

EX NO:05

Encrypt and decrypt a message by implementing Rail fence transposition technique.

DATE:22.02.24

Aim: To write a c program to encrypt and decrypt a message by implementing Rail fence transposition technique.

Algorithm:

1. Include necessary header files (<stdio.h> and <string.h>).
2. Declare character arrays for the original message (msg), encryption key (key), new key (newKey), encrypted message (encryptedMsg), and decrypted message (decryptedMsg).
3. Declare integer variables msgLen, keyLen, i, and j for storing lengths and loop indices.
4. Initialize msg and key with the original message and encryption key.
5. Calculate msgLen and keyLen using strlen.
6. Use a loop to generate the new key (newKey) based on the original key (key).
7. Initialize i and j to 0.
8. Use a loop to iterate over each character in the original message (msg).
9. Combine it with the corresponding character from the new key (newKey) using modular arithmetic.
10. Add a null terminator at the end of the decrypted message.

Program:

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 100
void encryptMessage(char *message, int rails, char *encryptedMessage) {
    int len = strlen(message), index = 0;
    for (int row = 0; row < rails; row++) {
        for (int i = row; i < len; i += 2 * (rails - 1)) {
            encryptedMessage[index++] = message[i];
            if (row != 0 && row != rails - 1 && i + 2 * (rails - row - 1) < len)
                encryptedMessage[index++] = message[i + 2 * (rails - row - 1)];
        }
    }
    encryptedMessage[len] = '\0';
}
void decryptMessage(char *encryptedMessage, int rails, char *decryptedMessage) {
    int len = strlen(encryptedMessage), index = 0;
    for (int row = 0; row < rails; row++) {
        for (int i = row; i < len; i += 2 * (rails - 1)) {
            decryptedMessage[i] = encryptedMessage[index++];
            if (row != 0 && row != rails - 1 && i + 2 * (rails - row - 1) < len)
                decryptedMessage[i + 2 * (rails - row - 1)] = encryptedMessage[index++];
        }
    }
    decryptedMessage[len] = '\0';
}
int main() {
    char message[MAX_LEN], encryptedMessage[MAX_LEN], decryptedMessage[MAX_LEN];
```

```

int rails;
printf("Enter the message to be encrypted: ");
fgets(message, sizeof(message), stdin);
printf("Enter the number of rails: ");
scanf("%d", &rails);
encryptMessage(message, rails, encryptedMessage);
printf("Encrypted message: %s\n", encryptedMessage);
decryptMessage(encryptedMessage, rails, decryptedMessage);
printf("Decrypted message: %s\n", decryptedMessage);
return 0;
}

```

Input and Output:

The screenshot shows a C++ IDE with the following components:

- Code Editor:** Contains the source code for `rail_fence.cpp`. The code implements the Rail Fence cipher encryption and decryption functions. It uses a zigzag pattern to fill the encrypted message and then reads it back to reconstruct the original message.
- Compiler:** Shows the compilation results. The output filename is `C:\Users\aiishwarya\Desktop\rail_fence.exe`. The compilation time is 1.42s.
- Console:** Displays the program's execution. It prompts the user to enter a message ("Enter the message to be encrypted: ") and the number of rails ("Enter the number of rails: "). The user enters "WORLD" and "2". The program outputs the encrypted message "WRDOL" and the decrypted message "WORLD".

Result: A c program to encrypt and decrypt a message by implementing Rail fence transposition technique is successfully executed.

SAVEETHA SCHOOL OF ENGINEERING

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES