|  |  |
| --- | --- |
| **Academic Year: 2024-25** | **Programme: BTECH CSE Cybersecurity** |
| **Year: 2nd** | **Semester: IV** |
| **StudentName: Jal Bafana** | **Batch: K1** |
| **Roll No: K005** | **Date of experiment: 02.01.2025** |
| **Faculty: Rejo Mathew** | **Signature with Date:** |

Experiment 1: Caesar Cipher

**Aim:**Toimplementshiftciphersandtostudyvarioustermsrelatedtocryptography.

# LearningOutcomes:

Aftercompletionofthisexperiment,studentshouldbeableto

1. Describebasicsymmetrickeyencryptionanddecryptionprocess.
2. Understandtheworkingofsubstitutionciphers.
3. Statelimitationsofshiftciphers.
4. Implementbruteforceattack

# Theory:

Followingarethebasictermsrelatedtocryptography

* PlainText:theoriginalinformation.
* Ciphertext:unintelligiblegibberishdata.Itistheoutputofencryptionprocess
* Encryptionistheprocessofconvertingordinaryinformation(plaintext)intounintelligiblegibberish(i.e.,ciphertext).
* Decryptionisthe reverse,in other words,moving fromtheunintelligible ciphertextbacktoplaintext.
* A cipher (or cipher) is a pair of algorithms which create the encryption and the reversingdecryption.Thedetailedoperationofacipheriscontrolledbothbythealgorithmandineachinstancebyakey.
* Key: This is a secret parameter (ideally known only to the communicants) for a specificmessageexchangecontext.Itisadiscreetdatasetthatcontroltheoperationofcryptographic algorithm

In cryptography, a **Caesar cipher**, also known as **Caesar's cipher**, the **shift cipher**, **Caesar'scode**or**Caesarshift**,isoneofthesimplestandmostwidelyknownencryptiontechniques.Itis a type of substitution cipher in which each letter in the plaintext is replaced by a letter somefixed number of positions down the alphabet. The method is named after Julius Caesar, whouseditinhisprivatecorrespondence.

The encryption canalsoberepresentedusing modulararithmetic by firsttransformingthelettersintonumbers,accordingtothescheme,A=0,B=1,...,Z=25.Encryptionofaletterbyashift*n*canbedescribedmathematicallyas,



Decryptionisperformedsimilarly,



Intheabove,theresultisintherange0...25.If*x+n*or*x-n*arenotintherange0...25,wehaveto subtract or add 26. The replacement remains the same throughout the message, so the cipherisclassedas atypeof*monoalphabeticsubstitution.*

*Example:*

PlaintextisHELLOWORLD

Keyis3.Changeeachlettertothethirdletterfollowingit(XgoestoA,YtoB,ZtoC)CiphertextisKHOORZRUOG

# Procedure:

**PartA:Encryption**

1. Acceptkeyandplaintextfromtheuser.
2. Encryptplaintextusingsubstitutioncipher
3. Outputciphertext

# PartB:Decryption

1. Acceptkeyandciphertextfromtheuser.
2. Decryptciphertextusingsubstitutioncipher
3. OutputPlaintext

# PartC:Bruteforceattack

1. Acceptciphertextfromtheuser.
2. Decryptciphertextusingsubstitutioncipher
3. OutputPlaintextforallpossiblekey

**Algorithm for Caesar Cipher:**

**Encryption**

1. Input:

- Plaintext message.

- A shift value (key) (k).

2. Process:

- For each character in the plaintext:

1. Check if the character is a letter.

2. Shift the character by (k) positions forward in the alphabet.

3. Wrap around to the beginning of the alphabet if the shift exceeds 'Z' (for uppercase) or 'z' (for lowercase).

4. Leave non-alphabet characters unchanged.

3. Output:

- Ciphertext message.

**Decryption**

1. Input:

- Ciphertext message.

- The same shift value (key) (k) used for encryption.

2. Process:

- For each character in the ciphertext:

1. Check if the character is a letter.

2. Shift the character by (k) positions backward in the alphabet.

3. Wrap around to the end of the alphabet if the shift goes below 'A' (for uppercase) or 'a' (for lowercase).

4. Leave non-alphabet characters unchanged.

3. Output:

- Decrypted plaintext message.

**Code: *type or copy your completed working code here***

*Note: Code should have proper comments*

**Encryption:**

#include<iostream>

#include<string.h>

using namespace std;

int main()

{

char ptext[100];

int key,len;

cout<<"Enter plain text = ";

cin.getline(ptext,100);

cout<<"\n Enter Key = ";

cin>>key;

len = strlen(ptext);

char ch;

for(int i = 0 ; ptext[i] != '\0' ; i++)

{

ch = ptext[i];

if (ch>= 'a' &&ch<= 'z')

{

ch = ch + key;

if (ch> 'z')

ch = ch - 'z' + 'a' - 1;

ptext[i] = ch;

}

else if (ch>= 'A' &&ch<= 'Z')

{

ch = ch + key;

if (ch> 'Z')

ch = ch - 'Z' + 'A' - 1;

ptext[i] = ch;

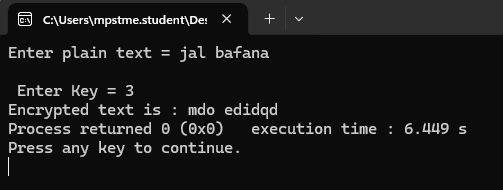
}

}

cout<<"Encrypted text is : "<<ptext;

return 0;

}

**

**Decryption:**

#include<iostream>

#include<string.h>

using namespace std;

int main()

{

char ctext[100];

int key,len;

cout<<"Enter cipher text = ";

cin.getline(ctext,100);

cout<<"\n Enter Key = ";

cin>>key;

len = strlen(ctext);

char ch;

for(int i = 0 ; ctext[i] != '\0' ; i++)

{

ch = ctext[i];

if (ch>= 'a' &&ch<= 'z')

{

ch = ch - key;

if (ch< 'a')

ch = ch + 'z' - 'a' + 1;

ctext[i] = ch;

}

else if (ch>= 'A' &&ch<= 'Z')

{

ch = ch - key;

if (ch< 'A')

ch = ch + 'Z' - 'A' + 1;

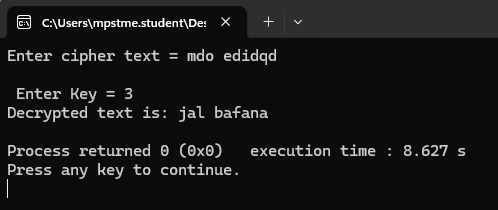
ctext[i] = ch;

}

}

cout<<"Decrypted text is: "<<ctext<<endl;

}



**Brute Force:**

#include<iostream>

#include<string.h>

using namespace std;

int main()

{

char ctext[100];

int key,len;

cout<<"Enter cipher text = ";

cin.getline(ctext,100);

len = strlen(ctext);

char ch;

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

{

for(int i = 0 ; ctext[i] != '\0' ; i++)

{

ch = ctext[i];

if (ch>= 'a' &&ch<= 'z')

{

ch = ch - key;

if (ch< 'a')

ch = ch + 'z' - 'a' + 1;

ctext[i] = ch;

}

else if (ch>= 'A' &&ch<= 'Z')

{

ch = ch - key;

if (ch< 'A')

ch = ch + 'Z' - 'A' + 1;

ctext[i] = ch;

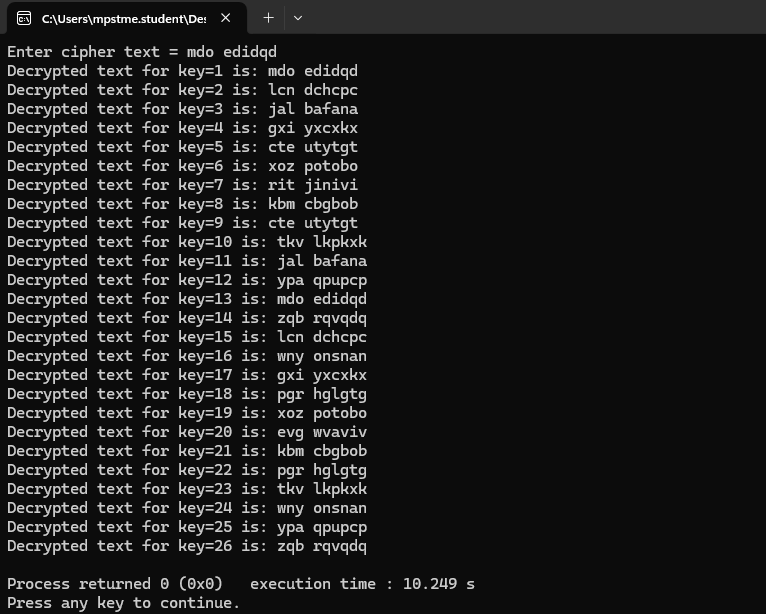
}

}

cout<<"Decrypted text for key="<< key+1 <<" is: "<<ctext<<endl;

}

}



**Questions:**

1. Describe the characteristics of a secure cryptographic algorithm?

A secure cryptographic algorithm has the following characteristics:

* **Confidentiality**: Ensures data is only accessible to authorized users.
* **Integrity**: Guarantees that data has not been altered during transit or storage.
* **Authentication**: Confirms the identity of the entities involved in the communication.
* **Non-repudiation**: Ensures that actions cannot be denied by the sender or receiver.
* **Key Strength**: Relies on complex and long keys that are computationally infeasible to guess or break.
* **Resilience to Attacks**: Designed to resist various attacks like brute force, replay, and side-channel attacks.
* **Efficiency**: Balances security with performance to work well in practical scenarios.
* **Randomness**: Generates unpredictable output, reducing the chance of pattern discovery.

1. What are some common tools used for brute force attacks, and how do they function?

Common tools for brute force attacks include:

* **Hydra**: A fast and flexible network login cracker that supports many protocols such as SSH, HTTP, and FTP.
* **John the Ripper**: A password-cracking tool that combines dictionary attacks with brute force to break encrypted passwords.
* **Hashcat**: A GPU-based password recovery tool that efficiently cracks hashed passwords.
* **Medusa**: A parallel, modular login brute-forcer designed for speed and reliability.
* **Aircrack-ng**: Specifically used for cracking Wi-Fi passwords by brute-forcing WPA/WPA2 keys. These tools function by systematically trying every possible combination of characters or using a pre-defined list of common passwords (dictionary) to gain unauthorized access.

1. How can frequency distribution analysis be used to identify patterns in ciphertext?

Frequency distribution analysis exploits the non-random nature of certain encryption methods. Steps include:

* Analyze the ciphertext for the frequency of characters or groups of characters (e.g., digraphs or trigraphs).
* Compare these frequencies with known language patterns, such as the frequency of 'e' in English text.
* Identify substitution patterns or repeating sequences that can hint at the encryption method or key used.
* This technique is particularly effective against classical ciphers like Caesar or Vigenère, which do not sufficiently randomize frequencies.

1. What are the key characteristics of a brute force attack, and how does it differ from a dictionary attack?

**Brute Force Attack:**

* **Exhaustive Approach**: Tests all possible combinations of characters to guess the password or key.
* **Time-Intensive**: Takes significant time for longer keys or complex passwords.
* **Guaranteed Success**: Will always find the password if given enough time and computational power.

**Dictionary Attack:**

* **Predefined List**: Uses a list of common words, phrases, or passwords to guess the correct one.
* **Faster but Limited**: Faster than brute force but limited to the entries in the dictionary.
* **No Guarantee**: Success depends on whether the target password exists in the list.

**Key Difference**: Brute force attacks are exhaustive and guarantee success, while dictionary attacks are faster but limited to a pre-existing list.

1. How can statistical analysis be used to detect anomalies in network traffic that may indicate a brute force attack?

Statistical analysis can help detect brute force attacks by identifying deviations from normal network behavior:

* **Connection Rate**: A high number of login attempts or connections within a short timeframe.
* **Failed Login Attempts**: An unusually high number of failed login attempts for specific accounts or services.
* **IP Address Patterns**: Multiple attempts from the same or a small range of IP addresses.
* **Payload Size**: Unusual sizes of login requests may indicate automated scripts.
* **Traffic Spike**: A sudden spike in traffic to authentication services can signal brute force activity. By monitoring and analyzing these patterns, security tools can generate alerts for suspected brute force attempts.

**Conclusion:** In the lab, we learned about the Caesar cipher, its encryption and decryption processes, and explored how brute force attack techniques can be used to break such ciphers. This provided insight into the vulnerabilities of simple encryption methods and the importance of stronger cryptographic algorithms.