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
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 электронного аналога машины «Энигма» для шифрования архивного файла.

1.2 Содержание работы

Для выполнения данной лабораторной работы необходимо решить следующие задачи:

1. реализовать в виде программы электронный аналог машины «Энигма»;
2. обеспечить шифрование архивного файла;
3. обеспечить расшифровку архивного файла;

2 Теоретическая часть

1. Определение информации, защиты информации, актив, информационная сфера, угроза, шифровальная машина «Энигма».

— **Информация** — это данные, которые могут быть восприняты, обработаны или использованы человеком или техническими средствами.

— **Защита информации** — это комплекс организационных, технических и программных мер, направленных на предотвращение несанкционированного доступа, искажения, утраты или уничтожения информации.

— **Актив** — это любой объект, обладающий ценностью для владельца (данные, оборудование, программное обеспечение и т.д.).

— **Информационная сфера** — это область деятельности, связанная с формированием, хранением, обработкой и использованием информации, а также воздействием информации на общество и человека.

— **Угроза** — это потенциальное событие или действие, которое может нанести ущерб информационным активам (кража данных, модификация, уничтожение).

— **Шифровальная машина «Энигма»** — электромеханическое устройство для шифрования текста, применявшееся Германией во время Второй мировой войны, основанное на многоалфавитной подстановке с использованием роторов.

2. Опишите алгоритм шифрования и дешифрования архивного файла.

Algorithm 1 Алгоритм шифрования и расшифровки произвольного файла

Файл настроек `settings.txt`, входной файл F_{in} Выходной файл F_{out} Считать параметры из `settings.txt`:

— типы роторов, их начальные позиции и кольцевые смещения;

— пары замен для коммутатора (plugboard);

— пары замен для отражателя (reflector).

Построить объекты: список роторов $R = (R_1, R_2, R_3)$, коммутатор P, отражатель M.
Открыть файл F_{in} в бинарном режиме и считать все байты в массив $D = (b_1, b_2, \ldots, b_n)$.
каждый байт $b_i \in D$ Выполнить шаг механизма роторов (правый всегда вращается, средний и левый — по условию попадания в позицию «notch»). Пропустить b_i через коммутатор: $c \leftarrow P(b_i)$. Последовательно пропустить c через роторы в прямом направлении (справа налево). Отразить результат через отражатель: $c \leftarrow M(c)$. Последовательно пропустить c через роторы в обратном направлении (слева направо). Снова пропустить c через коммутатор: $c \leftarrow P(c)$. Записать c в выходную последовательность D'.
Записать массив D' в файл F_{out} в бинарном режиме.

3 Практическая часть.

Листинг 3.1 – Файл `rotor.py`, описывающий поведение роторов

```
ALPHABET = list(range(256))

class Rotor:
    def __init__(self, wiring, notch, ring_setting=0, position=0):
        self.wiring = wiring
        self.notch = notch
        self.ring_setting = ring_setting
        self.position = position

    def step(self):
        self.position = (self.position + 1) % 256
        return self.position == self.notch

    def encode_forward(self, c: int) -> int:
        idx = (c + self.position - self.ring_setting) % 256
        encoded = self.wiring[idx]
        out = (encoded - self.position + self.ring_setting) % 256
        return out

    def encode_backward(self, c: int) -> int:
        idx = (c + self.position - self.ring_setting) % 256
        encoded_idx = self.wiring.index(idx)
        out = (encoded_idx - self.position + self.ring_setting) % 256
        return out
```

Листинг 3.2 – Файл `reflector.py`, описывающий поведение рефлектора

```
class Reflector:
    def __init__(self, pairs=None):
        self.mapping = {i: i for i in range(256)}
        if pairs:
            for a, b in pairs:
                self.mapping[a] = b
                self.mapping[b] = a

    def reflect(self, c: int) -> int:
        return self.mapping.get(c, c)
```

Листинг 3.3 – Файл `plugboard.py`, описывающий поведение плагборда

```
class Plugboard:
    def __init__(self, pairs=None):
        self.mapping = {i: i for i in range(256)}
        if pairs:
            for a, b in pairs:
                self.mapping[a] = b
                self.mapping[b] = a

    def encode(self, c: int) -> int:
        return self.mapping.get(c, c)
```

Листинг 3.4 – Файл `main.py`,

```python
from rotor import Rotor
from reflector import Reflector
from plugboard import Plugboard
import random

def read_settings(filename):
    with open(filename, "r") as f:
        lines = [line.strip() for line in f if line.strip()]

    rotor_types = lines[0].split()
    positions = [int(x) for x in lines[1].split()]
    ring_settings = [int(x) for x in lines[2].split()]

    plug_idx = next(i for i, line in enumerate(lines) if
        line.startswith("Plugboard"))
    refl_idx = next(i for i, line in enumerate(lines) if
        line.startswith("Reflector"))

    plug_pairs = [tuple(map(int, line.split())) for line in
        lines[plug_idx+1:refl_idx]]
    refl_pairs = [tuple(map(int, line.split())) for line in
        lines[refl_idx+1:]]

    return rotor_types, positions, ring_settings, plug_pairs,
        refl_pairs

def build_rotor(rotor_type, ring_setting, position):
    rng = random.Random(int(rotor_type))
    wiring = list(range(256))
    rng.shuffle(wiring)
    notch = rng.randint(0, 255)
    return Rotor(wiring, notch, ring_setting, position)

def build_machine(rotor_types, positions, ring_settings, plug_pairs,
    refl_pairs):
    rotors = [build_rotor(rtype, ring_settings[i], positions[i]) for
        i, rtype in enumerate(rotor_types)]
    plugboard = Plugboard(plug_pairs)
    reflector = Reflector(refl_pairs)
    return rotors, plugboard, reflector
```

```
34
35 def step_rotors(rotors):
36     right, middle, left = rotors
37     middle_on_notch = middle.position == middle.notch
38     right.step()
39     if right.position == right.notch or middle_on_notch:
40         middle.step()
41         if middle.position == middle.notch:
42             left.step()
43
44 def encode_message(data: bytes, rotors, plugboard, reflector) -> bytes:
45     out = bytearray()
46     for byte in data:
47         step_rotors(rotors)
48         c = plugboard.encode(byte)
49         for rotor in reversed(rotors):
50             c = rotor.encode_forward(c)
51         c = reflector.reflect(c)
52         for rotor in rotors:
53             c = rotor.encode_backward(c)
54         c = plugboard.encode(c)
55         out.append(c)
56     return bytes(out)
57
58 if __name__ == "__main__":
59     rotor_types, positions, ring_settings, plug_pairs, refl_pairs =
          read_settings("settings.txt")
60     rotors, plugboard, reflector = build_machine(rotor_types,
          positions, ring_settings, plug_pairs, refl_pairs)
61
62     with open("archive.rar", "rb") as f:
63         data = f.read()
64
65     result = encode_message(data, rotors, plugboard, reflector)
66
67     with open("archive.rar", "wb") as f:
68         f.write(result)
69
70     print("Файл␣зашифрован.")
```

4 Пример работы программы

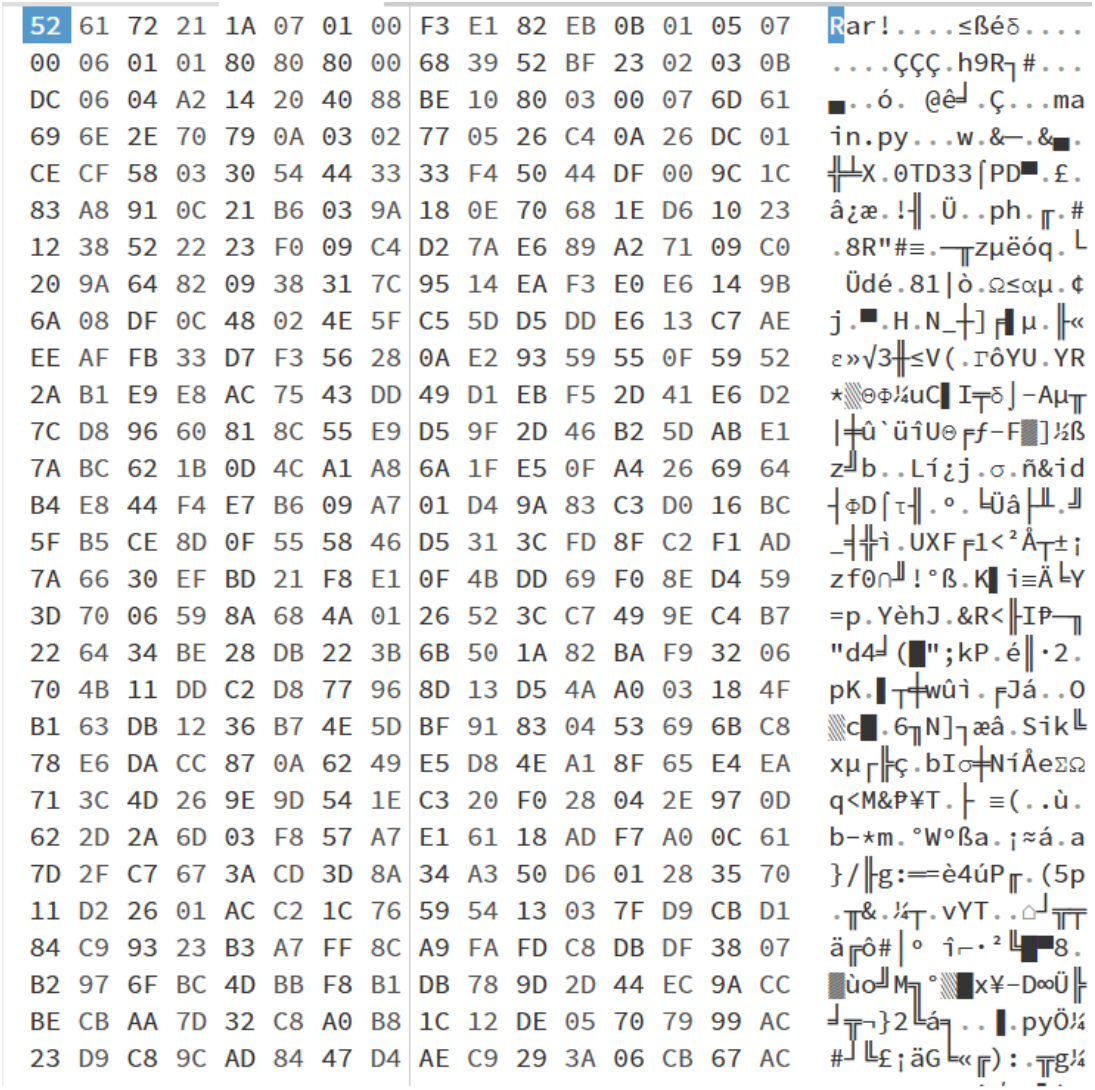

Рисунок 4.1 – Хекс-дамп файла до шифрования

Рисунок 4.2 – Хекс-дамп файла после шифрования