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](#_Toc157722973)

[2 Краткие теоретические сведения 4](#_Toc157722974)

[3 Результаты выполнения лабораторной работы 6](#_Toc157722975)

[Выводы](#_Toc157722976) 7

[Список использованных источников 8](#_Toc157722977)

[Приложение А (обязательное) Листинг исходного кода 9](#_Toc157722978)

# **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 раздельных 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.

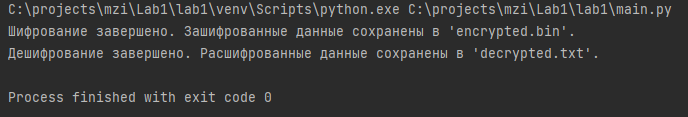


Рисунок 3.1 – Вывод консоли

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()