



# TEB2093 Computer Security - Lab 03

---

## Members

- Ammar Farhan Bin Mohamad Rizam (22006911)
- Amisya Fareezan Binti Mohd Fadhil (22007082)
- Ahmad Anas Bin Azhar (22005996)
- Muhammad Hanis Afifi Bin Azmi (22001602)

## Table of Contents

- [TEB2093 Computer Security - Lab 03](#)
  - [Members](#)
  - [Table of Contents](#)
  - [Task 1](#)
    - [Monoalphabetic substitution cipher](#)
  - [Task 2](#)
    - [Playfair Cipher](#)
  - [Task 3](#)
    - [Vigenere Cipher](#)
  - [Task 4](#)
    - [Rail Fence Cipher](#)

## Task 1

### Monoalphabetic substitution cipher

#### Code:

```
#include <iostream>
#include <string.h>

using namespace std;

int main() {
    cout << "Enter the message:\n";
    char msg[100];
    cin.getline(msg, 100); // take the message as input

    int i, j, length, choice, key;
    cout << "Enter key(0-25): ";
    cin >> key; // take the key as input
```

```
length = strlen(msg);
cout << "Enter your choice \n1. Encryption \n2. Decryption \n";
cin >> choice;

if (choice == 1) { // for encryption
    char ch;

    for(int i = 0; msg[i] != '\0'; ++i) {
        ch = msg[i];

        // encrypt for lowercase letter
        if (ch >= 'a' && ch <= 'z') {
            ch = ch + key;

            if (ch > 'z') {
                ch = ch - 'z' + 'a' - 1;
            }

            msg[i] = ch;
        }

        // encrypt for uppercase letter
        else if (ch >= 'A' && ch <= 'Z') {
            ch = ch + key;

            if (ch > 'Z') {
                ch = ch - 'Z' + 'A' - 1;
            }

            msg[i] = ch;
        }
    }

    printf("Encrypted message: %s", msg);
} else if (choice == 2) { // for decryption
    char ch;
    for(int i = 0; msg[i] != '\0'; ++i) {
        ch = msg[i];

        // decrypt for lowercase letter
        if (ch >= 'a' && ch <= 'z') {
            ch = ch - key;

            if (ch < 'a') {
                ch = ch + 'z' - 'a' + 1;
            }

            msg[i] = ch;
        } else if (ch >= 'A' && ch <= 'Z') { // decrypt for uppercase
letter
            ch = ch - key;

            if (ch < 'A') {
                ch = ch + 'Z' - 'A' + 1;
            }
        }
    }
}
```

```

        }

        msg[i] = ch;
    }
}

cout << "Decrypted message: " << msg;
}
}

```

### Encryption:

```

PS C:\Users\HANIS\Documents\cpp> cd "c:\Users\HANIS\I
Enter the message:
ilovecomputerscience
Enter key(0-25): 3
Enter your choice
1. Encryption
2. Decryption
1
Encrypted message: loryhfrpsxwhuvflhqfh
PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab>

```

### Decryption:

```

PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab> cd "c
{ .\mono }
Enter the message:
loryhfrpsxwhuvflhqfh
Enter key(0-25): 3
Enter your choice
1. Encryption
2. Decryption
2
Decrypted message: ilovecomputerscience
PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab>

```

### Explanation:

This C++ program implements a Caesar Cipher, a type of monoalphabetic substitution cipher, to encrypt and decrypt messages using a shift key. The program first prompts the user to enter a message (msg) and a shift key (key) between 0 and 25, then asks whether they want to encrypt or decrypt the message.

In encryption, each letter is shifted forward by the key value; if it exceeds 'z' (for lowercase) or 'Z' (for uppercase), it wraps around to the beginning of the alphabet, while non-alphabetic characters remain unchanged. Decryption reverses this process by shifting characters backward, ensuring proper wrap-around behavior. The encrypted or decrypted message is then displayed. For example, with a plaintext

"ilovecomputerscience" and key 3, the encrypted text becomes "loryhfrpsxwhuvflhqfh", which when decrypted returns to the original text. Though simple, this cipher introduces basic cryptographic concepts but is easily breakable with frequency analysis

## Task 2

### Playfair Cipher

#### Code:

```
#include <iostream>
#include <string>

using namespace std;

class playfair {
public:
    void doIt(string k, string t, bool ij, bool e) {
        createGrid(k, ij);
        getTextReady(t, ij, e);

        if (e) {
            doIt(1);
        } else {
            doIt(-1);
        }

        display();
    }

private:
    string _txt;
    char _m[5][5];

    void doIt(int dir) {
        int a, b, c, d;
        string ntxt;

        for(string::const_iterator ti = _txt.begin(); ti != _txt.end();
            ti++) {
            if (getCharPos( *ti++, a, b )) {
                if (getCharPos(*ti, c, d)) {
                    if (a == c) {
                        ntxt += getChar(a, b + dir);
                        ntxt += getChar(c, d + dir);
                    } else if (b == d) {
                        ntxt += getChar(a + dir, b);
                        ntxt += getChar(c + dir, d);
                    } else {
                        ntxt += getChar(c, b);
                        ntxt += getChar( a, d );
                    }
                }
            }
        }
    }
};
```

```
        }
    }
    _txt = txt;
}

void display() {
    cout << "\n\n OUTPUT:\n======" << endl;
    string::iterator si = _txt.begin();
    int cnt = 0;

    while(si != _txt.end()) {
        cout << *si;
        si++;
        cout << *si << " ";
        si++;

        if (++cnt >= 26) {
            cout << endl;
            cnt = 0;
        }
    }

    cout << endl << endl;
}

char getChar(int a, int b) {
    return _m[(b + 5) % 5][(a + 5) % 5];
}

bool getCharPos(char l, int &a, int &b) {
    for(int y = 0; y < 5; y++) {
        for( int x = 0; x < 5; x++ ) {
            if (_m[y][x] == l) {
                a = x;
                b = y;
                return true;
            }
        }
    }

    return false;
}

void getTextReady(string t, bool ij, bool e) {
    for(string::iterator si = t.begin(); si != t.end(); si++) {
        *si = toupper(*si);

        if (*si < 65 || *si > 90) {
            continue;
        }

        if (*si == 'J' && ij) {
            *si = 'I';
        }
    }
}
```

```
        } else if (*si == 'Q' && !ij) {
            continue;
        }

        _txt += *si;
    }

    if (e) {
        string ntxt = "";
        size_t len = _txt.length();

        for(size_t x = 0; x < len; x += 2) {
            ntxt += _txt[x];

            if (x + 1 < len) {
                if (_txt[x] == _txt[x + 1]) {
                    ntxt += 'X';
                }

                ntxt += _txt[x + 1];
            }
        }

        _txt = ntxt;
    }

    if (_txt.length() & 1) {
        _txt += 'X';
    }
}

void createGrid( string k, bool ij ) {
    if (k.length() < 1) {
        k = "MONARCHY";
    }

    k += "ABCDEFGHIIJKLMNOPQRSTUVWXYZ"; string nk = "";

    for(string::iterator si = k.begin(); si != k.end(); si++) {
        *si = toupper(*si);

        if (*si < 65 || *si > 90) {
            continue;
        }

        if ((*si == 'J' && ij) || (*si == 'Q' && !ij)) {
            continue;
        }

        if (nk.find( *si ) == -1) {
            nk += *si;
        }
    }
}
```

```

        copy(nk.begin(), nk.end(), &_m[0][0]);
    }
};

int main(int argc, char* argv[]) {
    string key, i, txt;
    bool ij, e;

    cout << "(E)ncode or (D)ecode? ";
    getline(cin, i);
    e = (i[0] == 'e' || i[0] == 'E');

    cout << "Enter a en/decryption key: ";
    getline(cin, key);

    cout << "I <-> J (Y/N): ";
    getline(cin, i);
    ij = (i[0] == 'y' || i[0] == 'Y');

    cout << "Enter the text: ";
    getline(cin, txt);

    playfair pf;
    pf.doIt(key, txt, ij, e);
    return system("pause");
}

```

### Encryption:

```

Decrypted message: Ilovecomputerscience
● PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab> cd
if ($?) { .\playfair }
(E)ncode or (D)ecode? e
Enter a en/decryption key: monarchy
I <-> J (Y/N): j
Enter the text: ilovecomputerscience

OUTPUT:
=====
GS Y0 CY NO KW ZJ AT DF BR YC

Press any key to continue . . .
○ PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab> █

```

### Decryption:

```

Press any key to continue . . .
● PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab> cd
  if ($?) { .\playfair }
(E)ncode or (D)ecode? d
Enter a en/decryption key: monarchy
I <-> J (Y/N): j
Enter the text: gsyocynokwzjatdfbryc

OUTPUT:
=====
IL OV EC OM PU TE RS CI EN CE

Press any key to continue . . .
○ PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab>

```

### Explanation:

Playfair Cipher, which encrypts text using pairs of letters based on a 5x5 grid generated from a keyword. The user chooses to encrypt (E) or decrypt (D), enters a key, selects whether to treat 'I' and 'J' as the same, and provides the message. The program removes duplicates from the key and fills the grid with unique letters (excluding 'Q'). During encryption, letters are paired, and specific Playfair rules shift them within the grid. Decryption reverses these shifts. For example, encrypting "ilovecomputerscience" with key "monarchy" gives "GS YO CY NO KW ZJ AT DF BR YC", which can be decrypted back. This cipher improves security over simple letter shifts.

## Task 3

### Vigenere Cipher

#### Code:

```

#include <iostream>
#include <string.h>

using namespace std;

int main() {
    char msg[] = "IloveComputerScience";
    char key[] = "LEMON";
    int msgLen = strlen(msg), keyLen = strlen(key), i, j;

    char newKey[msgLen], encryptedMsg[msgLen], decryptedMsg[msgLen];

    // generating new key
    for(i = 0, j = 0; i < msgLen; ++i, ++j) {
        if (j == keyLen) {
            j = 0;
        }
    }

```



```

        newKey[i] = key[j];
    }

    newKey[i] = '\0';

    // encryption
    for(i = 0; i < msgLen; ++i) {
        encryptedMsg[i] = ((msg[i] + newKey[i]) % 26) + 'A';
    }

    encryptedMsg[i] = '\0';

    // decryption
    for(i = 0; i < msgLen; ++i) {
        decryptedMsg[i] = (((encryptedMsg[i] - newKey[i]) + 26) % 26) +
'A';
    }

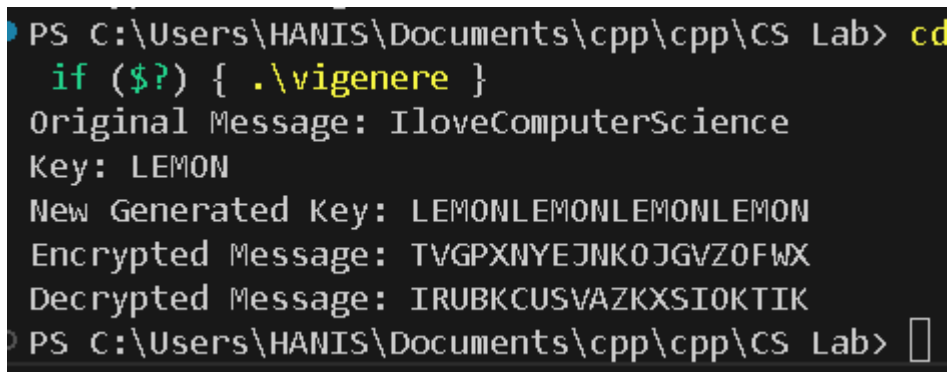
    decryptedMsg[i] = '\0';

    cout << "Original Message: " << msg;
    cout << "\nKey: " << key;
    cout << "\nNew Generated Key: " << newKey;
    cout << "\nEncrypted Message: " << encryptedMsg;
    cout << "\nDecrypted Message: " << decryptedMsg;

    return 0;
}

```

### Encryption:



```

PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab> cd
if ($?) { .\vigenere }
Original Message: IloveComputerScience
Key: LEMON
New Generated Key: LEMONLEMONLEMONLEMON
Encrypted Message: TVGPXNYEJNKOJGVZOFWX
Decrypted Message: IRUBKCU SVAZKXSIQKTIK
PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab>

```

### Explanation:

Vigenere Cipher, an encryption method using a repeating key sequence. The user inputs a message and a key, and the program generates a new key that matches the message length by repeating the original key. Each letter in the message is encrypted by shifting it based on the corresponding letter in the key, and decryption reverses this process. For example, using the key "LEMON" on "IloveComputerScience", the generated key becomes "LEMONLEMONLEMONLEMON", producing an encrypted message "TVGPXNYEJNKOJGVZOFWX" and a decrypted message "IRUBKCU SVAZKXSIQKTIK", though there might be a case mismatch. The Vigenère Cipher is more secure than simple substitution ciphers as it avoids direct frequency analysis.

## Task 4

### Rail Fence Cipher

**Code:**

```
#include <bits/stdc++.h>

using namespace std;

int main() {
    int t, n, m, i, j, k, sum = 0;

    string s;
    cout << "Enter the message" << '\n';
    cin >> s;

    cout << "Enter key" << '\n';
    cin >> n;

    vector<vector<char>> a(n,vector<char>(s.size(),' '));

    j = 0;

    int flag = 0;

    for (i = 0; i < s.size(); i++) {
        a[j][i] = s[i];
        if (j == n - 1) {
            flag = 1;
        } else if (j == 0) {
            flag = 0;
        }

        if (flag == 0) {
            j++;
        } else {
            j--;
        }
    }

    for (i = 0; i < n; i++) {
        for (j = 0; j < s.size(); j++) {
            if (a[i][j] != ' ') {
                cout << a[i][j];
            }
        }
    }
    cout << '\n';
    return 0;
}
```

**Encryption:**

```
PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab> cd  
{ .\rail }  
Enter the message  
ilovecomputerscience  
Enter key  
3  
ieprelvcmuesineootcc  
PS C:\Users\HANIS\Documents\cpp\cpp\CS Lab>
```

**Explanation:**

Rail Fence Cipher, a transposition cipher that rearranges message characters based on a given key (number of rails). The user inputs a message and a key, and the program writes the message in a zigzag pattern across the specified number of rails. It then reads the characters row-wise to form the encrypted message. For example, with the message "ilovecomputerscience" and a key of 3, the output is "ieprelvcmuesineootcc", showing how characters are reordered. The Rail Fence Cipher is simple but effective for basic encryption