

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы защиты информации»

ОТЧЕТ
к лабораторной работе № 4
на тему «Асимметричная криптография. Алгоритм Мак-Элиса»

Выполнил

Е. А. Киселева

Проверил

А. В. Герчик

Минск 2024

СОДЕРЖАНИЕ

1 Постановка задачи.....	3
2 Краткие теоретические сведения.....	4
3 Результаты выполнения лабораторной работы.....	5
Выводы	6
Список использованных источников	7
Приложение А (обязательное) Листинг исходного кода	8

1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является изучение теоретических сведений и реализация криптостойкого программного средства шифрования и дешифрования текстовых файлов при помощи Криптосистемы Мак-Элиса.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В системе Мак-Элиса параметрами системы, общими для всех абонентов, являются числа k , n , t . Для получения открытого и соответствующего секретного ключа каждому из абонентов системы следует осуществить следующие действия:

1 Выбрать порождающую матрицу $G = G_{kn}$ двоичного (n,k) -линейного кода, исправляющего t ошибок, для которого известен эффективный алгоритм декодирования.

2 Случайно выбрать двоичную невырожденную матрицу $S = S_k$.

3 Случайно выбрать подстановочную матрицу $P = P_n$.

4 Вычислить произведение матриц $G_1 = S \cdot G \cdot P$.

Открытым ключом является пара (G_1, t) , секретным – тройка (S, G, P) .

Для того чтобы зашифровать сообщение M , предназначенное для абонента А, абоненту В следует выполнить следующие действия:

- представить M в виде двоичного вектора длины k ;
- выбрать случайный бинарный вектор ошибок Z длиной n , содержащий не более t единиц;
- вычислить бинарный вектор $C = M * G_A + Z$ и направить его абоненту А.

Получив сообщение C , абонент А вычисляет вектор $C_1 = C * P^{-1}$, с помощью которого, используя алгоритм декодирования кода с порождающей матрицей G , получает далее векторы M_1 и $M = M_1 * S^{-1}$.

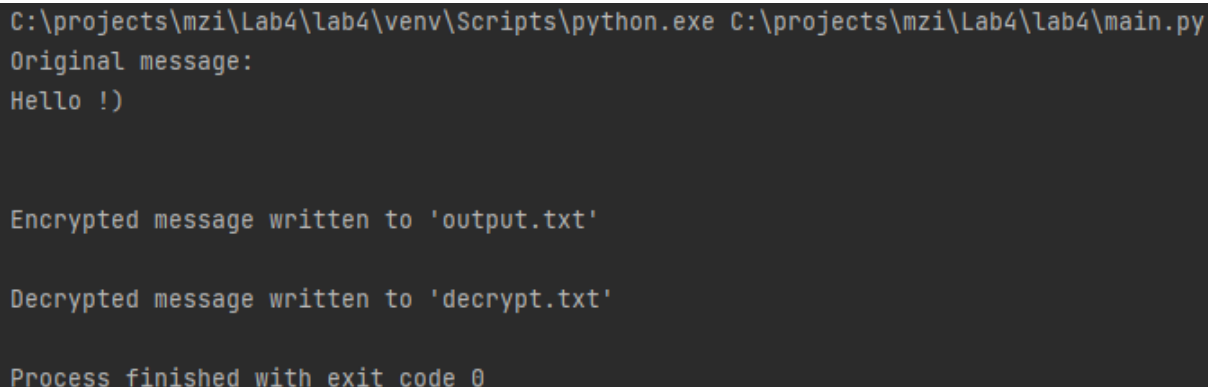
В качестве кода, исправляющего ошибки в системе Мак-Элиса, можно использовать код Гоппы. Известно, что для любого неприводимого полинома $g(x)$ степени t над полем $GF(2^m)$ существует бинарный код Гоппы длины $n = 2^m$ и размерности $k \geq n - mt$, исправляющий до t ошибок включительно, для которого имеется эффективный алгоритм декодирования.

Рекомендуемые параметры этой системы – $n = 1024$, $t = 38$, $k > 644$ – приводят к тому, что открытый ключ имеет размер около 219 бит, а длина сообщения увеличивается при шифровании примерно в 1,6 раза, в связи с чем данная система не получила широкого распространения.

3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи криптосистемы Мак-Элиса.

Начальный текст находится в файле input.txt. После запуска программа шифрует исходный файл и зашифрованные данные заносит в файл output.txt. Сразу после этого программа расшифровывает данные из файла output.txt и заносит расшифрованный текст в файл decrypt.txt. Работа программы представлен на рисунке 3.1.



```
C:\projects\mzi\Lab4\lab4\venv\Scripts\python.exe C:\projects\mzi\Lab4\lab4\main.py
Original message:
Hello !)

Encrypted message written to 'output.txt'

Decrypted message written to 'decrypt.txt'

Process finished with exit code 0
```

Рисунок 3.1 – Вывод консоли

Таким образом, в ходе данной лабораторной работы было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи криптосистемы Мак-Элиса.

ВЫВОДЫ

В ходе данной лабораторной работы были изучены теоретические сведения и реализовано криптостойкое программное средство шифрования и дешифрования текстовых файлов при помощи Криптосистемы Мак-Элиса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Криптосистема Мак-Элиса [Электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/kriptosistema-mak-elisa-i-problemy-eyo-vnedreniya>. – Дата доступа: 08.10.2024.

[2] Криптосистема Мак-Элиса в атаках декодирования классической информации [Электронный ресурс]. – Режим доступа: <https://web.snauka.ru/issues/2020/06/92527>. – Дата доступа: 09.10.2024.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг исходного кода

Листинг 1 – Программный код файла main

```
import os
import numpy as np
import base64

input_file_path = "input.txt"
output_file_path = "output.txt"
decrypt_file_path = "decrypt.txt"

if not os.path.exists(input_file_path):
    print(f"Input file '{input_file_path}' not found.")
else:
    with open(input_file_path, "r", encoding="utf-8") as file:
        original_message = file.read()
    print("Original message:\n" + original_message)

    def generate_random_matrix(rows, columns):
        return np.random.randint(2, size=(rows, columns), dtype=np.uint8)

    def encrypt(message, generator_matrix):
        message_bytes = message.encode('utf-8')
        encrypted_bytes = bytearray(len(message_bytes))
        rows, columns = generator_matrix.shape

        for i in range(len(message_bytes)):
            encrypted_bytes[i] = message_bytes[i] ^ generator_matrix[i %
rows, i % columns]

        return base64.b64encode(encrypted_bytes).decode('utf-8')

    def decrypt(encrypted_message, generator_matrix):
        encrypted_bytes = base64.b64decode(encrypted_message)
        decrypted_bytes = bytearray(len(encrypted_bytes))
        rows, columns = generator_matrix.shape

        for i in range(len(encrypted_bytes)):
            decrypted_bytes[i] = encrypted_bytes[i] ^ generator_matrix[i %
rows, i % columns]

        return decrypted_bytes.decode('utf-8')

    # Размеры матрицы
    n = 512
    k = 256
    generator_matrix = generate_random_matrix(k, n)

    encrypted_message = encrypt(original_message, generator_matrix)

    with open(output_file_path, "w", encoding="utf-8") as file:
        file.write(encrypted_message)
    print(f"\nEncrypted message written to '{output_file_path}'")

    decrypted_message = decrypt(encrypted_message, generator_matrix)

    with open(decrypt_file_path, "w", encoding="utf-8") as file:
        file.write(decrypted_message)
    print(f"\nDecrypted message written to '{decrypt_file_path}'")
```