МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)

Факультет информационных технологий
Кафедра «Инфокогнитивные технологии»

Лабораторная работа 3

По дисциплине «Защита информации»
Направление подготовки 09.03.03 «Прикладная информатика»
Профиль «Корпоративные информационные системы»

Выполнил:

студент группы 201-361

Погудин Александр

Москва 2023

Цель работы: написать программу, шифрующую изображение tux.png с помощью шифра AES. Режимы шифрования: ECB, CBC, CFB и OFB. Сравните скорости выполнения алгоритмов и результаты шифрования.

Введение

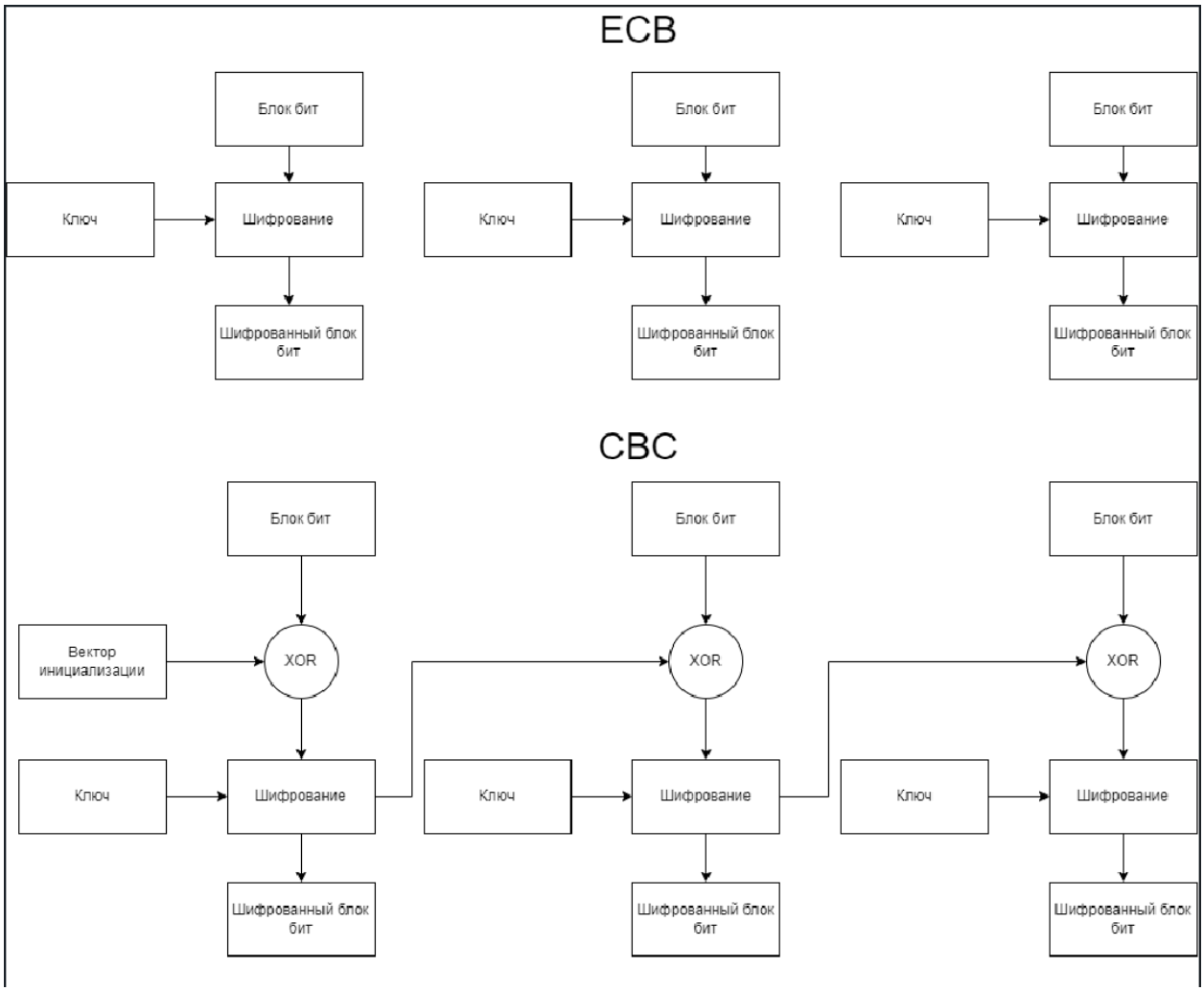
AES (Advanced Encryption Standard) - это блочный алгоритм шифрования симметричного ключа, который может работать в различных режимах шифрования. Режимы шифрования влияют на способ, в котором данные шифруются, и они обеспечивают различные уровни защиты и уникальности выходных данных. Рассмотрим основные режимы шифрования AES:

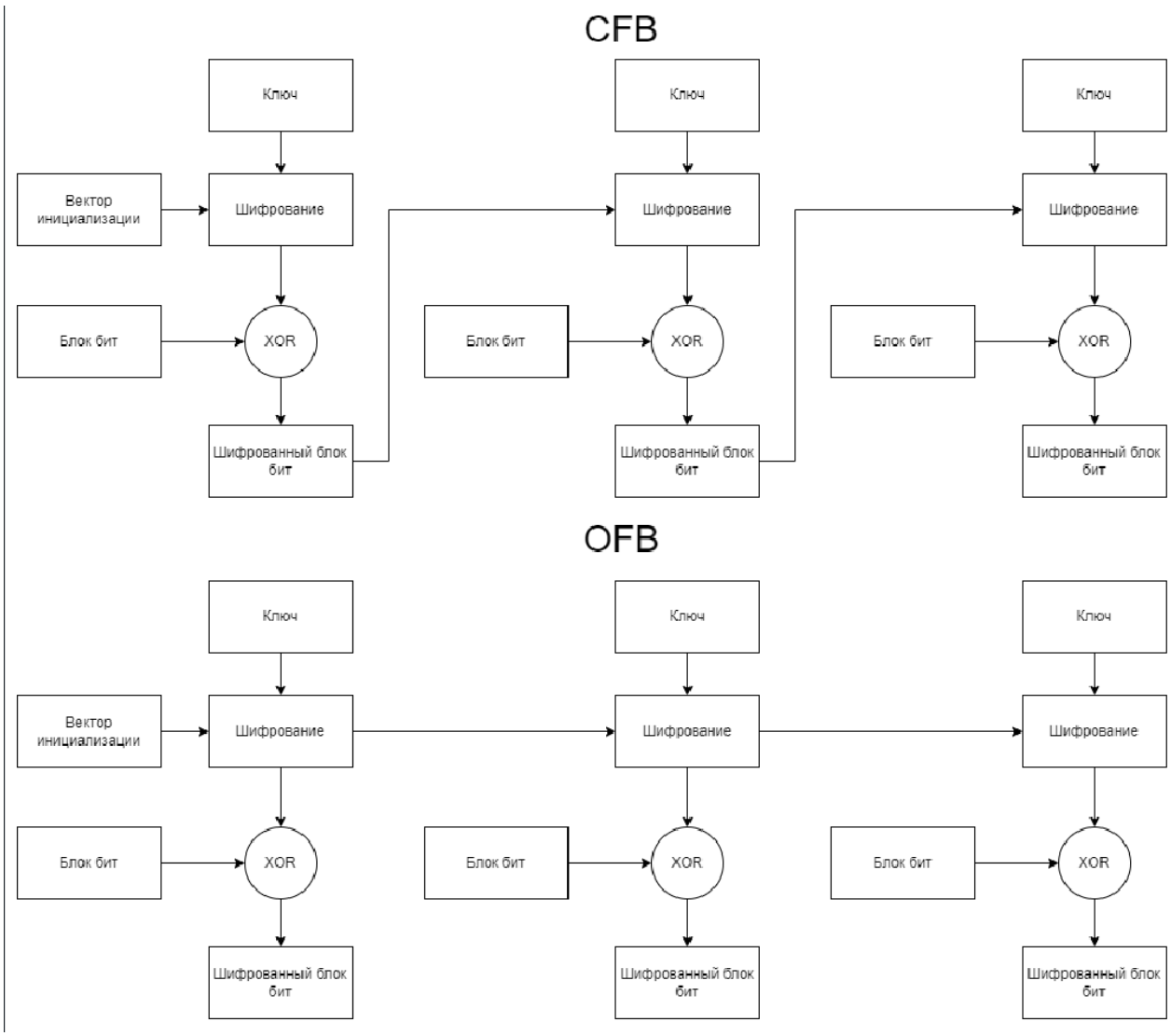
1. ECB (Electronic Codebook) - это самый простой режим шифрования. В этом режиме каждый блок данных шифруется независимо друг от друга с использованием одного и того же ключа. Проблема заключается в том, что если даже два блока данных являются идентичными, то они будут зашифрованы в одно и то же значение, что делает этот режим шифрования уязвимым для атак, например, для атак на частоту появления символов. ECB обычно не рекомендуется для секретности данных.

2. CBC (Cipher-Block Chaining) - в этом режиме каждый блок открытого текста комбинируется с предыдущим блоком зашифрованной информации (вектором инициализации) перед шифрованием. Таким образом, каждый блок открытого текста зависит от всех предыдущих блоков. Это делает CBC намного более безопасным, чем ECB. Однако, вектор инициализации должен быть случайным и непредсказуемым, иначе атакующий может использовать уязвимость.

3. CFB (Cipher Feedback) - в этом режиме шифр преобразует блоки открытого текста в псевдослучайную последовательность один за другим. Затем, игнорируя часть псевдослучайной последовательности, алгоритм шифрования производит операцию XOR с открытым текстом, чтобы создать зашифрованный текст. Такой режим позволяет производить шифрование потока данных произвольной длины.

4. OFB (Output Feedback) - этот режим также преобразует блоки открытого текста в псевдослучайную последовательность, которая никогда не используется для шифрования. Затем, так же как в режиме CFB, выполняется операция XOR с открытым текстом, чтобы создать зашифрованный текст. Разница заключается в том, что вывод псевдослучайной последовательности подается на следующую операцию шифрования в режиме OFB, в то время как в режиме CFB псевдослучайная последовательность выходит из предыдущего зашифрованного блока. OFB предоставляет простой способ шифрования данных потока, поскольку он позволяет работать с любой длиной сообщения.




CFB
Ключ
Ключ
Ключ
Вектор инициализации
Шифрование
Шифрование
Шифрование
Блок бит
XOR
Блок бит
XOR
Блок бит
XOR
Шифрованный блок бит
Шифрованный блок бит
Шифрованный блок бит
OFB
Ключ
Ключ
Ключ
Вектор инициализации
Шифрование
Шифрование
Шифрование
Блок бит
XOR
Блок бит
XOR
Блок бит
XOR
Шифрованный блок бит
Шифрованный блок бит
Шифрованный блок бит

Программа

```
from PIL import Image
from Crypto.Cipher import AES
from Crypto import Random
from datetime import datetime

key_generator = lambda size: Random.new().read(size)

checkImage = lambda image_b: image_b + b"\x00" * (16 - len(image_b) % 16)

RGB = lambda data: tuple(zip([data[i] for i in range(0, len(data)) if i % 3 == 0],
                             [data[i] for i in range(0, len(data)) if i % 3 == 1],
                             [data[i] for i in range(0, len(data)) if i % 3 == 2]))

def create_result_image(result, cipher):
    image_ecb = Image.new(image.mode, image.size)
    image_ecb.putdata(result)
    image_ecb.save(f"tux{cipher}.png", "PNG")

cipher = lambda mode, mode_str: create_result_image(RGB(AES.new(key, mode)
                                .encrypt(checkImage(image_bytes))[:len(image_bytes)]), mode_str)

image = Image.open("tux.png")
image_bytes = image.convert("RGB").tobytes()
key = key_generator(16)

start_time = datetime.now()
cipher(AES.MODE_ECB, "ECB")
print("Шифр ECB выполнен. \nВремя:", datetime.now() - start_time, end = "\n\n")

start_time = datetime.now()
cipher(AES.MODE_CBC, "CBC")
print("Шифр CBC выполнен. \nВремя:", datetime.now() - start_time, end = "\n\n")

start_time = datetime.now()
cipher(AES.MODE_CFB, "CFB")
print("Шифр CFB выполнен. \nВремя:", datetime.now() - start_time, end = "\n\n")

start_time = datetime.now()
cipher(AES.MODE_OFB, "OFB")
print("Шифр OFB выполнен. \nВремя:", datetime.now() - start_time, end = "\n\n")
```

Описание программы

1. Сначала производится импорт необходимых библиотек: PIL (Python Imaging Library) для работы с изображениями, Crypto и Random для генерации ключей шифрования и режима ECB, datetime для работы с датами и временем.

2. Затем объявляется lambda-функция для генерации ключа шифрования, используя модуль Random из Crypto.

3. Объявляется lambda-функция checkImage, которая добавляет символы нуля в конец массива байтов изображения, чтобы гарантировать, что они кратны 16.

4. Объявляется lambda-функция RGB, которая используется для переконвертации массива байтов изображения в список RGB-цветов пикселей.

5. Создается функция create_result_image, которая принимает результат шифрования (result) и режим шифрования (cipher), а затем сохраняет новое изображение в формате PNG в текущей директории.

6. Объявляется lambda-функция cipher, которая принимает режим шифрования (mode) и строку с названием шифрования (mode_str). Она вызывает функцию AES.new, чтобы создать новый экземпляр объекта AES, передав в него ключ шифрования (key) и режим шифрования (mode). Затем она вызывает функцию encrypt, чтобы зашифровать изображение, используя предварительно объявленную функцию checkImage. Результат шифрования нужно перевести в RGB формат и передать в функцию create_result_image вместе с режимом шифрования (mode_str).

7. Производится шифрование изображения разными режимами AES: ECB, CBC, CFB и OFB.

8. Сначала открывается изображение "tux.png" при помощи метода open() из библиотеки PIL.

9. Затем, изображение преобразуется в массив байтов в формате RGB (каждый пиксель изображения представлен тройкой байтов: красный, зеленый и синий) с помощью методов convert() и tobytes().

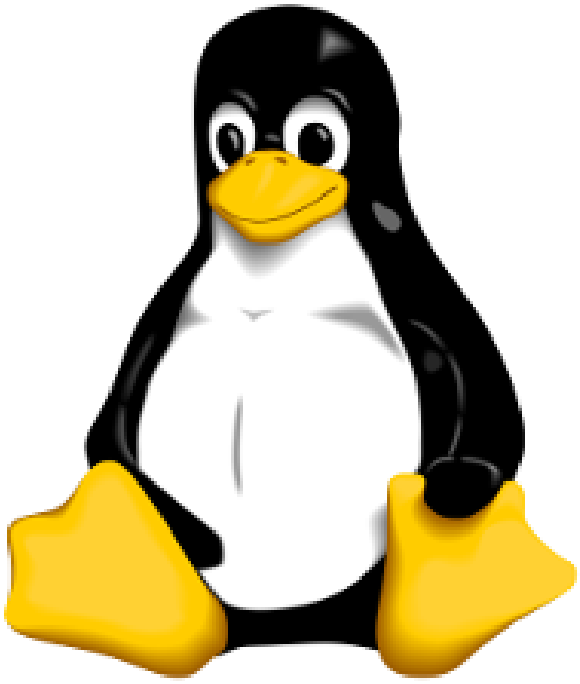
10. Генерируется ключ шифрования длиной 16 байт (128 бит), используя lambda-функцию key_generator, объявленную ранее.

11. Затем вызывается функция cipher(), которая была объявлена ранее, четыре раза с различными режимами шифрования и названиями этих режимов в качестве аргументов.

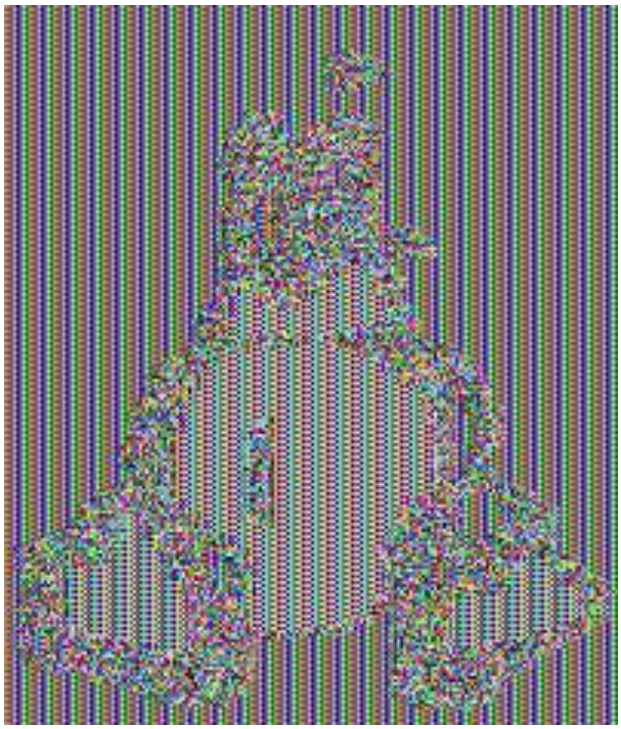
12. После каждого вызова функции cipher(), вычисляется время выполнения кода с помощью datetime.now(), всех вызовов и выводится на экран вместе с названием режима шифрования.

Результат работы программы

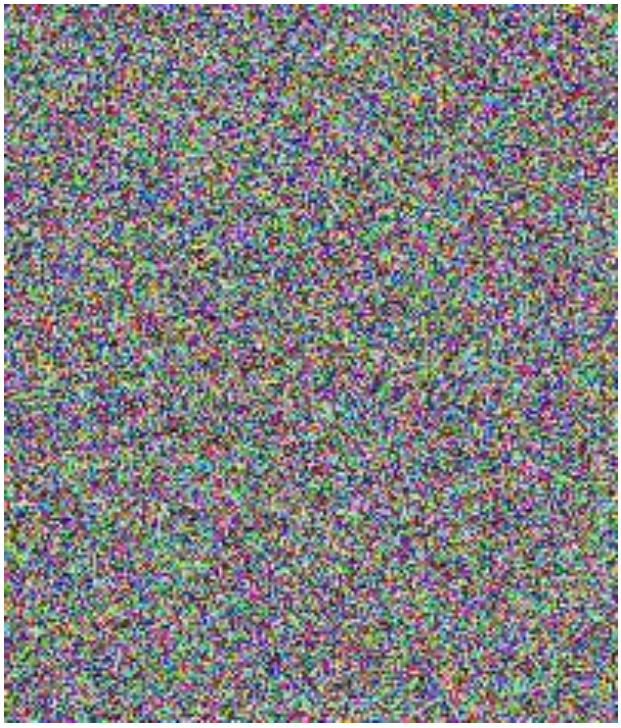
Исходное изображение



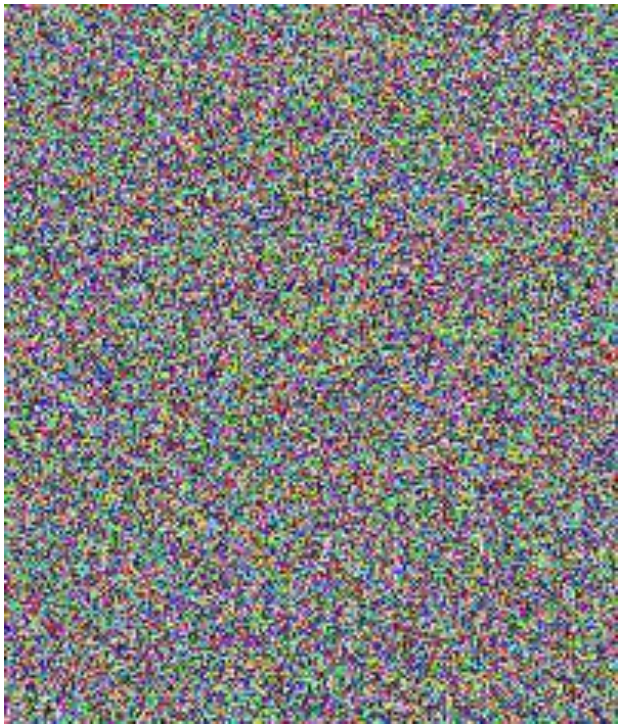
Режим ECB



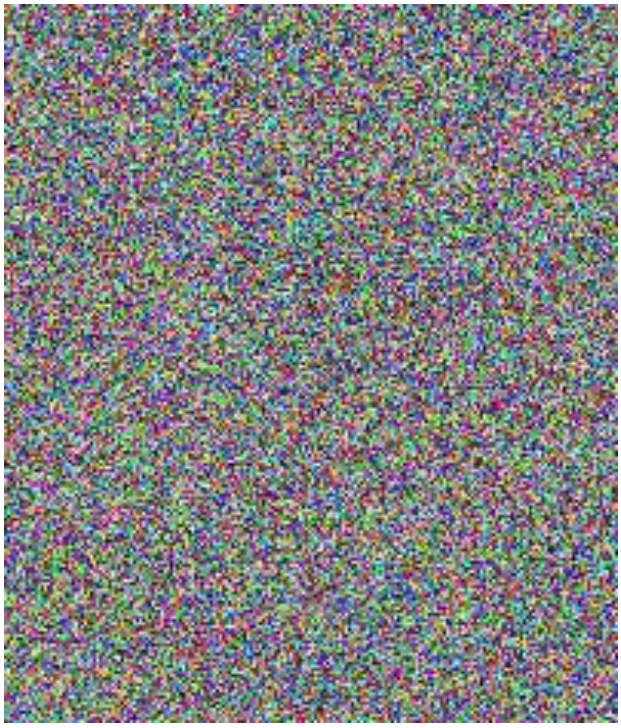
Режим CBC



Режим CFB



Режим OFB



Время шифрования

```
Шифр ECB выполнен.
Время: 0:00:00.031929

Шифр CBC выполнен.
Время: 0:00:00.039376

Шифр CFB выполнен.
Время: 0:00:00.038864

Шифр OFB выполнен.
Время: 0:00:00.034400
```

Вывод

В общем, код использует библиотеки PIL и Crypto, чтобы демонстрировать пример работы с AES в различных режимах шифрования на изображении в формате PNG.

Перед шифрованием изображения используется lambda-функция для генерации ключа шифрования длиной 16 байт, а также определены lambda-функции для обработки данных. Одна из таких функций добавляет нули в конец массива байта изображения, чтобы гарантировать, что они кратны 16.

Затем, производится вызов функции cipher() четыре раза с различными режимами шифрования. После каждого вызова измеряется время выполнения с помощью модуля datetime. Результаты каждого вызова функции выводятся на экран, каждый раз с указанием режима шифрования.

Общий вывод состоит в том, что код представляет собой простой пример использования AES для шифрования изображений в разных режимах. Он может быть использован для демонстрации различных режимов шифрования в AES и для сравнения времени выполнения для каждого режима.