ЛАБОРАТОРНАЯ РАБОТА №3

“Поточное симметричное шифрование”

по дисциплине

‘Информационная безопасность’

Вариант 5

Выполнил:
Соболев Иван
Александрович
Группа: P34312

Преподаватель:
Маркина Татьяна
Анатольевна

Санкт-Петербург, 2024

Цель работы

Изучение структуры и основных принципов работы современных алгоритмов поточного симметричного шифрования, приобретение навыков программной реализации поточных симметричных шифров.

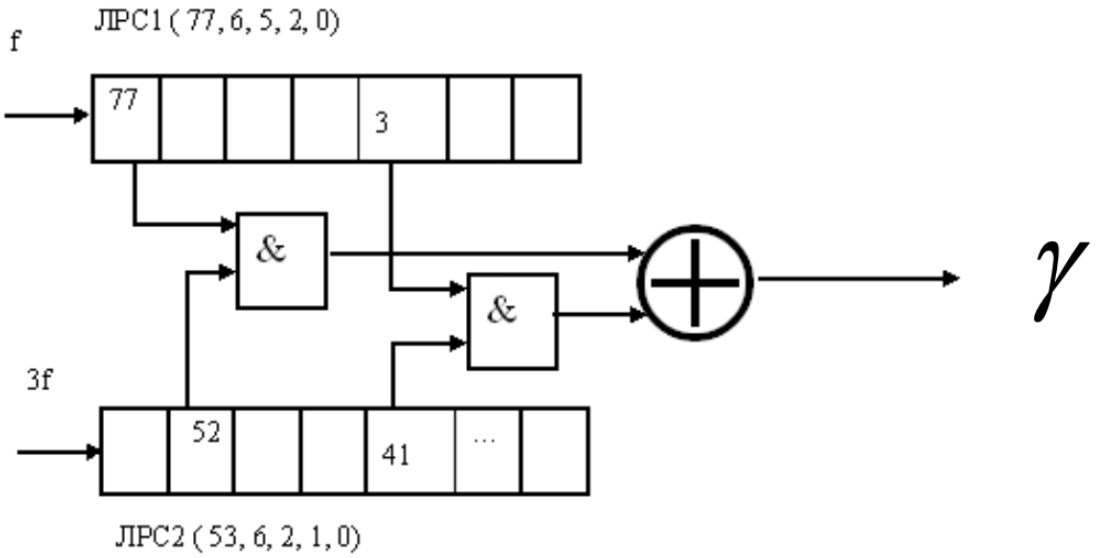
Программные и аппаратные средства

Для выполнения лабораторной работы был использован компьютер со следующими характеристиками:

- Процессор: Apple M2
- Видеокарта: Apple M2
- Объем оперативной памяти: 8GB
- Использована операционная система: macOS 14.4.1
- Версия Python: 3.13

Задание

Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной нелинейной схемы, использующей разные частоты тактирования ЛРС.



Листинг разработанной программы

main.py

```
from encoder import encode
from io_utils import read_from_file, read_filename, write_to_file, print_red


def main():
    try:
        input_file = read_filename(True)
        output_file = read_filename(False)
        if input_file is not None and output_file is not None:
            data = read_from_file(input_file)
            encoded = encode(data)
            write_to_file(output_file, encoded)
```

```
except Exception as e:
      print_red(f'Ошибка чтения из файла: {str(e)}')

if __name__ == "__main__":
   main()
```

encoder.py

```
from LFSR import LFSR

INIT_STATE_LFSR1 =
'10010111111001000011001100100011100111000111101010110110110010110000011011111 1'
INIT_STATE_LFSR2 = '10000011111100011110000100101011100101000100011011 0011'

seed1 = [int(bit) for bit in INIT_STATE_LFSR1]
seed2 = [int(bit) for bit in INIT_STATE_LFSR2]
lfsr1 = LFSR(seed1, [75, 6, 5, 2, 0])
lfsr2 = LFSR(seed2, [53, 6, 2, 1, 0])


def nonlinear_schema(i):
   and_result = lfsr1.data[77] & lfsr2.data[52]
   and_result_2 = lfsr1.data[3] & lfsr2.data[41]
   lfsr1.shift_right()  # Обновляем LFSR1 каждый бит
   if i % 3 == 0:
      lfsr2.shift_right()  # Обновляем LFSR2 каждый третий бит
   return and_result ^ and_result_2


def encode(text: bytes) -> bytes:
   encoded = []
   i = 0
   for byte in text:
      binary_string = format(byte, '08b')
      for bit in binary_string:
         gamma = nonlinear_schema(i)
         encoded.append(int(bit) ^ gamma)
         i += 1
   transformed_text_bytes = [
      int(''.join(map(str, encoded[i:i + 8])), 2) for i
      in range(0, len(encoded), 8)
   ]
   return bytes(transformed_text_bytes)
```

io_utils.py

```
import os
from typing import Optional


def print_red(message: str) -> None:
    """Функция для вывода ошибок"""
    print(f"\033[91m{message}\033[0m")


def print_green(message: str) -> None:
    """Функция для вывода ответов сервиса"""
    print(f"\033[92m{message}\033[0m")


def read_filename(is_input) -> str:
    """Функция чтения имени файла"""
    if is_input:
        filename = input("Введите названия входного файла: ")
    else:
        filename = input("Введите названия выходного файла: ")
    if not filename.strip():
        print_red("Ошибка: имя файла не должно быть пустым.")
        return None
    return filename


def read_from_file(filename) -> Optional[bytes]:
    """Функция чтения из файла"""
    try:
        if not os.path.isfile(filename):
            print_red(f"Ошибка: файл '{filename}' не найден.")
            return None
        with open(filename, "rb") as f:
            buf = f.read()
            if not buf.strip():
                print_red("Ошибка: файл пуст.")
                return None
            return buf
    except Exception as e:
        print_red(f'Ошибка чтения из файла: {str(e)}')
        return None


def write_to_file(filename, content):
    """Функция записи в файл"""
    try:
        with open(filename, 'wb') as file:
            file.write(content)
```

```
print_green(f"Данные успешно записаны в файл: {filename}")
except IOError as e:
  print_red(f"Ошибка записи в файл {filename}: {e}")
except Exception as e:
  print_red(f"Произошла ошибка: {e}")
```

LFSR.py

```
class LFSR:
  def __init__(self, seed: list[int], taps: list[int]):
    self.taps = taps
    self.data = seed

  def shift_right(self):
    new_bit = 0
    for tap in self.taps:
      new_bit ^= self.data[tap]
    for i in range(len(self.data) - 1):
      self.data[i] = self.data[i + 1]
    self.data[-1] = new_bit
```

Результаты работы программы

Исходный текст:
В одной из отдаленных улиц Москвы, в сером доме с белыми колоннами, антресолью и покривившимся балконом, жила некогда барыня, вдова, окруженная многочисленною дворней.

Шифрование текста:

```
/Users/isobolev/Desktop/itmo/itmo-4course/information_sec/1_3/.venv/bin/python /Users/isobolev/Desktop/itmo/itmo-4course/information_sec/1_3/main.py
Введите названия входного файла: input
Введите названия выходного файла: encoded.bin
Данные успешно записаны в файл: encoded.bin
```

Результат шифрования:

```
ă1◆@◆◆r"Ÿd◆◆◆ENQI"◆◆Z◆◆◆\↵gU◆]◆=◆
◆5(TbG◆◆◆◆◆7◆`N◆DLE◆q3FFEOT))◆2◆<◆◆%V◆ETB◆
◆◆◆◆V◆◆5◆S ◆Tg◆b◆◆VTR◆1=!◆y2◆◆◆ŗ◆DC3◆I◆8◆◆NAK◆|◆◆',◆ÿGSc◆◆p◆◆◆!◆◆◆S.dSI◆sRSAMDC1@◆DC11Z◆&n◆↵◆R◆◆EMr◆NAK◆@◆◆◆◆◆?]◆ ◆◆SOH◆◆kACK◆◆]-[ESC◆
◆3◆◆◆>◆◆◆[◆◆DC1
◆◆◆◆◆◆◆r>
```

Дешифрование текста:

```
/Users/isobolev/Desktop/itmo/itmo-4course/information_sec/1_3/.venv/bin/python /Users/isobolev/Desktop/itmo/itmo-4course/information_sec/1_3/main.py
Введите названия входного файла: encoded.bin
Введите названия выходного файла: output
Данные успешно записаны в файл: output
```

Дешифрованный текст:
В одной из отдаленных улиц Москвы, в сером доме с белыми колоннами, антресолью и покривившимся балконом, жила некогда барыня, вдова, окруженная многочисленною дворней.