Lab Assignment 1: Implementing Traditional Substitution Ciphers

Objective

The purpose of this lab is to deepen your understanding of traditional substitution ciphers by implementing them programmatically. You will gain hands-on experience with symmetric cryptographic techniques, understand their mechanics, and analyze their strengths and limitations.

Instructions

You are required to implement the following traditional substitution ciphers in your preferred programming language (Java, Python, or C):

1. Caesar Cipher
2. Vigenère Cipher
3. Atbash Cipher
4. Affine Cipher
5. Playfair Cipher
6. Hill Cipher (Matrix-based substitution)

Detailed Descriptions and Mathematical Formulations

1. Caesar Cipher:

- **Description:** The Caesar cipher is a substitution cipher that shifts each letter in the plaintext by a fixed number of positions down the alphabet. The decryption reverses the shift.
- **Mathematical Formulation:**

$$C = (P + k) \bmod 26$$

$$P = (C - k) \bmod 26$$

where P is the plaintext character, C is the ciphertext character, and k is the shift key.

Example Test Case:

- Input: `HELLO`

- Key: `3`
- Expected Output (Ciphertext): `KHOOR`
- Actual Output: `KHOOR`
- Decrypted Plaintext: `HELLO`

2. Vigenère Cipher:

- **Description:** The Vigenère cipher is a polyalphabetic substitution cipher that uses a keyword to shift the alphabet for each character of the plaintext.
- **Mathematical Formulation:**

$$C_i = (P_i + K_i) \bmod 26$$

$$P_i = (C_i - K_i) \bmod 26$$

where P_i is the ith character of the plaintext, C_i is the ith character of the ciphertext, and K_i is the ith character of the repeating keyword.

Example Test Case:

- Input: `ATTACKATDAWN`
- Key: `LEMON`
- Expected Output (Ciphertext): `LXFOPVEFRNHR`
- Actual Output: `LXFOPVEFRNHR`
- Decrypted Plaintext: `ATTACKATDAWN`

3. Atbash Cipher:

- **Description:** The Atbash cipher is a monoalphabetic substitution cipher that maps each letter to its reverse counterpart in the alphabet (e.g., A $\rightarrow$ Z, B $\rightarrow$ Y).
- **Mathematical Formulation:**

$$C = 25 - P$$

where P is the position of the plaintext letter in the alphabet (0-indexed), and C is the position of the ciphertext letter.

Example Test Case:

- Input: `HELLO`
- Expected Output (Ciphertext): `SVOOL`
- Actual Output: `SVOOL`
- Decrypted Plaintext: `HELLO`

4. Affine Cipher:

- **Description:** The Affine cipher uses linear transformation to substitute characters, defined by two keys a and b. It is essential that a and 26 are coprime.
- **Mathematical Formulation:**

$$C = (aP + b) \bmod 26$$

$$P = a^{-1}(C - b) \bmod 26$$

where a^{-1} is the modular multiplicative inverse of a modulo 26.

Example Test Case:

- Input: `HELLO`
- Keys: `a=5, b=8`
- Expected Output (Ciphertext): `RCLLA`
- Actual Output: `RCLLA`
- Decrypted Plaintext: `HELLO`

5. Playfair Cipher:

- **Description:** The Playfair cipher encrypts digraphs (pairs of letters) instead of single characters using a 5x5 matrix constructed from a keyword.
- **Rules:**
  - If both letters are in the same row, replace them with the letters to their immediate right.
  - If both letters are in the same column, replace them with the letters immediately below.
  - If the letters form a rectangle, replace them with the letters on the same row but at the opposite corners.

Example Test Case:

- Input: `HELLO`
- Key: `MONARCHY`
- Expected Output (Ciphertext): `CFSPM`
- Actual Output: `CFSPM`
- Decrypted Plaintext: `HELLO`

6. Hill Cipher:

- **Description:** The Hill cipher uses matrix multiplication to encrypt plaintext characters in blocks.

- **Mathematical Formulation:**

$$C = KP \bmod 26$$

$$P = K^{-1}C \bmod 26$$

where K is the encryption key matrix, P is the plaintext vector, and C is the ciphertext vector.

Example Test Case:

- Input: `HELP`
- Key Matrix:

$$\begin{bmatrix} 1 & 8 \\ 8 & 5 \end{bmatrix}$$

- Expected Output (Ciphertext): `NYBH`
- Actual Output: `NYBH`
- Decrypted Plaintext: `HELP`

Requirements

1. **Implementation:**
   - Each cipher must support both encryption and decryption functionality.
   - Ensure proper input validation for edge cases such as:
     - Non-alphabetic characters.
     - Mixed-case inputs.
     - Empty strings.
2. **Testing:**
   - Test your implementation with at least **three distinct test cases** for each cipher.
   - For each test case, document:
     - Input plaintext.
     - Encryption key or configuration.
     - Expected ciphertext.
     - Actual ciphertext and decrypted plaintext.
3. **Output Snapshots:**
   - Include clear screenshots of the program output for each test case.
   - Snapshots should show both encryption and decryption processes.
4. **Report Section:**
   - Summarize your findings for each cipher in a short paragraph.
   - Discuss any challenges or observations, such as how the cipher handles edge cases.

Submission Guidelines

- Submit a zip file containing:
  - Source code for all ciphers.
  - A report in PDF format with your findings and Output screenshots for all test cases.
- Ensure your code is well-documented and follows best practices.

Grading Criteria

Your assignment will be graded based on the following:

1. **Correctness (40%):** Does the implementation meet the requirements and produce correct results?
2. **Completeness (30%):** Are all ciphers implemented with both encryption and decryption?
3. **Testing and Documentation (20%):** Are test cases, snapshots, and observations comprehensive and well-documented?
4. **Code Quality (10%):** Is the code clean, modular, and well-commented?