KLE Society's

KLE Technological University

Open Ended (OE) Assessment Report

On

ENCRYPTION & DECRYPTION SYSTEM

Object Oriented Programming (20ECSC204)
Object Oriented Programming Lab (20ECSP203)

Submitted by

Name	Roll no	SRN
ABHISHEK AJATDESAI	43	02FE22BCS402
CHINMAY PARANJAPE	44	02FE22BCS403
PRATHAM SHINDE	52	02FE22BCS411
SUJAY PATRA	57	02FE22BCS416
Team Number: 8		

Faculty In-charge:

Prof. Savita Bagewadi

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

Academic year 2022-23

INTRODUCTION

PROBLEM STATEMENT

The provided code implements a message encryption and decryption system using the concepts of object-oriented programming. The user is given the option to encrypt or decrypt a message.

The code supports two encryption algorithms: XOR cipher and a common algorithm. The XOR cipher encryption uses a secret key to perform encryption and decryption, while the common algorithm adds 1 to each character of the message for encryption and subtracts 1 for decryption.

The program prompts the user to enter their choice for encryption or decryption and then asks for the message and optional secret key. Based on the user's inputs, the program creates an appropriate object of either XORCipher or CommonAlgorithm class to perform the encryption or decryption.

Exception handling is implemented to handle potential errors during user input and memory allocation.

The program provides the encrypted or decrypted message as output, depending on the user's choice, and gives the option to continue or exit the program after each encryption or decryption operation.

FEATURES OF APPLICATION:

Message Encryption & Decryption: The application allows the user to encrypt messages using either the XOR cipher algorithm or a common algorithm and then Decrypt the message.

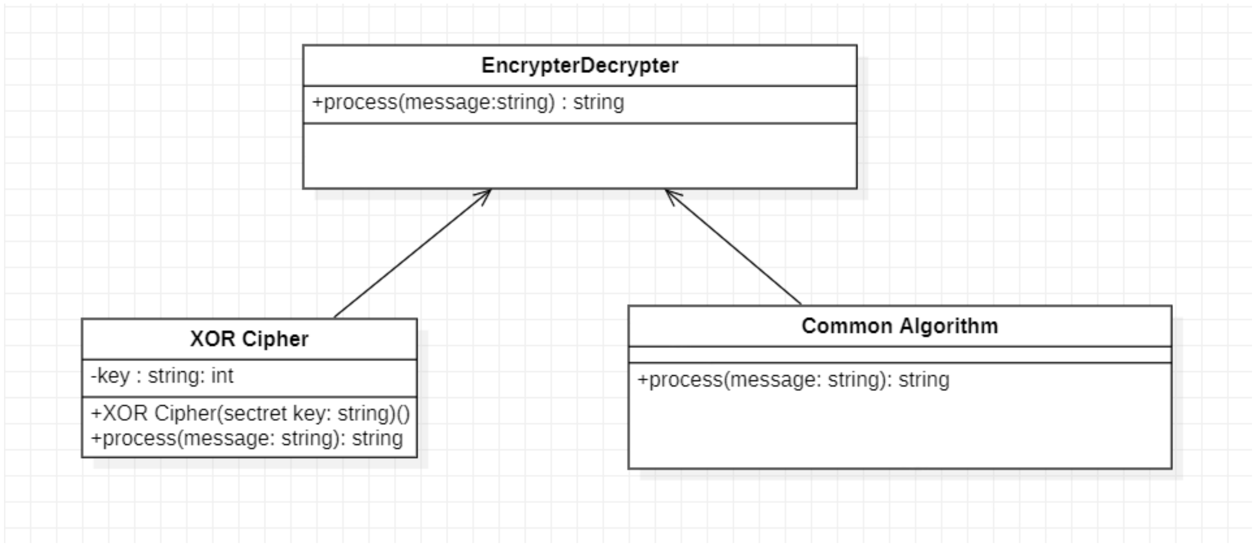
XOR Cipher Encryption: The XOR cipher algorithm provides a secure encryption method using a secret key. The application supports encrypting messages using this algorithm.

Common Algorithm Encryption: The application includes a common encryption algorithm that adds 1 to each character of the message for encryption. This provides a simple encryption method when a secret key is not provided.

User Input Validation: The application validates user input to ensure the correct choice ('E' for encryption or 'D' for decryption) is entered. It also handles invalid inputs and prompts the user to enter a valid choice.

Secret Key Option: The application allows the user to provide a secret key for encryption, which is used in the XOR cipher algorithm. If no secret key is provided, the application uses a common algorithm for encryption.

CLASS DIAGRAM:


EncrypterDecrypter
+process(message:string) : string
XOR Cipher
-key : string: int
+XOR Cipher(sectret key: string)()
+process(message: string): string
Common Algorithm
+process(message: string): string

CLASS DESCRIPTION:

Class EncryptorDecryptor:

Description: This is an abstract base class that defines the interface for encryption and decryption operations. It declares two pure virtual functions, encrypt and decrypt, which are meant to be overridden by derived classes.

Responsibilities: Provides an interface for encryption and decryption operations.

Class XORCipher:

Description: This derived class inherits from EncryptorDecryptor and implements encryption and decryption using the XOR cipher algorithm with a secret key.

Responsibilities: Performs encryption and decryption using the XOR cipher algorithm.

Class CommonAlgorithm:

Description: This derived class also inherits from EncryptorDecryptor and implements encryption and decryption using a common algorithm where each character of the message is shifted by 1.

Responsibilities: Performs encryption and decryption using a common algorithm.

CODE:

```
#include <iostream>
#include <string>
using namespace std;

class EncryptorDecryptor {
public:
    virtual string process(const string& message) = 0;
};

class XORCipher : public EncryptorDecryptor {
private:
    string key;

public:
    XORCipher(const string& secretKey) {
        key = secretKey;
    }

    string process(const string& message) override {
        string processedMessage = message;
        for (int i = 0; i < message.length(); ++i) {
            processedMessage[i] ^= key[i % key.length()];
        }
        return processedMessage;
    }
};

class CommonAlgorithm : public EncryptorDecryptor {
public:
    string process(const string& message) override {
        string processedMessage = message;
        for (int i = 0; i < message.length(); ++i) {
            processedMessage[i] += 1;
        }
        return processedMessage;
    }
};

int main() {
    string message;
    string key;
```

```
char choice;
int ch=1;

while(ch!=0)
{


try {
    cout << "Enter 'E' to encrypt or 'D' to decrypt a message: ";
    cin >> choice;
    cin.ignore();  // Ignore the newline character

    if (choice != 'E' && choice != 'e' && choice != 'D' && choice != 'd') {
        throw invalid_argument("Invalid choice. Please choose 'E' to encrypt or 'D' to decrypt.");
    }

    EncryptorDecryptor* processor = nullptr;

    if (choice == 'E' || choice == 'e') {
        cout << "Enter the message to encrypt: ";
        getline(cin, message);

        cout << "Enter the secret key (leave empty for common encryption): ";
        getline(cin, key);

        if (!key.empty()) {
            processor = new XORCipher(key);
        } else {
            processor = new CommonAlgorithm();
        }

        string encrypted = processor->process(message);
        cout << "Encrypted message: " << encrypted << endl;
    } else {
        cout << "Enter the message to decrypt: ";
        getline(cin, message);

        cout << "Enter the secret key (leave empty for common encryption): ";
        getline(cin, key);

        if (!key.empty()) {
            processor = new XORCipher(key);
        } else {
            processor = new CommonAlgorithm();
        }
```

```
            string decrypted = processor->process(message);
            cout << "Decrypted message: " << decrypted << endl;
        }

        delete processor;
    } catch (const exception& e) {
        cerr << "Error: " << e.what() << endl;
    }

    cout<<"\nPress 1 to Continue ; Press 0 to Exit\n"<<endl;
    cin>>ch;
    }
    return 0;
}     cin >> employeeID;

    cout << "Enter waiter's salary: ";
    cin >> salary;

    cout << "Enter the number of tables served by the waiter: ";
    cin >> tablesServed;

    cout << endl;

    Employee* waiter = new Waiter(name, employeeID, salary, tablesServed);

    cout << "Enter chef's name: ";
    cin.ignore();
    getline(cin, name);

    cout << "Enter chef's employee ID: ";
    cin >> employeeID;

    cout << "Enter chef's salary: ";
    cin >> salary;

    cout << "Enter the number of meals cooked by the chef: ";
    cin >> mealsCooked;

    cout << endl;



Employee* chef = new Chef(name, employeeID, salary, mealsCooked);
```

```
    cout << "Enter manager's name: ";
    cin.ignore();
    getline(cin, name);

    cout << "Enter manager's employee ID: ";
    cin >> employeeID;

    cout << "Enter manager's salary: ";


    cin >> salary;

    cout << endl;

    Employee* manager = new Manager(name, employeeID, salary);

    cout << endl;

    cout << "----- Employee Information -----" << endl;

    waiter->displayInformation();
    cout << "Total Pay: $" << waiter->calculatePay() << endl;
    cout << endl;

    chef->displayInformation();
    cout << "Total Pay: $" << chef->calculatePay() << endl;
    cout << endl;

    manager->displayInformation();
    cout << "Total Pay: $" << manager->calculatePay() << endl;

    delete waiter;
    delete chef;
    delete manager;

    return 0;
}
```

OUTPUT:

```
Enter 'E' to encrypt or 'D' to decrypt a message: E

Enter the message to encrypt: OOP with C++

Enter the secret key (leave empty for common encryption): 11

Encrypted message: ~~a◀FXEY◀r→→


Press 1 to Continue ; Press 0 to Exit


1

Enter 'E' to encrypt or 'D' to decrypt a message: D

Enter the message to decrypt: ~~a◀FXEY◀r→→

Enter the secret key (leave empty for common encryption): 11

Decrypted message: OOP with C+
```