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 №3.1
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 симметричного шифрования (AES). Шифрование и расшифровка произвольных архивных файлов.

1.2 Содержание работы

Для выполнения данной лабораторной работы необходимо решить следующие задачи:

1. реализовать программу шифрования симметричным алгоритмом AES;
2. обеспечить шифрование и расшифровку произвольного архивного файла (zip, rar или др.) с использованием разработанной программы;
3. предусмотреть работу программы с пустым и однобайтовым архивом;
4. протестировать работу программы на архивных файлах различных форматов.

2 Теоретическая часть

Вопросы для защиты работы

1. Перечислите подсистемы защиты информации в автоматизированных системах.

В автоматизированных системах можно выделить следующие подсистемы защиты информации:

— **Подсистема разграничения доступа** — контроль прав пользователей и администраторов.

— **Подсистема аутентификации и идентификации** — проверка подлинности пользователей.

— **Подсистема криптографической защиты** — шифрование данных, защита при передаче и хранении.

— **Подсистема регистрации и учёта событий** — ведение журналов доступа и действий.

— **Подсистема антивирусной и сетевой защиты** — предотвращение несанкционированного доступа и заражения.

2. Опишите алгоритм шифрования AES.

Algorithm 1 Алгоритм шифрования AES

Require: Входной файл F_{in}, секретный ключ K длиной 128, 192 или 256 бит
Ensure: Зашифрованный файл F_{enc}

Разбить F_{in} на блоки по 128 бит.
for каждый блок M_i **do**
    Выполнить начальное сложение блока с ключом (*AddRoundKey*).
    for $i = 1$ **to** $N_r - 1$ **do** ▷ N_r — число раундов (10, 12 или 14)
        Выполнить побайтовую подстановку (*SubBytes*).
        Переставить строки матрицы (*ShiftRows*).
        Смешать столбцы матрицы (*MixColumns*).
        Выполнить *AddRoundKey* с подключом K_i.
    end for
    Выполнить финальный раунд: *SubBytes*, *ShiftRows*, *AddRoundKey*.
    Записать зашифрованный блок в F_{enc}.
end for
Сохранить результат в F_{enc}.

3. Назовите особенности алгоритма симметричного шифрования архивных файлов.

— Архивы представляют собой двоичные данные, поэтому их нужно обрабатывать в бинарном режиме.

— Для AES используется блочная структура (128 бит), поэтому данные архивов необходимо дополнять до кратности блоку.

— Даже небольшая ошибка в шифровании или расшифровке может повредить весь архив, что делает контроль целостности критически важным.

— Шифрование производится уже после упаковки, поэтому степень сжатия архива не изменяется.

— Для повышения криптостойкости AES применяется в режиме CBC или аналогичных режимах с инициализационным вектором.

3 Практическая часть.

Листинг 3.1 – Файл `main.py`,

```python
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import os

def pad(data: bytes) -> bytes:
    padding_len = 16 - (len(data) % 16)
    return data + bytes([padding_len] * padding_len)

def unpad(data: bytes) -> bytes:
    padding_len = data[-1]
    return data[:-padding_len]

def encrypt_file(input_file: str, output_file: str, key: bytes):
    cipher = AES.new(key, AES.MODE_CBC)
    with open(input_file, "rb") as f:
        plaintext = f.read()

    padded_data = pad(plaintext)
    ciphertext = cipher.encrypt(padded_data)

    with open(output_file, "wb") as f:
        f.write(cipher.iv + ciphertext)

def decrypt_file(input_file: str, output_file: str, key: bytes):
    with open(input_file, "rb") as f:
        iv = f.read(16)
        ciphertext = f.read()

    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_data = cipher.decrypt(ciphertext)
    unpadded_data = unpad(decrypted_data)

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


if __name__ == "__main__":
    key = b"thisisasecretkey"
```

```
39
40      encrypt_file("input.zip", "encrypted.bin", key)
41      decrypt_file("encrypted.bin", "output.zip", key)
42
43      print("Шифрование и расшифровка завершены.")
```

4 Пример работы программы

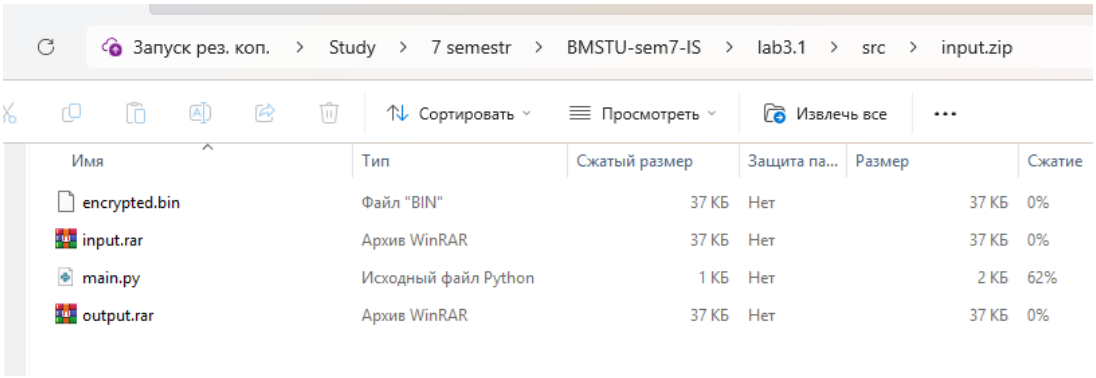

Рисунок 4.1 – Архив до шифрования

В результате шифрования получаются бинарный файл следующего содержания:

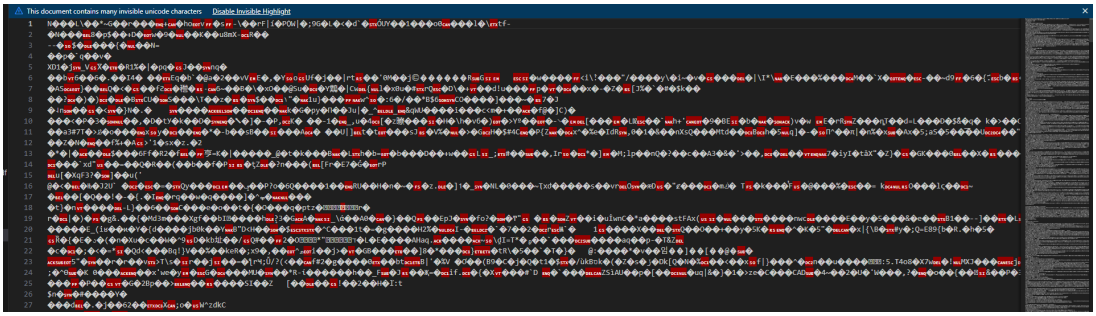

Рисунок 4.2 – Бинарный файл

После, расшифруем бинарный файл:

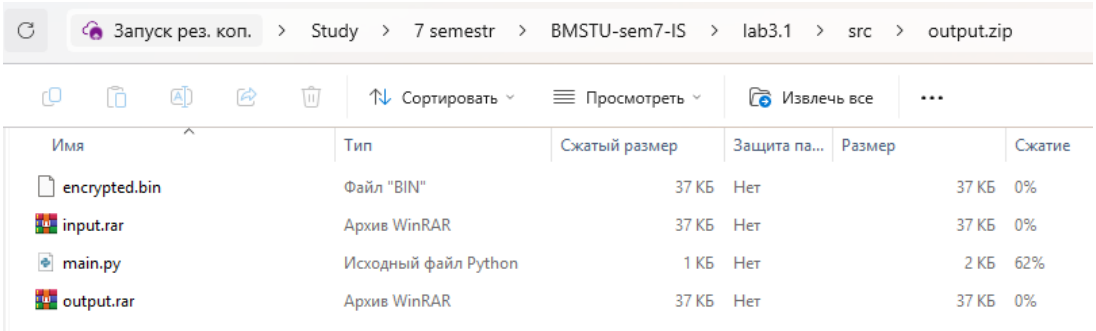

Рисунок 4.3 – Расшифрованный архив