Academic Year: 2024-25	**Programme: B.Tech CSE Cybersecurity**
Year: 2nd	**Semester: IV**
Student Name: Arjun Mehta	**Batch: K1**
Roll No: K036	**Date of experiment: 02.01.2025**
Faculty: Rejo Mathew	**Signature with Date:**

Experiment 1: Caesar Cipher

Aim: To implement shift ciphers and to study various terms related to cryptography.

Learning Outcomes:
After completion of this experiment, student should be able to

1. Describe basic symmetric key encryption and decryption process.
2. Understand the working of substitution ciphers.
3. State limitations of shift ciphers.
4. Implement brute force attack

Theory:
Following are the basic terms related to cryptography

- Plain Text: the original information.
- Cipher text: unintelligible gibberish data. It is the output of encryption process
- Encryption is the process of converting ordinary information (plaintext) into unintelligible gibberish (i.e., cipher text).
- Decryption is the reverse, in other words, moving from the unintelligible ciphertext back to plaintext.
- A cipher (or cipher) is a pair of algorithms which create the encryption and the reversingdecryption. The detailed operation of a cipher is controlled both by the algorithm andin each instance by a key.
- Key: This is a secret parameter (ideally known only to the communicants) for a specific message exchange context. It is a discreet data set that control the operation of cryptographic algorithm

In cryptography, a **Caesar cipher**, also known as **Caesar's cipher**, the **shift cipher**, **Caesar's code** or **Caesar shift**, is one of the simplest and most widely known encryption techniques. Itis a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. The method is named after Julius Caesar, who used it in his private correspondence.

The encryption can also be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,..., Z = 25. Encryption of a letter by a shift *n* can be described mathematically as,

$$E_n(x) = (x + n) \mod 26.$$

Decryption is performed similarly,

$$D_n(x) = (x - n) \mod 26.$$

In the above, the result is in the range 0...25. If *x+n* or *x-n* are not in the range 0...25, we have to subtract or add 26. The replacement remains the same throughout the message, so the cipher is classed as a type of *monoalphabetic substitution.*

Example:
Plaintext is HELLO WORLD
Key is 3. Change each letter to the third letter following it (X goes to A, Y to B, Z to C)
Ciphertext is KHOOR ZRUOG

Procedure:

Part A: Encryption

1. Accept key and plain text from the user.
2. Encrypt plaintext using substitution cipher
3. Output cipher text

Part B: Decryption

1. Accept key and cipher text from the user.
2. Decrypt cipher text using substitution cipher
3. Output Plain text

Part C: Brute force attack

1. Accept cipher text from the user.
2. Decrypt cipher text using substitution cipher
3. Output Plain text for all possible key

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

In [1]:
```
#Encryption

ptext = input("Enter plain text: ")
key = int(input("Enter Key: "))
ptext = list(ptext)

for i in range(len(ptext)):
    ch = ptext[i]

    if 'a' <= ch <= 'z':
        ch = chr(((ord(ch) - ord('a') + key) % 26) + ord('a'))
        ptext[i] = ch
    elif 'A' <= ch <= 'Z':
        ch = chr(((ord(ch) - ord('A') + key) % 26) + ord('A'))
        ptext[i] = ch

encrypted_text = ''.join(ptext)
print("Encrypted text is:", encrypted_text)
```

```
Enter plain text: Arjun Mehta
Enter Key: 3
Encrypted text is: Dumxq Phkwd
```

In [2]:
```
#Decryption

ctext = input("Enter cipher text: ")
key = int(input("Enter Key: "))
ctext = list(ctext)

for i in range(len(ctext)):
    ch = ctext[i]

    if 'a' <= ch <= 'z':
        ch = chr(((ord(ch) - ord('a') - key) % 26) + ord('a'))
        ctext[i] = ch
    elif 'A' <= ch <= 'Z':
        ch = chr(((ord(ch) - ord('A') - key) % 26) + ord('A'))
        ctext[i] = ch

decrypted_text = ''.join(ctext)
print("Decrypted text is:", decrypted_text)
```

```
Enter cipher text: Dumxq Phkwd
Enter Key: 3
Decrypted text is: Arjun Mehta
```

```
In [3]: #Brute Force

ctext = input("Enter cipher text: ")
ctext = list(ctext)

for key in range(26):
    for i in range(len(ctext)):
        ch = ctext[i]

        if 'a' <= ch <= 'z':
            ch = chr(((ord(ch) - ord('a') - (key+1)) % 26) + ord('a'))
            ctext[i] = ch
        elif 'A' <= ch <= 'Z':
            ch = chr(((ord(ch) - ord('A') - (key+1)) % 26) + ord('A'))
            ctext[i] = ch

    decrypted_text = ''.join(ctext)
    print(f"Decrypted text for key={key+1} is: {decrypted_text}")
```

```
Enter cipher text: Dumxq Phkwd
Decrypted text for key=1 is: Ctlwp Ogjvc
Decrypted text for key=2 is: Arjun Mehta
Decrypted text for key=3 is: Xogrk Jbeqx
Decrypted text for key=4 is: Tkcng Fxamt
Decrypted text for key=5 is: Ofxib Asvho
Decrypted text for key=6 is: Izrcv Umpbi
Decrypted text for key=7 is: Bskvo Nfiub
Decrypted text for key=8 is: Tkcng Fxamt
Decrypted text for key=9 is: Kbtex Wordk
Decrypted text for key=10 is: Arjun Mehta
Decrypted text for key=11 is: Pgyjc Btwip
Decrypted text for key=12 is: Dumxq Phkwd
Decrypted text for key=13 is: Qhzkd Cuxjq
Decrypted text for key=14 is: Ctlwp Ogjvc
Decrypted text for key=15 is: Newha Zrugn
Decrypted text for key=16 is: Xogrk Jbeqx
Decrypted text for key=17 is: Gxpat Sknzg
Decrypted text for key=18 is: Ofxib Asvho
Decrypted text for key=19 is: Vmepi Hzcov
Decrypted text for key=20 is: Bskvo Nfiub
Decrypted text for key=21 is: Gxpat Sknzg
Decrypted text for key=22 is: Kbtex Wordk
Decrypted text for key=23 is: Newha Zrugn
Decrypted text for key=24 is: Pgyjc Btwip
Decrypted text for key=25 is: Qhzkd Cuxjq
Decrypted text for key=26 is: Qhzkd Cuxjq
```

Questions:

1. Describe the characteristics of a secure cryptographic algorithm?
2. What are some common tools used for brute force attacks, and how do they function?
3. How can frequency distribution analysis be used to identify patterns in ciphertext?
4. What are the key characteristics of a brute force attack, and how does it differ from a dictionary attack?
5. How can statistical analysis be used to detect anomalies in network traffic that may indicate a brute force attack?

1. **Characteristics of a secure cryptographic algorithm**:

- **Confidentiality**: Ensures that information is only readable by authorized parties.
- **Integrity**: Makes sure the data is not altered during transmission.
- **Authentication**: Verifies the identity of the sender and receiver.
- **Non-repudiation**: Prevents someone from denying the authenticity of their actions.
- **Resistance to attacks**: Hard to break or reverse without the key, even using advanced computing power.

2. **Common tools for brute force attacks**:

- **John the Ripper**: A tool that tries many combinations of passwords to find the correct one.
- **Hashcat**: A tool for cracking password hashes by trying all possible combinations.
- **Aircrack-ng**: Used for cracking WEP and WPA-PSK passwords by trying various combinations. These tools work by systematically testing every possible password until the correct one is found.

3. **Frequency distribution analysis in ciphertext**:

- It looks at how often certain letters or patterns appear in the ciphertext.
- In languages like English, some letters (like 'e' and 't') appear more often. By matching these common letters with those in the ciphertext, you can guess the key or plaintext.

4. **Brute force attack vs. dictionary attack**:

- **Brute force attack**: Tries every possible combination of characters until the correct one is found. It is time-consuming but guarantees success.
- **Dictionary attack**: Uses a list of common passwords or words to guess the correct one. It is faster than brute force but can fail if the password isn't in the list.

5. **Statistical analysis to detect brute force attacks**:

- It looks for unusual patterns, like too many login attempts in a short time or attempts with many different passwords.
- Anomalies, such as a sudden increase in failed login attempts, can indicate a brute force attack is happening.

Conclusion: *Have understood and implemented the cipher of this lab successfully.*