МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по практическому заданию №13**

**по курсу**

**«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»**

Работу выполнил

Студенты 46 группы

Прозоров М.С.

Преподаватель:

Крамаренко А.А.

Краснодар 2024

**Постановка задачи**

Реализовать шифрование и расшифрование AES в режиме CTR.

Принцип работы AES в режиме CTR

Шаг 1: Инициализация

Перед началом шифрования необходимо инициализировать несколько компонентов:

- Ключ шифрования: Секретный ключ, известный только отправителю и получателю. Длина ключа может быть 128, 192, или 256 бит.

- Счетчик (Counter, CTR): Значение счетчика, которое изменяется для каждого блока данных. Обычно начинается с инициализирующего вектора (IV), который может быть открытым, но должен быть уникальным для каждой операции шифрования.

Шаг 2: Генерация ключевого потока

- Шифрование счетчика: для каждого блока данных AES шифрует текущее значение счетчика с использованием заданного ключа шифрования. Результатом шифрования является блок ключевого потока.

- Инкремент счетчика: после шифрования значения счетчика его значение увеличивается (обычно на единицу), чтобы гарантировать уникальность ключевого потока для каждого блока данных.

Шаг 3: Шифрование данных

- XOR данных и ключевого потока: Данные (плейнтекст) XOR`ятся с ключевым потоком, полученным на предыдущем шаге. Результатом этой операции является зашифрованный текст (шифртекст).

Шаг 4: Передача данных

- Зашифрованный текст и IV (или начальное значение счетчика) передаются получателю. IV необходим получателю для генерации того же ключевого потока.

Расшифрование данных

Процесс расшифрования в режиме CTR аналогичен процессу шифрования и выполняется в обратном порядке. Получатель использует тот же ключ шифрования и IV для генерации того же ключевого потока.

Шаг 1: Генерация ключевого потока

Получатель генерирует ключевой поток, шифруя значение счетчика (начиная с IV) с использованием того же ключа шифрования.

Шаг 2: Расшифрование данных

Зашифрованный текст (шифртекст) XOR'ится с ключевым потоком для восстановления исходных данных (плейнтекста).

**Вывод**

Реализовал шифрование и расшифрование AES в режиме CTR.

**Листинг:**

from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes

from cryptography.hazmat.backends import default\_backend

import os

def aes\_ctr\_encrypt(plaintext, key):

# Генерируем случайный инициализирующий вектор (IV)

iv = os.urandom(16)

print(f"IV: {iv}")

# Создаем объект шифра

cipher = Cipher(algorithms.AES(key), modes.CTR(iv), backend=default\_backend())

encryptor = cipher.encryptor()

# Шифруем данные

ciphertext = encryptor.update(plaintext) + encryptor.finalize()

# Возвращаем инициализирующий вектор и шифртекст

return (iv, ciphertext)

def aes\_ctr\_decrypt(iv, ciphertext, key):

# Создаем объект шифра с тем же ключом и IV

cipher = Cipher(algorithms.AES(key), modes.CTR(iv), backend=default\_backend())

decryptor = cipher.decryptor()

# Расшифровываем данные

plaintext = decryptor.update(ciphertext) + decryptor.finalize()

return plaintext

# Пример использования

key = os.urandom(32) # AES-256 требует ключ длиной 32 байта

print(f'key: {key}')

plaintext = b"AES Test"

print(f'Текст: {plaintext}')

iv, ciphertext = aes\_ctr\_encrypt(plaintext, key)

print("Зашифрованный текст:", ciphertext)

decrypted\_text = aes\_ctr\_decrypt(iv, ciphertext, key)

print("Расшифрованный текст:", decrypted\_text)