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 № 2

на тему «Симметричная криптография. СТБ 34.101.31-2011»

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 4](#_Toc157722975)

[Выводы](#_Toc157722976) 7

[Список использованных источников 7](#_Toc157722977)

[Приложение А (обязательное) Листинг исходного кода 8](#_Toc157722978)

# **1 ПОСТАНОВКА ЗАДАЧИ**

Целью выполнения данной лабораторной работы является реализация программных средств шифрования и дешифрования текстовых файлов при помощи СТБ 34.101.31-2011 в режиме гаммирования с обратной связью.

# **2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

СТБ 34.101.31-2011 «Криптографические алгоритмы шифрования и контроля целостности» - государственный стандарт Республики Беларусь, 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.

Выделяют четыре режима работы СТБ 34.101.31-2011:

– простой замены;

– сцепление блоков;

– гаммирование с обратной связью;

– режим счетчика.

В алгоритме шифрования гаммирование с обратной связью исходное сообщение разбивается на блоки, которые обрабатываются по одному. Шифрование и расшифрование производятся так, что каждый зашифрованный блок зависит от предыдущих блоков. Гамма формируется на основе предыдущего блока зашифрованных данных, так что результат шифрования текущего блока зависит также и от предыдущих блоков.

Шифрование начинается с использования синхропосылки S, которая подаётся на вход шифрующей функции, чтобы получить первый блок гаммы. Этот блок комбинируется с первым блоком исходного текста для получения первого зашифрованного блока. Затем каждый последующий блок зашифрованного текста подаётся на вход шифрующей функции, и результат снова используется для гаммирования с новым блоком исходного текста. Обратная связь между блоками усиливает криптостойкость.

Алгоритм шифрования следующий:

1 Установить 𝑌0 ← 𝑆.

2 Для 𝑖 = 1, 2, … , 𝑛 выполнить: 𝑌𝑖 ← 𝑋𝑖 ⊕ 𝐿|𝑋𝑖|(𝐹𝜃(𝑌𝑖−1)).

3 Установить 𝑌 ← 𝑌1 ‖ 𝑌2 ‖ . . . ‖ 𝑌𝑛.

4 Возвратить 𝑌 .

Расшифрование в режиме гаммирования с обратной связью происходит аналогично процессу шифрования, за исключением того, что шифротекст используется для восстановления исходных данных. Начальным значением также служит синхропосылка S, и дальнейшие шаги включают те же операции, что и при шифровании, но с использованием полученного шифротекста.

Режим гаммирования с обратной связью по стандарту СТБ 34.101.31-2011 обеспечивает высокий уровень криптостойкости при шифровании потоков данных и сообщений. Это делает его удобным для применения в ситуациях, где данные передаются небольшими блоками, а также для работы с файлами произвольной длины.

# **3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ             РАБОТЫ**

В ходе выполнения лабораторной было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи СТБ 34.101.31-2011 в режиме гаммирования с обратной связью.

Начальный текст находится в файле input.txt. После шифрования зашифрованная информация помещается в файл encrypted.txt. После дешифрования из файла encrypted.txt информация помещается в файл decrypted.txt. Пользователь может выбрать операцию, которую будет выполнять программа. 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.

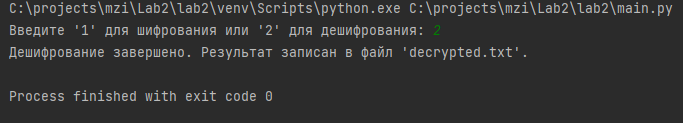


Рисунок 3.1 – Вывод консоли

Таким образом, в ходе данной лабораторной работы было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи СТБ 34.101.31-2011 в режиме гаммирования с обратной связью.

# **ВЫВОДЫ**

В ходе данной лабораторной работы было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи СТБ 34.101.31-2011 в режиме гаммирования с обратной связью.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Алгоритм шифрования СТБ 34.101.31-2011 [Электронный ресурс]. – Режим доступа: https://apmi.bsu.by/assets/files/std/belt-spec27.pdf. – Дата доступа: 07.09.2024.

[2] О шифровании СТБ 34.101.31-2011 [Электронный ресурс]. – Режим доступа: https://apmi.bsu.by/resources/std. – Дата доступа: 07.09.2024.

## **ПРИЛОЖЕНИЕ А**

## **(обязательное)**

## **Листинг исходного кода**

Листинг 1 – Программный код файла main

import os

# выполняет преобразование на основе предыдущего зашифрованного блока - XOR с ключом

def apply\_F\_theta(prev\_Y, key):

return prev\_Y ^ int(key, 2)

# выборка первых Xi\_length бит из Z

def mask\_L\_Xi(value, bit\_length):

mask = (1 << bit\_length) - 1 # Генерация маски для битовой длины, гарантирует, что количество бит соответствует длине текущего блока текста.

return value & mask

def encryption\_algorithm(input\_path, output\_path, key, sync\_pos):

if not os.path.isfile(input\_path):

print(f"Файл '{input\_path}' не найден. Операция шифрования невозможна.")

return

# Чтение входного файла как строки и преобразование в список кодов символов (ASCII)

with open(input\_path, 'r', encoding='utf-8') as infile:

plaintext = infile.read()

blocks = [ord(char) for char in plaintext] # Символы переводим в их ASCII-коды

# Начальное значение Y0 (синхропосылка)

prev\_Y = int(sync\_pos, 2)

encrypted = []

block\_length = max(len(bin(block)) - 2 for block in blocks) # Определяем длину блока (в битах)

for block in blocks:

# Преобразование через F\_theta

gamma = apply\_F\_theta(prev\_Y, key)

# Применение маски к блоку

masked\_gamma = mask\_L\_Xi(gamma, block\_length)

# XOR между текущим блоком и гаммой

encrypted\_block = block ^ masked\_gamma

encrypted.append(encrypted\_block)

# Обновление предыдущего зашифрованного блока для следующего шага

prev\_Y = encrypted\_block

# Записываем зашифрованные значения в файл, преобразуя обратно в символы

with open(output\_path, 'w', encoding='utf-8') as outfile:

outfile.write(''.join(chr(b) for b in encrypted))

print(f"Шифрование завершено. Результат записан в файл '{output\_path}'.")

def decryption\_algorithm(input\_path, output\_path, key, sync\_pos):

if not os.path.isfile(input\_path):

print(f"Файл '{input\_path}' не найден. Операция дешифрования невозможна.")

return

# Чтение входного файла и преобразование символов обратно в ASCII-коды

with open(input\_path, 'r', encoding='utf-8') as infile:

encrypted = infile.read()

blocks = [ord(char) for char in encrypted]

# Начальное значение Y0 (синхропосылка)

prev\_Y = int(sync\_pos, 2)

decrypted = []

block\_length = max(len(bin(block)) - 2 for block in blocks) # Определяем длину блока

for block in blocks:

# Преобразование через F\_theta

gamma = apply\_F\_theta(prev\_Y, key)

# Применение маски

masked\_gamma = mask\_L\_Xi(gamma, block\_length)

# XOR между зашифрованным блоком и гаммой

decrypted\_block = block ^ masked\_gamma

decrypted.append(decrypted\_block)

# Обновление предыдущего блока

prev\_Y = block

# Записываем расшифрованные данные в файл

with open(output\_path, 'w', encoding='utf-8') as outfile:

outfile.write(''.join(chr(b) for b in decrypted))

print(f"Дешифрование завершено. Результат записан в файл '{output\_path}'.")

def main():

operation = input("Введите '1' для шифрования или '2' для дешифрования: ").strip()

input\_filename = 'input.txt'

encrypted\_filename = 'encrypted.txt'

decrypted\_filename = 'decrypted.txt'

key = '101010'

sync\_pos = '11111111'

if operation == '1':

encryption\_algorithm(input\_filename, encrypted\_filename, key, sync\_pos)

elif operation == '2':

decryption\_algorithm(encrypted\_filename, decrypted\_filename, key, sync\_pos)

else:

print("Неверный выбор. Введите '1' для шифрования или '2' для дешифрования.")

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

main()