Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Дисциплина «Высокопроизводительные вычисления»

**Лабораторная работа № 3**

**Вариант 10**

**Реализация генератора псевдослучайных чисел**

Выполнил:

Студент группы ИВТАСбд-42

Сулейманов М.З.

Проверил:

Мартынов А.И.

Ульяновск

2024

**Цель работы**

Научиться реализовывать потоковый алгоритм шифрования с использованием генераторов псевдослучайных чисел, а также получить опыт реализации собственных функций хеширования текстовых паролей.

**Задание**

Реализовать приложение с графическим интерфейсом, позволяющее выполнять следующие действия:

1. Шифровать и дешифровать текстовые и двоичные файлы с помощью потокового шифрования и генератора псевдослучайных чисел, разработанного в предыдущей лабораторной работе.

2. Сохранять зашифрованные/дешифрованные данные в файл

3. Хешировать текстовый пароль, который используется при шифровании для инициализации генератора псевдослучайных чисел с помощью функции хеширования, указанной в варианте.

**Хэширование пароля**

Для хэширования пароля была использован метод SHA256 из библиотеки hashlib для python. Оно же использовалось в качестве начального приближения для генерации маски. Далее маска побитово сравнивается с значениями исходного текста, при шифрации эта операция изменит исходный текст на зашифрованный и наоборот при дешифрации.

**Листинг**

|  |
| --- |
| import tkinter as tk  from tkinter import filedialog, messagebox  import hashlib  sequence = []  def generate\_mask(X0, a, m, length):      global sequence      sequence = [X0]      for i in range(1, length):          Xn = (a \* sequence[-1]) % m          sequence.append(Xn)      mask = bytearray([bit % 256 for bit in sequence])        return mask    def hash\_password\_print(password):      return hashlib.sha1(password.encode()).hexdigest()  def encrypt\_file(filename, mask, password):      hashed\_password = int(hash\_password\_print(password),16)      print(hashed\_password)      mode = 'rb'      output\_mode = 'wb'      with open(filename, mode) as input\_file, open(f"{filename}.out", output\_mode) as output\_file:          while chunk := input\_file.read(1024):              transformed\_chunk = bytearray()              for i in range(len(chunk)):                  transformed\_chunk.append(chunk[i] ^ mask[i % len(mask)] ^ (hashed\_password & 0xFF))                  hashed\_password = (hashed\_password + 1) % 40  # обновление хэша              output\_file.write(transformed\_chunk)      return f"{filename}.out"  def decrypt\_file(filename, mask, password):      hashed\_password = int(hash\_password\_print(password),16)      mode = 'rb'      output\_mode = 'wb'      with open(filename, mode) as input\_file, open(f"{filename}.decrypted", output\_mode) as output\_file:          while chunk := input\_file.read(1024):              transformed\_chunk = bytearray()              for i in range(len(chunk)):                  transformed\_chunk.append(chunk[i] ^ mask[i % len(mask)] ^ (hashed\_password & 0xFF))                  hashed\_password = (hashed\_password + 1) % 256  # обновление хэша              output\_file.write(transformed\_chunk)      return f"{filename}.decrypted"  def process\_file(encrypt=True):      try:            a = 16807          m = 2147483647          length = 100000          password = entry\_password.get()          if not password:              messagebox.showwarning("Ошибка", "Поле пароля не должно быть пустым.")              return          filename = filedialog.askopenfilename(title="Выберите файл для шифрования/дешифрования.")          if not filename:              messagebox.showwarning("Ошибка", "Выберите файл.")              return          X0 = int(hash\_password\_print(password),16)          # генерация маски          mask = generate\_mask(X0, a, m, length)          if encrypt:                output\_file = encrypt\_file(filename, mask, password)              messagebox.showinfo("Успех", f"Файл зашифрован: {output\_file}")              hashed\_password = hash\_password\_print(password)              messagebox.showinfo("Пароль захэширован",                                  f"Хэш вашего пароля: {hashed\_password}")          else:                output\_file = decrypt\_file(filename, mask, password)              messagebox.showinfo("Успех", f"Файл дешифрован: {output\_file}")      except ValueError:          messagebox.showerror("Ошибка", "Пожалуйста, проверьте введённые данные.")    root = tk.Tk()  root.title("Шифрование/дешифрование")  frame = tk.Frame(root)  frame.pack(padx=10, pady=10)  root.geometry("400x400")  label\_password = tk.Label(frame, text="Введите пароль:")  label\_password.pack()  entry\_password = tk.Entry(frame, show="\*")  entry\_password.pack()  button\_encrypt = tk.Button(frame, text="Шифровать", command=lambda: process\_file(encrypt=True))  button\_encrypt.pack(pady=5)  button\_decrypt = tk.Button(frame, text="Дешифровать", command=lambda: process\_file(encrypt=False))  button\_decrypt.pack(pady=5)  root.mainloop() |