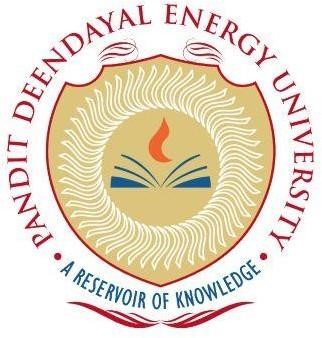
**PANDIT DEENDAYAL ENERGY UNIVERSITY SCHOOL OF TECHNOLOGY**



**Course: Information Security**

**Course Code: 20CP304P**

**LAB MANUAL**

**B.Tech. (Computer Engineering)**

**Semester 5**

|  |  |
| --- | --- |
| **Submitted To:** | **Submitted By:** |
| Dr. Rutvij H. Jhaveri | Dev Patel |
|  | 22BCP185 |
|  | G5 Batch |

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.**  **No.** | **List of experiments** | **Date** | **Sign** |
| **1** | Study and Implement a program for Caesar Cipher |  |  |
| **2** | Study and Implement a program for 5x5 Playfair Cipher to encrypt and decrypt the message. |  |  |
| **3** | Study and Implement a program for Rail Fence Cipher with columnar transposition |  |  |
| **4** | Study and implement a program for columnar Transposition Cipher |  |  |
| **5** | Study and implement a program for Vigenère Cipher |  |  |
| **6** | Study and Implement a program for n-gram Hill Cipher |  |  |
| **7** | Study and Use of RSA algorithm (encryption and decryption) |  |  |
| **8** | Study and implement a program of the Digital Signature with RSA algorithm (Reverse RSA) |  |  |
| **9** | Study and Use of Diffie-Hellman Key Exchange |  |  |
| **10** | Design cipher with detailed explanation. Sample as follows.   1. Write a program to encrypt the plaintext with the given key. E.g. plaintext GRONSFELD with the key 1234. Add 1 to G to get H (the letter 1 rank after G is H in the alphabet), then add 2 to C or E (the letter 2 ranks after C is E), and so on. Use smallest letter from plaintext as filler. 2. Encrypt the input words PLAINTEXT= RAG   BABY to obtain CIPHERTEXT = SCJ DDFD |  |  |

# Experiment – 1

• **Aim :** Study and implement a program for Caesar Cipher.

➢ **Caesar Cipher :**

• **Code :**

#include <stdio.h>

int main()

{

    char a[100];

    int key;

    printf("Enter the Text for Encryption: ");

    fgets(a, sizeof(a), stdin);

    printf("Enter the key to Encrypt the data : ");

    scanf("%d", &key);

    for (int i = 0; a[i] != '\0'; i++)

    {

        if (a[i] == '\n')

        {

            a[i] = '\0';

            break;

        }

    }

    for (int i = 0; a[i] != '\0'; i++)

    {

        if (a[i] == ' ')

        {

            continue;

        }

        else if (a[i] >= 'A' && a[i] <= 'Z')

        {

            a[i] = ((a[i] - 'A' + key) % 26) + 'A';

        }

        else if (a[i] >= 'a' && a[i] <= 'z')

        {

            a[i] = ((a[i] - 'a' + key) % 26) + 'a';

        }

        else if (a[i] >= '0' && a[i] <= '9')

        {

            a[i] = ((a[i] - '0' + key) % 10) + '0';

        }

    }

    printf("Encrypted Text: %s\n", a);

    printf("Enter the key to Decrypted the data : ");

    scanf("%d", &key);

    for (int i = 0; a[i] != '\0'; i++)

    {

        if (a[i] == ' ')

        {

            continue;

        }

        else if (a[i] >= 'A' && a[i] <= 'Z')

        {

            a[i] = ((a[i] - 'A' - key + 26) % 26) + 'A';

        }

        else if (a[i] >= 'a' && a[i] <= 'z')

        {

            a[i] = ((a[i] - 'a' - key + 26) % 26) + 'a';

        }

        else if (a[i] >= '0' && a[i] <= '9')

        {

            a[i] = ((a[i] - '0' - key + 10) % 10) + '0';

        }

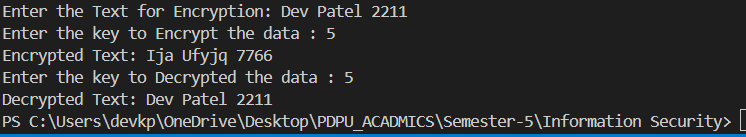
    }

    printf("Decrypted Text: %s\n", a);

    return 0;

}

• **Output :**



➢ **Refined Caesar Cipher :**

In this refined approach I had used two digit key and let’s say first digit is x and second digit is y. So the given string is encrypted using x and y alternatively as a key also decrypted in the same manner.

• **Code :**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void encryption(char \*plain\_text, int key)

{

    int key1 = key / 100;

    int key2 = (key / 10) % 10;

    int key3 = key % 10;

    int count = 0;

    for (int i = 0; plain\_text[i] != '\0'; i++)

    {

        if (isalpha(plain\_text[i]))

        {

            char base = isupper(plain\_text[i]) ? 'A' : 'a';

            int shift;

            if (count % 3 == 0)

            {

                shift = key1;

            }

            else if (count % 3 == 1)

            {

                shift = key2;

            }

            else

            {

                shift = key3;

            }

            plain\_text[i] = (plain\_text[i] - base + shift) % 26 + base;

            count++;

        }

    }

}

void decryption(char \*cipher\_text, int key)

{

    int key1 = key / 100;

    int key2 = (key / 10) % 10;

    int key3 = key % 10;

    int count = 0;

    for (int i = 0; cipher\_text[i] != '\0'; i++)

    {

        if (isalpha(cipher\_text[i]))

        {

            char base = isupper(cipher\_text[i]) ? 'A' : 'a';

            int shift;

            if (count % 3 == 0)

            {

                shift = key1;

            }

            else if (count % 3 == 1)

            {

                shift = key2;

            }

            else

            {

                shift = key3;

            }

            cipher\_text[i] = (cipher\_text[i] - base - shift + 26) % 26 + base;

            count++;

        }

    }

}

int main()

{

    char text[1000];

    int key;

    printf("Enter input string: ");

    fgets(text, sizeof(text), stdin);

    text[strcspn(text, "\n")] = 0; // Remove newline if present

    printf("Plain Text is: %s\n", text);

    printf("Enter three digit key: ");

    scanf("%d", &key);

    encryption(text, key);

    printf("Encrypted Text is: %s\n", text);

    decryption(text, key);

    printf("Decrypted Text is: %s\n", text);

    return 0;

}

• **Output :**

