VIT®

BHOPAL

Vigenere Cipher Encoder & Decoder

Project Report

Made and submitted by:

Name:Shivansh Tembhare

Reg.no.:-25BCE11233

Computer science & engineering(CORE)

Date:25-11-2025

Project Report: Vigenère Cipher Implementation

1. Introduction:

The Vigenère Cipher is a method of encrypting alphabetic text by using a series of interwoven Caesar ciphers, based on the letters of a keyword. It is a form of polyalphabetic substitution . This project provides a Python script to automate this process, allowing users to encrypt plain text and decrypt cipher text using a command-line interface.

2. Objective:

The primary objective of this script is to:

- Securely transform readable text (plaintext) into unreadable format (ciphertext) using a specific key.
- Reverse the process to retrieve the original message.
- Handle case sensitivity and non-alphabetic characters (numbers, punctuation) gracefully.

3. Mathematical Logic:

The core logic relies on modular arithmetic. Each letter is treated as a number ($A=0, B=1, ..., Z=25$).

- Encryption Formula:

C_i = (P_i + K_i) \mod 26

Where C_i is the encrypted letter, P_i is the plaintext letter, and K_i is the key letter.

- Decryption Formula:

P_i = (C_i - K_i) \mod 26

Note: In Python, the ord() function is used to get the ASCII value, and chr() is used to convert it back to a character.

4. Code Structure & Analysis:

The script is divided into three distinct sections based on the provided file.

A. Function: vigenere_encrypt(plain_text, key)

This function handles the conversion of plaintext to ciphertext.

- **Key Extension:** The key is repeated to match the length of the input string to ensure every character has a corresponding shift value.
- **Case Handling:** The code explicitly checks if plain_text[i].isupper() to ensure uppercase letters remain uppercase and lowercase remain lowercase.
- **Non-Alpha Handling:** Characters that are not letters (e.g., numbers, punctuation) are appended to the string without modification.

B. Function: vigenere_decrypt(cipher_text, key)

This function reverses the encryption process.

- It mirrors the encryption logic but subtracts the key's shift value instead of adding it.
- It ensures the result wraps around the alphabet correctly using the modulo operator % 26.

C. Main Execution Loop

The script includes a while loop that serves as the user interface:

1. **Input:** Prompts the user to choose 'e' (encrypt) or 'd' (decrypt).
2. **Text Splitting:** The script splits the input text into a list of words using x.split(" ").
3. **Iteration:** It iterates through this list, applying the vigenere_encrypt or vigenere_decrypt function to **each word individually**.
4. **Output:** The processed words are printed sequentially.
5. **Continuation:** The user is asked if they wish to continue (y/n).

5. Features & Observations:

Feature	Description
Polyalphabetic	Uses a keyword to shift letters, making it stronger than a standard Caesar cipher.
Case Preservation	"Hello" becomes "Rijvs" (if key is 'K'), preserving the capital 'H'.
Special Character Safety	Symbols like !, ?, or numbers are ignored and preserved in the output.
Word-Based Processing	**Crucial Observation:** The script restarts the key pattern for every new word because of the L=x.split(" ") line. Standard Vigenère usually flows the key continuously across spaces. This is a unique implementation detail of this specific script.

6. Sample Execution Flow:

Scenario: Encryption

User Input: e

Text: ATTACK DAWN

Key: KEY

Internal Logic:

- Word 1: "ATTACK" encrypted with "KEYKEY" = LXREGM
- Word 2: "DAWN" encrypted with "KEYK" = NEAL

Output: LXREGM NEAL

---

7. Conclusion

This Python project successfully implements a working Vigenère cipher. It demonstrates valid use of string manipulation, modular arithmetic, and control flow structures. While the "word-splitting" logic differs slightly from the classical cryptographic definition (where the key flows across spaces), the script functions effectively as a symmetric encryption tool for educational purposes.