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: Pereteatcu Arina FAF - 223,

Checked: asist. univ.,
Dumitru Nirca

Chișinău 2024

Topic: Polyalphabetic Ciphers

Tasks:

Task 3.2: Implement the Vigenère algorithm in one of the programming languages for messages in Romanian (31 letters), with these letters encoded as numbers 0, 1, ... 30. The character values in the text are between 'A' and 'Z', 'a' and 'z', and no other values are allowed. If the user enters other values, they should be prompted with the correct character range. The key length must not be less than 7. Encryption and decryption will be performed according to the formulas in the mathematical model presented above.

In the message, spaces should first be removed, and then all letters should be transformed to uppercase. The user should be able to choose the operation – encryption or decryption – and input the key, message, or ciphertext, and receive either the ciphertext or the decrypted message as output.

Theoretical notes:

Because monoalphabetic ciphers' frequency distribution mirrors that of the alphabet they use, they are vulnerable. A more regular distribution that doesn't provide the cryptanalyst any information makes a cipher more cryptographically secure. Combining high and low distributions is one method of flattening the distribution. The combination of the high frequency of T and the low frequency of X results in a more reasonable distribution for both "a" and "b" if the letter T is occasionally encrypted as "a" and other times as "b," and if the letter X is also occasionally encrypted as "a" and other times as "b." By employing two distinct encryption alphabets—one for characters in even positions in the plaintext and another for characters in odd positions—two distributions can be merged.

Vigenere Cipher

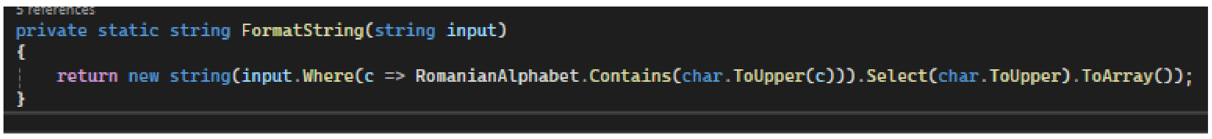
The Vigenère cipher, often mistakenly attributed to Blaise de Vigenère in the 19th century, was actually first described by Giovan Battista Bellaso in his 1553 book *La cifra del*. Vigenère later created a similar but stronger cipher in 1586.

Key Features:

- **Polyalphabetic Nature**: The Vigenère cipher employs multiple shifts for encryption, unlike the simpler Caesar cipher, which uses a single shift. This complexity makes it harder to break through frequency analysis since each letter can be encoded differently based on its position and the key.
- **Encryption Process**: Each letter of the plaintext is shifted by a number defined by a corresponding letter in the key. The encryption formula is:
  ci=(mi+ki)mod 26c_i = (m_i + k_i) \mod 26ci=(mi+ki)mod26
  where cic_ici is the ciphertext letter, mim_imi is the plaintext letter, and kik_iki is the key letter.
- **Increased Security**: The method of multiple shifts increases security for two main reasons:
  1. The key length remains unknown to attackers.
  2. The number of potential combinations grows significantly with the key length; for instance, with a key length of 5, there would be 265=11,881,37626^5 = 11,881,376265=11,881,376 possible combinations.
- **Decryption Process**: Decryption is performed similarly, but the key is subtracted from the ciphertext:
  mi=(ci−ki)mod 26m_i = (c_i - k_i) \mod 26mi=(ci−ki)mod26
- **Tabula Recta**: To facilitate encryption, a table called the Tabula Recta can be used. This table displays all possible shifts, allowing easy lookup of the ciphertext letter based on the plaintext letter and the corresponding key letter.

Implementation

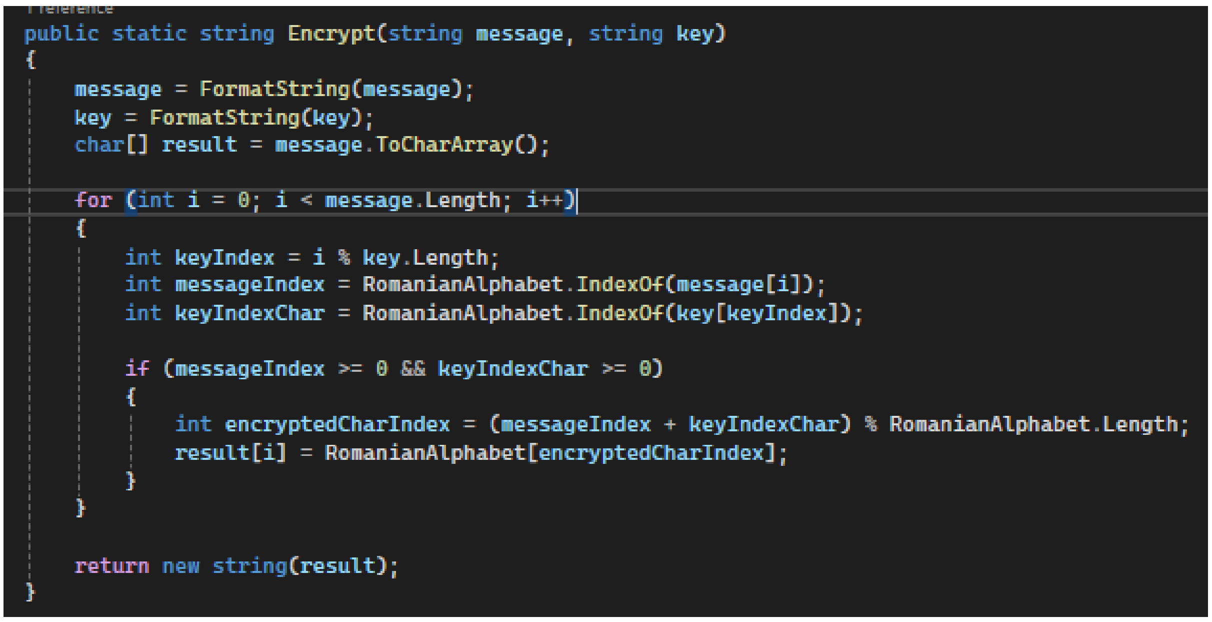
My implementation contains a public class called VigenereCipher and contains a constant string of the Romanian alphabet.



```
private static string FormatString(string input)
{
    return new string(input.Where(c => RomanianAlphabet.Contains(char.ToUpper(c))).Select(char.ToUpper).ToArray());
}
```

Figure1. Format

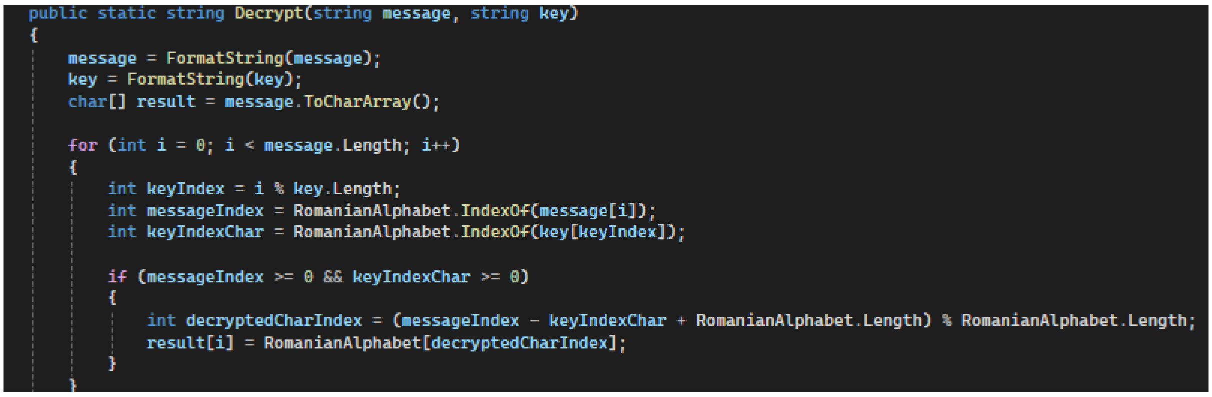
This method filters the characters in input and converts the input to upper-case.



```
public static string Encrypt(string message, string key)
{
    message = FormatString(message);
    key = FormatString(key);
    char[] result = message.ToCharArray();

    for (int i = 0; i < message.Length; i++)
    {
        int keyIndex = i % key.Length;
        int messageIndex = RomanianAlphabet.IndexOf(message[i]);
        int keyIndexChar = RomanianAlphabet.IndexOf(key[keyIndex]);

        if (messageIndex >= 0 && keyIndexChar >= 0)
        {
            int encryptedCharIndex = (messageIndex + keyIndexChar) % RomanianAlphabet.Length;
            result[i] = RomanianAlphabet[encryptedCharIndex];
        }
    }

    return new string(result);
}
```

Figure2. Encryption

The encrypt method loops over each character of the imputed message and determines the current index of the key, it checks if the message and key are valid and calculates the index of the encrypted character by adding the message and key indices, then taking the modulus to wrap around the alphabet length.



```
public static string Decrypt(string message, string key)
{
    message = FormatString(message);
    key = FormatString(key);
    char[] result = message.ToCharArray();

    for (int i = 0; i < message.Length; i++)
    {
        int keyIndex = i % key.Length;
        int messageIndex = RomanianAlphabet.IndexOf(message[i]);
        int keyIndexChar = RomanianAlphabet.IndexOf(key[keyIndex]);

        if (messageIndex >= 0 && keyIndexChar >= 0)
        {
            int decryptedCharIndex = (messageIndex - keyIndexChar + RomanianAlphabet.Length) % RomanianAlphabet.Length;
            result[i] = RomanianAlphabet[decryptedCharIndex];
        }
    }
```

Figure 3.Decryption

This method is responsible for decrypting the message. It checks for valid characters, then the index of the decrypted character is calculated by subtracting the key index from the message index and using modulus to wrap around.

Conclusion

In summary, during this laboratory work I applied the Vigenère cipher, a popular polyalphabetic encryption method designed for the Romanian alphabet. Using a methodical approach that included input validation and character formatting, the project's main goal was to show how well the Vigenère algorithm works for safely encrypting and decrypting data.

All things considered, this lab exercise not only strengthened my comprehension of polyalphabetic ciphers and how they are used, but it also gave me hands-on experience with programming ideas related to cryptography. The experiment demonstrated the value of human interaction, input validation, and methodical character handling—all crucial elements in the creation of safe cryptographic systems.