**INDEX**

|  |  |  |
| --- | --- | --- |
| **SL** | **Name of the Experiment** | **Page No.** |
|  | Write a program to implement encryption and decryption using Caesar cipher. |  |
|  | Write a program to implement encryption and decryption using Mono-Alphabetic cipher. |  |
|  | Write a program to implement encryption and decryption using Brute force attack cipher. |  |
|  | Write a program to implement encryption and decryption using Hill cipher. |  |
|  | Write a program to implement encryption using Playfair cipher. |  |
|  | Write a program to implement decryption using Playfair cipher. |  |
|  | Write a program to implement encryption using Poly-Alphabetic cipher. |  |
|  | Write a program to implement decryption using Poly-Alphabetic cipher. |  |
|  | Write a program to implement encryption using Vernam cipher. |  |
|  | Write a program to implement decryption using Vernam cipher. |  |

**Experiment No-01**

**Experiment Name:** Write a program to implement encryption and decryption using Caesar cipher.

**Theory:** The Caesar cipher operates on the principle of substitution. In this method, each letter of the plaintext (the original message) is replaced by a letter that is a fixed number of positions away in the alphabet. The number of positions shifted is called the "key" or "shift value." For example, with a key of 3, the letter 'A' would be replaced by 'D', 'B' would become 'E', and so on. The alphabet is treated cyclically, meaning that after 'Z', it wraps around to 'A'.

**Encryption Process**

The encryption process for the Caesar cipher involves the following steps:

**Choosing the Key**: Select a shift value (key), which determines how far each character will be shifted. This key must be shared securely between the sender and receiver.

**Substitution:** If the character is a letter, shift it forward in the alphabet by the key value. If the shift goes past 'Z' for uppercase letters or 'z' for lowercase letters, it wraps around to the beginning of the alphabet. On-alphabetical characters (such as digits, punctuation, and spaces) are typically left unchanged.

**Generating the Ciphertext:** The transformed characters are combined to produce the ciphertext, which is an unreadable format of the original message without the key.

**Decryption Process**

Decryption reverses the encryption process. The receiver, who knows the key, can shift the characters in the ciphertext backward by the same key value to retrieve the original plaintext. The decryption steps are

**Using the Same Key:** Use the same shift value that was used for encryption.

**Reverse Substitution:** For each character in the ciphertext:

If the character is a letter, shift it backward in the alphabet by the key value. If the shift goes past 'A' or 'a', it wraps around to the end of the alphabet.

Non-alphabetical characters remain unchanged.

**Recovering the Plaintext:** The resulting characters are combined to obtain the original message.

**Encryption Formula**

C=(P+K) mod n here, P and C are numerical representations of the plaintext and ciphertext characters, respectively, and the modulo operation ensures the result wraps around the alphabet.

**Decryption Formula:**

P=(C−K) mod n the modulo operation ensures the decryption correctly wraps around the alphabet.

Example:

Plaintext: HELLO WORLD

Cipher text: KHOOR ZRUOG

Here shifts key is 3

Explanation:

The letter 'H' shifted 3 positions becomes 'K'.

The letter 'E' shifted to 3 positions becomes 'H'.

The letter 'L' shifted 3 positions becomes 'O', and so on.

Spaces remain unchanged.

**Source code:**

def caesar\_encrypt(plaintext, shift):

encrypted\_text = ""

for char in plaintext:

if char.isalpha():

if char.islower():

encrypted\_text += chr((ord(char) + shift - ord('a')) % 26 + ord('a'))

else:

encrypted\_text += chr((ord(char) + shift - ord('A')) % 26 + ord('A'))

else:

encrypted\_text += char

return encrypted\_text

def caesar\_decrypt(ciphertext, shift):

decrypted\_text = ""

for char in ciphertext:

if char.isalpha():

if char.islower():

decrypted\_text += chr((ord(char) - shift - ord('a')) % 26 + ord('a'))

else:

decrypted\_text += chr((ord(char) - shift - ord('A')) % 26 + ord('A'))

else:

decrypted\_text += char

return decrypted\_text

plaintext = "mustofa"

shift = 3

ciphertext = caesar\_encrypt(plaintext, shift)

print(f"Caesar Cipher Encryption: {ciphertext}")

plaintext = caesar\_decrypt(ciphertext, 3)

print(f"Caesar Cipher Decryption: {plaintext}")

**Output:**

Enter plaintext: mustofa

Caesar Cipher Encryption: pxvwrid

Caesar Cipher Decryption: mustofa

**Experiment No-02**

**Experiment Name:** Write a program to implement encryption and decryption using Mono-Alphabetic cipher.

**Theory:** The Caesar cipher operates on the principle of substitution. In this method, each letter of the plaintext (the original message) is replaced by a letter that is a fixed number of positions away in the alphabet. The number of positions shifted is called the "key" or "shift value." For example, with a key of 3, the letter 'A' would be replaced by 'D', 'B' would become 'E', and so on. The alphabet is treated cyclically, meaning that after 'Z', it wraps around to 'A'.

**Encryption Process**

The encryption process for the Caesar cipher involves the following steps:

**Choosing the Key**: Select a shift value (key), which determines how far each character will be shifted. This key must be shared securely between the sender and receiver.

**Substitution:** If the character is a letter, shift it forward in the alphabet by the key value. If the shift goes past 'Z' for uppercase letters or 'z' for lowercase letters, it wraps around to the beginning of the alphabet. On-alphabetical characters (such as digits, punctuation, and spaces) are typically left unchanged.

**Generating the Ciphertext:** The transformed characters are combined to produce the ciphertext, which is an unreadable format of the original message without the key.

**Decryption Process**

Decryption reverses the encryption process. The receiver, who knows the key, can shift the characters in the ciphertext backward by the same key value to retrieve the original plaintext. The decryption steps are

**Using the Same Key:** Use the same shift value that was used for encryption.

**Reverse Substitution:** For each character in the ciphertext:

If the character is a letter, shift it backward in the alphabet by the key value. If the shift goes past 'A' or 'a', it wraps around to the end of the alphabet.

Non-alphabetical characters remain unchanged.

**Recovering the Plaintext:** The resulting characters are combined to obtain the original message.

**Encryption Formula**

C=(P+K) mod n here, P and C are numerical representations of the plaintext and ciphertext characters, respectively, and the modulo operation ensures the result wraps around the alphabet.

**Decryption Formula:**

P=(C−K) mod n the modulo operation ensures the decryption correctly wraps around the alphabet.

Example:

Plaintext: HELLO WORLD

Cipher text: KHOOR ZRUOG

Here shifts key is 3

Explanation:

The letter 'H' shifted 3 positions becomes 'K'.

The letter 'E' shifted to 3 positions becomes 'H'.

The letter 'L' shifted 3 positions becomes 'O', and so on.

Spaces remain unchanged.

**Source code:**

def caesar\_encrypt(plaintext, shift):

encrypted\_text = ""

for char in plaintext:

if char.isalpha():

if char.islower():

encrypted\_text += chr((ord(char) + shift - ord('a')) % 26 + ord('a'))

else:

encrypted\_text += chr((ord(char) + shift - ord('A')) % 26 + ord('A'))

else:

encrypted\_text += char

return encrypted\_text

def caesar\_decrypt(ciphertext, shift):

decrypted\_text = ""

for char in ciphertext:

if char.isalpha():

if char.islower():

decrypted\_text += chr((ord(char) - shift - ord('a')) % 26 + ord('a'))

else:

decrypted\_text += chr((ord(char) - shift - ord('A')) % 26 + ord('A'))

else:

decrypted\_text += char

return decrypted\_text

plaintext = "mustofa"

shift = 3

ciphertext = caesar\_encrypt(plaintext, shift)

print(f"Caesar Cipher Encryption: {ciphertext}")

plaintext = caesar\_decrypt(ciphertext, 3)

print(f"Caesar Cipher Decryption: {plaintext}")

**Output:**

Enter plaintext: mustofa

Caesar Cipher Encryption: pxvwrid

Caesar Cipher Decryption: mustofa