министерство науки и высшего образования российской федерации

федеральное государственное автономное государственное учреждение высшего образования

«санкт-петербургский государственный университет аэрокосмического приборостроения»

кафедра № 42

отчет

защищен с оценкой

преподаватель

Старший преподаватель С.Ю. Гуков

должность, уч.степень подпись, дата Инициалы, фамилия

отчет о лабораторной работе № 9

Асимметричный алгоритм шифрования RSA

По курсу: Алгоритмы и структуры данных

работу выполнил

Студент гр. № 4329 27.12.2024 Д.М.Онопричук\_\_\_\_

Санкт-Петербург

2024

Цель и задание

Программно реализовать на любом языке программирования ассиметричный алгоритм шифрования RSA. Продемонстрировать на примерах его работу по зашифровыванию и расшифровыванию текста. Разрешается реализовать как версию с пользовательским интерфейсом, так и консольную версию с дружелюбным командно-текстовым интерфейсом.

краткое описание хода разработки

Мой код реализует простое приложение для шифрования и расшифровки текста с использованием алгоритма RSA в графическом интерфейсе на базе библиотеки tkinter. Основные функции включают вычисление наибольшего общего делителя (НОД), проверку чисел на простоту, генерацию случайных простых чисел для формирования ключей, а также сами функции шифрования и расшифровки текста. Публичный и приватный ключи генерируются случайным образом, или могут быть введены пользователем вручную, с обязательной проверкой на совпадение модуля в обоих ключах.

Интерфейс включает текстовые поля для ввода исходного и зашифрованного текста, а также кнопки для выполнения шифрования и расшифровки. Пользователи могут задавать свои ключи, и также поддерживается копирование, вырезание и вставка через контекстное меню. Программа позволяет работать с текстами, шифруя и расшифровывая их с использованием алгоритма RSA, обеспечивая удобный и интуитивно понятный интерфейс для взаимодействия.

исходный код программы

import random

import tkinter as tk

from tkinter import messagebox

# Функции RSA

#НОД

def gcd(a, b):

    while b:

        a, b = b, a % b

    return a

#проверка на праймовость

def is\_prime(num):

    if num <= 1:

        return False

    for i in range(2, int(num\*\*0.5) + 1):

        if num % i == 0:

            return False

    return True

#генерация праймовых кандидатов

def generate\_prime\_candidate():

    while True:

        candidate = random.randint(100, 999)

        if is\_prime(candidate):

            return candidate

#генерация ключей

def generate\_keys():

    p = generate\_prime\_candidate()

    q = generate\_prime\_candidate()

    while q == p:

        q = generate\_prime\_candidate()

    n = p \* q

    phi\_n = (p - 1) \* (q - 1)

    e = random.randint(2, phi\_n - 1)

    while gcd(e, phi\_n) != 1:

        e = random.randint(2, phi\_n - 1)

    d = pow(e, -1, phi\_n)

    return ((e, n), (d, n))  # публичный ключ(e, n), закрытый ключ(d, n)

def encrypt(public\_key, plaintext):

    e, n = public\_key

    return [pow(ord(char), e, n) for char in plaintext]

def decrypt(private\_key, ciphertext):

    d, n = private\_key

    return ''.join([chr(pow(char, d, n)) for char in ciphertext])

# поддержка копирования, вырезания и вставки через меню

def create\_context\_menu(widget):

    menu = tk.Menu(widget, tearoff=0)

    menu.add\_command(label="Копировать", command=lambda: widget.event\_generate("<<Copy>>"))

    menu.add\_command(label="Вырезать", command=lambda: widget.event\_generate("<<Cut>>"))

    menu.add\_command(label="Вставить", command=lambda: widget.event\_generate("<<Paste>>"))

    def show\_menu(event):

        menu.post(event.x\_root, event.y\_root)

    widget.bind("<Button-3>", show\_menu)  #как оказалось, баттон 3 это пкм

# интерфейс

#генерация ключей

def generate\_keys\_ui():

    global public\_key, private\_key

    public\_key, private\_key = generate\_keys()

    public\_key\_label.config(text=f"Открытый ключ: {public\_key}")

    private\_key\_label.config(text=f"Закрытый ключ: {private\_key}")

    messagebox.showinfo("Ключи сгенерированы", "Ключи успешно сгенерированы!")

    print(public\_key,private\_key)

#добавление своих ключей

def set\_custom\_keys():

    try:

        global public\_key, private\_key

        e, n = map(int, public\_key\_entry.get().split(','))

        d, n\_private = map(int, private\_key\_entry.get().split(','))

        if n != n\_private:

            messagebox.showerror("Ошибка", "Модуль n в открытом и закрытом ключах должен совпадать!")

            return

        public\_key = (e, n)

        private\_key = (d, n)

        public\_key\_label.config(text=f"Открытый ключ: {public\_key}")

        private\_key\_label.config(text=f"Закрытый ключ: {private\_key}")

        messagebox.showinfo("Успешно", "Ключи добавлены!")

    except ValueError:

        messagebox.showerror("Ошибка", "Введите ключи в формате: число,число")

        print(public\_key,private\_key)

#шифровка текста

def encrypt\_text():

    plaintext = input\_text.get("1.0", "end-1c")

    if not public\_key:

        messagebox.showerror("Ошибка", "Сначала сгенерируйте ключи или задайте свои!")

        return

    ciphertext = encrypt(public\_key, plaintext)

    encrypted\_text.delete("1.0", "end")

    encrypted\_text.insert("1.0", ' '.join(map(str, ciphertext)))

    print(' '.join(map(str, ciphertext)))

#расшифровка текста

def decrypt\_text():

    ciphertext = encrypted\_text.get("1.0", "end-1c")

    if not private\_key:

        messagebox.showerror("Ошибка", "Сначала сгенерируйте ключи или задайте свои!")

        return

    try:

        ciphertext\_list = list(map(int, ciphertext.split()))

        plaintext = decrypt(private\_key, ciphertext\_list)

        decrypted\_text.delete("1.0", "end")

        decrypted\_text.insert("1.0", plaintext)

    except ValueError:

        messagebox.showerror("Ошибка", "Некорректный зашифрованный текст!")

# Создание окна

root = tk.Tk()

root.title("9 лаба")

# прописал интерфейс, на счет "дружелюбности" не уверен,

# ибо он тварь та еще, ему прописывать копирование и вставку отдельно надо, оказывается, но шо поделать, живем же как-то

frame = tk.Frame(root)

frame.pack(pady=10, padx=10)

generate\_button = tk.Button(frame, text="Сгенерировать ключи", command=generate\_keys\_ui)

generate\_button.grid(row=0, column=0, columnspan=2, pady=5)

public\_key\_label = tk.Label(frame, text="Открытый ключ: ---")

public\_key\_label.grid(row=1, column=0, columnspan=2, pady=5)

private\_key\_label = tk.Label(frame, text="Закрытый ключ: ---")

private\_key\_label.grid(row=2, column=0, columnspan=2, pady=5)

# ввод своих клбчей

custom\_keys\_label = tk.Label(frame, text="Задать свои ключи:")

custom\_keys\_label.grid(row=3, column=0, columnspan=2, pady=5)

public\_key\_entry = tk.Entry(frame, width=30)

public\_key\_entry.insert(0, "Введите открытый ключ (e,n)")

public\_key\_entry.grid(row=4, column=0, pady=5)

private\_key\_entry = tk.Entry(frame, width=30)

private\_key\_entry.insert(0, "Введите закрытый ключ (d,n)")

private\_key\_entry.grid(row=4, column=1, pady=5)

set\_keys\_button = tk.Button(frame, text="Добавить ключи", command=set\_custom\_keys)

set\_keys\_button.grid(row=5, column=0, columnspan=2, pady=5)

# шифровка и расшифровка

input\_text\_label = tk.Label(frame, text="Текст для шифрования:")

input\_text\_label.grid(row=6, column=0, pady=5)

input\_text = tk.Text(frame, height=3, width=40)

input\_text.grid(row=6, column=1, pady=5)

create\_context\_menu(input\_text)

encrypt\_button = tk.Button(frame, text="Зашифровать", command=encrypt\_text)

encrypt\_button.grid(row=7, column=0, pady=5)

encrypted\_text = tk.Text(frame, height=3, width=40)

encrypted\_text.grid(row=7, column=1, pady=5)

create\_context\_menu(encrypted\_text)

decrypt\_button = tk.Button(frame, text="Расшифровать", command=decrypt\_text)

decrypt\_button.grid(row=8, column=0, pady=5)

decrypted\_text = tk.Text(frame, height=3, width=40)

decrypted\_text.grid(row=8, column=1, pady=5)

create\_context\_menu(decrypted\_text)

# глобальные переменные

public\_key = None

private\_key = None

# запуск интерфейса

root.mainloop()

Результаты работы с примерами

Ниже приведены результаты работы программы с различными наборами входных данных

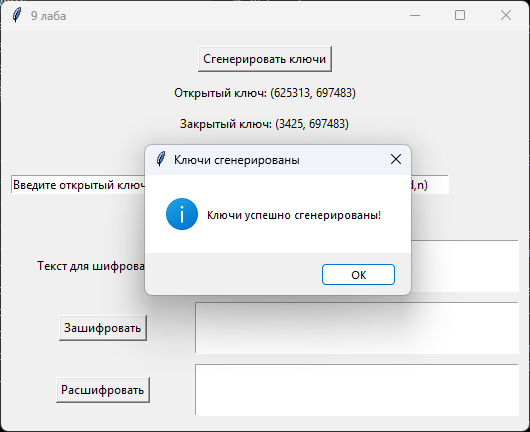


Рисунок 1 – Результаты работы программы

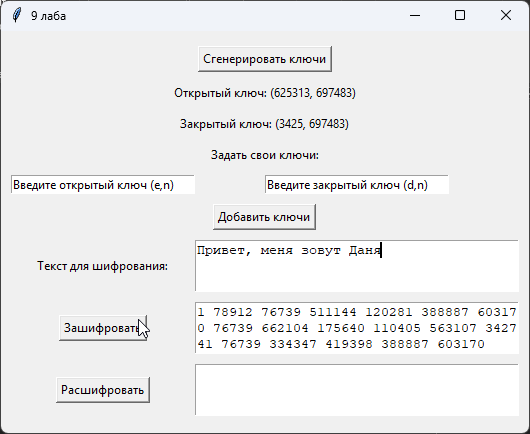


Рисунок 2 – Результаты работы программы

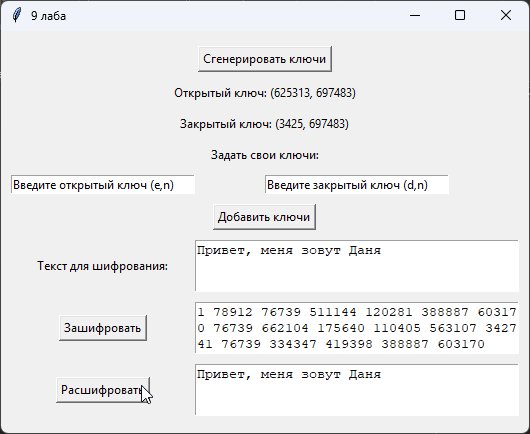
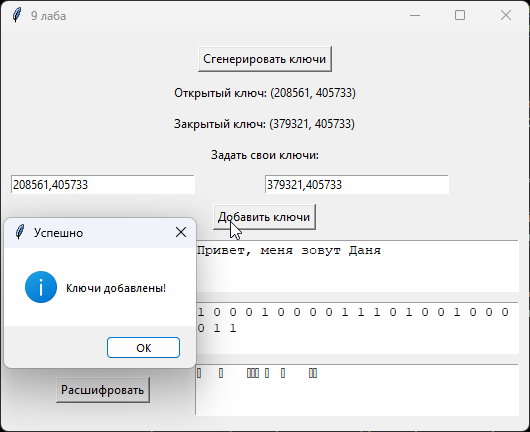


Рисунок 3 – Результаты работы программы



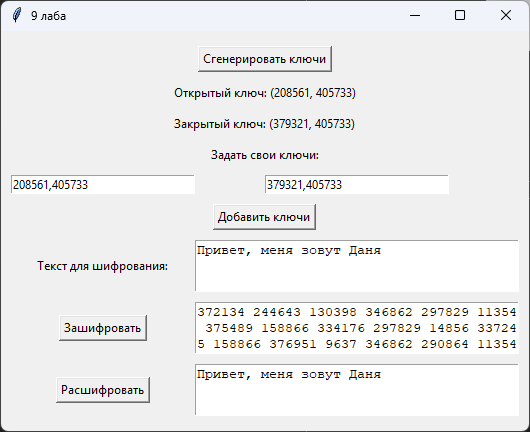


Рисунок 4 – Результаты работы программы

Выводы

В ходе выполнения лабораторной работы удалось реализовать программу, выполняющую шифрование и расшифровку текста с использованием алгоритма RSA. Программа генерирует публичные и приватные ключи, а также позволяет пользователю вводить собственные ключи для шифрования и расшифровки. В интерфейсе реализованы функции для копирования, вырезания и вставки текста, а также кнопки для выполнения операций шифрования и расшифровки. Все операции выполняются с использованием стандартных криптографических методов RSA, что позволяет эффективно обеспечивать безопасность данных.