

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж



Лабораторна робота №2

з дисципліни: «Технології захисту інформації»

на тему: «Блокове шифрування інформації та шифри моноалфавітної заміни»

Варіант №6

Виконала:

студентка групи ІТ-32

Моляща Ю.А.

Прийняв:

Табачишин Д.Р.

Львів - 2023

Лабораторна робота №2

«Блокове шифрування інформації та шифри моноалфавітної заміни»

Мета роботи: навчитися опрацьовувати (шифрувати та дешифрувати) файли з допомогою блокового шифрування інформації, методів моноалфавітної заміни та методу гамування.

Завдання на лабораторну роботу.

Завдання 1. Написати програму на мові C++ (чи іншій за згодою викладача) яка виконує шифрування повідомлення методом накладення гамми використовуючи таблицю 1. Гамма шифру і повідомлення вибираються згідно з варіантом з табл. 3.

Гамма: БЛАГОВИД. Повідомлення: ПІРЕНИЇ_2500_ТОНН.

Завдання 2. Використовуючи алфавіт (табл. 1), дешифрувати повідомлення, яке зашифроване методом гамування. Гамма шифру та повідомлення вибираються згідно з варіантом з таблиці 4.

Гамма: ДАНТУР. Повідомлення: СВИЙБІПІР03122М.

Завдання 3. Оформити алгоритм функціонування програми та здійснити його опис.

Завдання 4. Оформити звіт в згідно вимог, результати подати у вигляді текстів програми, відповідних пояснень та скріншотів.

Хід роботи.

Код програми:

```
# String to store ukrainian alphabet
alphabet = "АБВГГДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ_0123456789"
ALPHABET_LENGTH = len(alphabet)
# Create a dictionary to map each character to its index
alphabet_dict = {char: str(i) for i, char in enumerate(alphabet, start=1)}
def encrypt(message, gamma):
    """
    Encrypt a message using gamma.

    Claculations for index of encrypted character:
        Add each pair of indices from message and gamma.

    Args:
        message (str): Message to be encrypted.
        gamma (str): Gamma to encrypt the message.

    Returns:
        encrypted_message (str): Encrypted message.
    """
    encrypted_message = ""
    gamma = check_length(message, gamma)

    # Convert each character to its index in the alphabet and store in a list
```

```

message_indices_list = [ int(alphabet_dict[char]) for char in message ]
gamma_indices_list = [ int(alphabet_dict[char]) for char in gamma ]

encrypted_indices_list = [
    (char_message + char_gamma) % ALPHABET_LENGTH
    for char_message, char_gamma in zip(message_indices_list,
gamma_indices_list)
]

for item in encrypted_indices_list:
    for key, value in alphabet_dict.items():
        if str(item) == value:
            index = key
            encrypted_message += index
return encrypted_message

def decrypt(message, gamma):
    """
    Decrypt a message using gamma.

    Claculations for index of decrypted character:
        Subtract each pair of indices from message and gamma.
        Add alphabet length to the result for possible negative values.
        Get modulo of the result with the length of the alphabet.
        If the result is 0 replace it with the length of the alphabet.

    Args:
        message (str): Message to be decrypted.
        gamma (str): Gamma to decrypt the message.

    Returns:
        decrypted_message (str): Decrypted message.
    """
    decrypted_message = ""
    gamma = check_length(message, gamma)

    # Convert each character to its index in the alphabet and store in a list
    message_indices_list = [ int(alphabet_dict[char]) for char in message ]
    gamma_indices_list = [ int(alphabet_dict[char]) for char in gamma ]

    decrypted_indices_list = [
        (char_message - char_gamma + ALPHABET_LENGTH) % ALPHABET_LENGTH
        if (char_message - char_gamma + ALPHABET_LENGTH) % ALPHABET_LENGTH != 0
        else ALPHABET_LENGTH
        for char_message, char_gamma in zip(message_indices_list,
gamma_indices_list)
    ]

    for item in decrypted_indices_list:

```

```

        for key, value in alphabet_dict.items():
            if str(item) == value:
                index = key
            decrypted_message += index

    return decrypted_message

def check_length(message, gamma):
    """
    Check if the length of gamma is equal to the length of message.
    If gamma is shorter than message, repeat gamma to match the length of
    message.
    If gamma is longer than message, trim gamma to match the length of message.

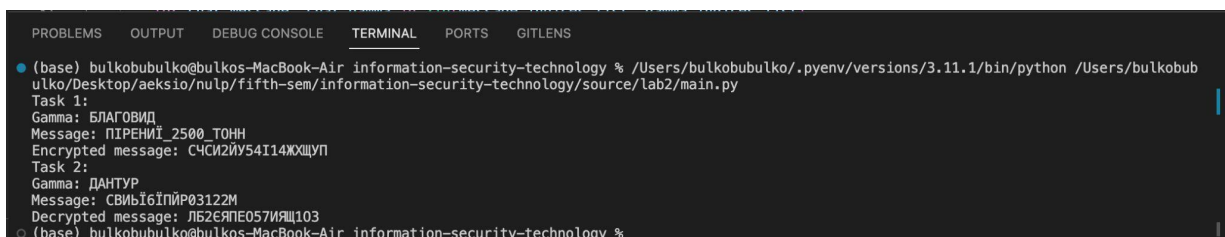
    Args:
        message (str): Message to be encrypted or decrypted.
        gamma (str): Gamma to encrypt or decrypt the message.

    Returns:
        gamma (str): Gamma with the same length as message.
    """
    if len(gamma) < len(message):
        gamma = gamma * (len(message) // len(gamma)) + gamma[:len(message) %
len(gamma)]
    else:
        gamma = gamma[:len(message)]
    return gamma

gamma = "БЛАГОВИД"
message = "ПІРЕНІЇ_2500_ТОНН"
print('Task 1:')
print(f'Gamma: {gamma}')
print(f'Message: {message}')
print(f'Encrypted message: {encrypt(message, gamma)}')
gamma2 = "ДАНТУР"
message2 = "СВІЙІ6ІПІР03122М"
print('Task 2:')
print(f'Gamma: {gamma2}')
print(f'Message: {message2}')
print(f'Decrypted message: {decrypt(message2, gamma2)}')

```

Приклад виконання зображено на Рис. 1.



```

(base) bulkobubulko@bulkos-MacBook-Air information-security-technology % /Users/bulkobubulko/Desktop/aeasio/nulp/fifth-sem/information-security-technology/source/lab2/main.py
Task 1:
Gamma: БЛАГОВИД
Message: ПІРЕНІЇ_2500_ТОНН
Encrypted message: СЧИ2ІУ54114ЮЩУП
Task 2:
Gamma: ДАНТУР
Message: СВІЙІ6ІПІР03122М
Decrypted message: ЛБ2ЄЯПЕ057ИЯЩ103
(base) bulkobubulko@bulkos-MacBook-Air information-security-technology %

```

Рис. 1. Приклад виконання

Посилання на GitHub-репозиторій: <https://github.com/bulkobubulko/nulp-ist>

Висновок: під час виконання лабораторної роботи було вивчено опрацювання (шифрувати та дешифрувати) файлів з допомогою блокового шифрування інформації, методів моноалфавітної заміни та методу гамування.