MINISTRY OF EDUCATION, CULTURE, AND RESEARCH OF THE REPUBLIC OF MOLDOVA

Technical University of Moldova
Faculty of Computers, Informatics, and Microelectronics
Department of Software Engineering and Automatics

Ion Vornicescu, FAF-231

Report

Laboratory Work No. 1

of Cryptography and Security

Checked by:
Maia Zaica, university assistant
DISA, FCIM, UTM

Chișinău – 2025

Contents

1 Algorithm Analysis

1.1 Objective

Study, implement, and test the classic **Caesar Cipher** algorithm and develop an improved variant that uses an alphabet permutation based on a keyword.

1.2 Tasks

- Implementation of the **Simple Caesar Cipher** , including key validation $k \in \{1, \ldots, 25\}$.
- Implementation of the **Permuted Caesar Cipher** , using a key k_2 to generate a permuted alphabet.
- Execution of all necessary validations: text normalization (uppercase, space removal, only A-Z letters) and key restrictions.
- Practical testing of the Permuted Caesar Cipher by exchanging keys and messages with a colleague.
- Formulation of conclusions based on the work performed.

1.3 Theoretical Notes

The Caesar Cipher (or shift cipher) is a symmetric, monoalphabetic substitution algorithm.

1.3.1 Simple Cipher

Encryption and decryption are based on a single key, k_1, which represents the number of shift positions in the standard alphabet.

- **Encryption Formula:** $C = (X + k_1) \pmod{26}$
- **Decryption Formula:** $M = (Y - k_1) \pmod{26}$

Where X and Y are the numerical representations (index 0-25) of the plaintext and ciphertext characters, respectively. The key space is extremely small (≤ 25), making the cipher vulnerable to a brute-force attack.

1.3.2 Permuted Cipher

To increase complexity, a second key, k_2 (a word), is introduced, which determines a new permuted alphabet. The shift k_1 is then applied to this alphabet.

- **Permuted Alphabet Generation:** Unique letters from k_2 (in order of appearance) are taken and added to the beginning of the alphabet; the remaining letters follow in alphabetical order.
- **Security:** The key space is considerably increased to $26! \times 25$, but the cipher remains monoalphabetic and, therefore, is still vulnerable to frequency analysis.

2 Technical Implementation

2.1 Code Structure

The implementation was developed in Python and includes separate functions for:

- Validating and retrieving the numerical key k_1 (between 1 and 25).
- Validating and retrieving the keyword k_2 (minimum 7 letters).
- Generating the permuted alphabet based on k_2.
- Normalizing the text (uppercase, space removal, and character validation).
- The main cipher/decipher function that applies the Caesar formula to the permuted alphabet.

2.2 Source Code

The following Python code implements the Permuted Caesar Cipher, covering the requirements of Tasks 1.1 and 1.2.

```
1  import sys
2  import string
3
4  ALFABET_STANDARD = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
5  N = len(ALFABET_STANDARD)
6
7  def get_valid_key1():
8      while True:
9          try:
10             k1 = input("Enter the numerical key k1 (between 1 and 25):
                   ")
11             k1 = int(k1)
12             if 1 <= k1 <= 25:
```

```
13                  return k1
14              else:
15                  print("Key k1 must be between 1 and 25 inclusive.")
16          except ValueError:
17              print("Invalid input. Key k1 must be an integer.")
18
19  def get_valid_key2():
20      while True:
21          k2 = input("Enter the keyword k2 (min. 7 letters): ").upper().
                replace(' ', '')
22
23          if len(k2) < 7:
24              print("Keyword k2 must have min. 7 characters.")
25              continue
26          if not k2.isalpha():
27              print("Keyword k2 must contain only letters.")
28              continue
29
30          return k2
31
32  def generate_permuted_alphabet(k2):
33      permuted = []
34
35      for char in k2:
36          if char not in permuted:
37              permuted.append(char)
38
39      for char in ALFABET_STANDARD:
40          if char not in permuted:
41              permuted.append(char)
42
43      return "".join(permuted)
44
45  def normalize_text(text):
46      text = text.upper().replace(' ', '')
47
48      for char in text:
49          if char not in ALFABET_STANDARD:
50              print(f"\nERROR: Character '{char}' is not allowed. Only A-
                    Z.")
51              return None
52
53      return text
54
55  def caesar_perm_cipher(text, k1, permuted_alphabet, operation):
56      result = ""
```

```
57
58      for char in text:
59          try:
60              x = permuted_alphabet.index(char)
61
62              if operation == 'c':
63                  new_x = (x + k1) % N
64              elif operation == 'd':
65                  new_x = (x - k1) % N
66
67              result += permuted_alphabet[new_x]
68
69          except ValueError:
70              result += char
71
72      return result
```

Listing 1: Implementation of the Permuted Caesar Cipher (Task 1.2)

```
--- Lucrare de laborator nr. 1: Cifrul lui Cezar cu Permutare (Sarcina 1.2) ---

Alegeți operația: [C]riptare / [D]ecriptare / [I]ește: c
Introduceți cheia numerică k1 (între 1 și 25): 12
Introduceți cuvântul-cheie k2 (min. 7 litere): abracadabra

Alfabet permutată (k2='ABRACADABRA'): ABRCDEFGHIJKLMNOPQSTUVWXYZ
Introduceți mesajul: afara ploua

--- REZULTAT ---
Cheie k1: 12
Cheie k2: ABRACADABRA
Mesaj original: AFARAPLOUA
Criptograma: LSLNLRYBFL
----------------
```

Figure 1: Program results for encryption

```
Alegeți operația: [C]riptare / [D]ecriptare / [I]ește: d
Introduceți cheia numerică k1 (între 1 și 25): 12
Introduceți cuvântul-cheie k2 (min. 7 litere): abracadabra

Alfabet permutată (k2='ABRACADABRA'): ABRCDEFGHIJKLMNOPQSTUVWXYZ
Introduceți criptograma: LSLNLRYBFL

--- REZULTAT ---
Cheie k1: 12
Cheie k2: ABRACADABRA
Criptograma introdusă: LSLNLRYBFL
Mesaj decriptat: AFARAPLOUA
----------------
```

Figure 2: Program results for decryption

3 Pair Testing

3.1 Methodology

A practical encryption/decryption test was conducted with a colleague, exchanging keys and cryptograms.

3.2 Encryption of Own Message

I encrypted my own message using my keys: $k_1 = 17$ and $k_2 = \text{SEMNATURA}$.

Table 1: Encryption of Own Message

Element	**Value**
Message	**IMPLEMENT**
Numerical Key (k_1)	17
Keyword (k_2)	**SEMNATURA**
Permuted Alphabet	S E M N A T U R B C D F G H I J K L O P Q V W X Y Z
Obtained Cryptogram	**TPDBOPOQW**

3.3 Decryption of Received Message

I decrypted the message received from a colleague, using the keys provided by them: $k_1 = 9$ and $k_2 = \text{CRIPTARE}$.

Table 2: Decryption of Received Message

Element	**Value**
Received Cryptogram	**LZLXHDL**
Received Numerical Key (k_1)	9
Received Keyword (k_2)	**CRIPTARE**
Colleague's Permuted Alphabet	C R I P T A E B D F G H J K L M N O Q S U V W X Y Z
Obtained Decrypted Message	**ANALIZA**
Expected Message	**ANALIZA**

3.4 Testing Results and Conclusions

The pair testing confirmed the functionality of the implementation, as the decrypted message coincided with the colleague's original plaintext message.

4 Conclusion

The laboratory work was successfully completed, achieving the implementation and testing objectives of the Caesar Cipher. The implementation of the Permuted Caesar Cipher was realized, which significantly increases complexity by using a modified alphabet generated by a second key (k_2). All requirements for text and key validation and normalization were respected.

The practical test validated the correctness of the algorithm, demonstrating that the encryption and decryption formulas, applied to the permuted alphabet, function as expected. Although the permuted variant improves resistance to a brute-force attack on the shift key, being a monoalphabetic cipher, it remains vulnerable to advanced cryptanalysis techniques based on frequency analysis.

References

[1] Khan Academy. (2025). *Călătorie în criptografie: Cifrul lui Cezar*. Available on: https://www.youtube.com/watch?v=sMOZf4GN3oc&t=3s

[2] Schneier, Bruce. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. 2nd Edition, John Wiley & Sons.

[3] Stallings, William. (2017). *Cryptography and Network Security: Principles and Practice*. 7th Edition, Pearson.