



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 $D = (b_1, b_2, \dots, b_n)$.

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

```
1 ALPHABET = list(range(256))
2
3 class Rotor:
4     def __init__(self, wiring, notch, ring_setting=0, position=0):
5         self.wiring = wiring
6         self.notch = notch
7         self.ring_setting = ring_setting
8         self.position = position
9
10    def step(self):
11        self.position = (self.position + 1) % 256
12        return self.position == self.notch
13
14    def encode_forward(self, c: int) -> int:
15        idx = (c + self.position - self.ring_setting) % 256
16        encoded = self.wiring[idx]
17        out = (encoded - self.position + self.ring_setting) % 256
18        return out
19
20    def encode_backward(self, c: int) -> int:
21        idx = (c + self.position - self.ring_setting) % 256
22        encoded_idx = self.wiring.index(idx)
23        out = (encoded_idx - self.position + self.ring_setting) % 256
24        return out
```

Листинг 3.2 – Файл reflector.py, описывающий поведение рефлектора

```
1 class Reflector:
2     def __init__(self, pairs=None):
3         self.mapping = {i: i for i in range(256)}
4         if pairs:
5             for a, b in pairs:
6                 self.mapping[a] = b
7                 self.mapping[b] = a
8
9     def reflect(self, c: int) -> int:
10        return self.mapping.get(c, c)
```

Листинг 3.3 – Файл plugboard.py, описывающий поведение плагборда

```
1 class Plugboard:
2     def __init__(self, pairs=None):
3         self.mapping = {i: i for i in range(256)}
4         if pairs:
5             for a, b in pairs:
6                 self.mapping[a] = b
7                 self.mapping[b] = a
8
9     def encode(self, c: int) -> int:
10        return self.mapping.get(c, c)
```

Листинг 3.4 – Файл main.py,

```
1 from rotor import Rotor
2 from reflector import Reflector
3 from plugboard import Plugboard
4 import random
5
6 def read_settings(filename):
7     with open(filename, "r") as f:
8         lines = [line.strip() for line in f if line.strip()]
9
10    rotor_types = lines[0].split()
11    positions = [int(x) for x in lines[1].split()]
12    ring_settings = [int(x) for x in lines[2].split()]
13
14    plug_idx = next(i for i, line in enumerate(lines) if
15                    line.startswith("Plugboard"))
16    refl_idx = next(i for i, line in enumerate(lines) if
17                    line.startswith("Reflector"))
18
19    plug_pairs = [tuple(map(int, line.split())) for line in
20                  lines[plug_idx+1:refl_idx]]
21    refl_pairs = [tuple(map(int, line.split())) for line in
22                  lines[refl_idx+1:]]
23
24    return rotor_types, positions, ring_settings, plug_pairs,
25           refl_pairs
26
27 def build_rotor(rotor_type, ring_setting, position):
28     rng = random.Random(int(rotor_type))
29     wiring = list(range(256))
30     rng.shuffle(wiring)
31     notch = rng.randint(0, 255)
32     return Rotor(wiring, notch, ring_setting, position)
33
34 def build_machine(rotor_types, positions, ring_settings, plug_pairs,
35                   refl_pairs):
36     rotors = [build_rotor(rtype, ring_settings[i], positions[i]) for
37               i, rtype in enumerate(rotor_types)]
38     plugboard = Plugboard(plug_pairs)
39     reflector = Reflector(refl_pairs)
40     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 =
60         read_settings("settings.txt")
61     rotors, plugboard, reflector = build_machine(rotor_types,
62         positions, ring_settings, plug_pairs, refl_pairs)
63
64     with open("archive.rar", "rb") as f:
65         data = f.read()
66
67     result = encode_message(data, rotors, plugboard, reflector)
68
69     with open("archive.rar", "wb") as f:
70         f.write(result)
71
72     print("Файл зашифрован.")

```


4 Пример работы программы

52	61	72	21	1A	07	01	00	F3	E1	82	EB	0B	01	05	07	Rar!....≤béδ....
00	06	01	01	80	80	80	00	68	39	52	BF	23	02	03	0BÇÇÇ.h9Rγ#...
DC	06	04	A2	14	20	40	88	BE	10	80	03	00	07	6D	61	■...ó. @é┘.Ç...ma
69	6E	2E	70	79	0A	03	02	77	05	26	C4	0A	26	DC	01	in.py...w.&—.&■.
CE	CF	58	03	30	54	44	33	33	F4	50	44	DF	00	9C	1C	┘.θTD33[PD■.£.
83	A8	91	0C	21	B6	03	9A	18	0E	70	68	1E	D6	10	23	â;æ.!┘.Ü...ph.┘.#
12	38	52	22	23	F0	09	C4	D2	7A	E6	89	A2	71	09	C0	.8R"#≡.┘zμëóq.┘
20	9A	64	82	09	38	31	7C	95	14	EA	F3	E0	E6	14	9B	Üdé.81 ð.Ω≤αμ.¢
6A	08	DF	0C	48	02	4E	5F	C5	5D	D5	DD	E6	13	C7	AE	j.■.H.N_┘]┘μ.┘«
EE	AF	FB	33	D7	F3	56	28	0A	E2	93	59	55	0F	59	52	ε»√3┘≤V(.ΓôYU.YR
2A	B1	E9	E8	AC	75	43	DD	49	D1	EB	F5	2D	41	E6	D2	*┘Φ½uC┘I┘δ┘-Αμ┘
7C	D8	96	60	81	8C	55	E9	D5	9F	2D	46	B2	5D	AB	E1	┘ù`üíU@┘f-F┘]½ß
7A	BC	62	1B	0D	4C	A1	A8	6A	1F	E5	0F	A4	26	69	64	z┘b...Lí; j.σ.ñ&id
B4	E8	44	F4	E7	B6	09	A7	01	D4	9A	83	C3	D0	16	BC	┘ΦD[┘┘.°.┘Üâ┘┘.┘
5F	B5	CE	8D	0F	55	58	46	D5	31	3C	FD	8F	C2	F1	AD	┘┘i.UXF┘1<²Ä┘±;
7A	66	30	EF	BD	21	F8	E1	0F	4B	DD	69	F0	8E	D4	59	z f0n┘!°ß.K┘ i≡Ä┘┘Y
3D	70	06	59	8A	68	4A	01	26	52	3C	C7	49	9E	C4	B7	=p.YèhJ.&R<┘IP┘┘
22	64	34	BE	28	DB	22	3B	6B	50	1A	82	BA	F9	32	06	"d4┘(┘";kP.é┘┘.2.
70	4B	11	DD	C2	D8	77	96	8D	13	D5	4A	A0	03	18	4F	pK.┘┘wùí.┘Já...0
B1	63	DB	12	36	B7	4E	5D	BF	91	83	04	53	69	6B	C8	┘c┘.6┘N┘]┘æâ.Sik┘┘
78	E6	DA	CC	87	0A	62	49	E5	D8	4E	A1	8F	65	E4	EA	xμ┘┘ç.bI┘┘NíÄεΣΩ
71	3C	4D	26	9E	9D	54	1E	C3	20	F0	28	04	2E	97	0D	q<M&P¥T.┘┘≡(. .ù.
62	2D	2A	6D	03	F8	57	A7	E1	61	18	AD	F7	A0	0C	61	b-*m.°W°ßa. ;~á.a
7D	2F	C7	67	3A	CD	3D	8A	34	A3	50	D6	01	28	35	70	} /┘g:≡=è4úP┘. (5p
11	D2	26	01	AC	C2	1C	76	59	54	13	03	7F	D9	CB	D1	.┘&.½┘.vYT. . .┘┘┘
84	C9	93	23	B3	A7	FF	8C	A9	FA	FD	C8	DB	DF	38	07	ä┘ô# ° í┘.²┘┘8.
B2	97	6F	BC	4D	BB	F8	B1	DB	78	9D	2D	44	EC	9A	CC	■ùσ┘M┘°┘┘x¥-D∞Ü┘┘
BE	CB	AA	7D	32	C8	A0	B8	1C	12	DE	05	70	79	99	AC	┘┘┘}2┘á┘. . .┘.pyÖ½
23	D9	C8	9C	AD	84	47	D4	AE	C9	29	3A	06	CB	67	AC	#┘┘┘; äG┘┘«┘┘): .┘g½

Рисунок 4.1 – Хекс-дамп файла до шифрования

