

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»

Выполнил

Е. А. Киселева

Проверил

Е. А. Лещенко

Минск 2024

СОДЕРЖАНИЕ

1 Постановка задачи.....	3
2 Краткие теоретические сведения.....	4
3 Результаты выполнения лабораторной работы.....	6
Выводы	7
Список использованных источников	8
Приложение А (обязательное) Листинг исходного кода	9

1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является реализация программных средств шифрования и дешифрования текстовых файлов при помощи стандарта шифрования ГОСТ 28147-89 в режиме гаммирования с обратной связью.

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.

Выделяют четыре режима работы ГОСТ 28147-89:

- простой замены;
- гаммирование;
- гаммирование с обратной связью;
- режим выработки имитовставки.

Алгоритм шифрования гаммирования с обратной связью похож на режим гаммирования, однако гамма формируется на основе предыдущего блока зашифрованных данных, так что результат шифрования текущего блока зависит также и от предыдущих блоков. По этой причине данный режим работы также называют гаммированием с зацеплением блоков.

Алгоритм шифрования следующий:

- 1 Синхропосылка заносится в регистры N1 и N2.
- 2 Содержимое регистров N1 и N2 шифруется в соответствии с алгоритмом простой замены. Полученный результат является 64-битным блоком гаммы.
- 3 Блок гаммы побитно складывается по модулю 2 с блоком открытого текста. Полученный шифротекст заносится в регистры N1 и N2.

4 Операции 2-3 выполняются для оставшихся блоков требующего шифрования текста.

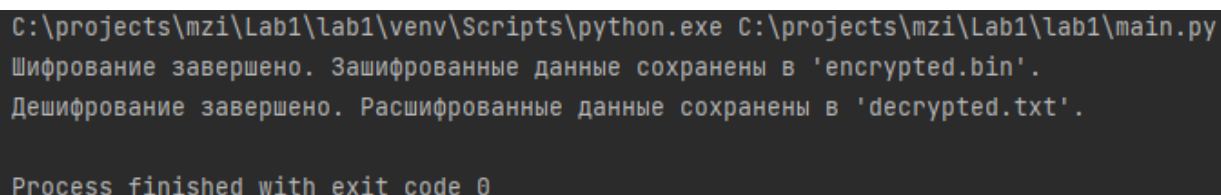
При изменении одного бита шифротекста, полученного с использованием алгоритма гаммирования с обратной связью, в соответствующем блоке расшифрованного текста меняется только один бит, так же затрагивается последующий блок открытого текста. При этом все остальные блоки остаются неизменными.

При использовании данного режима следует иметь в виду, что синхропосылку нельзя использовать повторно (например, при шифровании логически отдельных блоков информации – сетевых пакетов, секторов жёсткого диска и т. п.). Это обусловлено тем, что первый блок шифр-текста получен всего лишь сложением по модулю два с зашифрованной синхропосылкой; таким образом, знание всего лишь 8 первых байт исходного и шифрованного текста позволяют читать первые 8 байт любого другого шифр-текста после повторного использования синхропосылки.

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 гаммирования с обратной связью.

Начальный текст находится в файле input.txt. После шифрования зашифрованная информация помещается в файл encrypted.bin, после чего она снова дешифруется и помещается в файл decrypted.txt. В консоль выводится информация о завершении шифрования и завершении дешифрования, а также в какие файлы сохраняются данные. Результат работы программы представлен на рисунке 3.1.



```
C:\projects\mzi\Lab1\lab1\venv\Scripts\python.exe C:\projects\mzi\Lab1\lab1\main.py
Шифрование завершено. Зашифрованные данные сохранены в 'encrypted.bin'.
Дешифрование завершено. Расшифрованные данные сохранены в 'decrypted.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 стандарта шифрования ГОСТ 28147-89 в режиме гаммирования с обратной связью.

ВЫВОДЫ

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

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Алгоритм шифрования ГОСТ 28147-89 [Электронный ресурс]. – Режим доступа: <https://kaf401.rloc.ru/Criptfiles/gost28147/GOST28147.htm>. – Дата доступа: 03.09.2024.

[2] О шифровании ГОСТ 28147-89 [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/256843/>. – Дата доступа: 03.09.2024.

[3] Реализация алгоритма шифрования по ГОСТ 28147-89 [Электронный ресурс]. – Режим доступа: <https://www.cyberforum.ru/csharpnet/thread1109400.html>. – Дата доступа: 03.09.2024.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг исходного кода

Листинг 1 – Программный код файла main

```
import os

# Стандартные S-боксы по ГОСТ 28147-89
S_BOX = [
    [4, 10, 9, 2, 13, 8, 0, 14, 6, 11, 1, 12, 7, 15, 5, 3],
    [14, 11, 4, 12, 6, 13, 15, 10, 2, 3, 8, 1, 0, 7, 5, 9],
    [5, 8, 1, 13, 10, 3, 4, 2, 14, 15, 12, 7, 6, 0, 9, 11],
    [7, 13, 10, 1, 0, 8, 9, 15, 14, 4, 6, 12, 11, 2, 5, 3],
    [6, 12, 7, 1, 5, 15, 13, 8, 4, 10, 9, 14, 0, 3, 11, 2],
    [4, 11, 10, 0, 7, 2, 1, 13, 3, 6, 8, 5, 9, 12, 15, 14],
    [13, 11, 4, 1, 3, 15, 5, 9, 0, 10, 14, 7, 6, 8, 2, 12],
    [1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12],
]

def f(right_part, key):
    temp = (right_part + key) % (1 << 32)
    result = 0
    for i in range(8):
        s_box_input = (temp >> (4 * i)) & 0b1111
        s_box_output = S_BOX[i][s_box_input]
        result |= s_box_output << (4 * i)
    result = ((result << 11) | (result >> (32 - 11))) % (1 << 32)
    return result

def generate_subkeys(master_key):
    assert len(master_key) == 32
    subkeys = []
    for i in range(8):
        subkey = int.from_bytes(master_key[i * 4:(i + 1) * 4],
byteorder='little')
        subkeys.append(subkey)
    return subkeys

#Функция режима шифрования с простой заменой
def encrypt_block(block, subkeys):
    n1 = int.from_bytes(block[:4], byteorder='little')
    n2 = int.from_bytes(block[4:], byteorder='little')
    for i in range(32):
        key = subkeys[i % 8]
        temp = n1
        n1 = n2 ^ f(n1, key)
        n2 = temp
    encrypted_block = n1.to_bytes(4, byteorder='little') + n2.to_bytes(4,
byteorder='little')
    return encrypted_block

def gost_ofb_encrypt(plaintext, master_key, iv):
    subkeys = generate_subkeys(master_key)
    output = b''
    gamma = iv
    for i in range(0, len(plaintext), 8):
        gamma = encrypt_block(gamma, subkeys) #Используем шифрование с
простой заменой для получения гаммы
```

```

        block = plaintext[i:i + 8]
        # Дополнение блока до 8 байт
        if len(block) < 8:
            block += b'\0' * (8 - len(block))
        encrypted_block = bytes(a ^ b for a, b in zip(block, gamma))
        output += encrypted_block
    return output

def gost_ofb_decrypt(ciphertext, master_key, iv, plaintext_len):
    subkeys = generate_subkeys(master_key)
    output = b''
    gamma = iv
    for i in range(0, len(ciphertext), 8):
        gamma = encrypt_block(gamma, subkeys)
        block = ciphertext[i:i + 8]
        decrypted_block = bytes(a ^ b for a, b in zip(block, gamma))
        output += decrypted_block

    # Обрезаем лишние символы, которые были добавлены при дополнении блока
    return output[:plaintext_len]

def load_file(file_name):
    with open(file_name, 'rb') as file:
        return file.read()

def save_file(file_name, data):
    with open(file_name, 'wb') as file:
        file.write(data)

def main():
    master_key = os.urandom(32)
    iv = os.urandom(8)

    # Загрузка исходного текста
    plaintext = load_file('input.txt')
    plaintext_len = len(plaintext) # Сохраняем исходную длину текста

    # Шифрование
    ciphertext = gost_ofb_encrypt(plaintext, master_key, iv)
    save_file('encrypted.bin', ciphertext)
    print("Шифрование завершено. Зашифрованные данные сохранены в 'encrypted.bin'.")

    # Дешифрование
    decrypted_text = gost_ofb_decrypt(ciphertext, master_key, iv,
    plaintext_len)
    save_file('decrypted.txt', decrypted_text)
    print("Дешифрование завершено. Расшифрованные данные сохранены в 'decrypted.txt'.")

if __name__ == '__main__':
    main()

```