

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 №1
на тему

**СИММЕТРИЧНАЯ КРИПТОГРАФИЯ.
СТАНДАРТ ШИФРОВАНИЯ ГОСТ 28147-89**

Студент

Д. С. Кончик

Преподаватель

Е. А. Лещенко

Минск 2024

СОДЕРЖАНИЕ

1 Постановка задачи.....	3
2 Краткие теоретические сведения.....	4
3 Результаты выполнения лабораторной работы.....	5
Выводы	15
Список использованных источников	7
Приложение А (обязательное) Листинг кода.....	8

1 ПОСТАНОВКА ЗАДАЧИ

Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи стандарта шифрования ГОСТ 28147-89 в режиме генерации имитоприставок.

2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

ГОСТ 28147-89 «Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования» - устаревший государственный стандарт СССР, описывающий алгоритм симметричного блочного шифрования и режимы его работы. Симметричные криптосистемы – способ шифрования, в котором для шифрования и расшифрования применяется один и тот же криптографический ключ. Блочный шифр – разновидность симметричного шифра, оперирующего группам бит фиксированной длины – блоками, характерный размер которых меняется в пределах 64-256 бит. Если исходный текст меньше размера блока, перед шифрованием его дополняют.

ГОСТ 28147-89 является примером DES-подобных криптосистем, созданных по классической итерационной схеме Фейстеля. DES – алгоритм для симметричного шифрования, разработанный фирмой IBM и утвержденный правительством США в 1977 году как официальный стандарт. Размер блока для DES равен 64 битам. В основе алгоритма лежит сеть Фейстеля с 16 циклами и ключом, имеющим длины 56 бит.

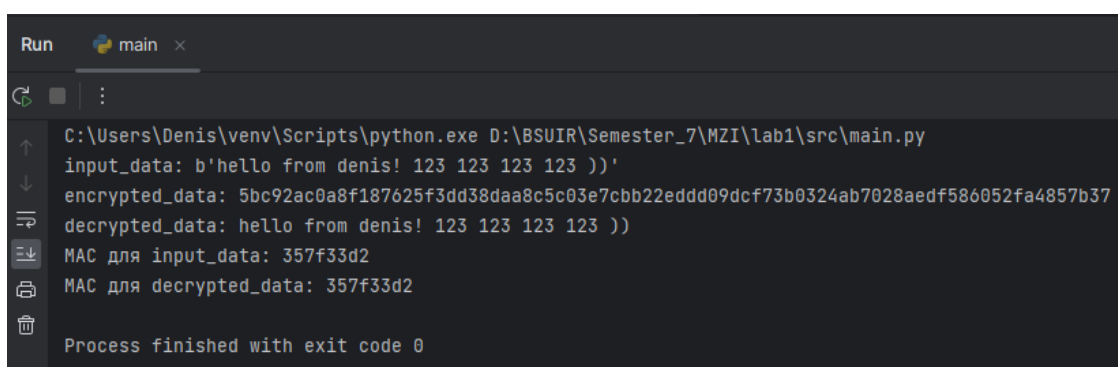
Сеть Фейстеля, или конструкция Фейстеля – один из методов построения блочных шифров. Сеть состоит из ячеек, называемых ячейками Фейстеля. На вход каждой ячейки поступают данные и ключ. На выходе каждой ячейки получают измененные данные и измененный ключ. Все ячейки однотипны, и считается, что сеть представляет собой определенную итерированную структуру. Ключ выбирается в зависимости от алгоритма шифрования или дешифрования и меняется при переходе от одной ячейки к другой. При шифровании и расшифровании выполняются одни и те же операции, отличается только порядок ключей.

Режим выработки имитовставки не является в общепринятом смысле режимом шифрования. При работе в режиме выработки имитовставки создается некоторый дополнительный блок, зависящий от всего текста и ключевых данных. Данный блок используется для проверки того, что в шифротекст случайно или преднамеренно не были внесены искажения. Это особенно важно для шифрования в режиме гаммирования, где злоумышленник может изменить конкретные биты, даже не зная ключа. Однако и при работе в других режимах вероятные искажения нельзя обнаружить, если в передаваемых данных нет избыточной информации.

3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи стандарта шифрования ГОСТ 28147-89 в режиме генерации имитоприставок.

Начальный текст находится в файле `input_file.txt`. После шифрования зашифрованная информация помещается в файл `encrypted_file.bin`, после чего она снова дешифруется и помещается в файл `decrypted_file.txt`. Генерация имитовставок проводится для данных `input_data` и `decrypted_data`. Если данные в каждом из файлов совпадают и их целостность не была нарушена, в консоли можно увидеть, что имитовставки (MAC) будут одинаковы. Вывод программы представлен на рисунке 1.



```
Run main x
C:\Users\Denis\venv\Scripts\python.exe D:\BSUIR\Semester_7\MZI\lab1\src\main.py
input_data: b'hello from denis! 123 123 123 123 ))'
encrypted_data: 5bc92ac0a8f187625f3dd38daa8c5c03e7cbb22eddd09dcf73b0324ab7028aedf586052fa4857b37
decrypted_data: hello from denis! 123 123 123 123 ))
MAC для input_data: 357f33d2
MAC для decrypted_data: 357f33d2
Process finished with exit code 0
```

Рисунок 1 – Вывод программы

Таким образом, в ходе данной лабораторной работы было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи стандарта шифрования ГОСТ 28147-89 в режиме генерации имитоприставок.

ВЫВОДЫ

В ходе данной лабораторной работы было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи стандарта шифрования ГОСТ 28147-89 в режиме генерации имитоприставок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Алгоритм шифрования ГОСТ 28147-89 [Электронный ресурс]. – Режим доступа: <https://kaf401.rloc.ru/Criptfiles/gost28147/GOST28147.htm>. – Дата доступа: 08.09.2024.

[2] О шифровании ГОСТ 28147-89 [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/256843/>. – Дата доступа: 08.09.2024.

[3] Реализация алгоритма шифрования по ГОСТ 28147-89 [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%93%D0%9E%D0%A1%D0%A2_28147-89.html. – Дата доступа: 08.09.2024.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг исходного кода

```
S_BOX = [
    [0x9, 0x6, 0x3, 0x2, 0x8, 0xB, 0x1, 0x7, 0xA, 0x4, 0xE, 0xF, 0xC, 0x0,
    0xD, 0x5], # S-box 0
    [0x3, 0x7, 0xE, 0x9, 0x8, 0xA, 0xF, 0x0, 0x5, 0x2, 0x6, 0xC, 0xB, 0x4,
    0xD, 0x1], # S-box 1
    [0xE, 0x4, 0x6, 0x2, 0xB, 0x3, 0xD, 0x8, 0xC, 0xF, 0x5, 0xA, 0x0, 0x7,
    0x1, 0x9], # S-box 2
    [0xE, 0x7, 0xA, 0xC, 0xD, 0x1, 0x3, 0x9, 0x0, 0x2, 0xB, 0x4, 0xF, 0x8,
    0x5, 0x6], # S-box 3
    [0xB, 0x5, 0x1, 0x9, 0x8, 0xD, 0xF, 0x0, 0xE, 0x4, 0x2, 0x3, 0xC, 0x7,
    0xA, 0x6], # S-box 4
    [0x3, 0xA, 0xD, 0xC, 0x1, 0x2, 0x0, 0xB, 0x7, 0x5, 0x9, 0x4, 0x8, 0xF,
    0xE, 0x6], # S-box 5
    [0x1, 0xD, 0x2, 0x9, 0x7, 0xA, 0x6, 0x0, 0x8, 0xC, 0x4, 0x5, 0xF, 0x3,
    0xB, 0xE], # S-box 6
    [0xB, 0xA, 0xF, 0x5, 0x0, 0xC, 0xE, 0x8, 0x6, 0x2, 0x3, 0x9, 0x1, 0x7,
    0xD, 0x4] # S-box 7
]
```

```
def cyclic_shift_left(value, shift):
    return ((value << shift) & 0xFFFFFFFF) | (value >> (32 - shift))
```

```
def F(block, key):
    temp = (block + key) & 0xFFFFFFFF
    result = 0
    for i in range(8):
        s_input = (temp >> (4 * i)) & 0xF
        s_output = S_BOX[i][s_input]
        result |= s_output << (4 * i)
    return cyclic_shift_left(result, 11)
```

```
def encrypt_block(block, key):
    N1, N2 = block[0], block[1]
    for i in range(32):
        round_key = key[i % 8]
        temp = F(N1, round_key)
        temp ^= N2
        N2, N1 = N1, temp
    return [N2, N1]
```

```
def decrypt_block(block, key):
    N1, N2 = block[0], block[1]
    for i in range(31, -1, -1):
        round_key = key[i % 8]
        temp = F(N1, round_key)
        temp ^= N2
        N2, N1 = N1, temp
    return [N2, N1]
```



```

def process_data(data, key, encrypt=True):
    padded_data = pad_data(data)
    result_data = []
    for i in range(0, len(padded_data), 8):
        block = [
            int.from_bytes(padded_data[i:i + 4], byteorder='little'),
            int.from_bytes(padded_data[i + 4:i + 8], byteorder='little')
        ]
        if encrypt:
            encrypted_block = encrypt_block(block, key)
        else:
            encrypted_block = decrypt_block(block, key)
        for value in encrypted_block:
            result_data.append(value.to_bytes(4, byteorder='little'))
    return b''.join(result_data)

def pad_data(data, block_size=8):
    padding_size = (block_size - (len(data) % block_size)) % block_size
    return data + bytes([0] * padding_size)

def unpad(data):
    i = len(data) - 1
    while i >= 0 and data[i] == 0:
        i -= 1
    return data[:i + 1]

def generate_mac(data, key, L=32):
    mac = [0x00000000, 0x00000000]
    block = [0, 0]
    for i in range(0, len(data), 8):
        block[0] = int.from_bytes(data[i:i + 4], byteorder='little')
        block[1] = int.from_bytes(data[i + 4:i + 8], byteorder='little')
        mac[0] ^= block[0]
        mac[1] ^= block[1]
        mac = encrypt_block(mac, key)
    mac_bytes = mac[0].to_bytes(4, byteorder='little') + mac[1].to_bytes(4,
byteorder='little')
    mac_final = mac_bytes[:L // 8]
    return mac_final

# Пример использования с файлами
if __name__ == "__main__":
    key = [
        0xA56BABCD, 0xDEF01234, 0x789ABCDE, 0xFEDCBA98,
        0x01234567, 0x89ABCDEF, 0x12345678, 0x9ABCDEF0
    ]

    # Чтение исходных данных из файла
    with open("input_file.txt", "rb") as infile:

```

```

    input_data = infile.read()

# Шифрование данных
encrypted_data = process_data(input_data, key, encrypt=True)

# Запись зашифрованных данных в файл
with open("encrypted_file.bin", "wb") as encrypted_file:
    encrypted_file.write(encrypted_data)

# Чтение зашифрованных данных из файла и расшифровка
with open("encrypted_file.bin", "rb") as encrypted_file:
    encrypted_data = encrypted_file.read()

decrypted_data = process_data(encrypted_data, key, encrypt=False)
decrypted_data = unpad(decrypted_data)

# Запись расшифрованных данных в файл
with open("decrypted_file.txt", "wb") as decrypted_file:
    decrypted_file.write(decrypted_data)

# Вывод всего
print(f"input_data: {input_data}")
print(f"encrypted_data: {encrypted_data.hex()}")
print(f"decrypted_data: {decrypted_data.decode('utf-8')}")
print(f"MAC для input_data: {generate_mac(input_data, key).hex()}")
print(f"MAC для decrypted_data: {generate_mac(decrypted_data,
key).hex()}")

```