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 Д.С. Шевцова

Проверил Е.А. Лещенко

Минск 2024

СОДЕРЖАНИЕ

[Введение 3](#_Toc158758843)

[1 Краткие теоретические сведения 4](#_Toc158758844)

[2 Результаты выполнения лабораторной работы](#_Toc158758845) 7

[Заключение](#_Toc158758846) 8

[Приложение А](#_Toc158758848) [(обязательное)](#_Toc158758849) [Листинг кода](#_Toc158758850) 9

## ВВЕДЕНИЕ

Целью данной лабораторной работы является реализация программных средств шифрования и дешифрования текстовых файлов при помощи стандарта шифрования ГОСТ 28147-89 в режиме простой замены.

## 1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

ГОСТ 28147-89 представляет собой симметричный 64-битовый блочный алгоритм с 256-битовым ключом.

Этот алгоритм криптографического преобразования данных предназначен для аппаратной и программной реализации, удовлетворяет криптографическим требованиям и не накладывает ограничений на степень секретности защищаемой информации.

Данные, подлежащие зашифровке, разбивают на 64-разрядные блоки.

Эти блоки разбиваются на два субблока N1 и N2 по 32 бит. На рисунке 1.1 представлена схема одного раунда замены.

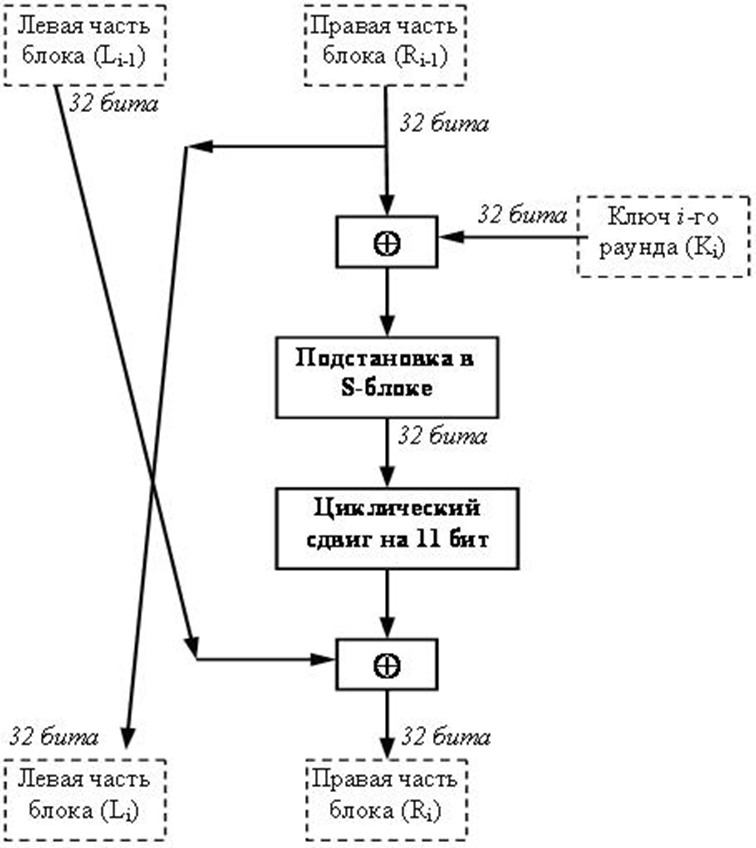


Рисунок 1.1 – Структура одного раунда ГОСТ 28147-89

На рисунке 1.2 представлена схема работы простого шифра замены.

Шифруемый блок данных разбивается на две части, которые затем обрабатываются как отдельные 32-битовые целые числа без знака. Сначала правая половина блока и подключ раунда складываются по модулю 2^32. Затем производится поблочная подстановка.

32-битовое значение, полученное на предыдущем шаге (обозначим его S), интерпретируется как массив из восьми 4-битовых блоков кода: S=(S0,S1,S2,S3,S4,S5,S6,S7).

Далее значение каждого из восьми блоков заменяется на новое, которое выбирается по таблице замен, представленной на рисунке 1.3.

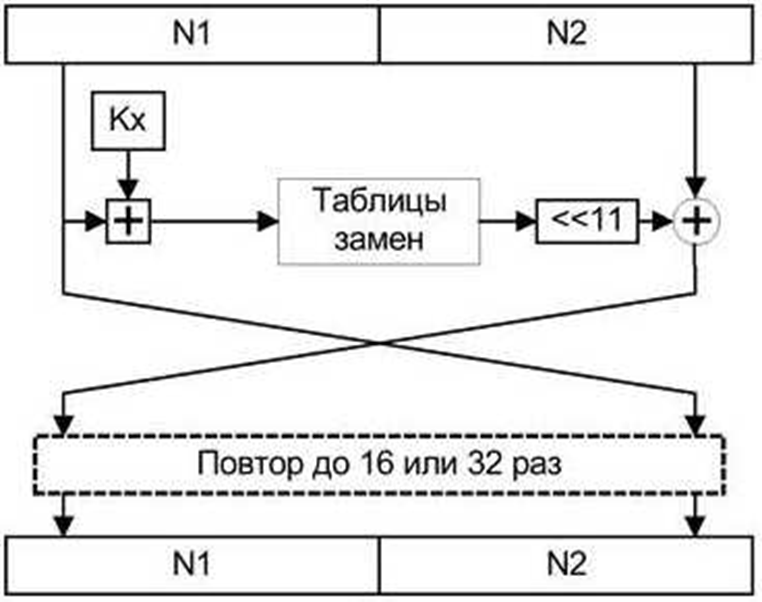


Рисунок 1.2 – Схема простого шифра замены ГОСТ 28147-89

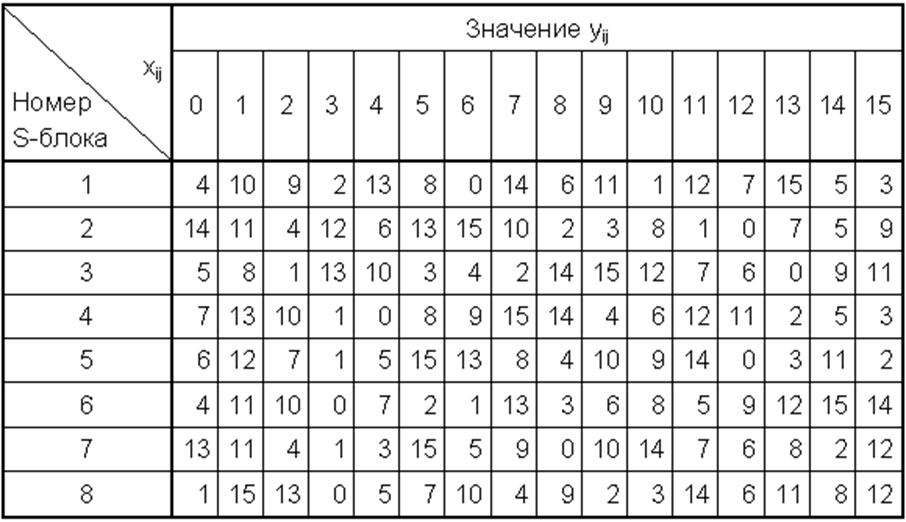


Рисунок 1.3 – Таблица замен ГОСТ 28147-89

В каждой строке таблицы замен записаны числа от 0 до 15 в произвольном порядке без повторений.

Значения элементов таблицы замен взяты от 0 до 15, так как в четырех битах, которые подвергаются подстановке, может быть записано целое число без знака в диапазоне от 0 до 15.

Значение блока S1 (четыре младших бита 32-разрядного числа S) заменится на число, стоящее на позиции, номер которой равен значению заменяемого блока.

Например, в этом случае S1=0 заменится на 4, если S1=1, то оно заменится на 10 и т.д.

После выполнения подстановки все 4-битовые блоки снова объединяются в единое 32-битное слово, которое затем циклически сдвигается на 11 битов влево. Наконец, с помощью побитовой операции "сумма по модулю 2" результат объединяется с левой половиной, вследствие чего получается новая правая половина Ri. Новая левая часть Li берется равной младшей части преобразуемого блока: Li= Ri-1. Полученное значение преобразуемого блока рассматривается как результат выполнения одного раунда алгоритма шифрования.

ГОСТ 28147-89 является блочным шифром, поэтому преобразование данных осуществляется блоками в так называемых базовых циклах. Базовые циклы заключаются в многократном выполнении для блока данных основного раунда, рассмотренного нами ранее, с использованием разных элементов ключа и отличаются друг от друга порядком использования ключевых элементов. Мною выполнялся режим простой замены. А именно все блоки шифруются независимо друг от друга с разными подключами в разных раундах. Для одинаковых блоков сообщения М блоки шифр текста будут одинаковыми.

# 2 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

На вход подается текстовый файл с содержимым для шифрации и для последующей расшифрации. На рисунке 2.1 представлен файл с текстом для шифрации, на рисунке 2.2 зашифрованный текст, а на рисунке 3.3 результат расшифрации.

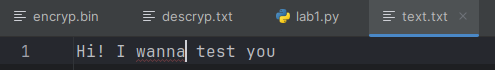


Рисунок 2.1 – Исходный текст

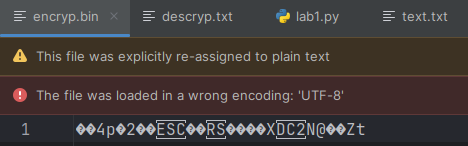


Рисунок 2.2 – Зашифрованный текст

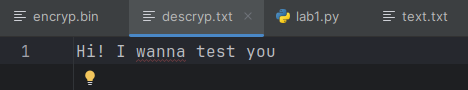


Рисунок 2.3 – Расшифрованный текст

## ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы были реализованы программные средства шифрования и дешифрования текстовых файлов при помощи стандарта шифрования ГОСТ 28147-89 режиме простой замены. Также были изучены теоретические сведения в данной области.

# **ПРИЛОЖЕНИЕ А**

## (обязательное)

## Листинг кода

lab1.py

import os

# Определяем размер файла

def filesize(filename):

return os.path.getsize(filename)

# Добавление padding (PKCS7) для заполнения до 8 байт

def add\_padding(data):

pad\_len = 8 - len(data) % 8

return data + bytes([pad\_len] \* pad\_len)

# Удаление padding (PKCS7) после расшифрования

def remove\_padding(data):

pad\_len = data[-1]

if 0 < pad\_len <= 8:

return data[:-pad\_len]

return data # Если что-то пошло не так, возвращаем данные как есть

# Функция, реализующая работу ГОСТ 28147-89 в режиме простой замены

def rpz(rezh, opener, saver):

# Таблица замен (8x16)

Tab\_Z = [

[0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xE, 0xF],

[0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xE, 0xF],

[0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xE, 0xF],

[0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xE, 0xF],

[0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xE, 0xF],

[0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xE, 0xF],

[0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xE, 0xF],

[0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xE, 0xF]

]

# Ключ

key = [

0x0123,

0x4567,

0x89AB,

0xCDEF,

0x0123,

0x4567,

0x89AB,

0xCDEF

]

N = bytearray(4) # 32-разрядный накопитель

n1 = n2 = SUM232 = 0 # Накопители N1, N2, и сумматор

with open(opener, "rb") as f\_begin, open(saver, "wb") as f\_end:

data = f\_begin.read()

# Добавляем padding при шифровании

if rezh == 1:

data = add\_padding(data)

block\_count = (len(data) \* 8 + 63) // 64 # Определим количество блоков

sh = min(len(data), 4)

sh1 = 0

flag = 0

# Чтение и преобразование блоков

for i in range(block\_count):

N[:] = bytearray(4) # Записываем в накопитель N1

if (sh1 + sh) < len(data):

N[:] = data[sh1:sh1 + sh]

sh1 += sh

else:

sh = len(data) - sh1

N[:] = data[sh1:sh1 + sh]

flag = 1

n1 = int.from\_bytes(N, 'little')

N[:] = bytearray(4) # Записываем в накопитель N2

if (sh1 + sh) < len(data):

N[:] = data[sh1:sh1 + sh]

sh1 += sh

else:

if not flag:

sh = len(data) - sh1

N[:] = data[sh1:sh1 + sh]

n2 = int.from\_bytes(N, 'little')

# 32 цикла простой замены

c = 0

for k in range(32):

if rezh == 1 and k == 24:

c = 7

elif rezh == 2 and k == 8:

c = 7

# Суммируем в сумматоре СМ1

SUM232 = (key[c] + n1) & 0xFFFFFFFF

# Заменяем по таблице замен

for q in range(4):

zam\_symbol = (SUM232 >> (q \* 8)) & 0xFF

first\_byte = (zam\_symbol & 0xF0) >> 4

second\_byte = zam\_symbol & 0x0F

first\_byte = Tab\_Z[2 \* q][first\_byte]

second\_byte = Tab\_Z[2 \* q + 1][second\_byte]

zam\_symbol = (first\_byte << 4) | second\_byte

SUM232 = (SUM232 & ~(0xFF << (q \* 8))) | (zam\_symbol << (q \* 8))

SUM232 = ((SUM232 << 11) & 0xFFFFFFFF) | (SUM232 >> 21) # Циклический сдвиг на 11

SUM232 ^= n2 # Складываем в сумматоре СМ2

if k < 31:

n2 = n1

n1 = SUM232

if rezh == 1:

if k < 24:

c = (c + 1) % 8

else:

c = (c - 1) % 8

else:

if k < 8:

c = (c + 1) % 8

else:

c = (c - 1) % 8

n2 = SUM232

# Вывод результата в файл

f\_end.write(n1.to\_bytes(4, 'little'))

f\_end.write(n2.to\_bytes(4, 'little'))

# Удаляем padding при дешифровании

if rezh == 2:

with open(saver, "rb") as f\_end:

decrypted\_data = f\_end.read()

decrypted\_data = remove\_padding(decrypted\_data)

with open(saver, "wb") as f\_end:

f\_end.write(decrypted\_data)

# Основная программа

def main():

# Шифруем файл text.txt в encryp.bin

rpz(1, 'text.txt', 'encryp.bin')

# Дешифруем файл encryp.bin в descryp.txt

rpz(2, 'encryp.bin', 'descryp.txt')

if \_\_name\_\_ == "\_\_main\_\_":

main()