**PANDIT DEENDAYAL ENERGY UNIVERSITY**

**SCHOOL OF TECHNOLOGY**



**Course: Information Security**

**Course Code: 20CP304P**

**LAB MANUAL**

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

**Semester 5**

|  |  |
| --- | --- |
| **Submitted To:**  Dr. Hargeet Kaur | **Submitted By:**  ADITYA JETHANI  21BCP166  G-5 |
|  |  |

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **List of experiments** | **Date** | **Sign** |
| **1** | Download and Practise Cryptool. |  |  |
| **2** | Study and implement Caesar cipher with encryption, decryption. |  |  |
| **3** | Program for Transposition(Columnar) cipher with encryption and decryption |  |  |
| **4** |  |  |  |
| **5** |  |  |  |
| **6** |  |  |  |
| **7** |  |  |  |
| **8** |  |  |  |
| **9** |  |  |  |
| **10** |  |  |  |

**PRACTICAL : 1**

**Aim**: Download and Practise Cryptool.

**Introduction** : CrypTool is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) project that is a free [e-learning](https://en.wikipedia.org/wiki/E-learning) software for illustrating [cryptographic and cryptanalytic concepts](https://en.wikipedia.org/wiki/Cryptography). According to "Hakin9",CrypTool is worldwide the most widespread e-learning software in the field of [cryptology](https://en.wikipedia.org/wiki/Cryptology).

They contain most [classical ciphers](https://en.wikipedia.org/wiki/Classical_cipher), as well as modern symmetric and [asymmetric cryptography](https://en.wikipedia.org/wiki/Public-key_cryptography) including [RSA](https://en.wikipedia.org/wiki/RSA_(algorithm)), [ECC](https://en.wikipedia.org/wiki/Elliptic_curve_cryptography), [digital signatures](https://en.wikipedia.org/wiki/Digital_signature), hybrid encryption, [homomorphic encryption](https://en.wikipedia.org/wiki/Homomorphic_encryption), and [Diffie–Hellman key exchange](https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange). Methods from the area of [quantum cryptography](https://en.wikipedia.org/wiki/Quantum_cryptography) (like [BB84 key exchange protocol](https://en.wikipedia.org/wiki/BB84)) and the area of [post-quantum cryptography](https://en.wikipedia.org/wiki/Post-quantum_cryptography) (like [McEliece](https://en.wikipedia.org/wiki/McEliece_cryptosystem" \o "McEliece cryptosystem), WOTS, [Merkle-Signature-Scheme](https://en.wikipedia.org/wiki/Merkle_signature_scheme), [XMSS, XMSS\_MT, and SPHINCS](https://en.wikipedia.org/wiki/Hash-based_cryptography)) are implemented. In addition to the algorithms, solvers (analysers) are included, especially for classical ciphers. Other methods (for instance [Huffman code](https://en.wikipedia.org/wiki/Huffman_coding), [AES](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard), [Keccak](https://en.wikipedia.org/wiki/SHA-3), [MSS](https://en.wikipedia.org/wiki/Merkle_signature_scheme)) are visualized.

**1.Caeser Cipher:**

The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It’s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

**2.** **Vigenère cipher:**

Vigenère Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Ceasar Ciphers. At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword.

**3.Vernam Cipher:**

Vernam Cipher is a method of encrypting alphabetic text. It is one of the Substitution techniques for converting plain text into cipher text. We take a key to encrypt the plain text whose length should be equal to the length of the plain text. Assign a number to each character of the plain-text and the key according to alphabetical order. Bitwise XOR both the number (Corresponding plain-text character number and Key character number). Subtract the number from 26 if the resulting number is greater than or equal to 26, if it isn’t then leave it.

**4.Simple Columnar Transposition:**

In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text. The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Width of the rows and the permutation of the columns are usually defined by a keyword. The permutation is defined by the alphabetical order of the letters in the keyword. Any spare spaces are filled with nulls or left blank or placed by a character. Finally, the message is read off in columns, in the order specified by the keyword.

**5.Mono Alphabetic Substitution:**

A monoalphabetic cipher is any cipher in which the letters of the plain text are mapped to cipher text letters based on a single alphabetic key. Examples of monoalphabetic ciphers would include the Caesar-shift cipher, where each letter is shifted based on a numeric key, and the atbash cipher, where each letter is mapped to the letter symmetric to it about the centre of the alphabet.

**Output:**

**A screenshot of a computer

Description automatically generated**

**Caeser Cipher**

**A screenshot of a computer

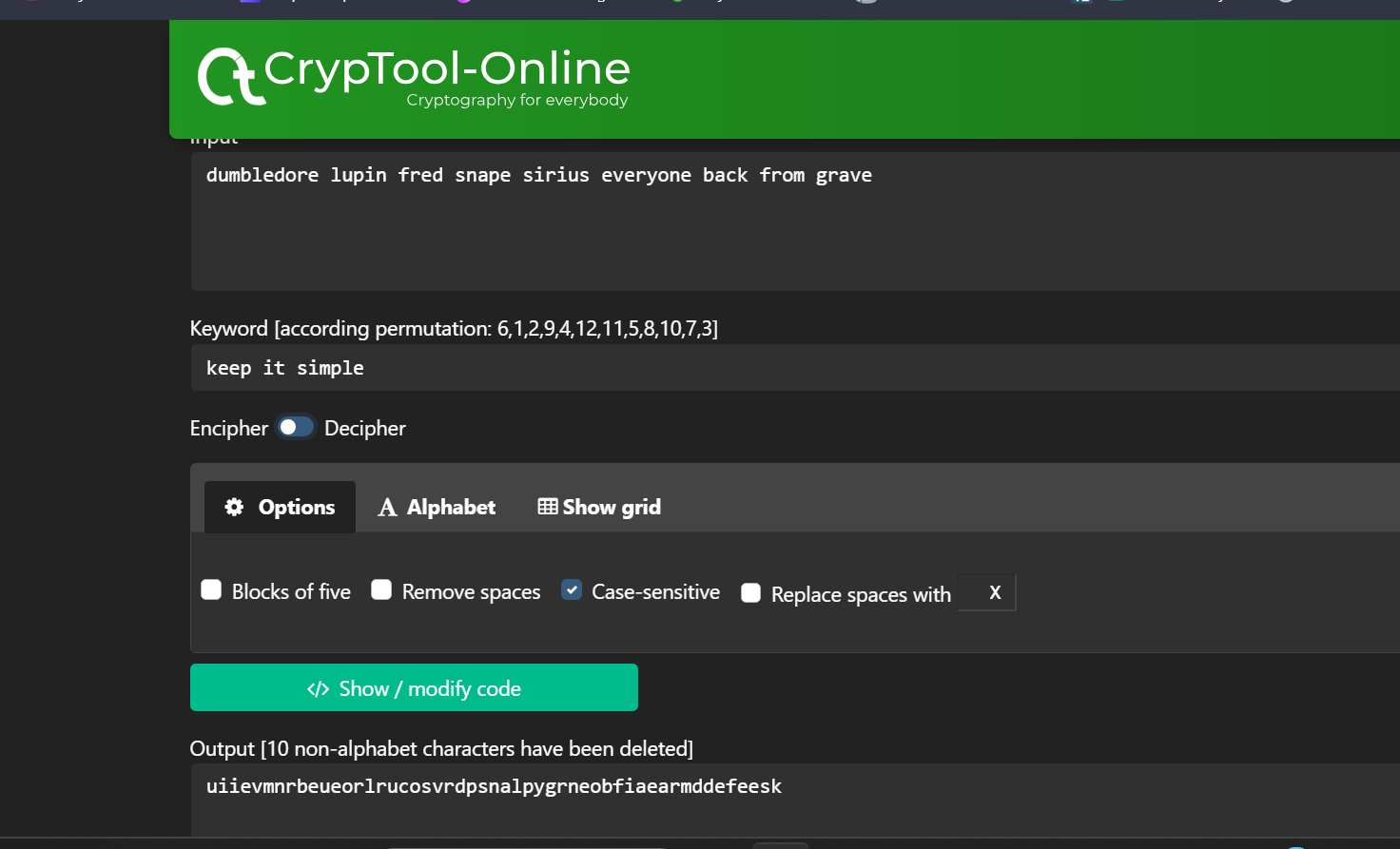
Description automatically generated**

**Vigenère cipher**

**A screenshot of a computer

Description automatically generated**

**Vernam Cipher**

****

**Simple Columnar Transposition**

**A screenshot of a computer

Description automatically generated**

**Mono Alphabetic Substitution**

**References :**

1. Wikipedia

(https://en.wikipedia.org/wiki/CrypTool#:~:text=CrypTool%20is%20an%20open%2Dsource,in%20the%20field%20of%20cryptology.&text=CrypTool%20implements%20more%20than%20400%20algorithms.)

1. GeeksforGeeks

(<https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/>)

**PRACTICAL : 2**

**Aim:** Study and implement Caesar cipher with encryption, decryption.

**Introduction:** The Caesar cipher is one of the earliest and simplest forms of encryption. This is a type of substitution cipher, where each character in the original message is moved to a number of places below the line.

For example, if there were 3 changes, A would be replaced by D, B by E, C by F, and so on. The first messages will be saved as "HELLO" and "KHOOR".

To encrypt a message using a Caesar cipher:

* Select shift number - This is the number of positions you shift each character in the list.
* Write a simple text message.
* Replace each letter in the message with the number you selected. So if the shift is 5, then A becomes F, B becomes G and so on.
* The resulting ciphertext is an encrypted message.

To dig into the Caesar Cipher:

* Displays the shift number originally used. Maybe they know this or maybe you should wonder. Take the cipher text and shift each character back to A by shift number. So if the variable was 3, then D would be A, E would be B, and so on.
* The resulting plaintext is the extracted original message.
* The Caesar cipher is easy to crack because there are only 25 possible changes. It does not offer the worst protection and is of particular historical interest today. However, it represents an early attempt to use alternatives to protect the message.

**Program Code:**

#include <bits/stdc++.h>

using namespace std;

string caesarEncrypt(const string &plaintext, int shift)

{

    string encryptedText = "";

    for (char c : plaintext)

    {

        if (isalpha(c))

        {

            char base = (islower(c)) ? 'a' : 'A';

            encryptedText += static\_cast<char>((c - base + shift) % 26 + base);

        }

        else

        {

            encryptedText += c;

        }

    }

    return encryptedText;

}

string caesarDecrypt(const string &ciphertext, int shift)

{

    return caesarEncrypt(ciphertext, 26 - shift);

}

int main()

{

    cout << "Caesar Cipher Encryption and Decryption" << endl;

    int choice;

    cout << "Select an option:" << endl;

    cout << "1. Encrypt" << endl;

    cout << "2. Decrypt" << endl;

    cout << "Enter your choice (1 or 2): ";

    cin >> choice;

    cin.ignore();

    string text;

    int shift;

    // Input text

    cout << "Enter the text: ";

    getline(cin, text);

    cout << "Enter the shift value (0-25): ";

    cin >> shift;

    if (shift < 0 || shift > 25)

    {

        cout << "Invalid shift value. Exiting..." << endl;

        return 1;

    }

    if (choice == 1)

    {

        string encryptedText = caesarEncrypt(text, shift);

        cout << "Encrypted text: " << encryptedText << endl;

    }

    else if (choice == 2)

    {

        string decryptedText = caesarDecrypt(text, shift);

        cout << "Decrypted text: " << decryptedText << endl;

    }

    else

    {

        cout << "Invalid choice. Exiting..." << endl;

        return 1;

    }

    return 0;

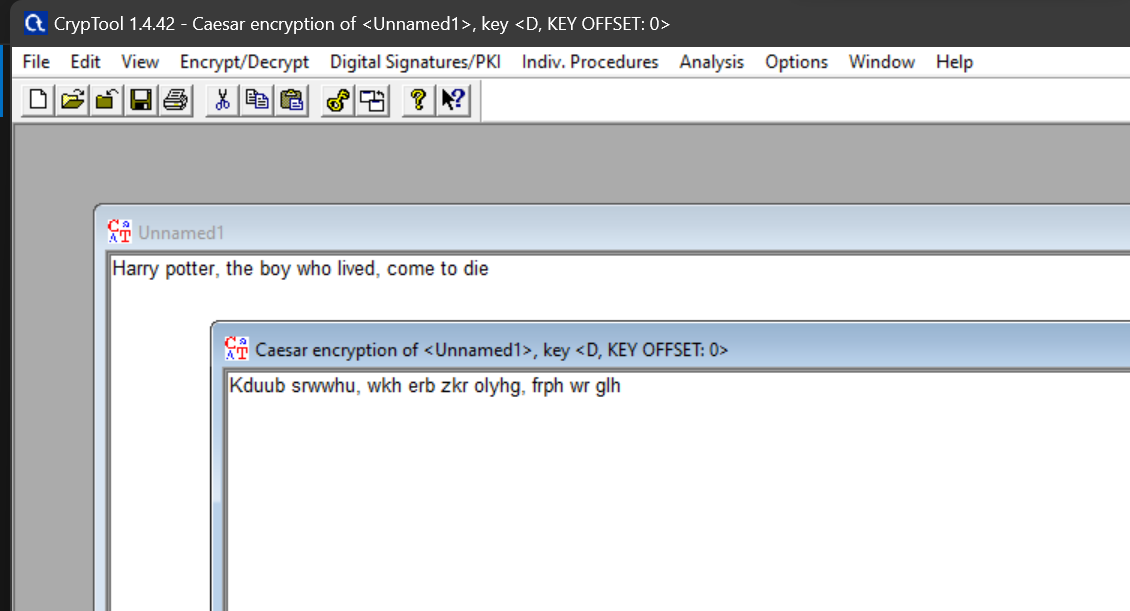
}

**Program Output(IDE):**

**A screen shot of a computer

Description automatically generated**

**Program Output (Cryptool):**

****

**DECRYPTION:**

**A screenshot of a computer

Description automatically generated**

**Cryptanalysis:**

**Time complexity:**

The time complexity of the Caesar cipher is O(n), where 'n' is the length of the input message (the number of characters in the plaintext or ciphertext). The number of characters processed is directly proportional to the length of the input message, the time complexity is linear, or O(n). Regardless of the shift value, the time complexity remains the same because the same amount of work is done for each character.

**Advantages:**

* Simple to implement - it just requires substituting each letter with another shifted letter. Easy to do by hand.
* Fast to encrypt and decrypt a message.
* Provides a slight amount of obfuscation of the original message.

**Disadvantages:**

* Very weak security - there are only 25 possible shifts so it is easy to brute force and crack the cipher.
* The frequency analysis of the cipher text allows the cipher to be cracked easily. More common letters like 'E' and 'T' will still appear frequently.
* The shift is reusable and needs to be protected. If the shift is known, the cipher can be decrypted trivially.
* It only works well for English text, not for other languages or symbols.
* Does not provide confidentiality, integrity, and authenticity in a message.

**Applications:**

* As a simple method to add basic encryption to messages. For example, Julius Caesar is said to have used it to communicate with his generals.
* As a teaching tool to illustrate encryption concepts. It provides a simple substitution cipher to demonstrate core encryption principles.
* To obscure words or passages in media to avoid filters or spoiling plot points.
* To provide a layer of simple obscurity for things like website or product codes.

**References:**

* Wikipedia(https://en.wikipedia.org/wiki/Caesar\_cipher)
* GeeksforGeeks(https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/)

**EXPERIMENT NO: 3**

**Aim:** Study and implement a program for Transposition(Columnar) cipher to encrypt and decrypt the message.

**Introduction:** In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword.

In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.

* The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order.
* Width of the rows and the permutation of the columns are usually defined by a keyword.
* Any spare spaces are filled with nulls or left blank or placed by a character (Example: \_).
* Finally, the message is read off in columns, in the order specified by the keyword.

To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length. Then they can write the message out in columns again, then re-order the columns by reforming the key word.

**Program Code:**

#include <bits/stdc++.h>

using namespace std;

string encryptSimpleColumnTransposition(const string& plainText, const string& key) {

    string encryptedText;

    int keyLength = key.length();

    int textLength = plainText.length();

    int numRows = (textLength + keyLength - 1) / keyLength;

    char matrix[numRows][keyLength];

    int index = 0;

    for (int i = 0; i < numRows; i++) {

        for (int j = 0; j < keyLength; j++) {

            if (index < textLength) {

                matrix[i][j] = plainText[index++];

            } else {

                matrix[i][j] = ' ';

            }

        }

    }

    string sortedKey = key;

    sort(sortedKey.begin(), sortedKey.end());

    int permutationIndex[keyLength];

    for (int i = 0; i < keyLength; i++) {

        permutationIndex[i] = key.find(sortedKey[i]);

    }

    for (int i = 0; i < keyLength; i++) {

        for (int j = 0; j < numRows; j++) {

            encryptedText += matrix[j][permutationIndex[i]];

        }

    }

    return encryptedText;

}

string decryptSimpleColumnTransposition(const string& encryptedText, const string& key) {

    string decryptedText;

    int keyLength = key.length();

    int textLength = encryptedText.length();

    int numRows = (textLength + keyLength - 1) / keyLength;

    char matrix[numRows][keyLength];

    string sortedKey = key;

    sort(sortedKey.begin(), sortedKey.end());

    int permutationIndex[keyLength];

    for (int i = 0; i < keyLength; i++) {

        permutationIndex[i] = key.find(sortedKey[i]);

    }

    int index = 0;

    for (int i = 0; i < keyLength; i++) {

        for (int j = 0; j < numRows; j++) {

            matrix[j][permutationIndex[i]] = encryptedText[index++];

        }

    }

    for (int i = 0; i < numRows; i++) {

        for (int j = 0; j < keyLength; j++) {

            decryptedText += matrix[i][j];

        }

    }

    return decryptedText;

}

int main() {

    string plainText;

    string key;

    cout << "Enter the plain text: ";

    getline(cin, plainText);

    cout << "Enter the key: ";

    getline(cin, key);

    string encryptedText = encryptSimpleColumnTransposition(plainText, key);

    cout << "Encrypted Text: " << encryptedText << endl;

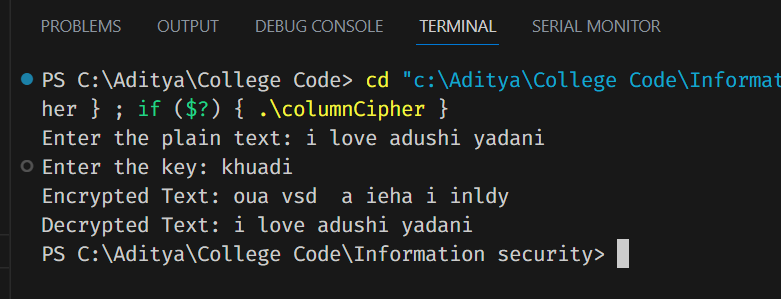
    string decryptedText = decryptSimpleColumnTransposition(encryptedText, key);

    cout << "Decrypted Text: " << decryptedText << endl;

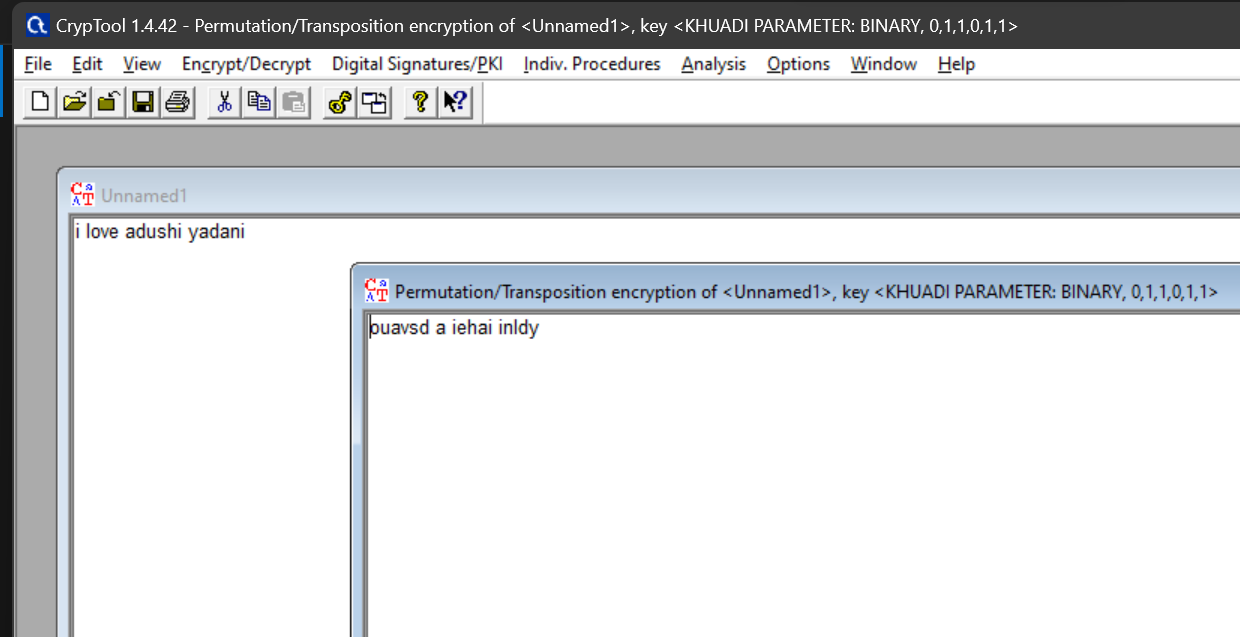
    return 0;

}

**Program Output(IDE):**

****

**Program Output (Cryptool):**

****

**A screenshot of a computer

Description automatically generated**

**Cryptanalysis:**

**Advantages of Columnar Transposition Cipher:**

* **Key Length:** Increasing the key length (number of columns) enhances the security of the cipher. Longer keys make brute force attacks more challenging.
* **Confusion :** The cipher exhibits confusion (hiding the relationship between plaintext and ciphertext) making it resistant to simple attacks.
* **Message Length Hiding:** The use of null characters or spaces to fill in the matrix hides the true length of the message, making it more challenging for attackers to determine the length of the original message.

**Disadvantages of Columnar Transposition Cipher:**

* **Vulnerable to Frequency Analysis:** While the cipher disrupts the frequency analysis of individual letters, longer repeated sequences or known plaintext can still provide enough information for frequency analysis to be effective.
* **Short Key Lengths:** If the key length is short, the number of possible key permutations is limited, making brute force attacks feasible.
* **Patterns in Ciphertext:** Depending on the key arrangement and original plaintext, there might be patterns in the ciphertext that attackers can exploit.

**Time Complexity:**

**Encrypt(encryptSimpleColumnTransposition):**

**Creating a matrix** requires iterating through each cell of the matrix, whose dimensions are **numRows** (indicated by message and key length) and **keyLength** (indicated by key length) This takes **O(numRows \* keyLength)** ) time.

**Formatting a key:** The key formation step takes **O(keyLength \* log(keyLength)**) time, which is the complex time for formatting keyLength objects.

**Creating an encrypted text:** Creating an encrypted text requires iterating through each column of the matrix (**keyLength** column), and for each column, to pass through **numRows** rows this takes **O(keyLength \* numRows)** ) time.

Overall, the main thing is that the matrix is ​​being built, which has a time complexity of **O(numRows \* keyLength).**

**Decrypt ( decryptSimpleColumnChange ):** . Building a matrix: Like encryption, it requires reconstruction of each matrix cell with **numRows** and **keyLength** dimensions, resulting in a time complexity of **O(numRows \* keyLength).**

**Key formation:** Key formation takes **O(keyLength \* log(keyLength)** time.

**Creating decrypted text:** Like encryption, creating decrypted text requires iterating through each row of the matrix (**numRows** rows), and in each row, passing through the **keyLength** columns takes **O(keyLength \* numRows)** time.

Overall, again, the key is building the matrix, which has a time complexity of **O(numRows \* keyLength).**

The time constraint for both encryption and decryption is the creation of the matrix, which takes **O(numRows \* keyLength)** time. Configuring the key and creating encryption or decryption also helps save time, but often.

**Applications:**

* **Steganography:** Steganography is the practice of concealing information within other, seemingly innocuous data. The columnar transposition cipher can be used to create steganographic messages by embedding a secret message within a larger text, making it less conspicuous.
* **Cryptography Education:** While not suitable for secure encryption in modern times, the columnar transposition cipher can be used as a stepping stone to teach basic cryptography concepts before moving on to more complex ciphers and encryption techniques.
* **Basic Message Privacy:** While not suitable for high-security scenarios, the columnar transposition cipher can provide a basic level of privacy for casual communication among individuals who are not well-versed in cryptography.
* **Puzzle and Games:** The cipher can be used in puzzles, escape room challenges, or treasure hunts. Players need to decrypt messages to advance in the game or solve puzzles.

**References:**

* Wikipedia(https://en.wikipedia.org/wiki/Transposition\_cipher)
* GeeksforGeeks(https://www.geeksforgeeks.org/columnar-transposition-cipher/