Academic Year: 2024-25	**Programme: BTECH-Cyber (CSE)**
Year: 2nd	**Semester: IV**
Student Name: Mishitha Verma	**Batch : K073**
Roll No: K073	**Date of experiment: 20/1/25**
Faculty: Rejo Mathew	**Signature with Date:**

Experiment 3: Affine Cipher

Aim: To study and implement Affine Cipher.

Learning Outcomes:

After completion of this experiment, student should be able to

1. Understand steps of Affine Cipher.
2. Implement Affine Cipher.
3. Understand variations of Affine Cipher and its effectiveness.

Theory:

The Affine cipher is a type of monoalphabetic substitution cipher, wherein each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter. The formula used means that each letter encrypts to one other letter, and back again, meaning the cipher is essentially a standard substitution cipher with a rule governing which letter goes to which. The whole process relies on working modulo m (the length of the alphabet used). In the affine cipher, the letters of an alphabet of size m are first mapped to the integers in the range 0 … m-1.

The 'key' for the Affine cipher consists of 2 numbers, we'll call them a and b. The following discussion assumes the use of a 26-character alphabet (m = 26). a should be chosen to be relatively prime to m (i.e. a should have no factors in common with m).

It uses modular arithmetic to transform the integer that each plaintext letter corresponds to into another integer that correspond to a ciphertext letter. The encryption function for a single letter is

Encryption:

$$E(x)= (ax + b) \bmod m$$

modulus m: size of the alphabet

a and b: key of the cipher.

a must be chosen such that a and m are coprime.

Decryption:

In deciphering the ciphertext, we must perform the opposite (or inverse) functions on the ciphertext to retrieve the plaintext. Once again, the first step is to convert each of the ciphertext letters into their integer values. The decryption function is

$$D(x)= a^{-1}(x-b) \bmod m$$

a^-1: modular multiplicative inverse of a modulo m. i.e., it satisfies the equation 1 = a a^-1 mod m.

To find a multiplicative inverse

We need to find a number x such that:

If we find the number x such that the equation is true, then x is the inverse of a, and we call it a^-1. The easiest way to solve this equation is to search each of the numbers 1 to 25, and see which one satisfies the equation.

```
[g,x,d] = gcd(a,m); % we can ignore g and d, we don't need them

x = mod(x,m);
```

If you now multiply x and a and reduce the result (mod 26), you will get the answer 1. Remember, this is just the definition of an inverse

i.e. if a*x = 1 (mod 26), then x is an inverse of a (and a is an inverse of x)

Encryption: For Key values a=17 and b=20

Original Text	T	W	E	N	T	Y		F	I	F	T	E	E	N
x	19	22	4	13	19	24		5	8	5	19	4	4	13
ax +b %26	5	4	10	7	5	12		1	0	1	5	10	10	7
Encrypted Text	F	E	K	H	F	M		B	A	B	F	K	K	H

Decryption: $a^{-1} = 23$

Encrypted Text	F	E	K	H	F	M		B	A	B	F	K	K	H
Encrypted Value	5	4	10	7	5	12		1	0	1	5	10	10	7
23*(x-b) %26	19	22	4	13	19	24		5	8	5	19	4	4	13
Original Text	T	W	E	N	T	Y		F	I	F	T	E	E	N

Steps to follow: Code has to be with comments

Encryption:

Code:

```
alphabets=("abcdefghijklmnopqrstuvwxyz")
p=input("Enter the Plain Text: ")
a=int(input("Enter a:"))
b=int(input("Enter b:"))



is_prime= True
for i in range(2, a):
    if a % i == 0:
        is_prime = False
        break


if not is_prime:
    print(f"Error: {a} is not a prime number, Please enter a prime number")
else:
    print(f"{a} is a prime number")

c=""

for char in p:
    if char.isalpha():
        charlower=char.lower()

        x = alphabets.index(charlower)

        encryptionval = (a * x + b) % 26

        encryption= alphabets[encryptionval]

        c = c + encryption

print("Encrypted text:", c)
```

Output:

```
> python -u "c:\Users\verma\OneDrive\Documents\clg stu
semester 4\itc\exp3\encryption.py"
Enter the Plain Text: MisHithA
Enter a:23
Enter b:18
23 is a prime number
Encrypted text: iuqxunxs
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp3>
Chat  Add Logs  CyberCoder  Improve Code  Share Code Link  Ln 31, Col 1  Spaces: 4  UTF-8  CRLF  {} Python  3.12.3  Go Live
```

```
> python -u "c:\Users\verma\OneDrive\Documents\clg stuff\
semester 4\itc\exp3\encryption.py"
Enter the Plain Text: MpStMe
Enter a:19
Enter b:20
19 is a prime number
Encrypted text: otyros
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp3>
```

```
That  Add Logs  CyberCoder  Improve Code  Share Code Link  Ln 39, Col 1  Spaces: 4  UTF-8  CRLF  {} Python  3.12.3  Go Live  AI Coc
```

```
> python -u "c:\Users\verma\OneDrive\Documents\clg stuff\
semester 4\itc\exp3\encryption.py"
Enter the Plain Text: kzeroseventythree
Enter a:19
Enter b:23
19 is a prime number
Encrypted text: fevidbvgvkuluaivv
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp3>
```

```
That  Add Logs  CyberCoder  Improve Code  Share Code Link  Ln 39, Col 1  Spaces: 4  UTF-8  CRLF  {} Python  3.12.3  Go Live  AI Coc
```

Decryption:

```
alphabet="abcdefghijklmnopqrstuvwxyz"
c= input("Enter Cipher Text: ")
a= int(input("Enter a: "))
b= int(input("Enter b: "))

for a  inv in range(2,25):
    if (a * a_inv) % 26 == 1:
        ainverse=a  inv


print(f"modular inverse of {a} is {ainverse}")

p = ""

for char in c:
    if char.isalpha():
        charlower=char.lower()

        y= alphabet.index(charlower)

        decryptedval = (ainverse*(y-b) % 26)

        decrypted = alphabet[decryptedval]

        p += decrypted

print("Decrypted Text: ",p)
```

Output:

```
> python -u "c:\Users\verma\OneDrive\Documents\clg stuff\
semester 4\itc\exp3\decryption.py"
Enter Cipher Text: iuqxunxs
Enter a: 23
Enter b: 18
modular inverse of 23 is 17
Decrypted Text:  mishitha
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp3>
```

Chat Add Logs CyberCoder Improve Code Share Code Link Ln 31, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.12.3 Go Live AI Code C

```
> python -u "c:\Users\verma\OneDrive\Documents\clg stuff\
semester 4\itc\exp3\decryption.py"
Enter Cipher Text: otyros
Enter a: 19
Enter b: 20
modular inverse of 19 is 11
Decrypted Text:  mpstme
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp3>
```

Chat Add Logs CyberCoder Improve Code Share Code Link Ln 12, Col 7 Spaces: 4 UTF-8 CRLF {} Python 3.12.3 Go Live AI Code

```
> python -u "c:\Users\verma\OneDrive\Documents\clg stuff\
semester 4\itc\exp3\decryption.py"
Enter Cipher Text: fevidbvgvkuluaivv
Enter a: 19
Enter b: 23
modular inverse of 19 is 11
Decrypted Text:  kzeroseventythree
PS C:\Users\verma\OneDrive\Documents\clg stuff\semester 4\itc\exp3>
```

Chat Add Logs CyberCoder Improve Code Share Code Link Ln 12, Col 7 Spaces: 4 UTF-8 CRLF {} Python 3.12.3 Go Live AI

Questions:

1. If the plaintext "HELLO" encrypts to "AXEEH", determine the keys a and b?

Ans: To determine the keys for the Affine cipher, we need to use the formula for encryption: $E(x)=(a \cdot x+b) \bmod 26$

where:

- x is the numerical equivalent of the plaintext letter (A = 0, B = 1, ..., Z = 25),
- E(x) is the numerical equivalent of the ciphertext letter
- a and b are the keys to be determined.

Given that "HELLO" encrypts to "AXEEH", we convert the letters into their corresponding numbers:

- "HELLO" → H (7), E (4), L (11), L (11), O (14)
- "AXEEH" → A (0), X (23), E (4), E (4), H (7)

We then set up the system of equations based on the Affine cipher formula:

$7a+b \equiv 0 \bmod 26$
$4a+b \equiv 23 \bmod 26$
$11a+b \equiv 4 \bmod 26$
$11a+b \equiv 4 \bmod 26$
$14a+b \equiv 7 \bmod 26$

From the first equation $7a+b \equiv 0 \bmod$, we can express b as:
$b = -7a \bmod 26$

or equivalently:
$b = 19a \bmod 26$

Substitute this expression for b into the second equation $4a+b \equiv 23 \bmod 26$:

$4a+19a \equiv 23 \bmod 26$

Simplifying:
$23a \equiv 23 \bmod 26$

So, a=1.

Substitute a=1 into the equation for b:

Final keys:
a=1
b=19

2. What are the common types of attacks on Affine Cipher?

Ans:
Brute Force Attack:
This means trying every possible combination of the keys aaa and bbb to decrypt the message. Since aaa must be a number that works well with 26 (coprime), and bbb can be any number from 0 to 25, there are 312 combinations to try.

Frequency Analysis:
In long messages, certain letters appear more often than others (like 'E' in English). Attackers can look at how often each letter appears in the ciphertext and guess which letters they might be, helping them figure out the encryption keys.

Known-Plaintext Attack:
If an attacker knows part of the original message (plaintext) and its encrypted version (ciphertext), they can work backward using the encryption formula to find out the keys aaa and bbb.

Chosen-Plaintext Attack:
In this attack, the attacker gets to pick some words or sentences (plaintext) and see how they are encrypted (ciphertext). By doing this, they can find the keys more easily.

3. What is a Three Pass Protocol? How does it help in Affine Cipher?

Ans: The Three Pass Protocol is a way for two people to send a secret message without sharing their secret keys. Here's how it works:
First Pass:
The sender locks the message using their own key and sends it to the receiver.
Second Pass:
The receiver locks the already locked message with their own key and sends it back to the sender.
Third Pass:
The sender unlocks it using their key, and then sends it back. The receiver then unlocks it with their key, and now the message is readable.
In the Affine Cipher, using the Three Pass Protocol means both people apply their own encryption. Even if someone sees the message during one of the passes, they can't unlock it because it's protected by both keys. This keeps the message safe.

4. Compare Vigenere Cipher and Affine Cipher w.r.t the following paper and summarize your answer
   R. I. Masya, R. F. Aji and S. Yazid, "Comparison of Vigenere Cipher and Affine Cipher in Three-pass Protocol for Securing Image," *2020 6th International Conference on Science and Technology (ICST)*, Yogyakarta, Indonesia, 2020, pp. 1-5, doi: 10.1109/ICST50505.2020.9732873.

Ans: This research studies how the Vigenère Cipher and Affine Cipher can be used with the Three-pass Protocol to protect image files. Cryptography, which is the method of keeping data safe, was used to apply these old encryption methods. The Three-pass Protocol is special because it lets people send data securely without sharing their encryption keys. It works in three steps, where both the sender and receiver use their own private keys to encrypt and decrypt the data. The Vigenère Cipher is an old method that mixes the original message with a repeated keyword to encrypt it. It adds variety to the encryption but has a weakness because the repeated keyword makes it easier to hack. The Affine Cipher uses math (modular arithmetic) to encrypt messages with linear formulas. It is easy to use but becomes unsafe if someone finds both the original and encrypted messages.
In this research, both ciphers were tested on image files. The tests measured how fast and efficient they were. The Vigenère Cipher was faster, which makes it better for tasks where speed is important. The Affine Cipher was slower but gave stronger security because of its math-based design. The study shows that there is a balance between speed and security when using these ciphers with the Three-pass Protocol for image encryption. Both ciphers work well to protect images, depending on what is needed for the situation.

Conclusion: *[Write your own conclusion regarding the lab performed]*

This program implements the Affine Cipher for encryption and decryption. The encryption uses the formula $E(x)=(a \cdot x+b) \bmod 26$, where a and b are the keys. The value of a must be prime for the encryption to work. Decryption is done by finding the modular inverse of a and applying the

formula $D(y)=a^{-1}\cdot(y-b) \bmod 26$. This method demonstrates the use of number theory in cryptography to securely encrypt and decrypt text.