

Ministry of Education, Culture and Research of the Republic of Moldova

Technical University of Moldova

Department of Software and Automation Engineering

**REPORT**

Laboratory work No. 3

**Discipline**: Cryptography and Security

Elaborated: FAF-221, Chichioi Iuliana

Checked: asist. univ.,Nirca Dumitru

Chișinău 2024

## Topic: Polyalphabetic ciphers (Playfair cipher) Tasks:

## Task 3.1: Implement the Playfair cipher algorithm in a programming language of your choice for messages in Romanian (using 31 letters). Character values in the text should be limited to 'A' through 'Z' and 'a' through 'z'; no other values are permitted. If the user inputs any other values, they should be prompted to use the correct character range. The key length must be at least 7 characters. The user should be able to select the operation—encryption or decryption—and input the key, message, or ciphertext, and receive the corresponding encrypted or decrypted message. The final phase of adding spaces according to language and message logic should be done manually.

## Objectives:

## Implement a Python-based Playfair cipher to encrypt and decrypt text.

## Demonstrate understanding of matrix manipulation and digraph encryption.

## Illustrate the encryption process with a step-by-step example.

## Theoretical notes: The Playfair cipher encrypts messages by pairing letters and using a 5x5 matrix based on a keyword. Here’s how it works:

## Matrix Creation: The keyword fills the 5x5 grid (removing duplicates), followed by any remaining letters of the alphabet. For Romanian, adjustments are made to fit all letters.

## Pairing Rules:

## Same letters in a pair? Insert an 'X' between them.

## Single letter left? Add an 'X' to complete the pair.

## Encryption:

## Same Row: Shift letters right.

## Same Column: Shift letters down.

## Rectangle: Swap columns of the paired letters.

## Decryption reverses these steps.

## Implementation: Creating the create\_playfair\_matrix function takes a keyword, removes duplicates, and fills a 5x5 matrix. The letter 'J' is substituted with 'I' to fit within the matrix.

## def create\_playfair\_matrix(key):

## key = ''.join(sorted(set(key.upper()), key=key.upper().index))

## matrix = [['' for \_ in range(5)] for \_ in range(5)]

## row, col = 0, 0

## for char in key:

## if char == 'J': char = 'I'

## matrix[row][col] = char

## col = (col + 1) % 5

## if col == 0: row += 1

## # Fill remaining cells

## for char in 'ABCDEFGHIKLMNOPQRSTUVWXYZ':

## if char not in key:

## matrix[row][col] = char

## col = (col + 1) % 5

## if col == 0: row += 1

## return matrix

## Formatting Plaintext for Encryption in the format\_plaintext function adjusts the plaintext by converting it to uppercase, inserting 'X' between identical letters in a digraph, and adding an extra 'X' if the last pair is incomplete. def format\_plaintext(plaintext):

## formatted, i = '', 0

## while i < len(plaintext):

## formatted += plaintext[i]

## if i + 1 < len(plaintext) and plaintext[i] == plaintext[i + 1]:

## formatted += 'X'

## elif i + 1 < len(plaintext):

## formatted += plaintext[i + 1]

## i += 1

## i += 1

## if len(formatted) % 2 != 0:

## formatted += 'X'

## return formatted Encrypting Plaintext with the encrypt\_playfair function applies Playfair encryption rules for each digraph in the plaintext. Depending on their positions in the matrix, the letters are shifted as per the Playfair rules.

## def encrypt\_playfair(plaintext, key):

## plaintext = format\_plaintext(plaintext.upper())

## matrix = create\_playfair\_matrix(key)

## ciphertext = ''

## for i in range(0, len(plaintext), 2):

## char1, char2 = plaintext[i], plaintext[i + 1]

## row1, col1, row2, col2 = None, None, None, None

## for r in range(5):

## for c in range(5):

## if matrix[r][c] == char1: row1, col1 = r, c

## elif matrix[r][c] == char2: row2, col2 = r, c

## if row1 == row2:

## ciphertext += matrix[row1][(col1 + 1) % 5] + matrix[row2][(col2 + 1) % 5]

## elif col1 == col2:

## ciphertext += matrix[(row1 + 1) % 5][col1] + matrix[(row2 + 1) % 5][col2]

## else:

## ciphertext += matrix[row1][col2] + matrix[row2][col1]

## return ciphertext So in this way we can see manually what the code do: Example

## Plaintext: HELLO

## Key: MONARCHY

## Step 1: Matrix Creation

## For the key MONARCHY, the matrix is constructed as follows:

| **M** | **O** | **N** | **A** | **R** |
| --- | --- | --- | --- | --- |
| **C** | **H** | **Y** | B | D |
| E | F | G | I | K |
| L | P | Q | S | T |
| U | V | W | X | Z |

## Step 2: Plaintext Formatting

## Formatted plaintext for HELLO becomes HE LX LO (adding 'X' between duplicate 'L's).

## Step 3: Encryption

## Pair 1: HE

## 'H' is at (1,1) and 'E' is at (2,0).

## Forming a rectangle, we swap columns to get CF.

## Pair 2: LX

## 'L' is at (3,0) and 'X' is at (4,3).

## Swapping columns, we get US.

## Pair 3: LO

## 'L' is at (3,0) and 'O' is at (0,1).

## Swapping columns, we get MP.

## Final Ciphertext: CFUSMP

## Conclusion:

## The Playfair cipher successfully encrypts plaintext by substituting pairs of letters based on a 5x5 matrix. This implementation formatted plaintext, applied encryption rules, and generated accurate ciphertext. Future enhancements could include broader input validation and additional tests.

## 

## 

## 

## 

## 