

Parul University

FACULTY OF ENGINEERING AND TECHNOLOGY

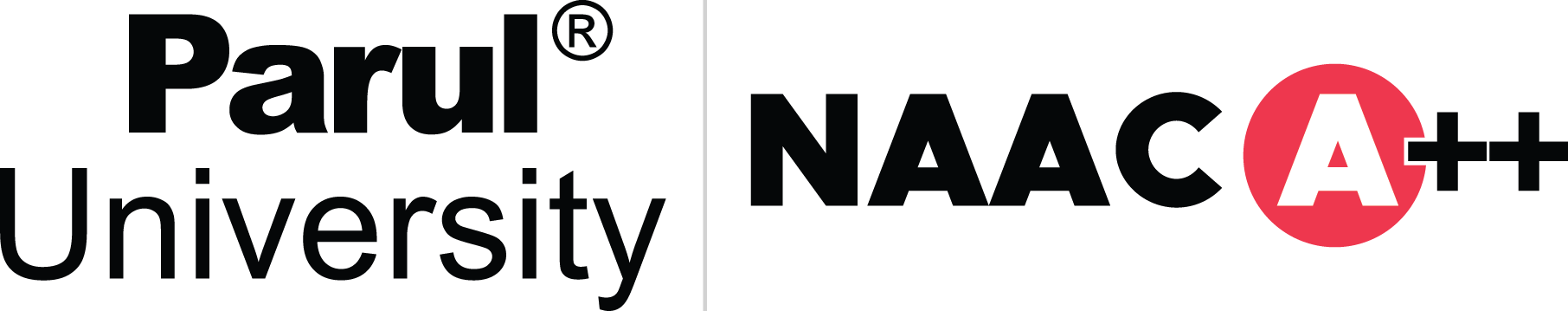
BACHELOR OF TECHNOLOGY

INFORMATION AND NETWORK SECURITY

(303105376)

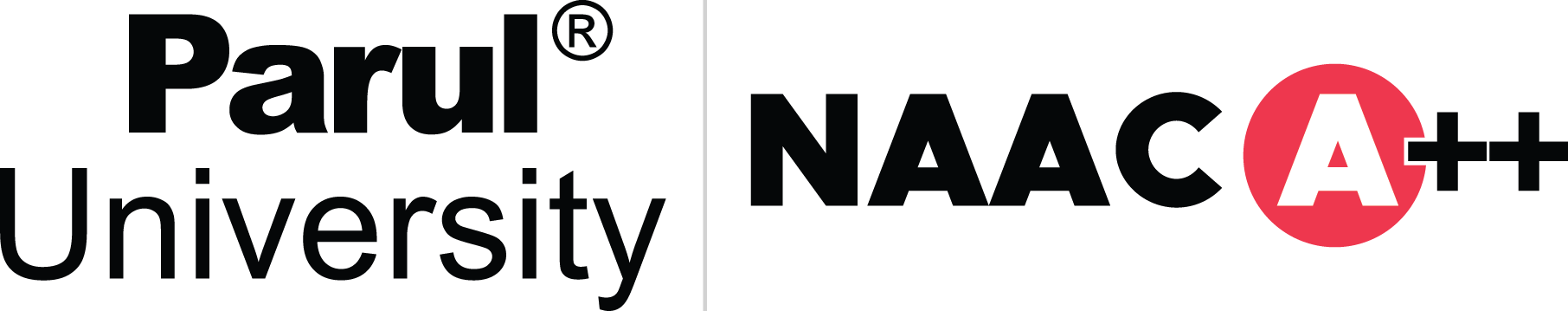
IV SEMESTER

Computer Science & Engineering Department



Laboratory Manual

Session 2025-26



Faculty of Engineering & Technology

Subject Name : INS Laboratory

Subject Code : 303105376

B.Tech CSE Year 4th Semester 7th

CERTIFICATE

This is to Certify that

Mr./Ms. …………………………………………………………………………………………………...

With enrolment no. ………………………………….has successfully completed his/her

Laboratory experiments in **Information and Network Security Laboratory (303105376)** From the department of …………………………………………..… during the academic year ………………………



Head of department: ……………………….

Date of Submission : ………………………. Staff In Charge: ……………………….

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr. No** | **Experimental Title** | **Page No** | | **Date of Completion** | **Sign** | **Marks (out of 10)** |
| **From** | **To** |
| 1. | Implement Caesar cipher encryption-decryption |  |  |  |  |  |
| 2. | Implement Monoalphabetic cipher encryption-decryption |  |  |  |  |  |
| 3. | Implement Playfair cipher encryption-decryption |  |  |  |  |  |
| 4. | Implement Polyalphabetic cipher encryption-decryption |  |  |  |  |  |
| 5. | Implement Hill cipher encryption-decryption |  |  |  |  |  |
| 6. | Implement Simple Transposition encryption-decryption |  |  |  |  |  |
| 7. | Implement One time pad encryption-decryption |  |  |  |  |  |
| 8. | Implement Diffi-Hellmen Key exchange Method |  |  |  |  |  |
| 9. | Implement RSA encryption-decryption algorithm |  |  |  |  |  |
| 10. | Demonstrate working of Digital Signature using Cryptool |  |  |  |  |  |

**TABLE OF CONTENT**

PRACTICAL - 1

**AIM** - Design test cases using Boundary value analysis.

**CODE :**

#include <stdio.h>

*int* main() {

*char* message[100];

*int* key, i, choice;

    printf("Enter a message (letters and spaces only): ");

    scanf(" %[^\n]", message); *// Read full line including spaces*

    printf("Enter key (number of positions to shift): ");

    scanf("*%d*", &key);

    printf("Choose:\n1. Encrypt\n2. Decrypt\nEnter your choice: ");

    scanf("*%d*", &choice);

    for (i = 0; message[i] != '\0'; i++) {

*char* ch = message[i];

        if (ch >= 'A' && ch <= 'Z') {

            if (choice == 1)

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

            else if (choice == 2)

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

        } else if (ch >= 'a' && ch <= 'z') {

            if (choice == 1)

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

            else if (choice == 2)

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

        }

*// If it's a space or any other character, we leave it unchanged*

    }

    if (choice == 1)

        printf("Encrypted message: *%s*\n", message);

    else if (choice == 2)

        printf("Decrypted message: *%s*\n", message);

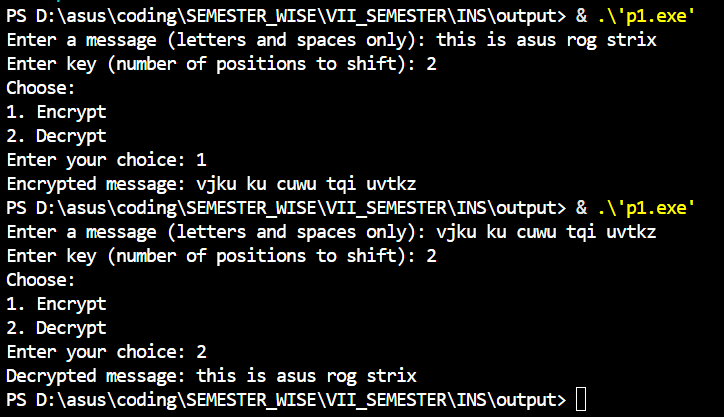
    else

        printf("Invalid choice!\n");

return 0;

}

**OUTPUT :**

****

PRACTICAL - 2

**AIM** - Design test cases using Boundary value analysis.

**CODE :**

#include <stdio.h>

#include <string.h>

*void* encrypt(*char* message[], *char* key[]) {

    for (*int* i = 0; message[i] != '\0'; i++) {

*char* ch = message[i];

        if (ch >= 'A' && ch <= 'Z') {

*// Uppercase encryption*

            message[i] = key[ch - 'A'];

        } else if (ch >= 'a' && ch <= 'z') {

*// Lowercase encryption (convert using same key)*

            message[i] = key[ch - 'a'] + 32;

        }

*// spaces and other characters are left unchanged*

    }

}

*void* decrypt(*char* message[], *char* key[]) {

*char* reverseKey[26];

*// Create reverse mapping for uppercase*

    for (*int* i = 0; i < 26; i++) {

        reverseKey[key[i] - 'A'] = 'A' + i;

    }

    for (*int* i = 0; message[i] != '\0'; i++) {

*char* ch = message[i];

        if (ch >= 'A' && ch <= 'Z') {

*// Uppercase decryption*

            message[i] = reverseKey[ch - 'A'];

        } else if (ch >= 'a' && ch <= 'z') {

*// Lowercase decryption (convert using reverse key)*

            message[i] = reverseKey[ch - 'a'] + 32;

        }

*// spaces and other characters are left unchanged*

    }

}

*int* main() {

*char* message[100];

*// Fixed substitution key (must be permutation of A-Z)*

*char* key[26] = {'Q','W','E','R','T','Y','U','I','O','P',

                    'A','S','D','F','G','H','J','K','L','Z',

                    'X','C','V','B','N','M'};

*int* choice;

    printf("Enter the message: ");

    scanf(" %[^\n]", message); *// Accept spaces in input*

    printf("Choose:\n1. Encrypt\n2. Decrypt\nEnter your choice: ");

    scanf("*%d*", &choice);

    if (choice == 1) {

        encrypt(message, key);

        printf("Encrypted message: *%s*\n", message);

    } else if (choice == 2) {

        decrypt(message, key);

        printf("Decrypted message: *%s*\n", message);

    } else {

        printf("Invalid choice!\n");

    }

    return 0;

}

**OUTPUT :**

