

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
2 Краткие теоретические сведения.....	4
3 Результаты выполнения лабораторной работы.....	5
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 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. Схема вычисления на  $i$ -ом такте зашифрования представлена на рисунке 2.1.

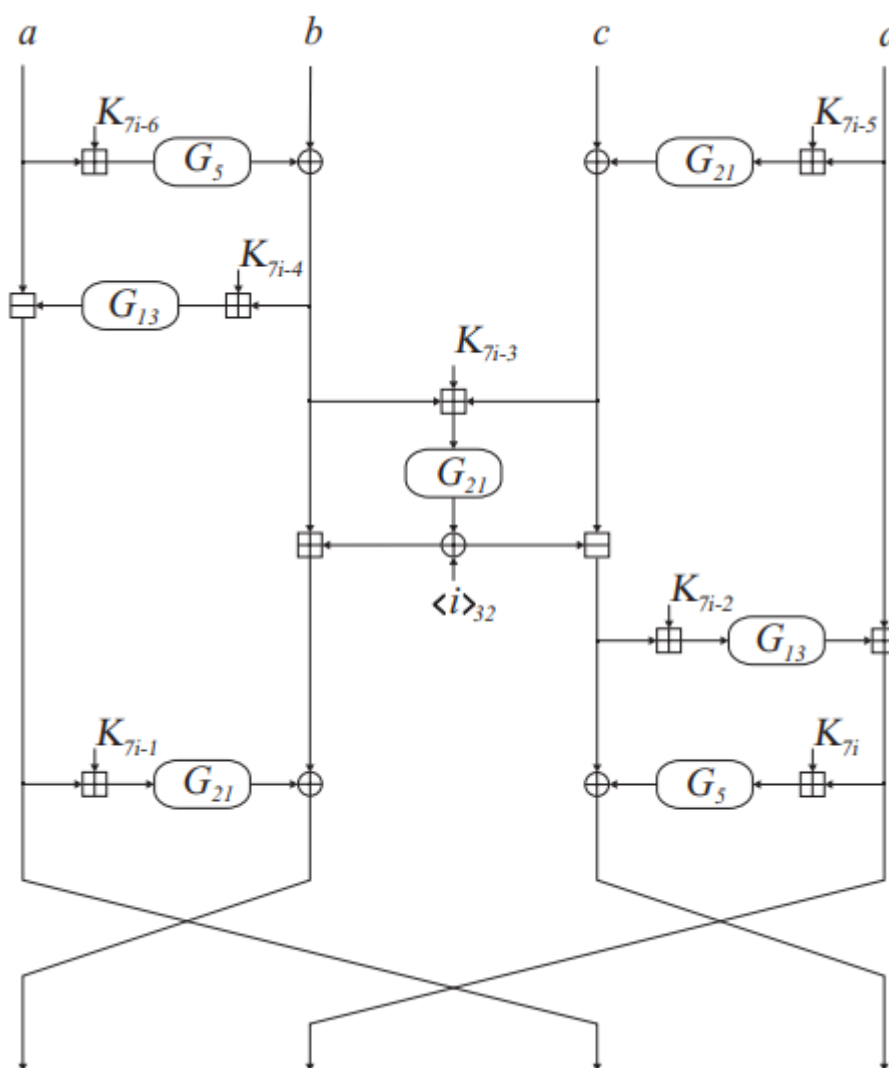


Рисунок 2.1 – Вычисления на  $i$ -ом такте зашифрования

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  $Y_0 \leftarrow S$ .
- 2 Для  $i = 1, 2, \dots, n$  выполнить:  $Y_i \leftarrow X_i \oplus L_{|X_i|}(F\theta(Y_{i-1}))$ .
- 3 Установить  $Y \leftarrow Y_1 \parallel Y_2 \parallel \dots \parallel Y_n$ .
- 4 Возвратить  $Y$ .

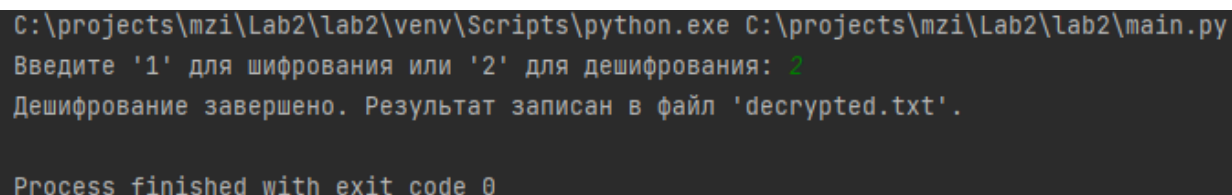
Расшифрование в режиме гаммирования с обратной связью происходит аналогично процессу шифрования, за исключением того, что шифротекст используется для восстановления исходных данных. Начальным значением также служит синхропосылка  $S$ , и дальнейшие шаги включают те же операции, что и при шифровании,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



```
C:\projects\mzi\Lab2\lab2\venv\Scripts\python.exe C:\projects\mzi\Lab2\lab2\main.py
Введите '1' для шифрования или '2' для дешифрования: 2
Дешифрование завершено. Результат записан в файл '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 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()

```