
              '4': "0100",
              '5': "0101",
              '6': "0110",
              '7': "0111",
              '8': "1000",
              '9': "1001",
              'A': "1010",
              'B': "1011",
              'C': "1100",
              'D': "1101",
              'E': "1110",
              'F': "1111"}
        bin = ""
        for i in range(len(s)):
              bin = bin + mp[s[i]]
        return bin
def bin2hex(s):
        mp = {"0000": '0',
              "0001": '1',
              "0010": '2',
              "0011": '3',
              "0100": '4',
              "0101": '5',
              "0110": '6',
              "0111": '7',
              "1000": '8',
              "1001": '9',
              "1010": 'A',
```

```
                "1011": 'B',
                "1100": 'C',
                "1101": 'D',
                "1110": 'E',
                "1111": 'F'}
        hex = ""
        for i in range(0, len(s), 4):
                ch = ""
                ch = ch + s[i]
                ch = ch + s[i + 1]
                ch = ch + s[i + 2]
                ch = ch + s[i + 3]
                hex = hex + mp[ch]
        return hex
def bin2dec(binary):
        binary1 = binary
        decimal, i, n = 0, 0, 0
        while(binary != 0):
                dec = binary % 10
                decimal = decimal + dec * pow(2, i)
                binary = binary//10
                i += 1
        return decimal
def dec2bin(num):
        res = bin(num).replace("0b", "")
        if(len(res) % 4 != 0):
                div = len(res) / 4
                div = int(div)
                counter = (4 * (div + 1)) - len(res)
                for i in range(0, counter):
                        res = '0' + res
        return res
def permute(k, arr, n):
        permutation = ""
```

```
    for i in range(0, n):
        permutation = permutation + k[arr[i] - 1]
    return permutation
def shift_left(k, nth_shifts):
    s = ""
    for i in range(nth_shifts):
        for j in range(1, len(k)):
            s = s + k[j]
        s = s + k[0]
        k = s
        s = ""
    return k
def xor(a, b):
    ans = ""
    for i in range(len(a)):
        if a[i] == b[i]:
            ans = ans + "0"
        else:
            ans = ans + "1"
    return ans
initial_perm = [58, 50, 42, 34, 26, 18, 10, 2,
                60, 52, 44, 36, 28, 20, 12, 4,
                62, 54, 46, 38, 30, 22, 14, 6,
                64, 56, 48, 40, 32, 24, 16, 8,
                57, 49, 41, 33, 25, 17, 9, 1,
                59, 51, 43, 35, 27, 19, 11, 3,
                61, 53, 45, 37, 29, 21, 13, 5,
                63, 55, 47, 39, 31, 23, 15, 7]
exp_d = [32, 1, 2, 3, 4, 5, 4, 5,
        6, 7, 8, 9, 8, 9, 10, 11,
        12, 13, 12, 13, 14, 15, 16, 17,
        16, 17, 18, 19, 20, 21, 20, 21,
        22, 23, 24, 25, 24, 25, 26, 27,
        28, 29, 28, 29, 30, 31, 32, 1]
```

```
per = [16, 7, 20, 21,
       29, 12, 28, 17,
       1, 15, 23, 26,
       5, 18, 31, 10,
       2, 8, 24, 14,
       32, 27, 3, 9,
       19, 13, 30, 6,
       22, 11, 4, 25]
sbox = [[[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],
         [0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],
         [4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],
         [15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]],

         [[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],
         [3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],
         [0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],
         [13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9]],

         [[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],
         [13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],
         [13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],
         [1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12]],

         [[7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],
         [13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],
         [10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],
         [3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14]],

         [[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],
         [14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],
         [4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],
         [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3]],

         [[12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
```

```
                [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
                [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
                [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13]],

                [[4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
                [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
                [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
                [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12]],

                [[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],
                [1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],
                [7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],
                [2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11]]]
final_perm = [40, 8, 48, 16, 56, 24, 64, 32,
                        39, 7, 47, 15, 55, 23, 63, 31,
                        38, 6, 46, 14, 54, 22, 62, 30,
                        37, 5, 45, 13, 53, 21, 61, 29,
                        36, 4, 44, 12, 52, 20, 60, 28,
                        35, 3, 43, 11, 51, 19, 59, 27,
                        34, 2, 42, 10, 50, 18, 58, 26,
                        33, 1, 41, 9, 49, 17, 57, 25]
def encrypt(pt, rkb, rk):
        pt = hex2bin(pt)
        pt = permute(pt, initial_perm, 64)
        print("After initial permutation", bin2hex(pt))
        left = pt[0:32]
        right = pt[32:64]
        for i in range(0, 16):
                right_expanded = permute(right, exp_d, 48)
                xor_x = xor(right_expanded, rkb[i])
                sbox_str = ""
                for j in range(0, 8):
                        row = bin2dec(int(xor_x[j * 6] + xor_x[j * 6 + 5]))
                        col = bin2dec(
```

```
                        int(xor_x[j * 6 + 1] + xor_x[j * 6 + 2] + xor_x[j * 6 + 3] + xor_x[j * 6 + 4]))
                    val = sbox[j][row][col]
                    sbox_str = sbox_str + dec2bin(val)
            sbox_str = permute(sbox_str, per, 32)
            result = xor(left, sbox_str)
            left = result
            if(i != 15):
                left, right = right, left
            print("Round ", i + 1, " ", bin2hex(left),
                " ", bin2hex(right), " ", rk[i])
    combine = left + right
    cipher_text = permute(combine, final_perm, 64)
    return cipher_text
pt = "123456ABCD132536"
key = "AABB09182736CCDD"
key = hex2bin(key)
keyp = [57, 49, 41, 33, 25, 17, 9,
            1, 58, 50, 42, 34, 26, 18,
            10, 2, 59, 51, 43, 35, 27,
            19, 11, 3, 60, 52, 44, 36,
            63, 55, 47, 39, 31, 23, 15,
            7, 62, 54, 46, 38, 30, 22,
            14, 6, 61, 53, 45, 37, 29,
            21, 13, 5, 28, 20, 12, 4]
key = permute(key, keyp, 56)
shift_table = [1, 1, 2, 2,
                    2, 2, 2, 2,
                    1, 2, 2, 2,
                    2, 2, 2, 1]
key_comp = [14, 17, 11, 24, 1, 5,
                    3, 28, 15, 6, 21, 10,
                    23, 19, 12, 4, 26, 8,
                    16, 7, 27, 20, 13, 2,
                    41, 52, 31, 37, 47, 55,
```

```
                    30, 40, 51, 45, 33, 48,
                    44, 49, 39, 56, 34, 53,
                    46, 42, 50, 36, 29, 32]
left = key[0:28]
right = key[28:56]
rkb = []
rk = []
for i in range(0, 16):
        left = shift_left(left, shift_table[i])
        right = shift_left(right, shift_table[i])
        combine_str = left + right
        round_key = permute(combine_str, key_comp, 48)
        rkb.append(round_key)
        rk.append(bin2hex(round_key))
print("Encryption")
cipher_text = bin2hex(encrypt(pt, rkb, rk))
print("Cipher Text : ", cipher_text)
print("Decryption")
rkb_rev = rkb[::-1]
rk_rev = rk[::-1]
text = bin2hex(encrypt(cipher_text, rkb_rev, rk_rev))
print("Plain Text : ", text)
```

OUTPUT:

```
Encryption
After initial permutation 14A7D67818CA18AD
Round  1   18CA18AD   5A78E394   194CD072DE8C
Round  2   5A78E394   4A1210F6   4568581ABCCE
Round  3   4A1210F6   B8089591   06EDA4ACF5B5
Round  4   B8089591   236779C2   DA2D032B6EE3
Round  5   236779C2   A15A4B87   69A629FEC913
Round  6   A15A4B87   2E8F9C65   C1948E87475E
Round  7   2E8F9C65   A9FC20A3   708AD2DDB3C0
Round  8   A9FC20A3   308BEE97   34F822F0C66D
```

Round 9 308BEE97 10AF9D37 84BB4473DCCC

Round 10 10AF9D37 6CA6CB20 02765708B5BF

Round 11 6CA6CB20 FF3C485F 6D5560AF7CA5

Round 12 FF3C485F 22A5963B C2C1E96A4BF3

Round 13 22A5963B 387CCDAA 99C31397C91F

Round 14 387CCDAA BD2DD2AB 251B8BC717D0

Round 15 BD2DD2AB CF26B472 3330C5D9A36D

Round 16 19BA9212 CF26B472 181C5D75C66D

Cipher Text : C0B7A8D05F3A829C

Decryption

After initial permutation 19BA9212CF26B472

Round 1 CF26B472 BD2DD2AB 181C5D75C66D

Round 2 BD2DD2AB 387CCDAA 3330C5D9A36D

Round 3 387CCDAA 22A5963B 251B8BC717D0

Round 4 22A5963B FF3C485F 99C31397C91F

Round 5 FF3C485F 6CA6CB20 C2C1E96A4BF3

Round 6 6CA6CB20 10AF9D37 6D5560AF7CA5

Round 7 10AF9D37 308BEE97 02765708B5BF

Round 8 308BEE97 A9FC20A3 84BB4473DCCC

Round 9 A9FC20A3 2E8F9C65 34F822F0C66D

Round 10 2E8F9C65 A15A4B87 708AD2DDB3C0

Round 11 A15A4B87 236779C2 C1948E87475E

Round 12 236779C2 B8089591 69A629FEC913

Round 13 B8089591 4A1210F6 DA2D032B6EE3

Round 14 4A1210F6 5A78E394 06EDA4ACF5B5

Round 15 5A78E394 18CA18AD 4568581ABCCE

Round 16 14A7D678 18CA18AD 194CD072DE8C

Plain Text : 123456ABCD132536

RESULT : Implementation of DES cipher algorithm is successfully completed.

EXERCISE IV

4. Apply AES algorithm for practical applications.

AIM :

To write AES algorithm for practical applications in python.

ALGORITHM :

Step – 1 : Start the Process.

Step – 2 : Open terminal and create files with the cipher name, like AES.py

Step – 3 : Create the files with the touch command :

touch AES.py

Step – 4 : Edit the files with nano CLI editor.

Step – 5 : To open the file type the editor along with file name & extension: *nano AES.py*

Step – 6 : From pycrypto library import AES.

Step – 7 : Feed the Key to be "**key = b'C&F)H@McQfTjWnZr'**".

Step – 8 : Create a cipher block with a new variable.

Step – 9 : Feed the plain text in a variable "**data**".

Step – 10 : Perform encryption with the encrypt aes keyword function and vice-versa for decryption.

Step – 11 : Print both plain text and Cipher text as the output.

Step – 12 : Run the python compiler for compiling the py files, use the command :

Python *AES.py*.

Step – 13 : Record the output in the observation.

Step – 14 : Stop the Process.

CODING :

```
from Crypto.Cipher import AES
key = b'C&F)H@McQfTjWnZr'
cipher = AES.new(key, AES.MODE_CCM)
data = "this is a sample AES encryption code".encode()
nonce = cipher.nonce
ciphertext = cipher.encrypt(data)
print("Cipher text:", ciphertext)
cipher = AES.new(key, AES.MODE_CCM, nonce=nonce)
plaintext = cipher.decrypt(ciphertext)
print("Plain text:", plaintext)
```

OUTPUT :

```
Cipher text: b':\x80\x98\x0f\xeb\xdc\xc78\xdd(oySl\x80\xdf!\x0e.\x9a+eL\x1a]W\xbd\xa6\xcc"6\xf5\x85
    xac\xb5\x13'
Plain text: b'this is a sample AES encryption code'
```

RESULT : Implementation of AES cipher algorithm is successfully completed.

EXERCISE V

5. Implement RSA Algorithm using HTML and JavaScript.

AIM :

To Implement RSA Algorithm using HTML and JavaScript.

ALGORITHM :

Step – 1 : Start the Process.

Step – 2 : Open terminal and create files with the cipher name, like RSA.html

Step – 3 : Create the files with the touch command :

touch RSA.html

Step – 4 : Edit the files with nano CLI editor.

Step – 5 : To open the type the editor along with file name & extension: *nano* RSA.html

Step – 6 : Define the headers and text for verbosity.

Step – 7 : Define the Javascript tags to include scripting capability to the code.

Step – 8 : Define the variables for key and input data, and then include Greatest Common Divisor function to the script.

Step – 9 : Include For looping and If conditions to check the parameters and perform encryption and key exchange between the Tabs.

Step – 10 : Open the RSA.html file in any browser to visualize and test the output.

Step – 11 : In the webpage, enter P & Q value and enter message to encrypt, then record the output in the observation note.

Step – 12 : Stop the Process.

CODING :

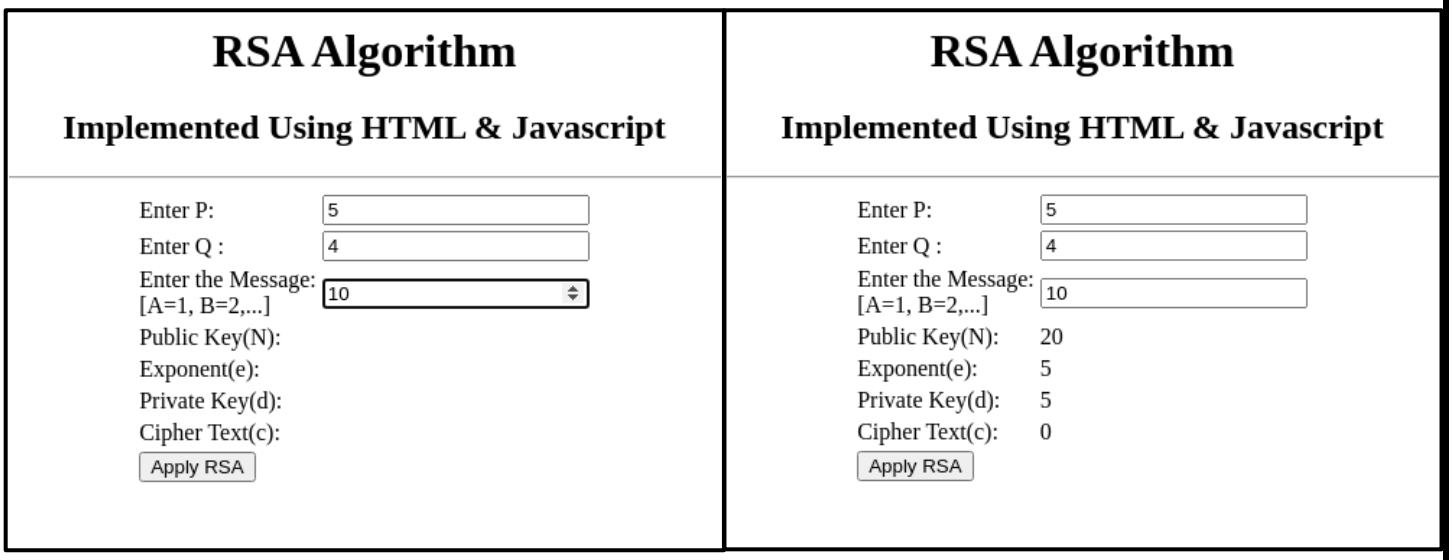
```
<html>
<head>
<title>RSA Encryption</title>
<meta name="viewport" content="width=device-width, initialscale=1.0">
</head>
<body>
<h1 style="text-align: center;">RSA Algorithm</h1>
<h2 style="text-align: center;">Implemented Using HTML & Javascript</h2>
<hr>
<table class="center">
<tr>
<td>Enter P:</td>
<td><input type="number" value="53" id="p"></td>
</tr>
<tr>
<td>Enter Q :</td>
<td><input type="number" value="59" id="q"></p>
</td>
</tr>
<tr>
<td>Enter the Message:<br>[A=1, B=2,...]</td>
<td><input type="number" value="89" id="msg"></p>
</td>
</tr>
<tr>
<td>Public Key(N):</td>
<td>
<p id="publickey(N)"></p>
</td>
</tr>
<tr>
<td>Exponent(e):</td>
<td>
```

```
<p id="exponent(e)"></p>
</td>
</tr>
<tr>
<td>Private Key(d):</td>
<td>
<p id="privatekey(d)"></p>
</td>
</tr>
<tr>
<td>Cipher Text(c):</td>
<td>
<p id="ciphertext(ct)"></p>
</td>
</tr>
<tr>
    <td><button onclick="RSA();">Apply RSA</button></td>
    </tr>
    </table>

    </body>
    <style>
        .center {
  margin-left: auto;
  margin-right: auto;
}
    </style>
    <script type="text/javascript">
    function RSA() {
    var gcd, p, q, no, n, t, e, i, x;
    gcd = function (a, b) { return (!b) ? a : gcd(b, a % b); };
    p = document.getElementById('p').value;
    q = document.getElementById('q').value;
    no = document.getElementById('msg').value;
```

```
    n = p * q;
    t = (p - 1) * (q - 1);
    for (e = 2; e < t; e++) {
    if (gcd(e, t) == 1) {
    break;
    }
    }
    for (i = 0; i < 10; i++) {
    x = 1 + i * t
    if (x % e == 0) {
    d = x / e;
    break;
    }
}
ctt = Math.pow(no, e).toFixed(0);
ct = ctt % n;
dtt = Math.pow(ct, d).toFixed(0);
dt = dtt % n;
document.getElementById('publickey(N)').innerHTML = n;
document.getElementById('exponent(e)').innerHTML = e;
document.getElementById('privatekey(d)').innerHTML = d;
document.getElementById('ciphertext(ct)').innerHTML = ct;
}
</script>
</html>
```

OUTPUT :



RESULT : Implementation of RSA cipher algorithm is successfully completed.

EXERCISE VI

6. Implement the Diffie-Hellman Key Exchange algorithm for a given problem.

AIM :

To Implement the Diffie-Hellman Key Exchange algorithm for a given problem.

ALGORITHM :

Step – 1: Start the Process.

Step – 2: Open terminal and create files with the cipher name, like D-H.py

Step – 3: Create the files with the touch command :

touch D-H.py

Step – 4: Edit the files with nano CLI editor.

Step – 5: To open the type the editor along with file name & extension: *nano* D-H.*py*

Step – 6: Define first function for prime checker with if conditions and for loop for cycling the check.

Step – 7: Define the second function for primitive root checker with for loops and check parameters.

Step – 8: Defining while loop for checking whether the user input is a prime or not.

Step – 9: Defining while loop for checking whether the user input is a primitive root or not.

Step – 10: Getting Private keys from user and using it to calculate Public keys.

Step – 11: Generating secret key for decryption

Step – 12: Defining print statements for getting the output on status of the key exchange and data ciphering.

Step – 13: Run the python compiler for compiling the py files, use the command :

Python D-H.py.

Step – 14: Stop the Process.

CODING :

```
# Diffie-Hellman Code
def prime_checker(p):
    if p < 1:
        return -1
    elif p > 1:
        if p == 2:
            return 1
        for i in range(2, p):
            if p % i == 0:
                return -1
            return 1
def primitive_check(g, p, L):
    for i in range(1, p):
        L.append(pow(g, i) % p)
    for i in range(1, p):
        if L.count(i) > 1:
            L.clear()
            return -1
        return 1
l = []
while 1:
    P = int(input("Enter P : "))
    if prime_checker(P) == -1:
        print("Number Is Not Prime, Please Enter Again!")
        continue
    break
while 1:
    G = int(input(f"Enter The Primitive Root Of {P} : "))
    if primitive_check(G, P, l) == -1:
        print(f"Number Is Not A Primitive Root Of {P}, Please Try Again!")
        continue
    break
```

```
# Private Keys
x1, x2 = int(input("Enter The Private Key Of User 1 : ")), int(
        input("Enter The Private Key Of User 2 : "))
while 1:
        if x1 >= P or x2 >= P:
                print(f"Private Key Of Both The Users Should Be Less Than {P}!")
                continue
        break
# Calculate Public Keys
y1, y2 = pow(G, x1) % P, pow(G, x2) % P
# Generate Secret Keys
k1, k2 = pow(y2, x1) % P, pow(y1, x2) % P
print(f"\nSecret Key For User 1 Is {k1}\nSecret Key For User 2 Is {k2}\n")
if k1 == k2:
        print("Keys Have Been Exchanged Successfully")
else:
        print("Keys Have Not Been Exchanged Successfully")
```

OUTPUT :

```
Enter P : 19
Enter The Primitive Root Of 19 : 3
Enter The Private Key Of User 1 : 5
Enter The Private Key Of User 2 : 7
Secret Key For User 1 Is 13
Secret Key For User 2 Is 13

Keys Have Been Exchanged Successfully
```

RESULT : Implementation of Deffie - Hellman algorithm is successfully completed.