**Experiment No. 1**

**Title: Substitution Cipher**

**Batch: B2 Roll No.: 16010420117 Experiment No.: 1**

**Aim:** To implement substitution ciphers – Affine and Vigenere cipher.

---

**Resources needed:** Windows/Linux.

---

**Theory**

**Pre Lab/ Prior Concepts:**

**Symmetric-key algorithms** are a class of algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of cipher text. The keys may be identical or there may be a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in comparison to public-key encryption. Symmetric-key encryption can use either stream ciphers or block ciphers. Transposition Cipher is block cipher. Ancient cryptographic systems are classified as: Substitution and Permutation Ciphers.

**Simple Substitution Cipher**

A substitution cipher replaces one symbol with another. Letters of plaintext are replaced by other letters or by numbers or symbols. In a particularly simple implementation of a simple substitution cipher, the message is encrypted by substituting the letter of the alphabet n places ahead of the current letter. For example, with n = 3, the substitution which acts as the key

plaintext: a b c d e f g h i j k l m n o p q r s t u v w x y z
ciphertext: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

The convention is plaintext will be in lowercase and the cipher text will be in uppercase. In this example, the key could be stated more succinctly as “3” since the amount of the shift is the key. Using the key of 3, we can encrypt the plaintext message: “fourscoreandsevenyearsago” by looking up each letter in the plaintext row and substituting the corresponding letter in the ciphertext row or by simply replacing each letter by the letter that is three positions ahead of it in the alphabet. In this particular example, the resulting cipher text is IRXUVFRUHDAGVHYHABHDUVDIR

To decrypt, we simply look up the ciphertext letter in the ciphertext row and replace it with the corresponding letter in the plaintext row, or simply shift each ciphertext letter backward by three. The simple substitution with a shift of three is known as the Caesar’s cipher because it was reputedly used with success by Julius Caesar.

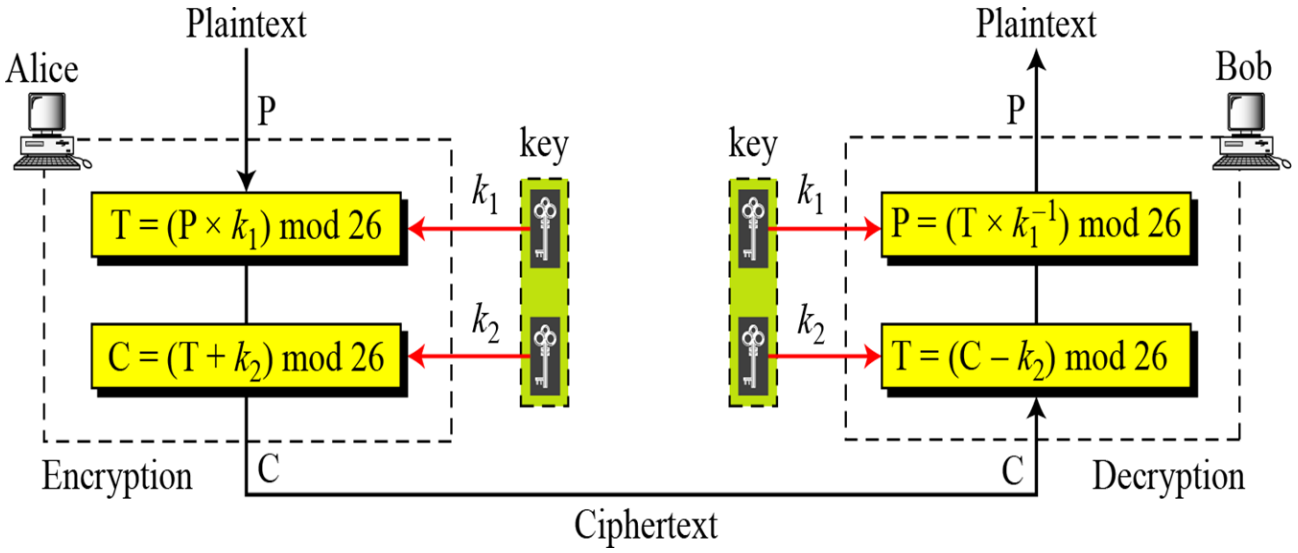
Substitution ciphers are classified as monoalphabetic and polyalphabetic substitution cipher. In monoalphabetic substitution cipher each occurrence of character is encrypted by same substitute character. In Polyalphabetic substitution cipher each occurrence of a character may have a different substitute due to variable Key.

**AFFINE CIPHER**

The Affine cipher is a type of monoalphabetic substitution cipher which uses a combination of Additive and Multiplicative Ciphers. Each letter is enciphered with the function (ax + b) mod 26, where b is the magnitude of the shift. The encryption function for a single letter is

C=(ax + b) mod m where $1 \le a \le m$, $1 \le b \le m$

where modulus m is the size of the alphabet and a and b are the keys of the cipher. The value a must be chosen such that a and m are coprime. The decryption function is $P = a^{-1}(c-b) \bmod m$, where $a^{-1}$ is the modular multiplicative inverse of a i.e., it satisfies the equation $a\,a^{-1} = 1 \bmod m$.




**Encryption: Key Values a=17, 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) mod 26 | 19 | 22 | 4 | 13 | 19 | 24 | | 5 | 8 | 5 | 19 | 4 | 4 | 13 |
| Decrypted Text | T | W | E | N | T | Y | | F | I | F | T | E | E | N |

**Vigenere Cipher**

Vigenere cipher is a polyalphabetic substitution cipher where each occurrence of a character may have a different substitute due to variable. A set of related monoalphabetic substitution rules are used. A key determines which rule to be used. The relationship between a character in the plaintext to a character in the cipher text is one-to-many.

$P = P_1P_2P_3 \ldots$ $C = C_1C_2C_3 \ldots$ $K = [(k_1, k_2, \ldots, k_m), (k_1, k_2, \ldots, k_m), \ldots]$

Encryption: $C_i = P_i + k_i$ Decryption: $P_i = C_i - k_i$

We can encrypt the message "She is listening" using the 6-character keyword "PASCAL".

| **Plaintext:** | s | h | e | i | s | l | i | s | t | e | n | i | n | g |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **P's values:** | 18 | 07 | 04 | 08 | 18 | 11 | 08 | 18 | 19 | 04 | 13 | 08 | 13 | 06 |
| **Key stream:** | ***15*** | ***00*** | ***18*** | ***02*** | ***00*** | ***11*** | ***15*** | ***00*** | ***18*** | ***02*** | ***00*** | ***11*** | ***15*** | ***00*** |
| **C's values:** | **07** | **07** | **22** | **10** | **18** | **22** | **23** | **18** | **11** | **6** | **13** | **19** | **02** | **06** |
| **Ciphertext:** | **H** | **H** | **W** | **K** | **S** | **W** | **X** | **S** | **L** | **G** | **N** | **T** | **C** | **G** |

**Activity:**
Implement the following substitution ciphers:

1. Affine Cipher
2. Vigenere Cipher

**Implementation:**
The program should have encryption function and decryption function for each cipher. Function should take message and a key as input from the user and display the expected output.

---

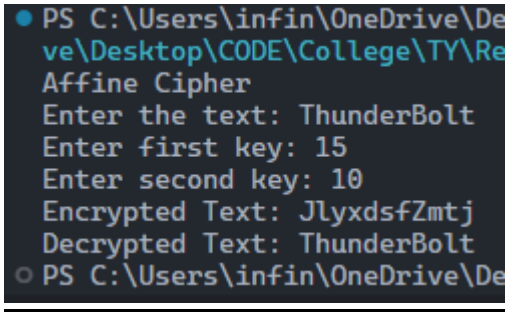
**Results:** (Program with output as per the format)

## Affine Cipher

```
def affine_encrypt(text, key):
    # C = (a*P + b) % 26

    encrypted_text = ''

    for t in text:
        if t!= " ":
            if t.islower():
                char = ((key[0]*(ord(t) - ord('a')) + key[1]) % 26) + ord('a')
                encrypted_text+= chr(char)
            elif t.isupper():
                char = ((key[0]*(ord(t) - ord('A')) + key[1]) % 26) + ord('A')
                encrypted_text+= chr(char)
        elif t==" ":
            encrypted_text+=" "

    return encrypted_text


# affine cipher decryption function
def affine_decrypt(cipher, key):
    # P = (a^-1 * (C - b)) % 26


    decrypted_text = ''

    key_inv = 0

    for i in range(26):
        flag = (key[0]* i) % 26

        if (flag == 1):
            key_inv = i


    for t in cipher:
        if t!= " ":
            if t.islower():
                char = (key_inv*(ord(t) - ord('a') - key[1]) % 26) + ord('a')
                decrypted_text+= chr(char)
            elif t.isupper():
                char = (key_inv*(ord(t) - ord('A') - key[1]) % 26) + ord('A')
                decrypted_text+= chr(char)
        elif t==" ":
            decrypted_text+=" "

    return decrypted_text
```

```
def main():

    print("Affine Cipher")
    text = input("Enter the text: ")
    k1 = int(input("Enter first key: "))
    k2 = int(input("Enter second key: "))
    key = [k1, k2]

    affine_encrypted_text = affine_encrypt(text, key)

    print('Encrypted Text: '+ affine_encrypted_text)
    print('Decrypted Text: '+ affine_decrypt(affine_encrypted_text, key))

main()
```

**Output:**



## Vignere Cipher

```
# Vigenere Cipher

def keyGenerationFunc(text, key):
    key = list(key)
    if len(text) == len(key):
        return key
    else:
        for i in range(len(text) - len(key)):
            key.append(key[i % len(key)])
    return "".join(key)

def vignere_encrypt(text, key):
    cipherText = []
    for i in range(len(text)):
        temp = ((ord(text[i]) + ord(key[i])) % 26) + 65
        cipherText.append(chr(temp))

    return "".join(cipherText)

def vignere_decrypt(cipherText, key):
    originalText = []
    for i in range(len(cipherText)):
        x = ((ord(cipherText[i]) - ord(key[i]) + 26) % 26) + 65
        originalText.append(chr(x))
    return("" . join(originalText))

print("Vignere Cipher")
text = input("Enter text: ").upper()
keyword = input("Enter Key: ")
key = keyGenerationFunc(text, keyword)
cipherText = vignere_encrypt(text,key)
print("Cipher text :", cipherText)
print("Decrypted Text :",vignere_decrypt(cipherText, key))
```

## Output:

```
PS C:\Users\infin\OneDrive\Des
ve\Desktop\CODE\College\TY\Ref
Vignere Cipher
Enter text: ThunderBolt
Enter Key: Soham
Cipher text : LBHTVWLOUDL
Decrypted Text : THUNDERBOLT
```

**Questions:**

1) Write down the flaws of Affine cipher and Vigenere Cipher:

**Ans:** The flaws of Affine Cipher is that the frequency distributions of symbols in the plaintext and in the ciphertext are identical, only the symbols having been relabeled. In fact, any structure or pattern in the plaintext is preserved intact in the ciphertext makes it easier to crack the cipher.

The drawback of Algorithm Vigenere Cipher is if the key length is smaller than the plaintext length, then the key will be repeated, because it most likely will produce the same ciphertext as long as the same plaintext.

**Outcomes: CO1:** Describe the basics of Information Security

**Conclusion:** We successfully learned and implemented Affine Cipher and Vignere Cipher

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**

**References: Books/ Journals/ Websites:**

1. Behrouz A. Forouzan, “Cryptography and Network Security”, Tata McGraw Hill
2. William Stalling, “Cryptography and Network Security”, Prentice Hall