Лабораторна робота №2 (Варіант 3)

Тема: Шифрування даних за допомогою генератора псевдовипадкових чисел.

Мета роботи: Ознайомлення з принципами шифрування методом гамування за допомогою лінійного конгруентного генератора псевдовипадкових чисел (ПВЧ) та реалізація програмного забезпечення для шифрування та дешифрування тексту.

Теоретичні відомості

Гамування — це метод шифрування, при якому до відкритого тексту додається псевдовипадкова послідовність, згенерована генератором ПВЧ. Для шифрування і дешифрування використовується одна й та сама псевдовипадкова послідовність.

Генератор ПВЧ описується рівнянням:

T(i+1)=(A×T(i)+C)mod MT(i+1) = (A \times T(i) + C) \mod MT(i+1)=(A×T(i)+C)modM

де:

- AAA, CCC — константи,
- MMM — модуль, що зазвичай дорівнює 2b2^b2b, де bbb — кількість розрядів (бітів),
- T(0)T(0)T(0) — початкове значення, яке виступає ключем.

Процес шифрування полягає в накладенні псевдовипадкових чисел на відкритий текст за допомогою побітової операції XOR. Аналогічно проводиться дешифрування.

Вхідні дані

- Початковий текст: "Привіт"
- Параметри генератора:
  - A=5A = 5A=5
  - C=3C = 3C=3
  - M=256M = 256M=256 (для 8-бітного кодування)
  - Початкове значення T(0)=7T(0) = 7T(0)=7

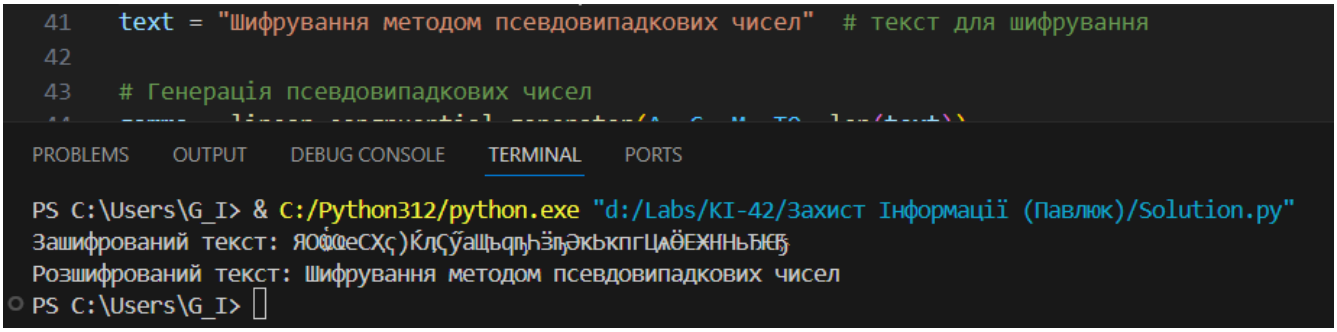
Алгоритм програми

1. Перетворення тексту: Початковий текст конвертується в двійкову послідовність.
2. Генерація псевдовипадкових чисел: Використовується лінійний конгруентний генератор для формування гами.
3. Шифрування: Накладання гами на текст за допомогою побітової операції XOR.
4. Дешифрування: Повторне застосування тієї ж гами для отримання початкового тексту.

```python
# Лінійний конгруентний генератор
def linear_congruential_generator(A, C, M, T0, length):
    numbers = [T0]
    for _ in range(1, length):
        T_next = (A * numbers[-1] + C) % M
        numbers.append(T_next)
    return numbers

# Функція для перетворення тексту в Unicode-коди (підтримка кирилиці)
def text_to_unicode(text):
    return [ord(char) for char in text]

# Функція для перетворення Unicode-кодів назад у текст
def unicode_to_text(unicode_list):
    return ''.join(chr(code) for code in unicode_list)

# Шифрування тексту
def encrypt(text, gamma):
    text_unicode = text_to_unicode(text)
    encrypted = []
    for i in range(len(text_unicode)):
        # Шифрування за допомогою операції XOR
        encrypted_char = text_unicode[i] ^ gamma[i % len(gamma)]
        encrypted.append(encrypted_char)
    return unicode_to_text(encrypted)

# Дешифрування тексту
def decrypt(encrypted_text, gamma):
    encrypted_unicode = text_to_unicode(encrypted_text)
    decrypted = []
    for i in range(len(encrypted_unicode)):
        decrypted_char = encrypted_unicode[i] ^ gamma[i % len(gamma)]
        decrypted.append(decrypted_char)
    return unicode_to_text(decrypted)

# Початкові параметри
A = 5  # константа A
C = 3  # константа C
M = 256  # модуль (2^8 для 8 розрядів)
T0 = 7  # початкове значення T(0)
text = "Шифрування методом псевдовипадкових чисел"  # текст для шифрування

# Генерація псевдовипадкових чисел
gamma = linear_congruential_generator(A, C, M, T0, len(text))
```

```python
# Шифрування
encrypted_text = encrypt(text, gamma)
print(f"Зашифрований текст: {encrypted_text}")

# Дешифрування
decrypted_text = decrypt(encrypted_text, gamma)
print(f"Розшифрований текст: {decrypted_text}")
```

Результат виконання програми:



Висновки

Під час виконання лабораторної роботи було опрацьовано та застосовано на практиці метод гамування за допомогою лінійного конгруентного генератора ефективно працює для шифрування даних. Зашифрований текст важко розкрити без знання початкових параметрів генератора ПВЧ. Однак, шифрування може бути зламане, якщо відомий хоча б фрагмент вихідного тексту.