

Ministerul Educatiei, Culturii și Cercetarii al Republicii Moldova Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică Departamentul Ingineria Software și Automatica

Report for the subject:

Cryptography and Security

***Laboratory work 1: Caesar’s Cipher***

Elaborated by:

Sirghi Tudor, gr. FAF-213

Verified by: Mîțu Cătălin

Chișinău – 2023

**Subject:** Caesar’s Cipher

## Objectives:

1. Using one of the programming languages, implement the Caesar algorithm for the English alphabet. Use only the letter encoding shown in Table 1 (it is not permitted to use ASCII or Unicode, which are specified by the programming language). Key values must range from 1 to 25, inclusive, and no other values are permitted. The assumed range of text character values is 'A' to 'Z', 'a' to 'z'; no other values are implied. If the user enters different values, the advised tuning will be correct.

Before encryption, the text is converted to all capital letters and the spaces are removed. The user will be able to select the operation (encryption or decryption), input the key, message, or cryptogram, and receive the corresponding decrypted cryptogram or message.

1. To implement the Caesar algorithm with two keys while adhering to the Task's conditions. In addition, key 2 must contain only Latin letters and be at least 7 characters long.

## Caesar’s Cipher

The encryption method employed in this cipher involves the substitution of each individual letter inside the plaintext with a distinct letter derived from a sequential displacement within the alphabet. The secret key, denoted as k, is a shared value used for both encryption and decryption processes. It represents the numerical shift applied to the alphabet, namely k∈{1, 2, 3,…, n–1}, where n is the length of the alphabet. The process of encrypting and decrypting a message using the Caesar cipher can be mathematically described by the following formulas:

c = ek(x) = x + k (mod n),

m = dk(y) = y – k (mod n),

The variables x and y reflect the numerical values assigned to the corresponding plaintext characters. The Modulo function, denoted as (a mod b), computes the remainder resulting from the division of the integer a by the integer b.

For example, for k = 3 we have (fig. 1):

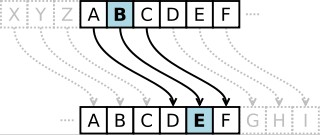


Figure 1 – Example of alphabetical displacement

To enhance the cryptoresistance of the Caesar cipher, an alphabet permutation can be applied using a keyword (not to be confused with the cipher's primary key). This key can be any sequence of alphabetic letters, either a vocabulary word or nonsense.

Let the second key be k2 =cryptography. We apply this key to the alphabet

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

obtaining:

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

## This new order was created by situating the letters of k2 at the beginning of the alphabet, followed by the remaining letters in their natural order. We will consider the fact that letters will not be repeated; if a letter appears multiple times, it will be placed only once.

## Results:

|  |
| --- |
| Select an operation:  --------------------  1 - Encryption  2 - Decryption  3 - Encryption with 2 keys  4 - Decryption with 2 keys  q - Exit  --------------------  The text for encryption/decryption must contain only Latin alphabet letters and optionally spaces.  The first key must be an integer between 1 and 25 and the second key must have at least 7 characters.  Enter your choice: |

When executing the code, the user is prompted with a menu, to chose an operation they would like to complete, along with an explanation of how the input should look.

**Results: Task 1**

When selecting option 1, which pertains to encryption, the user is prompted to provide a numeric key ranging from 1 to 25and the text for encryption,. Subsequently, the application proceeds to cleanse and check the provided input, subsequently executing Caesar cipher encryption using the selected key, and ultimately presenting the encrypted message. The program initiates a request for user input and subsequently provides the output as a response. (Table 1)

|  |
| --- |
| Choose the operation from the list above:  "1" for Encrypt  "2" for Decrypt  Input:1  Type the key for your encryption/decryption  It must be between 1 and 25: 4  Please, enter your message:I Love UTM  Result: MPSZIYXQ |

Table 1 – Encryption using 1 key

When selecting option 2, which pertains to the decryption process, the user is required to submit both the encrypted text and a numeric decryption key between the range of 1 to 25. The application performs input validation, does the reversal of the Caesar cipher decryption, and presents the resultant original text. The program initiates a request for user input and afterwards provides the decrypted message as output. (Table 2)

|  |
| --- |
| Choose the operation from the list above:  "1" for Encrypt  "2" for Decrypt  Input:2  Type the key for your encryption/decryption  It must be between 1 and 25: 4  Please, enter your message:MPSZIYXQ  Result: ILOVEUTM |

Table 2 – Decryption using 1 key

**Results: Task 2**

Also, like in previous example, user have 2 choices: ‚1’ for Encryption and ‚2’ for Decryption. It also requires a number key between 1 and 25, a string key with more than 7 charachters and text message for Encryption/Decryption

In first exemple we can see the application performs input validation, generates a shifted alphabet according to the second key, then applies encryption to the text using both keys. The output exhibits the encrypted message alongside the alphabet that has undergone a change. (Table 3)

|  |
| --- |
| Choose the operation from the list above:  "1" for Encrypt  "2" for Decrypt  Input:1  Type the key for your encryption/decryption  It must be between 1 and 25: 9  Type the second key, it must to:  1.Use only English letters from A to Z.  2.To be a string more than 7 charachters.  Your second key: Solution  Enter message: We dont have a problem without a solution  Your Modified alphabet is: SOLUTINABCDEFGHJKMPQRVWXYZ  Result: IRQDJGXKTRKOUDMERSIHGXDFGKCDEFGHDJ |

Table 3 – Encryption using 2 keys (number key and string key)

In second exemple we can see the application performs input validation, reconstructs the shifted alphabet using the second key, decrypts the text using both keys, and afterwards presents the original message. (Table 4)

|  |
| --- |
| Choose the operation from the list above:  "1" for Encrypt  "2" for Decrypt  Input:2  Type the key for your encryption/decryption  It must be between 1 and 25: 9  Type the second key, it must to:  1.Use only English letters from A to Z.  2.To be a string more than 7 charachters.  Your second key: Solution  Enter message: IRQDJGXKTRKOUDMERSIHGXDFGKCDEFGHDJ  Your Modified alphabet is: SOLUTINABCDEFGHJKMPQRVWXYZ  Result: WEDONTHAVEAPROBLEMWITHOUTASOLUTION |

Table 4 – Decryption using 2 keys (number key and string key)

For wrong cases, such as: Invalid number for key; the string is less than 7 characters; you choose wrong operation; the message is not using latin charchter, we have a specific message as output: (Table 5)

|  |
| --- |
| Result: Your operation can not be Processed.  Please, enter "1" for Encryption or "2" for Decryption. |
| Wrong key. Key must be between 1 and 25. |
| Invalid key. Second key must be at least 7 characters long. |
| Result: Only English alphabet characters (A-Z) are allowed. |

Table 5 – Response for wrong input form user

**Conclusion:**

Throughout this lab project, I gained a comprehensive understanding of the fundamental principles underlying Caesar's encryption technique. As part of this learning journey, I successfully developed a software application that effectively implements the Caesar cipher algorithm. This application enables the encryption and decryption of messages using either a single key or a pair of keys.

While initially, employing Caesar's cipher with two keys may appear straightforward, it introduces a significant level of complexity that greatly enhances security. In comparison to the single-key variant, the dual-key version significantly increases the challenge of deciphering protected messages. For instance, attempting a brute-force attack on this dual-key encryption would necessitate exploring an astounding number of 26!25 distinct possibilities to unveil the concealed message.

This enhancement in cryptography underscores the critical importance of employing robust encryption techniques, particularly when safeguarding sensitive data is paramount. The dual-key Caesar cipher not only fortifies security but also serves as an instructive tool for delving deeper into the intricacies of encryption. It illuminates the challenging balance between security and user-friendliness within the realm of cryptography, rendering it an intriguing subject for further exploration.

GitHub link for repository:  
<https://github.com/TudorSIRGHI/CS>