|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Information Security (116U01L602)** | **Semester:** | **VI** |
| **Date of Performance:** | **15 / 01 / 2025** | **DIV/ Batch No:** | **A-3** |
| **Student Name:** | **Kashish Mamania** | **Roll No:** | **16010122104** |

|  |
| --- |
| **Title: Encryption-Decryption programs using classical cryptography** |

|  |
| --- |
| **Objectives:** |
| To write a program to convert plain text into cipher text using Caesar cipher and Transposition cipher |

|  |
| --- |
| **Expected Outcome of Experiment:** |
| **CO1** |

|  |
| --- |
| **Books/ Journals/ Websites referred:** |
|  |

|  |
| --- |
| **Pre Lab/ Prior Concepts:** |
|  |

|  |
| --- |
| **New Concepts to be learned:** |
| Caesar Cipher, Matrix Transposition Cipher, Columnar Transposition Cipher, Shift Cipher, Mono-alphabetic Substitution Cipher |

|  |
| --- |
| **Abstract:** |
| This laboratory experiment explores the design and implementation of encryption-decryption programs using **classical cryptography**, focusing on foundational techniques such as substitution and transposition ciphers. The lab emphasizes hands-on programming to demonstrate core principles of symmetric-key cryptography, where a single key is used for both encryption and decryption. |

|  |
| --- |
| **Related Theory:** |
| **1. Caesar Cipher**   * **Definition**: A substitution cipher where each letter in the plaintext is shifted a fixed number of positions down the alphabet. It’s named after Julius Caesar, who used it with a shift of 3. * **Key**: The shift value (e.g., shift of 3). * **Example**:   + Plaintext: HELLO   + Shift: 3   + Ciphertext: KHOOR (H→K, E→H, L→O, etc.). * **Mathematical Formula**:   + Encryption: E(x)=(x+k)mod  26*E*(*x*)=(*x*+*k*)mod26   + Decryption: D(x)=(x−k)mod  26*D*(*x*)=(*x*−*k*)mod26 * **Vulnerability**: Only 25 possible keys, making it vulnerable to brute-force attacks.   **2. Shift Cipher**   * **Definition**: A generalization of the Caesar Cipher where letters are shifted by any integer value (not just 3). The Caesar Cipher is a specific case of a shift cipher. * **Key**: The shift value k*k*. * **Example**:   + Plaintext: ATTACK   + Shift: 7   + Ciphertext: HAAHJR (A→H, T→A, etc.). * **Use Case**: Simple encryption but insecure for modern standards.   **3. Mono-alphabetic Substitution Cipher**   * **Definition**: A substitution cipher where each plaintext letter is mapped to a fixed ciphertext letter using a scrambled alphabet. Unlike the Caesar cipher, the mapping is arbitrary. * **Key**: A permutation of the alphabet (e.g., A→X, B→Z, C→M, etc.). * **Example**:   + Plaintext: SECRET   + Key: XZMBYACDEFGHIJKLNOPQRSTUVW   + Ciphertext: LBMYBO. * **Vulnerability**: Frequency analysis (e.g., E is the most common letter in English).   **4. Matrix Transposition Cipher**   * **Definition**: A transposition cipher where plaintext is written row-wise into a matrix and read column-wise to produce ciphertext. The key is the matrix dimensions or column order. * **Example**:   + Plaintext: ENEMYATTACKSTONIGHT   + Matrix (4x5):   E N E M Y  A T T A C  K S T O N  I G H T \_   * + Ciphertext: EAKI NTST GETO YCHTN (read columns top to bottom). * **Decryption**: Rearrange ciphertext into the matrix and read rows.   **5. Columnar Transposition Cipher**   * **Definition**: A transposition cipher where columns are reordered based on a keyword. The key determines the column permutation. * **Key**: A keyword (e.g., HACK → columns sorted alphabetically as 3,1,2,4). * **Example**:   + Plaintext: LORD RAMA WAS A GOOD KING   + Keyword: HACK (columns ordered as 3,1,2,4)   + Matrix:   L O R D  R A M A  W A S \_  A G O O  D K I N  G \_ \_ \_   * + Ciphertext: R D L O A A R M S \_ W A O O A G I N D K \_ \_ G \_ (read columns 3,1,2,4). * **Vulnerability**: Frequency of letters remains unchanged, but permutation adds complexity.   **Key Differences**   | **Cipher Type** | **Method** | **Key Mechanism** | **Security Level** | | --- | --- | --- | --- | | Caesar Cipher | Substitution | Fixed shift (e.g., 3) | Very low (25 keys) | | Shift Cipher | Substitution | Any integer shift | Low | | Mono-alphabetic Substitution | Substitution | Arbitrary alphabet permutation | Moderate (26! keys) | | Matrix Transposition | Transposition | Matrix dimensions | Moderate (depends on key) | | Columnar Transposition | Transposition | Keyword-based column order | Moderate | |

|  |
| --- |
| **Implementation Details:** |
| Caesar Cipher:  def encrypt\_shift(plaintext, key):      ciphertext = ""      for char in plaintext.lower():          if char.isalpha():              shifted = chr((ord(char) - ord('a') + key) % 26 + ord('a'))              ciphertext += shifted          else:              ciphertext += char      return ciphertext.upper()  def decrypt\_shift(ciphertext, key):      plaintext = ""      for char in ciphertext.lower():          if char.isalpha():              shifted = chr((ord(char) - ord('a') - key) % 26 + ord('a'))              plaintext += shifted          else:              plaintext += char      return plaintext  # Example usage  key = 15  message = "hello"  encrypted = encrypt\_shift(message, key)  decrypted = decrypt\_shift(encrypted, key)  print(f"Original: {message}")  print(f"Encrypted: {encrypted}")  print(f"Decrypted: {decrypted}")  Matrix Transposition Cipher:  import math  def encrypt\_transposition(plaintext, key):      # Remove spaces and convert to uppercase      plaintext = ''.join(plaintext.split()).upper()        # Calculate number of columns and rows      cols = len(key)      rows = math.ceil(len(plaintext) / cols)        # Pad the plaintext if necessary      plaintext += '\_' \* (cols \* rows - len(plaintext))        # Create the matrix      matrix = [[''] \* cols for \_ in range(rows)]      index = 0      for i in range(rows):          for j in range(cols):              matrix[i][j] = plaintext[index]              index += 1        # Print the encryption matrix      print("Encryption Matrix:")      for row in matrix:          print(row)        # Read off the columns according to the key      ciphertext = ''      for col in key:          col\_index = int(col) - 1          ciphertext += ''.join(matrix[row][col\_index] for row in range(rows))        return ciphertext  def decrypt\_transposition(ciphertext, key):      cols = len(key)      rows = math.ceil(len(ciphertext) / cols)        # Create the matrix      matrix = [[''] \* cols for \_ in range(rows)]        # Fill the matrix column by column according to the key      index = 0      for col in key:          col\_index = int(col) - 1          for row in range(rows):              if index < len(ciphertext):                  matrix[row][col\_index] = ciphertext[index]                  index += 1        # Print the decryption matrix      print("Decryption Matrix:")      for row in matrix:          print(row)        # Read off the matrix row by row      plaintext = ''      for row in matrix:          plaintext += ''.join(row)        # Remove padding      plaintext = plaintext.rstrip('\_')        return plaintext  # Get user input  plaintext = input("Enter the message to encrypt: ")  key = input("Enter the key (e.g., 3142 for a 4-column transposition): ")  # Encrypt the message  encrypted = encrypt\_transposition(plaintext, key)  print(f"Encrypted message: {encrypted}")  # Decrypt the message  decrypted = decrypt\_transposition(encrypted, key)  print(f"Decrypted message: {decrypted}") |

|  |
| --- |
| **Results/Output:** |
| Caesar Cipher output:    Matrix Transposition Cipher output: |

|  |
| --- |
| **Conclusion:** |
| This lab demonstrates the practical implementation of classical cryptographic techniques, including substitution (shift/monoalphabetic) and transposition ciphers, while highlighting their vulnerabilities to brute-force and frequency-based attacks. The exercises reinforce the foundational principles of symmetric-key cryptography and underscore the necessity for robust modern encryption methods. |

|  |
| --- |
| **Post-Lab Questions:** |
| 1.1 Points of difference between mono-alphabetic cipher and poly-alphabetic cipher:   | **Mono-alphabetic Cipher** | **Poly-alphabetic Cipher** | | --- | --- | | In monoalphabetic cipher, every symbol that is in plain text gets mapped to fixed symbols that are in cipher texts. | Polyalphabetic cipher is based on substitution with the help of multiple substitution alphabets. | | There is a one-to-one relationship between the characters that exist in plain text and the characters that exist in cipher text. | There is a one-to-many relationship between plain text characters and cipher text characters. | | It is a simple substitution cipher. | It is multiple substitutions cipher. | | Monoalphabetic ciphers are not that strong as compared to polyalphabetic cipher. | Polyalphabetic ciphers are much stronger.[14](https://www.geeksforgeeks.org/difference-between-monoalphabetic-cipher-and-polyalphabetic-cipher/) | | Contains frequency of letters same as the message. | It does not contain the same frequency of letters as in the message. |   1.2 Working of a rail-fence cipher:  In the rail fence cipher, the plaintext is written downwards diagonally on successive "rails" of an imaginary fence, then moving up when the bottom rail is reached, down again when the top rail is reached, and so on until the whole plaintext is written out. The ciphertext is then read off in rows.  Example: Let's encrypt the message "WE ARE DISCOVERED. RUN AT ONCE." using 3 rails.  W . . . E . . . C . . . R . . . U . . . O . . . . E . R . D . S . O . E . E . N . T . N . E . . . A . . . I . . . V . . . D . . . A . . . C .  The ciphertext is then read off row by row: WECRUOERDSOEEntneaivdac  1.3 Three applications of cryptography:   1. Secure Communication and Messaging: Cryptography is used to ensure the confidentiality and integrity of digital communication, including email, instant messaging, and voice calls. End-to-end encryption ensures that only the intended recipients can access and decipher the messages. 2. Online Banking and Financial Transactions: Cryptography secures online banking, payment systems, and financial transactions by encrypting sensitive data such as account numbers, passwords, and transaction details. It helps prevent unauthorized access and fraud. 3. Healthcare and Medical Records: Cryptography safeguards electronic health records and medical data, ensuring patient privacy and compliance with healthcare regulations like the Health Insurance Portability and Accountability Act (HIPAA). |

|  |
| --- |
| **Virtual Lab:** |
| Breaking Shift Cipher: |