Name: Devanshu Surana
Roll No.: 23
Prn:1032210755
Panel: C batch:C1

Lab A2: Implement simple DES symmetric key algorithm using python or java or C++

Objective of Lab

1. To study understand and implement DES symmetric key algorithm

Theory

The Data Encryption Standard (DES) is a symmetric key encryption algorithm that was widely used for secure data encryption but is now considered obsolete due to its relatively short key length (56 bits). However, understanding DES can be helpful for learning about encryption principles and algorithms. Note that DES has been succeeded by more secure encryption algorithms like AES (Advanced Encryption Standard).

Here's a high-level explanation of how DES works, along with a simple Python implementation:

Understanding DES:

1. Key Generation:
   - A 64-bit encryption key is provided, but only 56 bits are used, with 8 bits used for parity checking.
   - The 56-bit key is divided into 16 subkeys, one for each of the 16 rounds of DES.

2. Initial Permutation (IP):
   - The 64-bit plaintext is initially permuted according to a fixed table.

3. Rounds (16 rounds):
   - The plaintext is divided into two 32-bit blocks, left and right.
   - The right block is expanded to 48 bits.
   - The expanded right block is XORed with the subkey for the current round.
   - The result is passed through a series of S-boxes, which substitute 48 bits with 32 bits.
   - The 32-bit outputs from the S-boxes are then permuted.
   - The left and right blocks are swapped.
   - This process is repeated for 16 rounds.

4. Final Permutation (FP):

- After all rounds are completed, the left and right blocks are combined and permuted according to another fixed table.

5. Cipher Text:
   - The final permutation is the ciphertext.

Python Implementation:
I'll provide a simple Python implementation of DES using the `pycryptodome` library, which provides cryptographic functions. You may need to install the library using `pip install pycryptodome`.

```
from Crypto.Cipher import DES
from Crypto.Random import get_random_bytes

# Initialize DES cipher with a random 8-byte key
key = get_random_bytes(8)
cipher = DES.new(key, DES.MODE_ECB)

# Input plaintext (64 bits)
plaintext = b'12345678'

# Encrypt
ciphertext = cipher.encrypt(plaintext)
print("Ciphertext:", ciphertext)

# Decrypt
decipher = DES.new(key, DES.MODE_ECB)
decrypted_text = decipher.decrypt(ciphertext)
print("Decrypted text:", decrypted_text)
```

Code

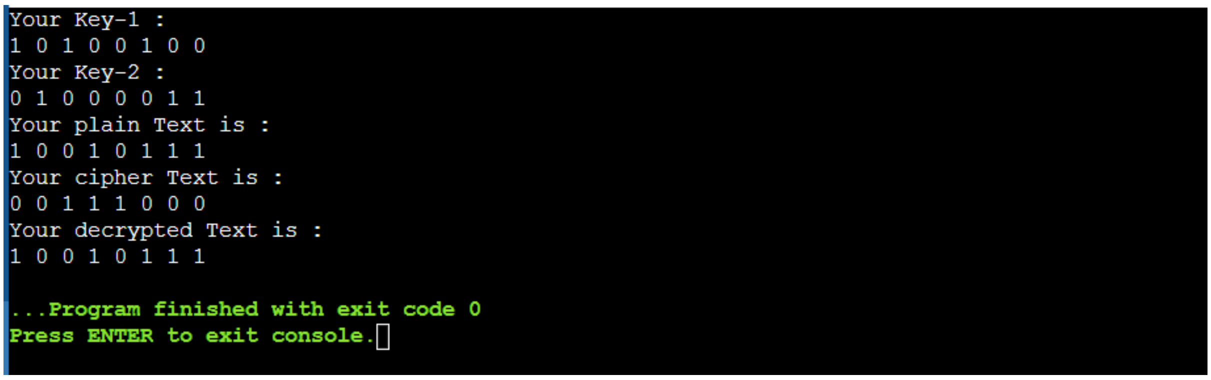
```
import java.io.*;

public class GFG {
	int key[] = {
		1, 0, 1, 0, 0, 0, 0, 0, 1, 0
	};
	int P10[] = { 3, 5, 2, 7, 4, 10, 1, 9, 8, 6 };
	int P8[] = { 6, 3, 7, 4, 8, 5, 10, 9 };
int key1[] = new int[8];
int key2[] = new int[8];

int[] IP = { 2, 6, 3, 1, 4, 8, 5, 7 };
int[] EP = { 4, 1, 2, 3, 2, 3, 4, 1 };
int[] P4 = { 2, 4, 3, 1 };
int[] IP_inv = { 4, 1, 3, 5, 7, 2, 8, 6 };

int[][] S0 = { { 1, 0, 3, 2 },
               { 3, 2, 1, 0 },
               { 0, 2, 1, 3 },
               { 3, 1, 3, 2 } };
int[][] S1 = { { 0, 1, 2, 3 },
               { 2, 0, 1, 3 },
               { 3, 0, 1, 0 },
               { 2, 1, 0, 3 } };

void key_generation()
{
        int key_[] = new int[10];

        for (int i = 0; i < 10; i++) {
                key_[i] = key[P10[i] - 1];
        }

        int Ls[] = new int[5];
        int Rs[] = new int[5];

        for (int i = 0; i < 5; i++) {
                Ls[i] = key_[i];
                Rs[i] = key_[i + 5];
        }

        int[] Ls_1 = shift(Ls, 1);
        int[] Rs_1 = shift(Rs, 1);

        for (int i = 0; i < 5; i++) {
                key_[i] = Ls_1[i];
                key_[i + 5] = Rs_1[i];
        }

        for (int i = 0; i < 8; i++) {
                key1[i] = key_[P8[i] - 1];
        }

        int[] Ls_2 = shift(Ls, 2);
        int[] Rs_2 = shift(Rs, 2);

        for (int i = 0; i < 5; i++) {
                key_[i] = Ls_2[i];
                key_[i + 5] = Rs_2[i];
        }

        for (int i = 0; i < 8; i++) {
                key2[i] = key_[P8[i] - 1];
        }

        System.out.println("Your Key-1 :");

        for (int i = 0; i < 8; i++)
                System.out.print(key1[i] + " ");

        System.out.println();
        System.out.println("Your Key-2 :");

        for (int i = 0; i < 8; i++)
                System.out.print(key2[i] + " ");
}

int[] shift(int[] ar, int n)
{
        while (n > 0) {
                int temp = ar[0];
                for (int i = 0; i < ar.length - 1; i++) {
                        ar[i] = ar[i + 1];
                }
                ar[ar.length - 1] = temp;
                n--;
        }
        return ar;
}

int[] encryption(int[] plaintext)
{
        int[] arr = new int[8];

        for (int i = 0; i < 8; i++) {
                arr[i] = plaintext[IP[i] - 1];
        }
        int[] arr1 = function_(arr, key1);

        int[] after_swap = swap(arr1, arr1.length / 2);

        int[] arr2 = function_(after_swap, key2);

        int[] ciphertext = new int[8];

        for (int i = 0; i < 8; i++) {
                ciphertext[i] = arr2[IP_inv[i] - 1];
        }

        return ciphertext;
}
String binary_(int val)
{
        if (val == 0)
                return "00";
        else if (val == 1)
                return "01";
        else if (val == 2)
                return "10";
        else
                return "11";
}

int[] function_(int[] ar, int[] key_)
{

        int[] l = new int[4];
        int[] r = new int[4];

        for (int i = 0; i < 4; i++) {
                l[i] = ar[i];
                r[i] = ar[i + 4];
        }
```

```
int[] ep = new int[8];

for (int i = 0; i < 8; i++) {
        ep[i] = r[EP[i] - 1];
}

for (int i = 0; i < 8; i++) {
        ar[i] = key_[i] ^ ep[i];
}

int[] l_1 = new int[4];
int[] r_1 = new int[4];

for (int i = 0; i < 4; i++) {
        l_1[i] = ar[i];
        r_1[i] = ar[i + 4];
}

int row, col, val;

row = Integer.parseInt("" + l_1[0] + l_1[3], 2);
col = Integer.parseInt("" + l_1[1] + l_1[2], 2);
val = S0[row][col];
String str_l = binary_(val);

row = Integer.parseInt("" + r_1[0] + r_1[3], 2);
col = Integer.parseInt("" + r_1[1] + r_1[2], 2);
val = S1[row][col];
String str_r = binary_(val);

int[] r_ = new int[4];
for (int i = 0; i < 2; i++) {
        char c1 = str_l.charAt(i);
        char c2 = str_r.charAt(i);
        r_[i] = Character.getNumericValue(c1);
        r_[i + 2] = Character.getNumericValue(c2);
}
int[] r_p4 = new int[4];
for (int i = 0; i < 4; i++) {
        r_p4[i] = r_[P4[i] - 1];
}
```

```
        for (int i = 0; i < 4; i++) {
                l[i] = l[i] ^ r_p4[i];
        }

        int[] output = new int[8];
        for (int i = 0; i < 4; i++) {
                output[i] = l[i];
                output[i + 4] = r[i];
        }
        return output;
}

int[] swap(int[] array, int n)
{
        int[] l = new int[n];
        int[] r = new int[n];

        for (int i = 0; i < n; i++) {
                l[i] = array[i];
                r[i] = array[i + n];
        }

        int[] output = new int[2 * n];
        for (int i = 0; i < n; i++) {
                output[i] = r[i];
                output[i + n] = l[i];
        }

        return output;
}


int[] decryption(int[] ar)
{
        int[] arr = new int[8];

        for (int i = 0; i < 8; i++) {
                arr[i] = ar[IP[i] - 1];
        }

        int[] arr1 = function_(arr, key2);
        int[] after_swap = swap(arr1, arr1.length / 2);

        int[] arr2 = function_(after_swap, key1);

        int[] decrypted = new int[8];

        for (int i = 0; i < 8; i++) {
            decrypted[i] = arr2[IP_inv[i] - 1];
        }

        return decrypted;
    }

    public static void main(String[] args)
    {

        GFG obj = new GFG();

        obj.key_generation();
        int[] plaintext = {
            1, 0, 0, 1, 0, 1, 1, 1
        };

        System.out.println();
        System.out.println("Your plain Text is :");
        for (int i = 0; i < 8; i++)
            System.out.print(plaintext[i] + " ");

        int[] ciphertext = obj.encryption(plaintext);

        System.out.println();
        System.out.println(
            "Your cipher Text is :");
        for (int i = 0; i < 8; i++)
            System.out.print(ciphertext[i] + " ");

        int[] decrypted = obj.decryption(ciphertext);

        System.out.println();
        System.out.println(
            "Your decrypted Text is :");
        for (int i = 0; i < 8; i++)
```

```
                              System.out.print(decrypted[i] + " ");
          }
}
```

Output Screen shots



```
Your Key-1 :
1 0 1 0 0 1 0 0
Your Key-2 :
0 1 0 0 0 0 1 1
Your plain Text is :
1 0 0 1 0 1 1 1
Your cipher Text is :
0 0 1 1 1 0 0 0
Your decrypted Text is :
1 0 0 1 0 1 1 1

...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion: Thus, we have successfully learned and implemented DES cipher.

FAQs:

1. What is the concept of fiestel cipher.
2. Draw and describe DES algorithm briefly
3. List and state broad level operations used internally in DES algorithm.
4. Compare various block ciphers such as DES, AES, Blowfish etc..
5. What are the Block cipher design guidelines.