CIS 7 Project Documentation

1. Team Name: Mehak Lohchan

2. Project Information

This project aims to solve the problem of decrypting messages that are encrypted using the Vigenère Cipher. The Vigenère Cipher is a polyalphabetic substitution cipher that utilizes a repeating keyword to encrypt alphabetic text. The process of encryption involves using the Vigenère table, which consists of letters of the alphabets written out twenty-six times in different rows and are shifted to the left. The main problem that this project will aim for is the decryption of messages encrypted with the Vigenère Cipher. Given a cipher-text and the corresponding key-word that will be used to encrypt the message from the user, the project provides a method to decrypt the cipher-text and retrieve the original plaintext message. The decryption process consists of finding the row in the Vigenère table that corresponds to the key-word, locates the position of each cipher-text in the row, and uses the columns label as the plaintext. The process is repeated for each of the letters in the message and key-word, which will then decrypt the entire message. Additionally, the project provides an implementation of the ASCII table which uses numerical values for letters of the alphabet and converts them into numbers. To do so, I will be implementing conditional and iterative loops to perform arithmetic operations for certain ranges of the ASCII table, followed by modulo 26 given that there are twenty-six characters of the alphabet. The project offers a solution that will allow users to decipher messages encrypted with the Vigenère Cipher, enabling them to retrieve the original plaintext and comprehend the content of the initial encoded message. In this project, discrete structures play a fundamental role in implementing the Vigenère Cipher. A structure such as the Vigenère table, is represented as a two-dimensional array, which means that there consists a row and a column. The Vigenère Cipher table facilitates the association between plaintext and cipher text letters throughout the encryption and decryption processes. Additionally. modular arithmetic is also utilized to handle the mathematical operations. By utilizing modular arithmetic operations, the project ensures that the range does not go out of bounds and remains in the range of the alphabet. These discrete structures and concepts form the foundation of the Vigenère Cipher project. However, with every program, comes limitations. One limitation to take into consideration is key length. The program assumes that the length of the keyword that is used for encryption is something that can be determined. If the exact length of the keyword is not provided, given that the keyword can be repeated as many times to match the length of the message, it becomes challenging to accurately decrypt the cipher-text. A way to improve the limitations of the program can be by expanding the program to support a larger and more broader range of characters for the keyword, including numbers, symbols, and non-alphabetic characters. Doing this will increase the key space and strengthen the encryption. This expansion would make it more difficult for attackers to decipher the cipher-text. Another way to improve the limitations can be by implementing efficient data structures for tasks such as keyword repetition and modular arithmetic operations. By implementing more structured algorithms, this can improve the programs efficiency.

3. Flowchart/Pseudocode

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

using namespace std;

// function checks if a given string only contains alphabetic letters
function isAlphabetic(str):
    for each character c in str:
        if c is not an alphabetic character:
            return false
    return true

// this function performs the encryption process
// it takes plaintext and keyword, and returns cipher text
function encrypt(plaintext, keyword):
    ciphertext = ""
    keyLen = length of keyword
    plainLen = length of plaintext

    for i = 0 to plainLen - 1:
        plainChar = plaintext[i]
        keyChar = keyword[i % keyLen]

        plainChar = convert plainChar to uppercase
        keyChar = convert keyChar to uppercase

        if plainChar is an alphabetic character:
            cipherChar = ((plainChar - 'A') + (keyChar - 'A')) mod 26 + 'A'
            append cipherChar to ciphertext
        else:
            append plainChar to ciphertext

    return ciphertext

// driver program — prompts user to enter input
function main():
    plaintext = ""
    keyword = ""

    while true:
        output "Enter the plaintext: "
        read plaintext
        if not isAlphabetic(plaintext):
            output "Invalid input! Please enter alphabetic characters only."
        else:
            break

    while true:
        output "Enter the keyword: "
        read keyword

        if not isAlphabetic(keyword):
            output "Invalid input! Please enter alphabetic characters only."
        else:
            break

    ciphertext = encrypt(plaintext, keyword)
    output "Ciphertext: " + ciphertext

    return 0
```