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-223, Iordan Liviu

Checked: asist. univ. Nirca Dumitru

Chișinău 2023

Topic: Vigenere:

The objective of this laboratory work was to implement the Vigenere cipher algorithm in a programming language for processing messages in Romanian. The character set for encoding was defined as 31 letters, ranging from 'A' to 'Z' and including specific Romanian characters 'Ă', 'Â', 'Î', 'Ș', and 'Ț'. The user was required to input messages that would be validated to only contain permitted characters. The key length for encryption and decryption needed to be at least 7 characters. Spaces were removed from the input, and all letters were transformed to uppercase before processing. The program had to allow users to choose between encryption and decryption, input the key and message or cryptogram, and return the processed output.

Theoretical notes:

The Vigenere cipher is a polyalphabetic substitution cipher that encrypts text using a series of Caesar ciphers based on the letters of a keyword. Each letter in the plaintext is shifted along the alphabet by a number of positions determined by the corresponding letter in the keyword. Decryption reverses this process, using the same keyword.

The algorithm follows these formulas:

- **Encryption**: $C_i = (P_i + K_i) \bmod N$
- **Decryption**: $P_i = (C_i - K_i + N) \bmod N$

Where:

- P_i and C_i are the numerical values of the plaintext and ciphertext characters.
- K_i is the numerical value of the corresponding keyword character.
- N_i is the size of the alphabet.

Implementation:

The Python code below presents a step-by-step implementation of the Vigenère cipher for Romanian text:

Snippet 1: Validating Input Characters

```
def is_valid_message(message):
    valid_chars = set("AĂÂBCDEFGHIÎJKLMNOPQRSȘTȚUVWXYZ")
    return all(char in valid_chars for char in message)
```

Explanation: This function checks if all characters in the provided message are within the allowed set of Romanian uppercase letters. It ensures that the input does not contain invalid characters, maintaining the algorithm's constraints.

Snippet 2: Preparing the Message

```
def prepare_message(message):
    message = message.replace(" ", "").upper()
    return ''.join(filter(lambda char: char in
"AĂÂBCDEFGHIÎJKLMNOPQRSȘTȚUVWXYZ", message))
```

Explanation: This function removes spaces from the message and converts all characters to uppercase. It also filters out any character not in the predefined valid set, ensuring compliance with input requirements.

Snippet 3: Generating the Key

```
def generate_key(msg, key):
    key = list(key)
    if len(msg) == len(key):
```

```
        return key
    else:
        for i in range(len(msg) - len(key)):
            key.append(key[i % len(key)])
    return "".join(key)
```

Explanation: The function extends the key to match the length of the message. This ensures that each character in the message has a corresponding character in the key during the encryption or decryption process.

Snippet 4: Encryption Logic

```
def encrypt_vigenere(msg, key):
    encrypted_text = []
    key = generate_key(msg, key)
    alphabet = "AĂÂBCDEFGHIÎJKLMNOPQRSȘTȚUVWXYZ"
    for i in range(len(msg)):
        char = msg[i]
        if char in alphabet:
            encrypted_index = (alphabet.index(char) +
alphabet.index(key[i])) % len(alphabet)
            encrypted_char = alphabet[encrypted_index]
        else:
            encrypted_char = char
        encrypted_text.append(encrypted_char)
    return "".join(encrypted_text)
```

Explanation: The encryption function iterates over the message, shifting each character by the position of the corresponding key character in the alphabet. The modulo operation ensures wrap-around at the end of the alphabet.

Snippet 5: Decryption Logic

```
def decrypt_vigenere(msg, key):
    decrypted_text = []
    key = generate_key(msg, key)
    alphabet = "AĂÂBCDEFGHIÎJKLMNOPQRSȘTȚUVWXYZ"
    for i in range(len(msg)):
        char = msg[i]
        if char in alphabet:
            decrypted_index = (alphabet.index(char) - 
alphabet.index(key[i]) + len(alphabet)) % len(alphabet)
            decrypted_char = alphabet[decrypted_index]
        else:
            decrypted_char = char
        decrypted_text.append(decrypted_char)
    return "".join(decrypted_text)
```

Explanation: The decryption function reverses the encryption process by subtracting the key character's position from the ciphertext character's position and normalizing with the modulo operation to handle the wrap-around.

Conclusion:

This laboratory work demonstrated the successful implementation of the Vigenère cipher for Romanian text, accommodating the unique characters specific to the language. The task enhanced understanding of polyalphabetic substitution techniques, modular arithmetic, and input validation. The project emphasized secure programming practices by enforcing strict input

controls and required the adaptation of the algorithm for a non-standard alphabet.