|  |  |
| --- | --- |
| **Academic Year: 2024-25** | **Programme: BTECH-Cyber (CSE)** |
| **Year: 2nd** | **Semester: IV** |
| **Student Name: Jal Bafana** | **Batch: K1** |
| **Roll No: K005** | **Date of experiment: 16.01.2025** |
| **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) mod 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

#include <iostream>

using namespace std;

int main() {

char text[100], encryptedText[100];

int textLen = 0,a,b;

cout << "Enter the text: ";

cin.getline(text, 100);

cout << "Enter first key: ";

cin >> a;

cout << "Enter second key: ";

cin >> b;

// Calculate the length of the text and key

while (text[textLen] != '\0') textLen++;

// Encrypt the text

for (int i = 0; i < textLen; i++)

{

// Encrypt alphabetic characters

if ((text[i] >= 'A' && text[i] <= 'Z') || (text[i] >= 'a' && text[i] <= 'z'))

{

// Encrypt uppercase letters

if (text[i] >= 'A' && text[i] <= 'Z')

{

encryptedText[i] = char(((a \* (text[i] - 'A') + b) % 26) + 'A');

}

// Encrypt lowercase letters

else

{

encryptedText[i] = char(((a \* (text[i] - 'a') + b) % 26) + 'a');

}

}

else

{

encryptedText[i] = text[i]; // Non-alphabetic characters remain unchanged

}

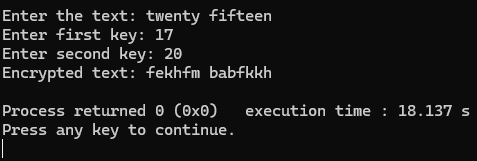
}

encryptedText[textLen] = '\0'; // Null-terminate the encrypted text

cout << "Encrypted text: " << encryptedText << endl;

return 0;

}

****

#include <iostream>

using namespace std;

int main() {

char text[100], decryptedText[100];

int textLen = 0,a,b,i,ainv;

cout << "Enter the text: ";

cin.getline(text, 100);

cout << "Enter first key: ";

cin >> a;

cout << "Enter second key: ";

cin >> b;

// Calculate the length of the text and key

while (text[textLen] != '\0') textLen++;

//calculating A inverse

for(ainv=2;ainv<27; ainv++)

{

if(((a\*ainv)%26) == 1)

{

break;

}

}

// Decrypt the text

for (i = 0; i < textLen; i++)

{

// Decrypt alphabetic characters

if ((text[i] >= 'A' && text[i] <= 'Z') || (text[i] >= 'a' && text[i] <= 'z'))

{

// Decrypt uppercase letters

if (text[i] >= 'A' && text[i] <= 'Z')

{

decryptedText[i] = char(((ainv \* ((text[i] - 'A') - b + 26)) % 26) + 'A');

}

// Decrypt lowercase letters

else

{

decryptedText[i] = char(((ainv \* ((text[i] - 'a') - b + 26)) % 26) + 'a');

}

}

else

{

decryptedText[i] = text[i]; // Non-alphabetic characters remain unchanged

}

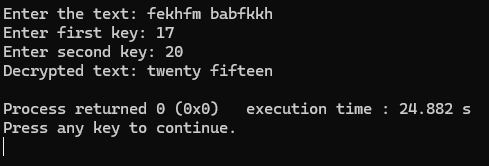
}

decryptedText[textLen] = '\0'; // Null-terminate the decrypted text

cout << "Decrypted text: " << decryptedText << endl;

return 0;

}

****

**Questions:**

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

a=1

b=19

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

* **Brute Force Attack**: This involves trying all possible values for the key aaa and bbb. Since aaa has 12 possible values (it must be coprime with 26), and bbb has 26 possible values, there are only 312 possible combinations to test.
* **Known Plaintext Attack**: If the attacker knows some part of the plaintext and the corresponding ciphertext, they can use this information to derive the keys aaa and bbb.
* **Frequency Analysis**: Since the Affine cipher is a monoalphabetic substitution cipher, it is vulnerable to frequency analysis, where common letters in the plaintext (like 'E' or 'T') might be matched with common ciphertext letters.
* **Chosen Plaintext Attack**: The attacker can choose a plaintext and get the corresponding ciphertext. This can help derive the keys using the Affine cipher equations.

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

A Three Pass Protocol is a cryptographic protocol used to enhance the security of communication by using encryption in multiple phases (three in total). This protocol provides the following steps:

1. **First Pass**: The sender encrypts the message using their public key and sends it to the receiver.
2. **Second Pass**: The receiver decrypts the message with their private key, and then re-encrypts it with the sender's public key.
3. **Third Pass**: The sender decrypts the message with their private key.

In the context of the Affine cipher, a Three Pass Protocol can help by preventing an attacker from directly obtaining the message or the encryption keys in a single pass. Each step adds an extra layer of security, making it more difficult to break the cipher even if the attacker has access to the ciphertext.

1. 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.

In the paper "Comparison of Vigenere Cipher and Affine Cipher in Three-pass Protocol for Securing Image," the authors analyze the performance of both ciphers when applied to image encryption. Here is a summary based on the comparison:

* **Security**: The Vigenère cipher is generally more secure than the Affine cipher because it uses a polyalphabetic substitution method, which makes it resistant to frequency analysis. The Affine cipher, being a monoalphabetic cipher, is more vulnerable to such attacks.
* **Efficiency**: The Affine cipher is easier to implement and faster than the Vigenère cipher, as it involves simple arithmetic operations. On the other hand, Vigenère requires more computation due to its complex key schedule.
* **Performance in Three-pass Protocol**: Both ciphers benefit from the Three Pass Protocol, but the Vigenère cipher, due to its more complex nature, tends to offer better security, albeit at the cost of efficiency.
* **Application to Image Encryption**: The paper discusses the use of both ciphers for encrypting images, where Vigenère provides better protection against statistical attacks, while Affine offers faster encryption but weaker security.

**Conclusion:** *In this lab, we learned to implement the Affine cipher for encryption and decryption, gaining hands-on experience with cryptographic algorithms and their vulnerabilities.*