**Cybersecurity Lab 1**

**Problem Statement – Implement Ceasar Cipher**

**Name –** Aaditya M. Patil **RollNo -** 58

**PRN –** 12210643

**Theoretical Explanation**

The Caesar cypher is a simple substitution cypher where each letter in the plaintext is shifted a certain number of places down or up the alphabet.

1. **Encryption Process**:
   * Each letter in the plaintext is shifted by a fixed number of positions (key) down the alphabet.
   * For example, with a shift of 3, 'A' becomes 'D', 'B' becomes 'E', ..., 'X' becomes 'A', 'Y' becomes 'B', and 'Z' becomes 'C'.
2. **Decryption Process**:
   * To decrypt, you shift each letter in the ciphertext in the opposite direction of the encryption shift.
3. **Key Management**:
   * The key specifies the number of positions each letter should be shifted.
   * In the Caesar cypher, the key is an integer from 0 to 25 (since there are 26 letters in the English alphabet).
4. **Security**:
   * Caesar cypher is vulnerable to brute-force attacks due to its small key space.
   * It can be easily broken using frequency analysis and known plaintext attacks.

**Code**

#include <iostream>

#include <string>

using namespace std;

// Function to encrypt plaintext using Caesar cipher

string encrypt(string plaintext, int key) {

    string ciphertext = "";

    for (char& ch : plaintext) {

        if (isalpha(ch)) {  // Process only alphabetic characters

            char base = isupper(ch) ? 'A' : 'a';

            ch = static\_cast<char>((ch - base + key) % 26 + base);  // Shift character

        }

        ciphertext += ch;

    }

    return ciphertext;

}

// Function to decrypt ciphertext using Caesar cipher

string decrypt(string ciphertext, int key) {

    return encrypt(ciphertext, 26 - key);  // Decrypt by shifting in opposite direction

}

int main() {

    string plaintext = "Hello, World!";

    int key = 3;

    // Encrypt plaintext

    string encrypted = encrypt(plaintext, key);

    cout << "Encrypted: " << encrypted << endl;

    // Decrypt ciphertext

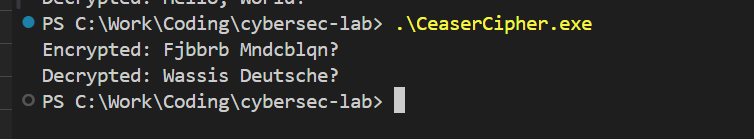
    string decrypted = decrypt(encrypted, key);

    cout << "Decrypted: " << decrypted << endl;

    return 0;

}

**Output**

****