

CNS PRACTICAL NO.3

Aim: Write a program in Python to perform Cryptanalysis or decoding of Vignere Cipher

Theory:

Key Concepts:

- Polyalphabetic Substitution Cipher: Unlike monoalphabetic substitution ciphers (like Caesar cipher) that use a fixed substitution pattern for each letter, Vigenère cipher uses multiple substitution patterns based on a keyword. This makes it more secure compared to simple substitution ciphers.
- Keyword: The keyword is a secret sequence of characters (usually letters) that determines the shift value for each letter in the plaintext. It's the core of the Vigenère cipher and dictates the encryption and decryption process.
- Encryption: To encrypt a message using the Vigenère cipher, each letter of the plaintext is shifted based on the corresponding letter in the keyword. The shift value for a letter is determined by its position in the alphabet (A=0, B=1, ..., Z=25).
- Decryption: To decrypt an encrypted message, the reverse process is applied. Each letter of the ciphertext is shifted back based on the corresponding letter in the keyword.

Advantages: Vigenère cipher is stronger than simple Caesar cipher due to its use of multiple substitution patterns. It's relatively easy to understand and implement.

Disadvantages: The security of Vigenère cipher depends on the length and secrecy of the keyword. If the keyword is short or not truly random, it's susceptible to attacks such as frequency analysis. It's also vulnerable to Kasiski examination and Friedman test if the keyword length is too short.

CODE

```
def generate_key(msg, key):  
    key = list(key)  
    if len(msg) == len(key):  
        return "".join(key)  
    else:  
        for i in range(len(msg) - len(key)):  
            key.append(key[i % len(key)])  
    return "".join(key)  
  
def encrypt_vigenere(msg, key):  
    encrypted_text = []  
    key = generate_key(msg, key)  
    for i in range(len(msg)):  
        char = msg[i]  
        if char.isupper():  
            encrypted_char = chr((ord(char) + ord(key[i].upper()) - 2 * ord('A')) % 26 + ord('A'))  
        else:  
            encrypted_char = chr((ord(char) + ord(key[i])) % 26 + ord('A'))  
        encrypted_text.append(encrypted_char)  
    return ''.join(encrypted_text)
```

```

    elif char.islower():
        encrypted_char = chr((ord(char) + ord(key[i].lower()) - 2 * ord('a')) % 26 + ord('a'))
    else:
        encrypted_char = char # Non-alphabetic characters are kept unchanged
    encrypted_text.append(encrypted_char)
return "".join(encrypted_text)

def decrypt_vigenere(msg, key):
    decrypted_text = []
    key = generate_key(msg, key)
    for i in range(len(msg)):
        char = msg[i]
        if char.isupper():
            decrypted_char = chr((ord(char) - ord(key[i].upper()) + 26) % 26 + ord('A'))
        elif char.islower():
            decrypted_char = chr((ord(char) - ord(key[i].lower()) + 26) % 26 + ord('a'))
        else:
            decrypted_char = char # Non-alphabetic characters are kept unchanged
        decrypted_text.append(decrypted_char)
    return "".join(decrypted_text)

# --- User Input ---
plaintext = input("Enter the plaintext message: ")
key = input("Enter the key: ")

# --- Encryption ---
encrypted = encrypt_vigenere(plaintext, key)
print(f"\nEncrypted Text: {encrypted}")

# --- Decryption ---
decrypted = decrypt_vigenere(encrypted, key)
print(f"Decrypted Text: {decrypted}")

```

OUTPUT

```

Enter the plaintext message: msw from inft
Enter the key: Vesit

Encrypted Text: hwo ymse bijl
Decrypted Text: msw from inft

```

Review Questions.

1. Discuss the Strengths and Weaknesses of the Vigenère Cipher

Strengths:

- **Polyalphabetic nature:** Unlike monoalphabetic ciphers, each letter can be encrypted differently depending on the key, making frequency analysis more difficult.
- **Simple and fast:** Easy to implement and encrypt/decrypt manually or in code.
- **Repeating key hides patterns** better than simple substitution.

Weaknesses:

- **Key repetition:** If the key is shorter than the plaintext, it repeats and introduces patterns that can be exploited (especially in long messages).
 - **Vulnerable to Kasiski and Friedman tests** if the key length is known or guessed.
 - **Not secure for modern usage:** It can be broken easily with modern cryptanalysis tools and techniques.
-

2. How Can the Vigenère Cipher Be Broken or attacked? (One Method)

Method: Kasiski Examination

The most famous and effective attack on the Vigenère cipher before computers.

Step-by-step breakdown:

1. **Look for repeated patterns in the ciphertext:**
 - These likely come from the same plaintext block encrypted with the same key fragment.
 - For example, repeated ciphertext strings like "XYL" may appear multiple times.
2. **Measure distances between these repetitions:**
 - Suppose "XYL" appears at positions 10 and 40 → distance = 30.

3. Find the GCD of several such distances:

- These distances are likely multiples of the key length. GCD of multiple such values gives the probable key length.

4. Break the ciphertext into parts:

- If key length = 3, divide the ciphertext into 3 groups:
 - Group 1: Characters encrypted by key[0]
 - Group 2: Characters encrypted by key[1]
 - Group 3: Characters encrypted by key[2]

5. Frequency analysis on each group:

- Each group is effectively a Caesar cipher.
- Attack each group using frequency analysis and guess the Caesar shift (key letter).

3. What is the significance of the key length in the security of the Vigenère cipher?

Key length plays a critical role in the strength of the cipher:

 Short keys:

- Lead to frequent repetition.
- Make the cipher vulnerable to pattern detection.
- Easy to attack using the Kasiski method.

 Long keys:

- If the key is as long as the message and random → the cipher becomes a one-time pad, which is:
 - Unbreakable (proven mathematically).
 - Requires a truly random key, used only once.

 Trade-off:

- Longer keys = more secure but harder to share/remember.
- Shorter keys = easier but much less secure.

Conclusion:

Longer and non-repeating keys greatly increase the cipher's resistance to attack.

4. Vigenère vs. Other Polyalphabetic Ciphers

Feature	Vigenère Cipher	Autokey Cipher	One-Time Pad
Key type	Repeating fixed key	Key starts with fixed key + message letters	Truly random, as long as message
Key repetition	Yes	Less repetition due to message-based key	No repetition
Security	Moderate (breakable with analysis)	Stronger than Vigenère	Perfect (if truly random and kept secret)
Ease of use	Simple	Slightly more complex	Impractical in large-scale communication

5. Encrypt "explanation" Using Vigenère Cipher and Key "leg"

Step 1: Repeat the key to match the length of the message

Plaintext: e x p l a n a t i o n (11 letters)
 Key: l e g l e g l e

Step 2: Use the formula:

$$\text{Encrypted character} = (\text{Plaintext} + \text{Key}) \bmod 26$$

Let's do this step by step (using 0-based index: A=0, B=1, ..., Z=25):

Plaintext	Key	P (0–25)	K (0–25)	(P + K) mod 26	Cipher
-----------	-----	----------	----------	----------------	--------

e		4	11	15	p
---	--	---	----	----	---

x	e	23	4	1	b
---	---	----	---	---	---

p	g	15	6	21	v
---	---	----	---	----	---

l		11	11	22	w
---	--	----	----	----	---

a	e	0	4	4	e
---	---	---	---	---	---

n	g	13	6	19	t
---	---	----	---	----	---

a		0	11	11	
---	--	---	----	----	--

t	e	19	4	23	x
---	---	----	---	----	---

i	g	8	6	14	o
---	---	---	---	----	---

o		14	11	25	z
---	--	----	----	----	---

n	e	13	4	17	r
---	---	----	---	----	---

Final Ciphertext: pbvwetlxozr