## Name: **Ketan Kunkalikar**

## Roll: **21CSE1016**

## Lab Subject: **Security Lab**

## Topic: **Shift Ciphers**

## Instructor: **Dr. Meenakshi Panda**

## Date: **11 September 2024**

## **What exactly are shift ciphers?**

## We learnt about Shift Ciphers in class today and were tasked with implementing encrypting/decrypting programs using the Shift Cipher.

## Shift ciphers, also known as Caesar ciphers, are a form of substitution cipher in cryptography. In this method, each letter in the plaintext is replaced by a letter a fixed number of positions down the alphabet. This "shift" value serves as the encryption key. For example, with a shift of 3, 'A' would be replaced by 'D', 'B' would become 'E', and so on. The cipher wraps around the alphabet, so 'X' would become 'A'.

## It is the first encryption algorithm we have studied so far, and it is easy to bruteforce it. Hence, the Caesar cipher is not used a lot today.

// 1. Write a program to encrypt the message "Are you Ready for class" using shift cipher with any key value. Then decrypt the message back to plain text.

#include <bits/stdc++.h>

using namespace std;

// Function to encrypt a message using shift cipher

string encrypt(const string &message, int key)

{

    string encrypted\_message = "";

    for (char c : message)

    {

        if (isalpha(c))

        {

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

            encrypted\_message += static\_cast<char>((c - base + key) % 26 + base);

        }

        else if (isdigit(c))

        {

            encrypted\_message += static\_cast<char>((c - '0' + key) % 10 + '0');

        }

        else

        {

            encrypted\_message += c;

        }

    }

    return encrypted\_message;

}

// Function to decrypt a message encrypted with shift cipher

string decrypt(const string &encrypted\_message, int key)

{

    string decrypted\_message = "";

    for (char c : encrypted\_message)

    {

        if (isalpha(c))

        {

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

            decrypted\_message += static\_cast<char>((c - base - key + 26) % 26 + base);

        }

        else if (isdigit(c))

        {

            decrypted\_message += static\_cast<char>((c - '0' - key + 10) % 10 + '0');

        }

        else

        {

            decrypted\_message += c;

        }

    }

    return decrypted\_message;

}

int main()

{

    string original\_message;

    cout << "Enter the message to encrypt: ";

    getline(cin, original\_message);

    int key;

    cout << "Enter the key value: ";

    cin >> key;

    // Encrypt the message

    string encrypted\_message = encrypt(original\_message, key);

    // Decrypt the message

    string decrypted\_message = decrypt(encrypted\_message, key);

    // Output results

    cout << "Original message: " << original\_message << "\n";

    cout << "Encrypted message: " << encrypted\_message << "\n";

    cout << "Decrypted message: " << decrypted\_message << "\n";

    return 0;

}

## OUTPUT:

## INPUT: Are you Ready for class

## 

## INPUT: I transferred Rs 2034 to you

## 

// 2. Write a program to find the key value of the given cipher text (JBCRCLQRWCRVNBJENBWRWN)

#include <bits/stdc++.h>

using namespace std;

// Function to decrypt a message with a given key

string decrypt(const string& encrypted\_message, int key) {

    string decrypted\_message = "";

    for (char c : encrypted\_message) {

        if (isalpha(c)) {

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

            decrypted\_message += static\_cast<char>((c - base - key + 26) % 26 + base);

        } else {

            decrypted\_message += c;

        }

    }

    return decrypted\_message;

}

// Function to calculate the frequency of each letter in a string

vector<int> calculateFrequency(const string& text) {

    vector<int> freq(26, 0);

    for (char c : text) {

        if (isalpha(c)) {

            freq[toupper(c) - 'A']++;

        }

    }

    return freq;

}

// Function to find the most likely key based on letter frequency

int findLikelyKey(const string& ciphertext) {

    vector<int> freq = calculateFrequency(ciphertext);

    int max\_freq = \*max\_element(freq.begin(), freq.end());

    int most\_common = distance(freq.begin(), find(freq.begin(), freq.end(), max\_freq));

    // Assuming that 'E' is the most common letter in English

    return (most\_common - ('E' - 'A') + 26) % 26;

}

int main() {

    const string ciphertext = "JBCRCLQRWCRVNBJENBWRWN";

    int likely\_key = findLikelyKey(ciphertext);

    string decrypted = decrypt(ciphertext, likely\_key);

    cout << "Ciphertext: " << ciphertext << "\n";

    cout << "Most likely key: " << likely\_key << "\n";

    cout << "Decrypted message: " << decrypted << "\n";

    cout << "\nAll possible decryptions:\n";

    for (int key = 0; key < 26; key++) {

        cout << "Key " << key << ": " << decrypt(ciphertext, key) << "\n";

    }

    return 0;

}

## OUTPUT:

## 

## NOTE: I have used a frequency analysis function over here, which assumes that the most common letter in the English Alphabet is ‘e’, and based on this clue, finds a key value which is the right key value.

## However, I learnt that this does not give the right answer.

## My T.A. then told me that frequency analysis is not very effective for small bodies of text, and can produce wrong results.

## **By observation alone, I can determine that from all the decryptions, the one with key value = 9 is the most appropriate, and is hence the right key value.**

// Q3. Repeat question 1, using the substitution cipher.

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

using namespace std;

// Function to generate a random substitution key

string generateKey() {

    string key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    srand(time(0));  // Seed the random number generator

    for (int i = 25; i > 0; i--) {

        int j = rand() % (i + 1);

        swap(key[i], key[j]);

    }

    return key;

}

// Function to encrypt a message using substitution cipher

string encrypt(const string& message, const string& key) {

    string encrypted\_message = "";

    for (char c : message) {

        if (isalpha(c)) {

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

            encrypted\_message += isupper(c) ? key[c - 'A'] : tolower(key[c - 'a']);

        } else {

            encrypted\_message += c;

        }

    }

    return encrypted\_message;

}

// Function to decrypt a message encrypted with substitution cipher

string decrypt(const string& encrypted\_message, const string& key) {

    string decrypted\_message = "";

    for (char c : encrypted\_message) {

        if (isalpha(c)) {

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

            size\_t index = key.find(toupper(c));

            decrypted\_message += (base == 'A') ? 'A' + index : 'a' + index;

        } else {

            decrypted\_message += c;

        }

    }

    return decrypted\_message;

}

int main() {

    const string original\_message = "Are you Ready for class";

    string key = generateKey();

    // Encrypt the message

    string encrypted\_message = encrypt(original\_message, key);

    // Decrypt the message

    string decrypted\_message = decrypt(encrypted\_message, key);

    // Output results

    cout << "Original message: " << original\_message << "\n";

    cout << "Key: " << key << "\n";

    cout << "Encrypted message: " << encrypted\_message << "\n";

    cout << "Decrypted message: " << decrypted\_message << "\n";

    return 0;

}

## **OUTPUT:**

## 