Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Безопасность и защита информации»

Профиль: «Компьютерные системы»

Семестр 6

ОТЧЕТ

по лабораторной работе №4

Тема: «Блочное шифрование XOR»

Выполнил: студент группы КС-21-1б

Чирков А.В. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Шереметьев В. Г. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

**ЦЕЛЬ РАБОТЫ**

Получить практические навыки по использованию метода блочного шифрования XOR, используя блоки длиной 32 бита, ключ длиной 16 бит.

**ЗАДАНИЕ**

Реализовать шифрование бинарного файла, используя метод блочного шифрования.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Шифрование методом XOR (исключающее ИЛИ) является одним из простейших алгоритмов шифрования. Он основан на логической операции XOR, которая работает следующим образом:

Если оба бита одинаковые, результат операции XOR равен 0.

Если оба бита разные, результат операции XOR равен 1.

Пример алгоритма шифрования XOR для текстового сообщения:

Исходное сообщение: "HELLO"

Ключ шифрования: "10101"

Преобразуем каждый символ в сообщении и ключе в бинарное представление:

"H" -> 01001000

"E" -> 01000101

"L" -> 01001100

"O" -> 01001111

Применяем операцию XOR к каждому байту исходного сообщения и соответствующему байту ключа:

"H" XOR "00010101" -> 01001000 XOR 00010101 = 1011101

"E" XOR "00010101" -> 01000101 XOR 00010101 = 1010000

"L" XOR "00010101" -> 01001100 XOR 00010101 = 1011001

"L" XOR "00010101" -> 01001100 XOR 00010101 = 1011001

"O" XOR "00010101" -> 01001111 XOR 00010101 = 1011010

Зашифрованное сообщение: "1011010 1010000 1011001 1011001 1011010"

Блочное шифрование - это метод шифрования, при котором данные разбиваются на фиксированные блоки определенного размера, которые затем обрабатываются шифровальным алгоритмом по отдельности. Каждый блок данных обрабатывается независимо друг от друга.

Алгоритм блочного шифрования XOR:

1. Выбирается ключ (в нашем случае до 16 бит)

2. Файл разбивается на блоки длиной 32 бита (в нашем случае)

3. Применяем операцию XOR к каждому байту исходного сообщения и соответствующему байту ключа

4. Записываем зашифрованное сообщение в файл

Дешифрование аналогично шифрованию - применяем операцию XOR к каждому байту зашифрованного сообщения и соответствующему байту ключа.

**ХОД РАБОТЫ**

При запуске программы, на экране появляется окно с полем ввода ключа и кнопками шифрования (Encrypt File) и дешифрования (Decrypt File) файла (рисунок 1).

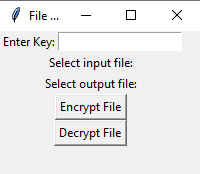


Рисунок 1 – Окно программы при запуске.

Вводим ключ. Пример работы программы представлен на рисунке 2.

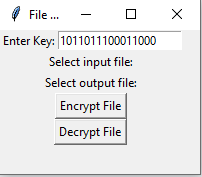


Рисунок 2 – Пример работы программы.

Нажимаем кнопку шифрования и выбираем файлы (рисунок 3).

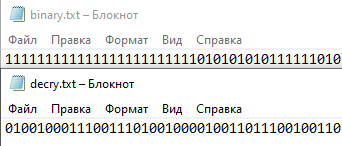


Рисунок 3 - Исходный и зашифрованный файлы

Нажимаем кнопку Decrypt и выбираем файлы для дешифрования (рисунок 4).

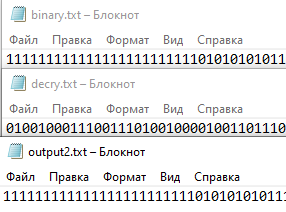


Рисунок 4 - Дешифрование файла

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

import tkinter as tk

from tkinter import filedialog

from functools import partial

def encrypt\_block(block, key):

while len(key) < len(block):

key += key

block\_int = int(block, 2)

key\_int = int(key[:len(block)], 2)

encrypted\_block\_int = block\_int ^ key\_int

encrypted\_block = bin(encrypted\_block\_int)[2:].zfill(len(block))

return encrypted\_block

def decrypt\_block(block, key):

while len(key) < len(block):

key += key

decrypted\_block\_int = int(block, 2) ^ int(key[:len(block)], 2)

decrypted\_block = bin(decrypted\_block\_int)[2:].zfill(len(block))

return decrypted\_block

def encrypt\_file(input\_file, output\_file, key):

with open(input\_file, 'r') as f\_in, open(output\_file, 'w') as f\_out:

while True:

block = f\_in.read(32)

if not block:

break

block += '0' \* (32 - len(block))

encrypted\_block = encrypt\_block(block, key)

f\_out.write(encrypted\_block)

print("Encryption completed.")

def decrypt\_file(input\_file, output\_file, key):

with open(input\_file, 'r') as f\_in, open(output\_file, 'w') as f\_out:

while True:

block = f\_in.read(32)

if not block:

break

decrypted\_block = decrypt\_block(block, key)

f\_out.write(decrypted\_block)

print("Decryption completed.")

def browse\_file(label):

filename = filedialog.askopenfilename()

label.config(text="Selected file: " + filename)

return filename

def encrypt\_selected\_file():

key = generate\_button\_entry.get().encode()

input\_file = browse\_file(input\_label)

output\_file = browse\_file(output\_label)

encrypt\_file(input\_file, output\_file, key)

def decrypt\_selected\_file():

key = generate\_button\_entry.get().encode()

input\_file = browse\_file(input\_label)

output\_file = browse\_file(output\_label)

decrypt\_file(input\_file, output\_file, key)

root = tk.Tk()

root.title("File Encryption")

generate\_button = tk.Label(root, text="Enter Key:")

generate\_button.grid(row=1, column=0)

generate\_button\_entry = tk.Entry(root)

generate\_button\_entry.grid(row=1, column=1)

input\_label = tk.Label(root, text="Select input file:")

input\_label.grid(row=5, column=0, columnspan=2)

output\_label = tk.Label(root, text="Select output file:")

output\_label.grid(row=6, column=0, columnspan=2)

encrypt\_button = tk.Button(root, text="Encrypt File", command=encrypt\_selected\_file)

encrypt\_button.grid(row=7, column=0, columnspan=2)

decrypt\_button = tk.Button(root, text="Decrypt File", command=decrypt\_selected\_file)

decrypt\_button.grid(row=8, column=0, columnspan=2)

root.mainloop()