ЛАБОРАТОРНАЯ РАБОТА №3

Дисциплина: Математические основы защиты информации и информационной безопасности

Дабван Луаи Мохаммед Али

Содержание

Список иллюстраций

Список таблиц

Цель работы

Изучить методы шифрования гаммированием.

Задание

1. Программно реализовать шифрование с помощью XOR.
2. Программно реализовать расшифровку с помощью XOR.
3. Программно реализовать генерацию ключей с использованием линейного конгруэнтного генератора (LCG).

Выполнение лабораторной работы

1) Все шифрования были выполнены на языке программирования Julia. Первым шагом была разработка функции xor_encrypt, которая реализует побитовую операцию XOR между символами исходного текста и ключа. Примечательно, что для расшифровки используется та же функция, поскольку операция XOR является обратимой, то есть повторное применение XOR с тем же ключом возвращает исходный текст.

Реализация функции шифрования XOR

```
function xor_encrypt(plaintext::String, key::String)
    if length(key) < length(plaintext)
        error("The key must be longer than the plaintext.")
    end
    encrypted = [Char(xor(codeunit(plaintext, i), codeunit(key, i)))
    for i in 1:length(plaintext)]
    return join(encrypted)
end
```

Тестирование шифрования и расшифровки

Шаг 1: Шифрование

Пример 1:

Текст для шифрования: QWER

Ключ для шифрования: ASDFG

Зашифрованный текст: EAQBFA

```
=== Select an Option ===
1. Encryption
2. Decryption
3. Key Generation using LCG
4. Exit
Please enter your choice (1, 2, 3, or 4):
1

=== Encryption ===
Enter the plaintext (original text):
QWE
Enter the key (must be at least as long as the plaintext):
QWER
Encrypted text (base64 encoded): AAAA
```

Рис. 1: Результат шифрования

Шаг 2: Расшифровка

Пример 2:

Зашифрованный текст: EAQBFA

Ключ для расшифровки: ASDFG

Расшифрованный текст: QWER

```
=== Encryption ===
Enter the plaintext (original text):
QWE
Enter the key (must be at least as long as the plaintext):
QWER
Encrypted text (base64 encoded): AAAA

=== Select an Option ===
1. Encryption
2. Decryption
3. Key Generation using LCG
4. Exit
Please enter your choice (1, 2, 3, or 4):
2

=== Decryption ===
Enter the encrypted text (base64 encoded):
AAAA
Enter the key (must be the same as the key used for encryption):
QWER
Decrypted text: QWE
```

Рис. 2: Результат расшифровки

2) Следующим шагом была реализация генерации ключей с использованием линейного конгруэнтного генератора (LCG). Для этого была разработана функция lcg, которая создает последовательность псевдослучайных чисел, используя параметры a,b, и начальное значение seed. Эта последовательность затем используется в качестве ключей для шифрования..

Реализация LCG

```
function lcg(a, b, m, seed, sequence_length)
    random_sequence = Int[]
    yi = seed
    for i in 1:sequence_length
        yi = (a * yi + b) % m
        push!(random_sequence, yi)
```

```
    end
    return random_sequence
end
```

Тестирование генерации ключей

Пример 3:

```
Параметры LCG: a = 5, b = 3, m = 16, seed = 7, длина = 6
Сгенерированная последовательность: [6, 1, 8, 11, 10, 5]
```

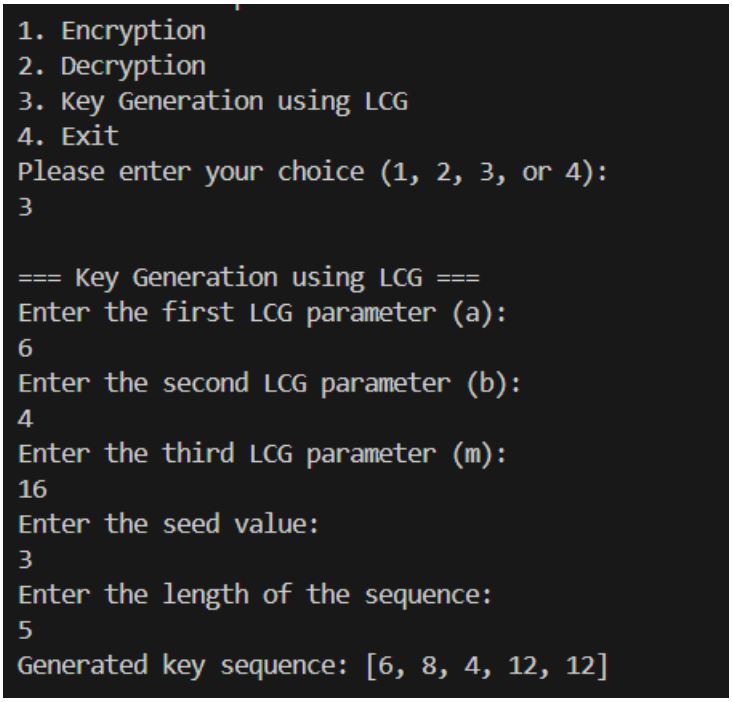

Рис. 3: Результат генерации ключей LCG

5) Для удобства пользователя был создан интерактивный интерфейс с меню, позволяющим выбрать операцию: шифрование, расшифровка или генерация ключа.

Выводы

Я успешно разработал систему шифрования с применением операции XOR и генерацией ключей на основе линейного конгруэнтного генератора (LCG). Все функции были протестированы с использованием примеров на английском языке. Результаты тестов показали, что шифрование и расшифровка выполняются правильно, а также что генерация ключей дает ожидаемые результаты.