| | |
|---|---|
| **Academic Year: 2024-25** | **Programme: BTECH-Cyber (CSE)** |
| **Year: 2nd** | **Semester: IV** |
| **Student Name: Arjun Mehta** | **Batch: K1** |
| **Roll No: k036** | **Date of experiment: 16-01-2025** |
| **Faculty: Rejo Matthew** | **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) mod 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

**Code:**

Code cell to mod inverse:

```
def mod_inverse(a,m):
    for x in range(1,m):
        if (a*x)%m==1:
            return x
    return None
```

Code cell for Input:

```
plaintext = input("Enter plaintext: ")
a = int(input("Enter first integer: "))
b = int(input("Enter second integer: "))
```

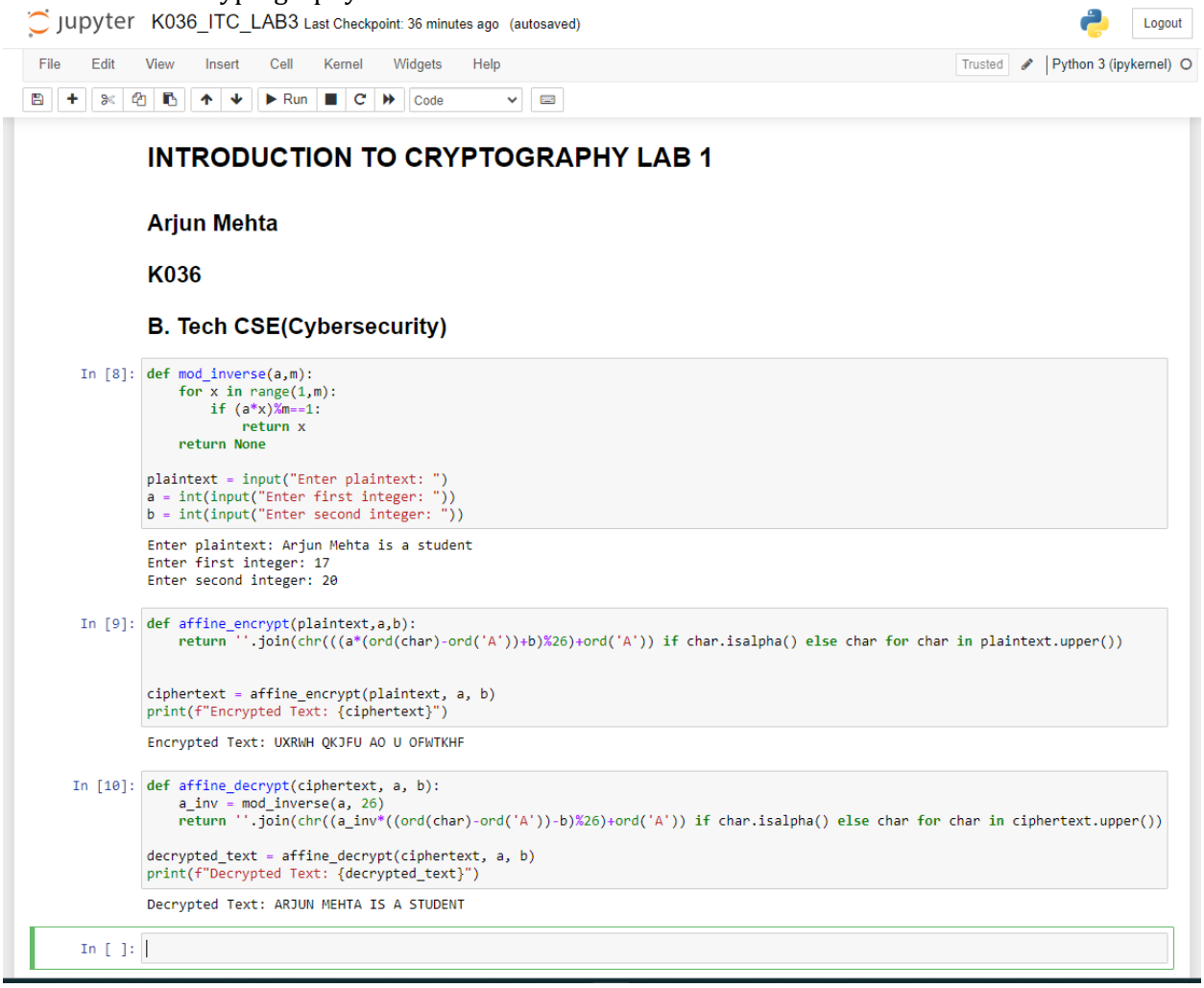
Code cell for encryption:

```
def affine_encrypt(plaintext,a,b):
    return ''.join(chr(((a*(ord(char)-ord('A'))+b)%26)+ord('A')) if char.isalpha() else char for char in plaintext.upper())
ciphertext = affine_encrypt(plaintext, a, b)
print(f"Encrypted Text: {ciphertext}")
```

Code cell for decryption:

```
def affine_decrypt(ciphertext, a, b):
    a_inv = mod_inverse(a, 26)
    return ''.join(chr((a_inv*((ord(char)-ord('A'))-b)%26)+ord('A')) if char.isalpha() else char for char in ciphertext.upper())
decrypted_text = affine_decrypt(ciphertext, a, b)
print(f"Decrypted Text: {decrypted_text}")
```

**OUTPUT:**




**Questions:**

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

a= 1
b= 19

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

Brute Force Attack: Since the key space of the Affine Cipher is relatively small (for a 26-letter alphabet, there are only 312 possible keys), an attacker can try all possible combinations of keys (a and b) to decrypt the ciphertext. This is feasible because the number of possible keys is not large enough to be computationally prohibitive.

Frequency Analysis: Like other substitution ciphers, the Affine Cipher is vulnerable to frequency analysis. In a sufficiently large ciphertext, the frequency of occurrence of each letter may reflect the frequency of the corresponding plaintext letters. By comparing the frequency distribution of the ciphertext with the known frequency distribution of the language (e.g., English), an attacker can deduce the likely values of a and b.

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

Three-Pass Protocol

The Three-Pass Protocol is a method that allows two people to exchange messages securely without sharing a common key.

Here's how it works in simple steps:

First Pass: The sender encrypts the message with their own key and sends it to the receiver.
Second Pass: The receiver encrypts the already encrypted message with their own key and sends it back to the sender.
Third Pass: The sender decrypts the double-encrypted message with their own key and sends it back to the receiver.
Final Decryption: The receiver decrypts the message with their own key to get the original message.
How It Helps in Affine Cipher
Using the Three-Pass Protocol with the Affine Cipher adds an extra layer of security. Here's why:

Both parties use their own keys to encrypt and decrypt the message.
An eavesdropper can't decrypt the message without knowing both keys.
Example in Affine Cipher
Sender's Encryption: Encrypt the message "HELLO" with sender's keys (a_1, b_1).
Receiver's Encryption: Encrypt the sender's encrypted message with receiver's keys (a_2, b_2).
Sender's Decryption: Decrypt the double-encrypted message with sender's keys (a_1^{-1}, -b_1).
Receiver's Decryption: Decrypt the message with receiver's keys (a_2^{-1}, -b_2) to get the original message "HELLO".
This method ensures that neither party can read the message without both encryption keys, making it more secure.

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.

Vigenere Cipher
Type: Uses multiple alphabets (polyalphabetic).
How It Works: Encrypts text using a keyword, shifting letters based on that keyword.
Strengths: More secure against attacks because it changes shifts throughout the text.
Weaknesses: Can still be broken if the keyword is short or if patterns are found.

Affine Cipher
Type: Uses one alphabet (monoalphabetic).
How It Works: Each letter is transformed using a simple math formula.
Strengths: Easy to understand and fast to encrypt/decrypt.
Weaknesses: Less secure because it uses a fixed pattern for each letter.

Comparison in Image Security
Both ciphers can be used together in a three-pass protocol to secure images during transmission.
The Vigenere Cipher is more secure but slower, while the Affine Cipher is faster but less secure.
Using both can improve overall security while still allowing for reasonable speed.

Conclusion
In short, the Vigenere Cipher offers better security, while the Affine Cipher is quicker.
Combining them can help protect images effectively during transmission.

**Conclusion:** *Have successfully applied, understood and implanted affine cipher in code and theory and answered the questions to my capacity.*