Ministry of Education of Republic of Moldova

Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engineering Department

Cryptography and Security

Laboratory work #1

Caesar Cipher

Author: Timur Cravtov
std. gr. FAF-231

Verified by: Maia Zaica
asist. univ

Chișinău
2025

Contents

1 Caesar Cipher

Caesar Cipher is one of the oldest and simplest methods of encryption. It is a type of substitution cipher where each letter in the plaintext is shifted a certain number of places down or up the alphabet. For example, with a shift of 3, 'A' would be replaced by 'D', 'B' would become 'E', and so on. The method is named after Julius Caesar, who reportedly used it to communicate securely with his generals [2].

Since this method is quite simple, it is also easy to break. A common way to decipher a message encoded with a Caesar Cipher is through brute force, trying all 25 possible shifts (since a shift of 26 would return the original text). Frequency analysis can also be used, as certain letters appear more frequently in the English language (like 'E', 'T', 'A', 'O', 'I', and 'N').

To secure the Caesar Cipher, one can use a permutation of the alphabet instead of a simple shift. This means that each letter is replaced by another letter according to a fixed permutation, making it more difficult to decipher without knowing the specific permutation used. In that way, there are 26! (factorial of 26) possible permutations, which significantly increases the complexity of breaking the cipher through brute force methods.

Other ways to enhance the security of the Caesar Cipher include using multiple shifts (a method known as the Vigenère cipher), or combining it with other encryption techniques. However, for modern applications, more advanced encryption methods are recommended due to the simplicity and vulnerabilities of the Caesar Cipher [3].

2 Implementation

The full code of the implementation can be found in the repository [1].

2.1 Simple Caesar Cipher

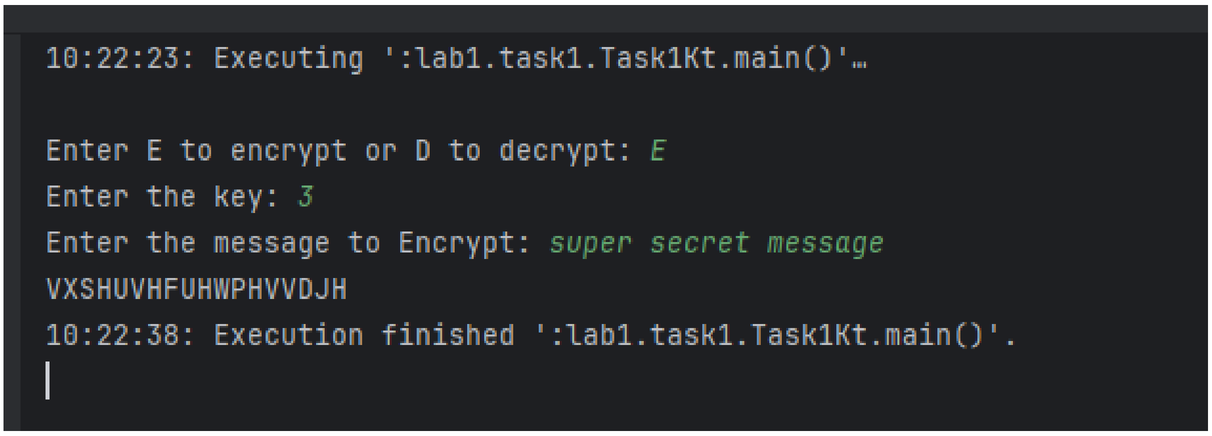
2.1.1 Encryption

To encrypt a String, an extension method for the String type is created. The method takes an integer parameter representing the shift value. First, it normalizes the string, removing spaces and making the string all uppercase. Next, it iterates through each character in the string and applies the shift accordingly.

The alphabet is represented as a string constant. For each character in the input string, the method finds its index in the alphabet, adds the shift value to this index, and uses modulo operation to wrap around if the index exceeds the length of the alphabet. The resulting character is then appended to a StringBuilder, which is returned as the final encrypted string.

```kotlin
fun String.encryptCaesar(key: Int): String {
    val normalized = this.normalizeForEncryption()
    val shift = (key % 26 + 26) % 26
    return buildString {
        for (char in normalized) {
            val index = code(char)
            if (index != -1) {
                val newIndex = (index + shift) % 26
                append(alphabet[newIndex])
            }
        }
    }
}
```

Listing 1: Kotlin code for Caesar Cipher encryption



```
10:22:23: Executing ':lab1.task1.Task1Kt.main()'…

Enter E to encrypt or D to decrypt: E
Enter the key: 3
Enter the message to Encrypt: super secret message
VXSHUVHFUHWPHVVDJH
10:22:38: Execution finished ':lab1.task1.Task1Kt.main()'.
```

Figure 1: Encryption example

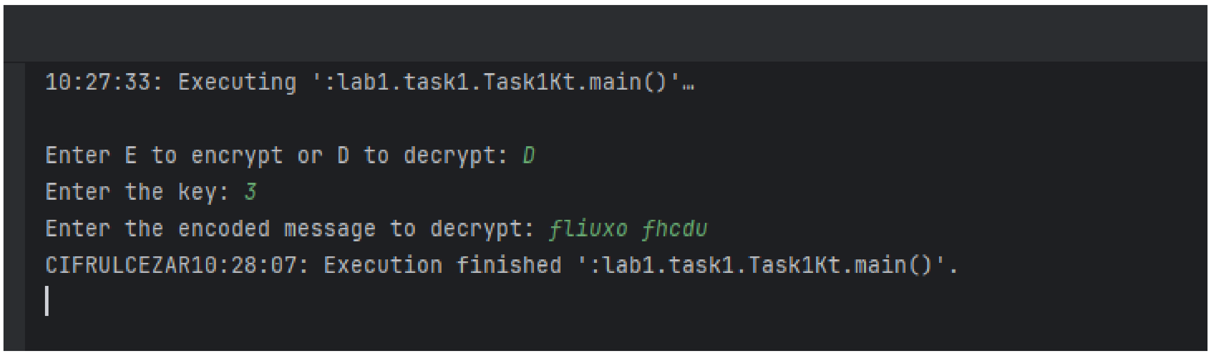
2.1.2 Decryption

Since the Caesar Cipher is symmetric, the decryption process is similar to encryption. The only difference is that the shift value is subtracted instead of added. The same extension method structure is used, iterating through each character and applying the reverse shift.

The inverse shift is calculated by subtracting the shift value from the character's index in the alphabet. The modulo operation ensures that the index wraps around correctly if it goes below zero.

```kotlin
fun String.decryptCaesar( key: Int): String {
    return this.encryptCaesar((26 - key) % 26)
}
```

Listing 2: Kotlin code for Caesar Cipher decryption



```
10:27:33: Executing ':lab1.task1.Task1Kt.main()'…

Enter E to encrypt or D to decrypt: D
Enter the key: 3
Enter the encoded message to decrypt: fliuxo fhcdu
CIFRULCEZAR10:28:07: Execution finished ':lab1.task1.Task1Kt.main()'.
```

Figure 2: Decryption example

2.2 Caesar Cypher with permutation

There are multiple ways to implement a Caesar Cipher with permutation. In laboratory below, the pernutation is based on the second key. It's letters are placed at the beginning of the alphabet, followed by the remaining letters in their original order. For example, if the key is ”KEYWORD”, the resulting alphabet would be ”KEYWORDABCFGHIJLMNPQSTUVXZ”. After the alphabet is generated, the encryption and decryption processes are similar to the simple Caesar Cipher, but using the permuted alphabet instead of the standard one.

3 Conclusions

References

[1] Laboratory work repository `https://github.com/TimurCravtov/CryptographyAndSecurityLabs`

[2] History of Caesar Cipher `https://www.cantorsparadise.com/the-history-of-codes-and-cryptography-from-caesar-cipher-to-quantum-encryption-2`

[3] Vigenère Cipher `https://www.math.stonybrook.edu/~scott/Book331/Improved_Caesar_like_cipher.html`