**ЛАБОРАТОРНАЯ РАБОТА №2**

“Блочное симметричное шифрование”

по дисциплине

‘Информационная безопасность’

Вариант 3Д

***Выполнил:***

Соболев Иван Александрович

**Группа:** P34312

***Преподаватель:***

Маркина Татьяна Анатольевна

Санкт-Петербург, 2024

# Цель работы

# Изучение структуры и основных принципов работы современных алгоритмов блочного симметричного шифрования, приобретение навыков программной реализации блочных симметричных шифров.

# Программные и аппаратные средства

Для выполнения лабораторной работы был использован компьютер со следующими характеристиками:

* Процессор: Apple M2
* Видеокарта: Apple M2
* Объем оперативной памяти: 8GB
* Использована операционная система: macOS 14.4.1
* Версия Python: 3.12

# Задание

# Реализовать систему симметричного блочного шифрования, позволяющую шифровать и дешифровать файл на диске с использованием заданного блочного шифра в заданном режиме шифрования.

Алгоритм: RC6  
Режим шифрования: OFB

# Листинг разработанной программы

io\_utils.py

import os  
from typing import Optional, Any  
  
ENCRYPT\_ACTION = "enc"  
DECRYPT\_ACTION = "dec"  
  
"""Функция для вывода ошибок"""  
def print\_red(message: str) -> None:  
 print(f"\033[91m{message}\033[0m")  
  
  
"""Функция чтения имени файла с открытым текстом из консоли"""  
def read\_filename() -> Optional[str]:  
 try:  
 filename = input("Введите названия файла с текстом для шифрования: ")  
 if not filename.strip():  
 print\_red("Ошибка: имя файла не должно быть пустым.")  
 return None  
  
 if not os.path.isfile(filename):  
 print\_red(f"Ошибка: файл '{filename}' не найден.")  
 return None  
 return filename  
 except Exception as e:  
 print\_red(f'Ошибка чтения из файла: {str(e)}')  
 return None  
  
  
"""Функция выбора действия"""  
def decide\_action() -> Any | None:  
 print('Выберите действие: \n'  
 '1. Зашифровать файл\n'  
 '2. Расшифровать файл')  
 try:  
 option = int(input("Введите номер: "))  
 if option not in [1, 2]:  
 print\_red(f'Ошибка: опция {option} недоступна!')  
 return None  
 if option == 1:  
 return ENCRYPT\_ACTION  
 elif option == 2:  
 return DECRYPT\_ACTION  
 else:  
 return None  
 except Exception:  
 print\_red(f'Ошибка: неверный ввод. Необходимо ввести целое число.')  
 return None

except Exception as e:  
 print\_red(f'Ошибка чтения из файла: {str(e)}')  
 return None  
  
  
"""Функция выбора действия"""  
def decide\_action() -> Any | None:  
 print('Выберите действие: \n'  
 '1. Зашифровать файл\n'  
 '2. Расшифровать файл')  
 try:  
 option = int(input("Введите номер: "))  
 if option not in [1, 2]:  
 print\_red(f'Ошибка: опция {option} недоступна!')  
 return None  
 if option == 1:  
 return ENCRYPT\_ACTION  
 elif option == 2:  
 return DECRYPT\_ACTION  
 else:  
 return None  
 except Exception:  
 print\_red(f'Ошибка: неверный ввод. Необходимо ввести целое число.')  
 return None

RC6.py

import struct  
  
w = 32 # длина слова в битах  
r = 20 # число раундов  
b = 16 # длина ключа  
P32 = 0xB7E15163  
Q32 = 0x9E3779B9  
LEFT = "LEFT"  
RIGHT = "RIGHT"  
  
"""Функция циклического сдвига"""  
def cyclic\_shift(x, y, direction):  
 y = y % w  
 if direction == LEFT:  
 return ((x << y) & (2 \*\* w - 1)) | (x >> (w - y))  
 if direction == RIGHT:  
 return (x >> y) | ((x << (w - y)) & (2 \*\* w - 1))  
  
  
"""Функция подготовки ключа"""  
def rc6\_key\_schedule(key):  
 L = [0] \* (b // 4)  
 for i in range(b - 1, -1, -1):  
 L[i // 4] = (L[i // 4] << 8) + key[i]  
  
 S = [P32]  
 for i in range(1, 2 \* r + 4):  
 S.append((S[i - 1] + Q32) % 2 \*\* w)  
  
 A = 0  
 B = 0  
 i = 0  
 j = 0  
 for \_ in range(3 \* max(b // 4, 2 \* r + 4)):  
 A = cyclic\_shift((S[i] + A + B) % 2 \*\* w, 3, LEFT)  
 S[i] = cyclic\_shift((S[i] + A + B) % 2 \*\* w, 3, LEFT)  
 B = cyclic\_shift((L[j] + A + B) % 2 \*\* w, (A + B) % w, LEFT)  
 L[j] = cyclic\_shift((L[j] + A + B) % 2 \*\* w, (A + B) % w, LEFT)  
 i = (i + 1) % (2 \* r + 4)  
 j = (j + 1) % (b // 4)  
 return S  
  
  
"""Функция RC6 шифрования"""  
def rc6\_encrypt(plaintext, S):  
 A, B, C, D = struct.unpack('<4I', plaintext)  
  
 B = (B + S[0]) % 2 \*\* w  
 D = (D + S[1]) % 2 \*\* w  
  
 for i in range(1, r + 1):  
 t = cyclic\_shift(B \* (2 \* B + 1) % 2 \*\* w, 5, LEFT) # 5 = log2(32)  
 u = cyclic\_shift(D \* (2 \* D + 1) % 2 \*\* w, 5, LEFT)  
 A = (cyclic\_shift(A ^ t, u, LEFT) + S[2 \* i]) % 2 \*\* w  
 C = (cyclic\_shift(C ^ u, t, LEFT) + S[2 \* i + 1]) % 2 \*\* w  
 A, B, C, D = B, C, D, A  
 A = (A + S[2 \* r + 2]) % 2 \*\* w  
 C = (C + S[2 \* r + 3]) % 2 \*\* w  
  
 return struct.pack('<4I', A, B, C, D)

S = [P32]  
 for i in range(1, 2 \* r + 4):  
 S.append((S[i - 1] + Q32) % 2 \*\* w)  
  
 A = 0  
 B = 0  
 i = 0  
 j = 0  
 for \_ in range(3 \* max(b // 4, 2 \* r + 4)):  
 A = cyclic\_shift((S[i] + A + B) % 2 \*\* w, 3, LEFT)  
 S[i] = cyclic\_shift((S[i] + A + B) % 2 \*\* w, 3, LEFT)  
 B = cyclic\_shift((L[j] + A + B) % 2 \*\* w, (A + B) % w, LEFT)  
 L[j] = cyclic\_shift((L[j] + A + B) % 2 \*\* w, (A + B) % w, LEFT)  
 i = (i + 1) % (2 \* r + 4)  
 j = (j + 1) % (b // 4)  
 return S  
  
  
"""Функция RC6 шифрования"""  
def rc6\_encrypt(plaintext, S):  
 A, B, C, D = struct.unpack('<4I', plaintext)  
  
 B = (B + S[0]) % 2 \*\* w  
 D = (D + S[1]) % 2 \*\* w  
  
 for i in range(1, r + 1):  
 t = cyclic\_shift(B \* (2 \* B + 1) % 2 \*\* w, 5, LEFT) # 5 = log2(32)  
 u = cyclic\_shift(D \* (2 \* D + 1) % 2 \*\* w, 5, LEFT)  
 A = (cyclic\_shift(A ^ t, u, LEFT) + S[2 \* i]) % 2 \*\* w  
 C = (cyclic\_shift(C ^ u, t, LEFT) + S[2 \* i + 1]) % 2 \*\* w  
 A, B, C, D = B, C, D, A  
 A = (A + S[2 \* r + 2]) % 2 \*\* w  
 C = (C + S[2 \* r + 3]) % 2 \*\* w  
  
 return struct.pack('<4I', A, B, C, D)

OFB.py

from RC6 import rc6\_key\_schedule, rc6\_encrypt  
  
"""Функция для заполнения текста до нужной длины  
 Заполняет числом равным длине финального паддинга"""  
def pad(plaintext):  
 padding\_len = 16 - (len(plaintext) % 16)  
 padding = bytes([padding\_len] \* padding\_len)  
 return plaintext + padding  
  
  
"""Функция для удаления падднига"""  
def unpad(padded\_plaintext):  
 padding\_len = padded\_plaintext[-1]  
 return padded\_plaintext[:-padding\_len]  
  
"""Функция операции XOR"""  
def xor\_bytes(a, b):  
 return bytes(x ^ y for x, y in zip(a, b))  
  
"""Функция для шифрования в режиме OFB"""  
def rc6\_ofb\_encrypt(key, iv, plaintext):  
 S = rc6\_key\_schedule(key)  
 ciphertext = bytearray()  
 feedback = iv  
 padded\_plaintext = pad(plaintext)  
  
 for i in range(0, len(padded\_plaintext), 16):  
 keystream = rc6\_encrypt(feedback, S)  
 block = padded\_plaintext[i:i + 16]  
 ciphertext\_block = xor\_bytes(block, keystream)  
 ciphertext.extend(ciphertext\_block)  
 feedback = keystream  
  
 return bytes(ciphertext)  
  
"""Функция для дешифрования в режиме OFB"""  
def rc6\_ofb\_decrypt(key, iv, ciphertext):  
 S = rc6\_key\_schedule(key)  
 plaintext = bytearray()  
 feedback = iv  
  
 for i in range(0, len(ciphertext), 16):  
 keystream = rc6\_encrypt(feedback, S)  
 block = ciphertext[i:i + 16]  
 plaintext\_block = xor\_bytes(block, keystream)  
 plaintext.extend(plaintext\_block)  
 feedback = keystream  
  
 return unpad(bytes(plaintext))

"""Функция операции XOR"""  
def xor\_bytes(a, b):  
 return bytes(x ^ y for x, y in zip(a, b))  
  
"""Функция для шифрования в режиме OFB"""  
def rc6\_ofb\_encrypt(key, iv, plaintext):  
 S = rc6\_key\_schedule(key)  
 ciphertext = bytearray()  
 feedback = iv  
 padded\_plaintext = pad(plaintext)  
  
 for i in range(0, len(padded\_plaintext), 16):  
 keystream = rc6\_encrypt(feedback, S)  
 block = padded\_plaintext[i:i + 16]  
 ciphertext\_block = xor\_bytes(block, keystream)  
 ciphertext.extend(ciphertext\_block)  
 feedback = keystream  
  
 return bytes(ciphertext)  
  
"""Функция для дешифрования в режиме OFB"""  
def rc6\_ofb\_decrypt(key, iv, ciphertext):  
 S = rc6\_key\_schedule(key)  
 plaintext = bytearray()  
 feedback = iv  
  
 for i in range(0, len(ciphertext), 16):  
 keystream = rc6\_encrypt(feedback, S)  
 block = ciphertext[i:i + 16]  
 plaintext\_block = xor\_bytes(block, keystream)  
 plaintext.extend(plaintext\_block)  
 feedback = keystream  
  
 return unpad(bytes(plaintext))

main.py

import os  
from OFB import rc6\_ofb\_encrypt, rc6\_ofb\_decrypt  
from io\_utils import decide\_action, ENCRYPT\_ACTION, DECRYPT\_ACTION, read\_filename, print\_red  
  
key = b'SuperSecretKey123'  
encrypted\_file = 'encrypted.bin'  
decrypted\_file = 'decrypted.txt'  
  
  
"""Функция для чтения и шифрования открытого текста"""  
def encrypt\_file(symmetric\_key, input\_file, output\_file):  
 global iv  
 try:  
 iv = generate\_iv()  
  
 with open(input\_file, 'rb') as f:  
 plaintext = f.read()  
  
 if not plaintext:  
 raise ValueError("Файл пустой.")  
  
 ciphertext = rc6\_ofb\_encrypt(symmetric\_key, iv, plaintext)  
  
 with open(output\_file, 'wb') as f:  
 f.write(iv + ciphertext)  
  
 print(f"Файл зашифрован и сохранен в: {output\_file}")  
  
 except (ValueError) as e:  
 print\_red(f"Ошибка: {e}")  
 except Exception as e:  
 print\_red(f"Произошла непредвиденная ошибка: {e}")  
  
"""Функция для расшифрования зашифрованного текста и записи результата"""  
def decrypt\_file(symmetric\_key, input\_file, output\_file):  
 global iv  
  
 try:  
 if not os.path.exists(input\_file):  
 raise FileNotFoundError(f"Файл {input\_file} не найден!")  
  
 if os.path.getsize(input\_file) == 0:  
 raise ValueError(f"Файл {input\_file} пустой.")  
  
 with open(input\_file, 'rb') as f:  
 iv = f.read(16)  
 ciphertext = f.read()  
  
 decrypted\_plaintext = rc6\_ofb\_decrypt(symmetric\_key, iv, ciphertext)  
  
 with open(output\_file, 'wb') as f:  
 f.write(decrypted\_plaintext)  
  
 print(f"Файл расшифрован и сохранен в: {output\_file}")  
  
 except (FileNotFoundError, ValueError) as e:  
 print\_red(f"Ошибка: {e}")  
 except Exception as e:  
 print\_red(f"Произошла непредвиденная ошибка: {e}")  
  
"""Функция для генерации вектора инициализации"""  
def generate\_iv():  
 return os.urandom(16)  
  
  
def main():  
 action = decide\_action()  
 if action == ENCRYPT\_ACTION:  
 input\_file = read\_filename()  
 if input\_file is not None:  
 encrypt\_file(key, input\_file, encrypted\_file)  
  
 elif action == DECRYPT\_ACTION:  
 decrypt\_file(key, encrypted\_file, decrypted\_file)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
  
iv = None

with open(input\_file, 'rb') as f:  
 plaintext = f.read()  
  
 if not plaintext:  
 raise ValueError("Файл пустой.")  
  
 ciphertext = rc6\_ofb\_encrypt(symmetric\_key, iv, plaintext)  
  
 with open(output\_file, 'wb') as f:  
 f.write(iv + ciphertext)  
  
 print(f"Файл зашифрован и сохранен в: {output\_file}")  
  
 except (ValueError) as e:  
 print\_red(f"Ошибка: {e}")  
 except Exception as e:  
 print\_red(f"Произошла непредвиденная ошибка: {e}")  
  
"""Функция для расшифрования зашифрованного текста и записи результата"""  
def decrypt\_file(symmetric\_key, input\_file, output\_file):  
 global iv  
  
 try:  
 if not os.path.exists(input\_file):  
 raise FileNotFoundError(f"Файл {input\_file} не найден!")  
  
 if os.path.getsize(input\_file) == 0:  
 raise ValueError(f"Файл {input\_file} пустой.")  
  
 with open(input\_file, 'rb') as f:  
 iv = f.read(16)  
 ciphertext = f.read()  
  
 decrypted\_plaintext = rc6\_ofb\_decrypt(symmetric\_key, iv, ciphertext)  
  
 with open(output\_file, 'wb') as f:  
 f.write(decrypted\_plaintext)  
  
 print(f"Файл расшифрован и сохранен в: {output\_file}")  
  
 except (FileNotFoundError, ValueError) as e:  
 print\_red(f"Ошибка: {e}")  
 except Exception as e:  
 print\_red(f"Произошла непредвиденная ошибка: {e}")  
  
"""Функция для генерации вектора инициализации"""  
def generate\_iv():  
 return os.urandom(16)  
  
  
def main():  
 action = decide\_action()  
 if action == ENCRYPT\_ACTION:  
 input\_file = read\_filename()  
 if input\_file is not None:  
 encrypt\_file(key, input\_file, encrypted\_file)  
  
 elif action == DECRYPT\_ACTION:  
 decrypt\_file(key, encrypted\_file, decrypted\_file)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
  
iv = None

def main():  
 action = decide\_action()  
 if action == ENCRYPT\_ACTION:  
 input\_file = read\_filename()  
 if input\_file is not None:  
 encrypt\_file(key, input\_file, encrypted\_file)  
  
 elif action == DECRYPT\_ACTION:  
 decrypt\_file(key, encrypted\_file, decrypted\_file)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
  
iv = None

# Результаты работы программы

Исходный текст:  
У-у-у-у-у-гу-гуг-гуу! О, гляньте на меня, я погибаю. Вьюга в подворотне ревёт мне отходную, и я вою с ней. Пропал я, пропал. Негодяй в грязном колпаке – повар столовой нормального питания служащих центрального совета народного хозяйства – плеснул кипятком и обварил мне левый бок.  
Какая гадина, а ещё пролетарий. Господи, боже мой – как больно! До костей проело кипяточком. Я теперь вою, вою, да разве воем поможешь.

Шифрование текста:  
A screen shot of a computer

Description automatically generated

Результат шифрования (содержимое файла encrypted.bin):

y ��dǴFW3o�g�F6�����J���@lAX�wD)�\_�r o?j�h/�m |���o�:W(Pn���~P�;3EQ�(�|��n\_�;xf`�V6��  
���0���mdk�‑�\_�UP�������a(ݱƟ�FU� =󂷣�5��d���Uј��<�E����ca���Uì�4��(�1��t��^�Dㇲr��߮��e��^�Lq��@Y��D�R���vԃ�\�vD  
ܤ�Z�0�ojjá�������:�r���\�����t4�q�F5��WtG���� H������������,��T���n���|?�߯�P���gNG�Ul�:�B�~%#�M�Sv�p��{Xe"�mMJ4ȺK�n!�F�:z��!r3{�F�����quܲ�] I ��%����%��6t҆��1���N�zG1B�M'�J���f��4)`=�����.n.A��]��{��m-5��戈b!�ީ��s�����w?������ROL&֙CJO�\Ϳu�  
�R��;xP�]�{�R��3g��2��U�‑˷8^�:�i�.-��  
\_���x�G\_��k�IT���ߝ�����5E����S�\h��j���mv�?\*�J��!�n����ެ�y׻g  
�#�3��֙�}U]��L�!�­�b�Lhq��(|x�‑���������Y����{�|�������yk$�(5vw�����G���n�r+��P��\*8ő�r�#�jsu���ΰP�,��ο��KN�

Дешифрование текста:  
A screen shot of a black background

Description automatically generated

Дешифрованный текст (содержимое файла decrypted.txt):  
У-у-у-у-у-гу-гуг-гуу! О, гляньте на меня, я погибаю. Вьюга в подворотне ревёт мне отходную, и я вою с ней. Пропал я, пропал. Негодяй в грязном колпаке – повар столовой нормального питания служащих центрального совета народного хозяйства – плеснул кипятком и обварил мне левый бок.  
Какая гадина, а ещё пролетарий. Господи, боже мой – как больно! До костей проело кипяточком. Я теперь вою, вою, да разве воем поможешь.