Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №2.1
по дисциплине «Защита информации»

Студент Лукьяненко В.А.

Группа ИУ7-71Б

Преподаватель Руденкова Ю.С.

2025 г.

1 Задание

1.1 Цель работы

Цель работы: разработка алгоритма шифрования с открытым ключом. Шифрование и расшифровка архивных файлов.

1.2 Содержание работы

Для выполнения данной лабораторной работы необходимо решить следующие задачи:

1. реализовать программу шифрования алгоритмом с открытым ключом;
2. использовать алгоритм RSA;
3. обеспечить шифрование и расшифровку файла архива (rar, zip или др.) с использованием разработанной программы;
4. предусмотреть работу программы с пустым и однобайтовым файлом.

2 Теоретическая часть

Вопросы для защиты работы

1. Алгоритм шифрования с открытым ключом архивного файла.

Algorithm 1 Алгоритм шифрования архивного файла RSA

Require: Архивный файл F_{in}, открытый ключ $K_{pub} = (e, n)$
Ensure: Зашифрованный файл F_{enc}
1: Разбить F_{in} на блоки длиной не более $|n|$ бит.
2: **for** каждый блок M_i **do**
3: Преобразовать M_i в число m.
4: Вычислить $c \equiv m^e \pmod{n}$.
5: Записать c в F_{enc}.
6: **end for**
7: Сохранить результат в F_{enc}.

2. Алгоритм расшифровки с открытым ключом архивного файла.

Algorithm 2 Алгоритм расшифровки архивного файла RSA

Require: Зашифрованный файл F_{enc}, закрытый ключ $K_{priv} = (d, n)$
Ensure: Исходный архивный файл F_{out}
1: Разбить F_{enc} на блоки.
2: **for** каждый блок C_i **do**
3: Преобразовать C_i в число c.
4: Вычислить $m \equiv c^d \pmod{n}$.
5: Преобразовать m в байтовую последовательность.
6: Записать результат в F_{out}.
7: **end for**
8: Сохранить результат в F_{out}.

3. Определение асимметричного шифрования и описание RSA.

Асимметричное шифрование — метод криптографии, при котором используются два ключа:

— открытый ключ, известный всем участникам и применяемый для шифрования;

— закрытый ключ, известный только владельцу и применяемый для расшифровки.

Алгоритм RSA:

1. Генерируется пара ключей: открытый (e, n) и закрытый (d, n).

2. Для шифрования сообщения m вычисляется $c \equiv m^e \pmod{n}$.

3. Для расшифровки вычисляется $m \equiv c^d \pmod{n}$.

Ключи создаёт **получатель**: он публикует открытый ключ для отправителей и хранит в секрете закрытый ключ, которым выполняется расшифровка.

3 Практическая часть.

Листинг 3.1 – Файл `main.py`,

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import os

def generate_keys(key_size=2048):
    key = RSA.generate(key_size)
    private_key = key.export_key()
    public_key = key.publickey().export_key()

    with open("private.pem", "wb") as f:
        f.write(private_key)
    with open("public.pem", "wb") as f:
        f.write(public_key)

def encrypt_file(input_file, output_file, public_key_file):
    with open(public_key_file, "rb") as f:
        public_key = RSA.import_key(f.read())
    cipher = PKCS1_OAEP.new(public_key)

    with open(input_file, "rb") as f:
        data = f.read()

    block_size = public_key.size_in_bytes() - 42
    encrypted = b""
    for i in range(0, len(data), block_size):
        block = data[i:i + block_size]
        encrypted += cipher.encrypt(block)

    with open(output_file, "wb") as f:
        f.write(encrypted)

def decrypt_file(input_file, output_file, private_key_file):
    with open(private_key_file, "rb") as f:
        private_key = RSA.import_key(f.read())
    cipher = PKCS1_OAEP.new(private_key)

    with open(input_file, "rb") as f:
        encrypted = f.read()
```

```
39
40      block_size = private_key.size_in_bytes()
41      decrypted = b""
42      for i in range(0, len(encrypted), block_size):
43          block = encrypted[i:i + block_size]
44          decrypted += cipher.decrypt(block)
45
46      with open(output_file, "wb") as f:
47          f.write(decrypted)
48
49
50  if __name__ == "__main__":
51      if not os.path.exists("private.pem") or not
            os.path.exists("public.pem"):
52          generate_keys()
53
54      encrypt_file("input.zip", "encrypted.bin", "public.pem")
55      decrypt_file("encrypted.bin", "output.zip", "private.pem")
56
57      print("Шифрование и дешифровка завершены.")
```

4 Пример работы программы

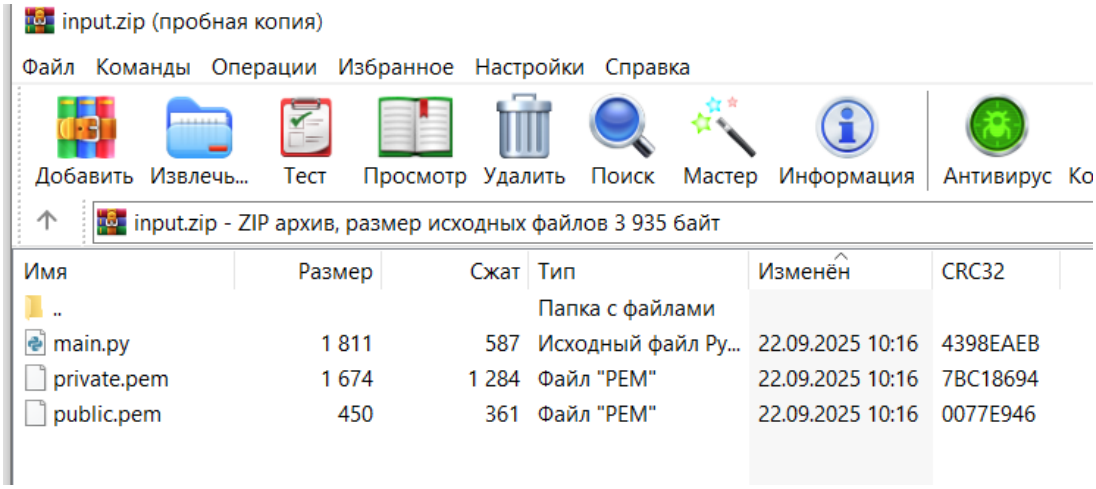

Рисунок 4.1 – Архив до шифрования

В результате шифрования получаются бинарный файл следующего содержания:

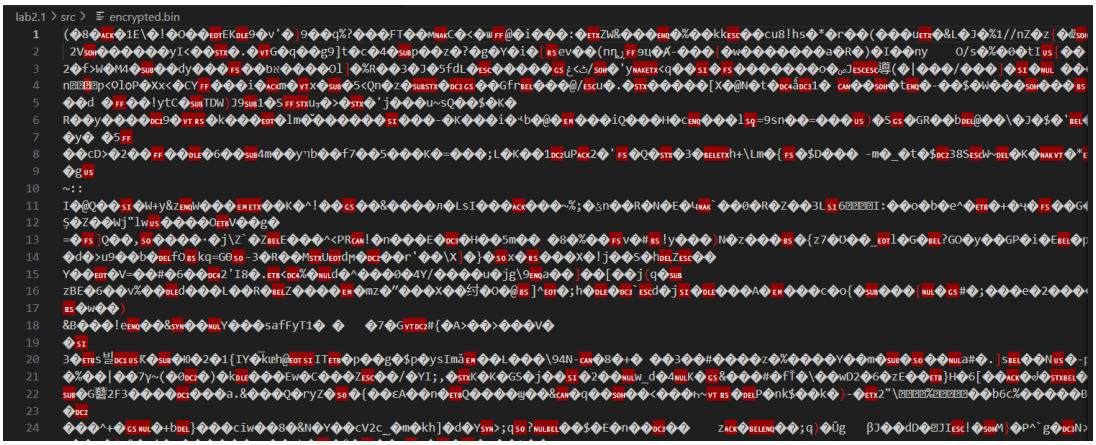

Рисунок 4.2 – Бинарный файл после шифрования

После, расшифруем бинарный файл:

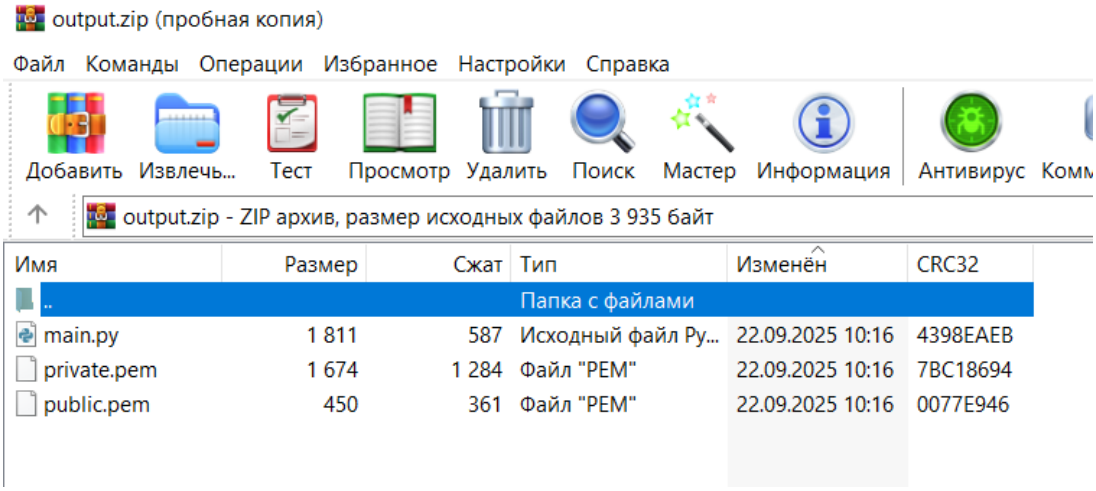

Рисунок 4.3 – Расшифрованный файл