MainPage/Info Security/Encript Lab 1.2

Университет ИТМО
Факультет ФПИ и КТ

Отчёт по лабораторной работе 2

«Блочное симметричное шифрование»

по дисциплину

«Информационная безопасность»

Студент: Чжоу Хунсян
Группа: P34131
Преподаватель:

Санкт-Петербург 2024

Цель работы

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

Варианты заданий

> Вариант: 23 % 10 = 3

Алгоритм: RC6
Режим шифрования: ECB, CBC, PCBC

Исходный код

Основные параметры для RC6

```
W = 32  # Длина слов
R = 20  # Количества цикл
BLOCK_SIZE = 16  # Размер блоков (Bytes)
```

Операция вращения для шифрования RC6

```
# Операция вращения бита влево
def rotate_left(x, n, w=W):
    return ((x << n) & ((1 << w) - 1)) | (x >> (w - n))


# Операция вращения бита вправо
def rotate_right(x, n, w=W):
    return (x >> n) | ((x << (w - n)) & ((1 << w) - 1))
```

Расширение ключ

```
def key_schedule(key: bytes) -> List[int]:
    P32 = 0xB7E15163
    Q32 = 0x9E3779B9
    S = [P32]
    for i in range(1, 2 * R + 4):
        S.append((S[i - 1] + Q32) & ((1 << W) - 1))

    L = [int.from_bytes(key[i:i + 4], 'little') for i in range(0, len(key), 4)]
    if len(L) * 4 < len(key):
        L.append(0)

    v = max(len(L), len(S))
    A = B = i = j = 0
    for _ in range(3 * v):
        A = S[i] = rotate_left((S[i] + A + B) & ((1 << W) - 1), 3)
        B = L[j] = rotate_left((L[j] + A + B) & ((1 << W) - 1), (A + B) % W)
        i = (i + 1) % len(S)
        j = (j + 1) % len(L)
    return S
```

Шифрование RC6

```
# Шифрование RC6
def rc6_encrypt_block(block: bytes, S: List[int]) -> bytes:
    A, B, C, D = struct.unpack('<IIII', block)
    B = (B + S[0]) & ((1 << W) - 1)
    D = (D + S[1]) & ((1 << W) - 1)
    for i in range(1, R + 1):
        t = rotate_left((B * (2 * B + 1)) & ((1 << W) - 1), 5)
        u = rotate_left((D * (2 * D + 1)) & ((1 << W) - 1), 5)
        A = (rotate_left(A ^ t, u % W) + S[2 * i]) & ((1 << W) - 1)
        C = (rotate_left(C ^ u, t % W) + S[2 * i + 1]) & ((1 << W) - 1)
        A, B, C, D = B, C, D, A
    A = (A + S[2 * R + 2]) & ((1 << W) - 1)
    C = (C + S[2 * R + 3]) & ((1 << W) - 1)
    return struct.pack('<IIII', A, B, C, D)
```

Дешифрование RC6

```
def rc6_decrypt_block(block: bytes, S: List[int]) -> bytes:
    A, B, C, D = struct.unpack('<IIII', block)
    C = (C - S[2 * R + 3]) & ((1 << W) - 1)
    A = (A - S[2 * R + 2]) & ((1 << W) - 1)
    for i in range(R, 0, -1):
        A, B, C, D = D, A, B, C
        u = rotate_left((D * (2 * D + 1)) & ((1 << W) - 1), 5)
        t = rotate_left((B * (2 * B + 1)) & ((1 << W) - 1), 5)
        C = rotate_right((C - S[2 * i + 1]) & ((1 << W) - 1), t % W) ^ u
        A = rotate_right((A - S[2 * i]) & ((1 << W) - 1), u % W) ^ t
    D = (D - S[1]) & ((1 << W) - 1)
    B = (B - S[0]) & ((1 << W) - 1)
    return struct.pack('<IIII', A, B, C, D)
```

Заполнение данных

Алгоритм блочного симметричного шифрования требует, чтобы длина данных открытого текста в байтах была целым кратным размеру блока, поэтому мы должны дополнять данные открытого текста перед шифрованием данных открытого текста.

Здесь мы применяем алгоритм PKCS#7

```
def pad(data: bytes, block_size: int) -> bytes:
    padding_len = block_size - len(data) % block_size
    return data + bytes([padding_len] * padding_len)


# Удаление заполнения данных
def unpad(data: bytes) -> bytes:
    return data[:-data[-1]]
```

Режим：ECB

```
def encrypt_ecb(data: bytes, key: bytes) -> bytes:
    S = key_schedule(key)
    data = pad(data, BLOCK_SIZE)
    encrypted = b""
    for i in range(0, len(data), BLOCK_SIZE):
        encrypted += rc6_encrypt_block(data[i:i + BLOCK_SIZE], S)
    return encrypted


def decrypt_ecb(data: bytes, key: bytes) -> bytes:
    S = key_schedule(key)
    decrypted = b""
    for i in range(0, len(data), BLOCK_SIZE):
        decrypted += rc6_decrypt_block(data[i:i + BLOCK_SIZE], S)
    return unpad(decrypted)
```

Режим：CBC

```
def encrypt_cbc(data: bytes, key: bytes, iv: bytes) -> bytes:
    S = key_schedule(key)
    data = pad(data, BLOCK_SIZE)
    encrypted = b""
    prev = iv
    for i in range(0, len(data), BLOCK_SIZE):
        block = bytes(a ^ b for a, b in zip(data[i:i + BLOCK_SIZE], prev))
        encrypted_block = rc6_encrypt_block(block, S)
        encrypted += encrypted_block
        prev = encrypted_block
    return encrypted


def decrypt_cbc(data: bytes, key: bytes, iv: bytes) -> bytes:
    S = key_schedule(key)
    decrypted = b""
    prev = iv
    for i in range(0, len(data), BLOCK_SIZE):
        block = rc6_decrypt_block(data[i:i + BLOCK_SIZE], S)
        decrypted += bytes(a ^ b for a, b in zip(block, prev))
        prev = data[i:i + BLOCK_SIZE]
    return unpad(decrypted)
```

Режим : PCBC

```
def encrypt_pcbc(data: bytes, key: bytes, iv: bytes) -> bytes:
    S = key_schedule(key)
    data = pad(data, BLOCK_SIZE)
    encrypted = b""
    prev_enc = iv
    prev_plain = iv
    for i in range(0, len(data), BLOCK_SIZE):
        block = bytes(a ^ b ^ c for a, b, c in zip(data[i:i + BLOCK_SIZE], prev_plain, pre
        encrypted_block = rc6_encrypt_block(block, S)
        encrypted += encrypted_block
        prev_enc, prev_plain = encrypted_block, data[i:i + BLOCK_SIZE]
    return encrypted


def decrypt_pcbc(data: bytes, key: bytes, iv: bytes) -> bytes:
    S = key_schedule(key)
    decrypted = b""
    prev_enc = iv
    prev_plain = iv
    for i in range(0, len(data), BLOCK_SIZE):
        block = rc6_decrypt_block(data[i:i + BLOCK_SIZE], S)
        decrypted_block = bytes(a ^ b ^ c for a, b, c in zip(block, prev_plain, prev_enc))
        decrypted += decrypted_block
        prev_enc, prev_plain = data[i:i + BLOCK_SIZE], decrypted_block
    return unpad(decrypted)


# Чтение файла
def read_file(filename):
    with open(filename, 'r', encoding='UTF-8') as file:
        return file.read()


# Запись файла
def write_file(filename, content):
    with open(filename, 'w', encoding='UTF-8') as file:
        file.write(content)
```

```
# Основная функция
if __name__ == "__main__":
    key_input = input("Please input keyword: ")
    # key = b"thisisakey123456" # 16 bytes Ключ
    key = key_input.encode('utf-8')
    iv = b"initialvector1234" # 16 bytes Вектор
    text = read_file("plaintext.txt")
    plaintext = text.encode('utf-8')
    print("PlainText:\n", plaintext.decode('utf-8'))

    print()

    ecb_encrypted = encrypt_ecb(plaintext, key)
    print("ECB Encrypt:\n", ecb_encrypted.hex())
    print("ECB Decrypt:\n", decrypt_ecb(ecb_encrypted, key).decode('utf-8'))

    print()

    cbc_encrypted = encrypt_cbc(plaintext, key, iv)
    print("CBC Encrypt:\n", cbc_encrypted.hex())
    print("CBC Decrypt:\n", decrypt_cbc(cbc_encrypted, key, iv).decode('utf-8'))

    print()

    pcbc_encrypted = encrypt_pcbc(plaintext, key, iv)
    print("PCBC Encrypt:\n", pcbc_encrypted.hex())
    print("PCBC Decrypt:\n", decrypt_pcbc(pcbc_encrypted, key, iv).decode('utf-8'))
```

Результаты работы программы

Пример Usage

```
"/Users/2398768715qq.com/GitHub/ITMO-Labs/InfoSecurity/Lab1.2/Lab2 RC6/.venv/bin/python" /
Please input keyword: КЛЮЧ
PlainText:
 В рамках нашей работы мы определили следующую методологическую базу:
Сначала мы провели тщательный анализ существующих решений по дизайну интерфейса. Это включ
На основе результатов анализа мы разработали теоретическую модель. Эта модель включала клю
С помощью разработанной модели мы приступили к экспериментальному проектированию. Был созд
Заключительным этапом было тестирование созданного прототипа. Мы провели серию тестов с уч

ECB Encrypt:
 0d03d15fe59e7b4752bbfff82e91d052b4f54fc423ebec1eeb8f02d059346ec3531753926d45770d14d834d3b
ECB Decrypt:
 В рамках нашей работы мы определили следующую методологическую базу:
Сначала мы провели тщательный анализ существующих решений по дизайну интерфейса. Это включ
На основе результатов анализа мы разработали теоретическую модель. Эта модель включала клю
С помощью разработанной модели мы приступили к экспериментальному проектированию. Был созд
Заключительным этапом было тестирование созданного прототипа. Мы провели серию тестов с уч

CBC Encrypt:
 94a44009bfbe5237f776604d37d120c860f3210957f6f0fbb313e5900690ddc5bf5105123aa6ebd935769567f
CBC Decrypt:
 В рамках нашей работы мы определили следующую методологическую базу:
Сначала мы провели тщательный анализ существующих решений по дизайну интерфейса. Это включ
На основе результатов анализа мы разработали теоретическую модель. Эта модель включала клю
С помощью разработанной модели мы приступили к экспериментальному проектированию. Был созд
Заключительным этапом было тестирование созданного прототипа. Мы провели серию тестов с уч

PCBC Encrypt:
 0d03d15fe59e7b4752bbfff82e91d05216ab0d967053b9b5166e6321b9be3ce143118fc8717b708399d5b132c
PCBC Decrypt:
 В рамках нашей работы мы определили следующую методологическую базу:
Сначала мы провели тщательный анализ существующих решений по дизайну интерфейса. Это включ
На основе результатов анализа мы разработали теоретическую модель. Эта модель включала клю
С помощью разработанной модели мы приступили к экспериментальному проектированию. Был созд
Заключительным этапом было тестирование созданного прототипа. Мы провели серию тестов с уч

Process finished with exit code 0
```

Вывод

В ходу лабораторной работы, я выучил несколько подходы для щифрования и дешифрования текст в файле.