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 №3
по дисциплине «Защита информации»

Тема Разработка шифровальной машины «Энигма»

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 произвольного файла.

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 произвольного файла с использованием разработанной программы;
4. предусмотреть работу программы с пустым, однобайтовым файлом.

2 Теоретическая часть

1. Алгоритм шифрования файла с использованием открытого ключа (RSA).

Algorithm 1 Алгоритм шифрования

Require: Входной файл F_{in}, открытый ключ K_{pub}
Ensure: Зашифрованный файл F_{enc}

1: Считать содержимое файла F_{in} поблочно (размер блока зависит от длины ключа).
2: **for** каждый блок M_i **do**
3: Преобразовать M_i в число m.
4: Вычислить $c \equiv m^e \pmod{n}$, где (e, n) — открытый ключ.
5: Записать c в выходной поток.
6: **end for**
7: Сохранить результат в F_{enc}.

2. Алгоритм расшифровки файла с использованием закрытого ключа (RSA).

Algorithm 2 Алгоритм расшифровки

Require: Зашифрованный файл F_{enc}, закрытый ключ K_{priv}
Ensure: Исходный файл F_{out}

1: Считать содержимое файла F_{enc} поблочно.
2: **for** каждый блок C_i **do**
3: Преобразовать C_i в число c.
4: Вычислить $m \equiv c^d \pmod{n}$, где (d, n) — закрытый ключ.
5: Преобразовать m в байтовую последовательность.
6: Записать её в выходной поток.
7: **end for**
8: Сохранить результат в F_{out}.

3. Определение асимметричного шифрования.

Асимметричное шифрование — это метод криптографии, в котором используются два ключа:

— открытый ключ, доступный любому пользователю и применяемый для шифрования;

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

Главное свойство: знание открытого ключа не позволяет вычислить закрытый ключ, что обеспечивает высокий уровень защиты.

3 Практическая часть.

Листинг 3.1 – Файл `main.py`,

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

def generate_keys():
    key = RSA.generate(2048)
    private_key = key.export_key()
    public_key = key.publickey().export_key()
    with open("private.pem", "wb") as priv_file:
        priv_file.write(private_key)
    with open("public.pem", "wb") as pub_file:
        pub_file.write(public_key)

def load_key(filename):
    with open(filename, "rb") as f:
        return RSA.import_key(f.read())

def encrypt_file(input_filename, output_filename,
   public_key_file="public.pem"):
    public_key = load_key(public_key_file)
    cipher = PKCS1_OAEP.new(public_key)

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

    encrypted_data = b""
    chunk_size = 190
    for i in range(0, len(data), chunk_size):
        encrypted_data += cipher.encrypt(data[i:i+chunk_size])

    with open(output_filename, "wb") as f:
        f.write(encrypted_data)

    print(f"Файл {input_filename} зашифрован в {output_filename}")

def decrypt_file(input_filename, output_filename,
   private_key_file="private.pem"):
    private_key = load_key(private_key_file)
    cipher = PKCS1_OAEP.new(private_key)
```

```
37
38      with open(input_filename, "rb") as f:
39          encrypted_data = f.read()
40
41      decrypted_data = b""
42      chunk_size = 256
43      for i in range(0, len(encrypted_data), chunk_size):
44          decrypted_data +=
                cipher.decrypt(encrypted_data[i:i+chunk_size])
45
46      with open(output_filename, "wb") as f:
47          f.write(decrypted_data)
48
49      print(f"Файл␣{input_filename}␣расшифрован␣в␣{output_filename}")
50
51
52  if __name__ == "__main__":
53      generate_keys()
54      encrypt_file("input.txt", "encrypted.bin")
55      decrypt_file("encrypted.bin", "decrypted.txt")
```

4 Пример работы программы

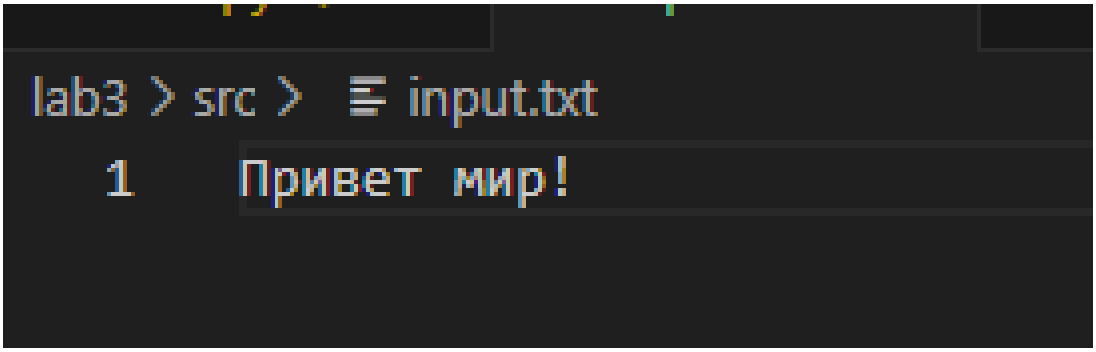

Рисунок 4.1 – Файл до шифрования

Рисунок 4.2 – Файл после шифрования

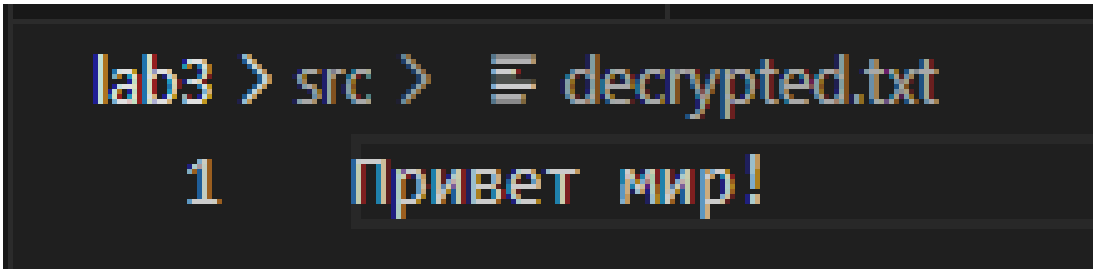

Рисунок 4.3 – Расшифрованный файл

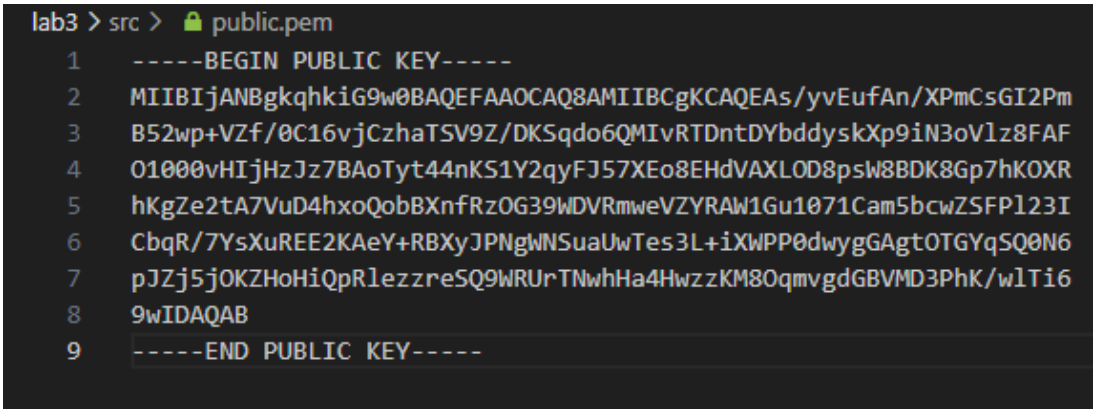

Рисунок 4.4 – Публичный ключ