Криптографические системы
Лабораторная работа №2
Блочное симметричное шифрование

ФИО студента: Готовко Алексей Владимирович
Вариант: 3
Учебная группа: P34101

1 Цель работы

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

2 Вариант задания

Реализовать систему симметричного блочного шифрования, позволяющую шифровать и дешифровать файл на диске с использованием блочного шифра `RC6` в режиме шифрования `PCBC`.

3 Исходный код класса-шифратора

`RC6Encryptor.py`

```python
import struct
from math import e, log2
from random import Random

from utils import round_to_odd, rotate_bits_left, rotate_bits_right


class RC6Encryptor:
    _word_size = 32  # word size in bits
    _block_size = 16  # pcbc block size in bytes
    _mask = 2 ** _word_size - 1
    _lg_w = int(log2(_word_size))
    _round_cnt = 20
    _key_size = 128  # key size in bytes

    # Magic constants for round keys scheduling
    _f = 1.6180339887498948482  # golden ratio
    _q = round_to_odd((_f - 1) * 2 ** _word_size)
    _p = round_to_odd((e - 2) * 2 ** _word_size)

    def __init__(self, keyword: str):
        self._rng = Random(keyword)
        self._initial_key_gen()
        self._round_keys_gen()
        self._initial_vector_gen()

    def _initial_key_gen(self) -> None:
        self._initial_key = bytearray(self._rng.getrandbits(8) for _ in
        ↪  range(self._key_size))

    def _round_keys_gen(self) -> None:
        self._init_key_parts = [int.from_bytes(self._initial_key[i:i + 1],
        ↪  byteorder="little") for i in
                                range(self._key_size)]
        c = len(self._init_key_parts)
        self._round_keys = [0] * (2 * self._round_cnt + 4)

        self._round_keys[0] = self._p
        for i in range(1, 2 * self._round_cnt + 3):
            self._round_keys[i] = self._round_keys[i - 1] + self._q

```

```
        a = b = i = j = 0
        v = 3 * max(c, 2 * self._round_cnt + 4)
        for _ in range(1, v):
            a = self._round_keys[i] = rotate_bits_left(self._round_keys[i] + a + b, 3,
            ↪  self._word_size)
            b = self._init_key_parts[j] = rotate_bits_left(self._init_key_parts[j] + a + b,
            ↪  a + b, self._word_size)
            i = (i + 1) % (2 * self._round_cnt + 4)
            j = (j + 1) % c

    def _initial_vector_gen(self) -> None:
        self._initial_vector = bytes(bytearray(self._rng.getrandbits(8) for _ in
        ↪  range(self._block_size)))

    def _encrypt_block(self, plain_data: bytes) -> bytes:
        a, b, c, d = struct.unpack("<4L", plain_data)
        b = (b + self._round_keys[0]) & self._mask
        d = (d + self._round_keys[1]) & self._mask
        for i in range(self._round_cnt):
            t = rotate_bits_left(b * (2 * b + 1), self._lg_w, self._word_size) & self._mask
            u = rotate_bits_left(d * (2 * d + 1), self._lg_w, self._word_size) & self._mask
            a = (rotate_bits_left(a ^ t, u, self._word_size) + self._round_keys[2 * i + 2])
            ↪  & self._mask
            c = (rotate_bits_left(c ^ u, t, self._word_size) + self._round_keys[2 * i + 3])
            ↪  & self._mask
            a, b, c, d = b, c, d, a
        a = (a + self._round_keys[2 * self._round_cnt + 2]) & self._mask
        c = (c + self._round_keys[2 * self._round_cnt + 3]) & self._mask
        return struct.pack("<4L", a, b, c, d)

    def _decrypt_block(self, encrypted_data: bytes) -> bytes:
        a, b, c, d = struct.unpack("<4L", encrypted_data)
        c = (c - self._round_keys[2 * self._round_cnt + 3]) & self._mask
        a = (a - self._round_keys[2 * self._round_cnt + 2]) & self._mask
        for i in range(self._round_cnt - 1, -1, -1):
            a, b, c, d = d, a, b, c
            u = rotate_bits_left(d * (2 * d + 1), self._lg_w, self._word_size) & self._mask
            t = rotate_bits_left(b * (2 * b + 1), self._lg_w, self._word_size) & self._mask
            c = rotate_bits_right((c - self._round_keys[2 * i + 3]) & self._mask, t,
            ↪  self._word_size) ^ u
            a = rotate_bits_right((a - self._round_keys[2 * i + 2]) & self._mask, u,
            ↪  self._word_size) ^ t
        d = (d - self._round_keys[1]) & self._mask
        b = (b - self._round_keys[0]) & self._mask
        return struct.pack("<4L", a, b, c, d)

    def pcbc_encrypt(self, plain_data: bytes) -> bytes:
        prev_encrypted_block = self._initial_vector
        encrypted_data = b''

        for i in range(0, len(plain_data), self._block_size):
            block = plain_data[i:i + self._block_size]
            if len(block) < self._block_size:
                block = block.ljust(self._block_size, b'\0')
            block_to_encrypt = bytes([x ^ y for x, y in zip(block, prev_encrypted_block)])
```

```
            encrypted_block = self._encrypt_block(block_to_encrypt)
            prev_encrypted_block = bytes([x ^ y for x, y in zip(block, encrypted_block)])
            encrypted_data += encrypted_block
        return encrypted_data

    def pcbc_decrypt(self, encrypted_data) -> bytes:
        prev_encrypted_block = self._initial_vector
        decrypted_data = b''

        for i in range(0, len(encrypted_data), self._block_size):
            block = encrypted_data[i:i + self._block_size]
            decrypted_block = self._decrypt_block(block)
            plaintext_block = bytes([x ^ y for x, y in zip(decrypted_block,
            ↪  prev_encrypted_block)])
            prev_encrypted_block = bytes([x ^ y for x, y in zip(plaintext_block, block)])
            decrypted_data += plaintext_block
        return decrypted_data.rstrip(b'\0')
```

4 Результат работы программы

4.1 stdout программы

```
Supply keyword: goofy ahh infosec lab
Provide file name: test_text.txt
Successfully encrypted data. Saved to: out/rc6_pcbc_encrypted.bin
Successfully decrypted data. Saved to: out/rc6_pcbc_decrypted.txt

Process finished with exit code 0
```

4.2 Исходный текст

```
xgodness@xgodness-pc:~/itmo/4-year/information-security$ cat test_text.txt
Ребята, не стоит вскрывать эту тему. Вы молодые, шутливые, вам всё легко. Это не то.
Это не микросервисы на спринге и даже не аннотации Балакшина.
Сюда лучше не лезть. Серьёзно, любой из вас будет жалеть.
Лучше закройте тему и забудьте, что тут писалось.
Я вполне понимаю, что данным сообщением вызову дополнительный интерес,
но хочу сразу предостеречь пытливых - стоп. Остальные просто не найдут.
```

4.3 Расшифрованный текст

```
xgodness@xgodness-pc:~/itmo/4-year/information-security$ cat out/rc6_pcbc_decrypted.txt
Ребята, не стоит вскрывать эту тему. Вы молодые, шутливые, вам всё легко. Это не то.
Это не микросервисы на спринге и даже не аннотации Балакшина.
Сюда лучше не лезть. Серьёзно, любой из вас будет жалеть.
Лучше закройте тему и забудьте, что тут писалось.
Я вполне понимаю, что данным сообщением вызову дополнительный интерес,
но хочу сразу предостеречь пытливых - стоп. Остальные просто не найдут.
```