Федеральное государственное бюджетное образовательное учреждение высшего образования «Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации».

## Кафедра «Прикладная Информатика»

ОТЧЕТ

О ПРОДЕЛАННОЙ ЛАБОРАТОРНОЙ РАБОТЕ №3

по курсу «Информационная безопасность»

Выполнил: студент группы

ИК-731

**(наименование группы)**

***Соколов Дмитрий Александрович***

**(Ф.И.О.)**

## Нижний Новгород

**2025 г.**

ЛАБОРАТОРНАЯ РАБОТА № 3

«Ассиметричные алгоритмы шифрования данных»

Цель работы: освоить методику работы ассиметричных алгоритмов шифрования, где существует два ключа – один для шифрования, другой для дешифрования.

Вариант № 18

**Задание**

Разработать визуальное приложение для шифрования/дешифрования изображений.

**Теория для создания приложения**

**Этап 1**

Необходимо сгенерировать ключи RSA.

1. Выбираются два различных простых числа p и q.
2. Вычисляется модуль: n = p \* q
3. Вычисляется функция Эйлера: ф(n) = (p-1)\*(q-1)
4. Выбирается число e, взаимно простое с ф(n), 1 < e < ф(n)
5. Вычисляется число d, такое что: e \* d ≡ 1 (mod ф(n))
6. Теперь у нас есть e,n – ключи для шифрования и d,n – закрытые ключи для дешифровки.

**Этап 2**

Шифрование данных

1. Для ускорения процесса я беру первые 200байт изображения
2. Байты преобразуются в 1 целое число
3. Применяется форму RSA для шифрования: с = me mod n (m – исходные данные), получаем c – зашифрованное число

**Этап 3**

Дешифрование данных

1. Из файла .enc берем зашифрованное число c
2. Применяем формулу m = cd mod n, d – закрытый ключ, который определили ранее.
3. Полученное число преобразуется обратно в байты.
4. Восстановленный фрагмент сохраняется в файл \_decrypted.jpg.

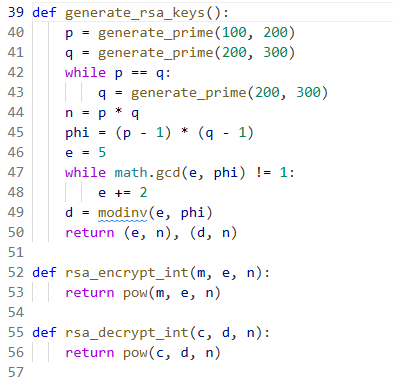
**Алгоритм работы программы**

1. Запуск программы.
2. Создание пары ключей RSA.
3. Выбираем изображение для демонстрации.
4. Программа шифрует первые 200 байт изображения с использованием открытого ключа.
5. Зашифрованный файл сохраняется на диск.
6. Для расшифровки берем файл .enc.
7. Программа расшифровывает данные с использованием закрытого ключа.
8. Восстановленный фрагмент сохраняется в новый файл и может быть открыт для просмотра (т.к шифруем часть изображения, то при расшифровке мы не получим полное изображение).

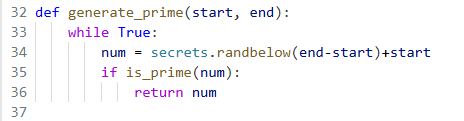
**Листинг программы**

Для реализации я выбрал Python и популярные библиотеку для создания интерфейса **tkinter и pillow для работы с изображениями.**

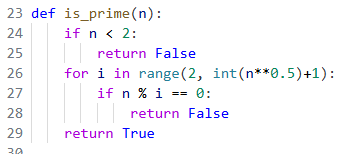
Функция генерации ключей RSA:



Необходимо получить 2 простых числа, используем вспомогательную функцию generate\_prime():



Далее в этой функции вызывается ещё одна функция, которая проверяет, что число действительно простое. is\_prime():



Далее в функции generate\_rsa\_keys(), запускается цикл пока p не будет равно q (обязательное условие из теории)

Далее в этой же функции считаем значения ключей по формулам из теории:

Считаем модуль и функция эйлера



Далее нужно выбрать e, удовлетворяющее условие φ(n), 1 < e < φ(n).

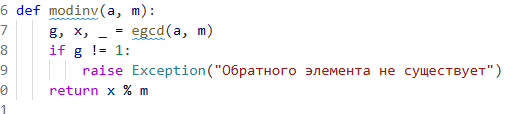


gcd в Python - это функция для вычисления наибольшего общего делителя

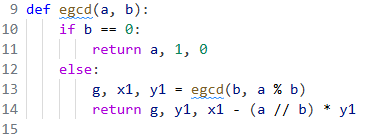
После того как нашли e, необходимо найти d:



Для этого вызываем функцию modinv(), которая ищет обратное по модулю:



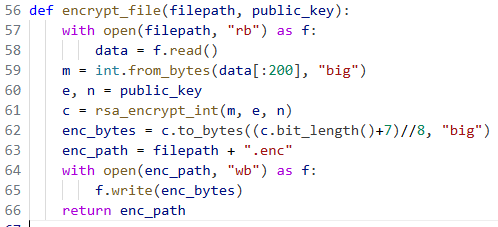
Эта функция использует ещё одну, именно egcd (это расширенный алгоритм Евклида, написанный на Python):



После выполнения данных функций, я получаю число d и формирую пары ключей:



Теперь все готово для шифрования, создаем функцию encrypt\_file(), которая принимает выбранный файл (точнее его путь на пк) и публичные ключи (e, n):



Алгоритм функции:

Открываем файл в байтовом представлении и складываем в переменную data:



Берем первые 200 байт и преобразовываем в целое число(будет достаточно для демонстрации):



M – это и есть данные для шифрации, а именно 200 байт нашего изображения.

Далее деструктурируем e,n из публичных ключей:



Получаем зашифрованное значение наших данных в переменную С:



Используем функцию rsa\_encrypt\_int():



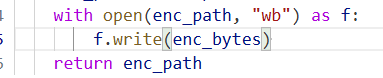
Преобразовываем в байты C:



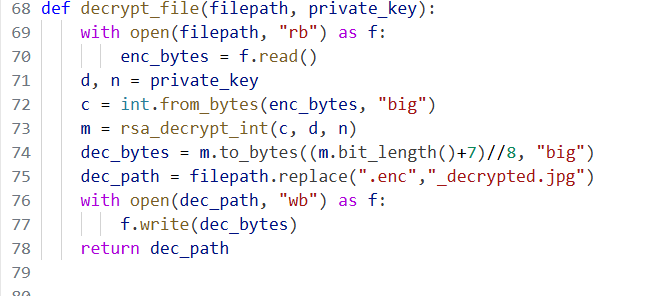
Создаем путь для сохранения:



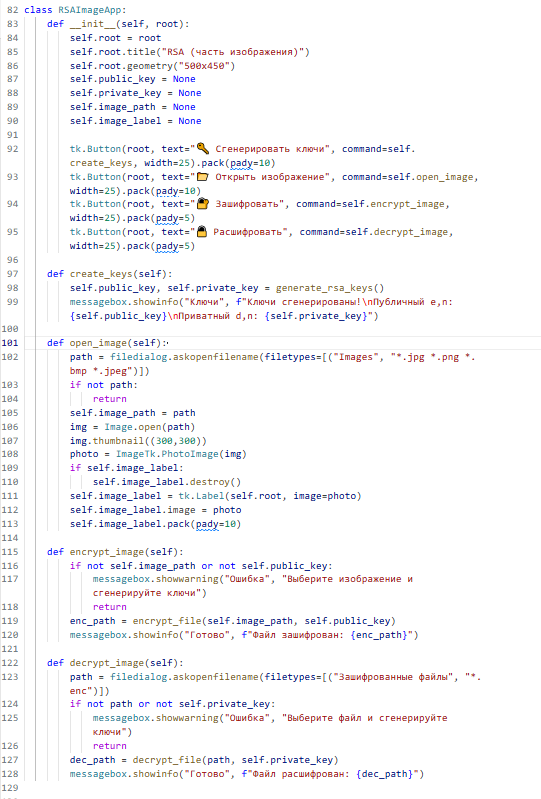
Сохраняем зашифрованную информацию:



Последняя функция, которая делает обратное действие, берет уже зашифрованный файл:

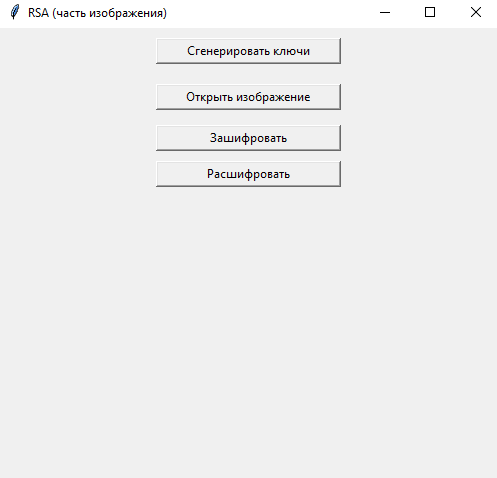


Функция не относящаяся к шифрации, которая используется для отрисовки интерфейса программы, написанная на tkinter:

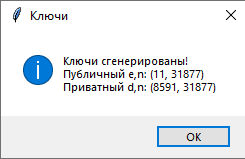


Результат работы программы:

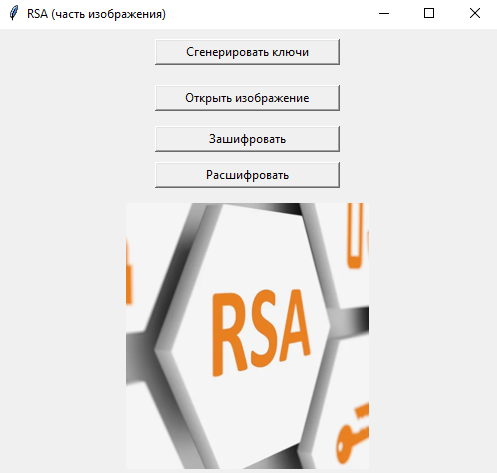
Вывод интерфейса:



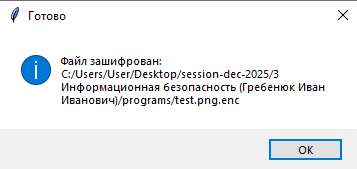
Нажатие на кнопку «Сгенерировать ключи», генерирует их:



Далее загружаем изображение, сохранил тестовое изображение в директорию проекта:



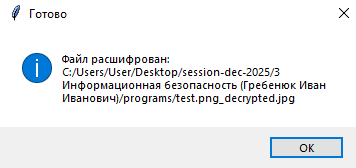
Нажимаем зашифровать:



Появился зашифрованный файл:



Пробуем расшифровать, выбираем зашифрованный файл, используем те же ключи (не генерируем новые):



Получаем часть расшифрованного изображения:



**Выводы**

В ходе лабораторной работы по шифрованию фрагмента изображения с использованием алгоритма RSA была изучена теория асимметричной криптографии и принципы работы открытого и закрытого ключей.

Мной были реализованы функции генерации ключей, шифрования и дешифрования, а также графический интерфейс для демонстрации работы алгоритма.

В работе подчеркнута важность выбора простых чисел и вычисления функции Эйлера для корректной работы алгоритма. Итогом стало создание наглядного приложения, демонстрирующего базовые возможности асимметричного шифрования и концепцию защиты данных.