Отчет к домашнему заданию 2 по дисциплине криптография.

Выполнил Хаукка С.И.

Задание 1
Файл 1.txt зашифрован с помощью AES-128 в режиме ECB на ключе `YELLOW SUBMARINE` и закодирован в base64. Примечание: буквы ключа заглавные, длина ровно 16 символов (байт) - замечательный ключ для AES-128.

Дешифруйте файл. В конце концов у вас есть ключ. Проще всего использовать OpenSSL::Cipher в режиме AES-128-ECB, но это не наш путь. Мы должны реализовать режим ECB сами, это пригодится нам в дальнейшем.

Хорошая новость в том, что для этого не нужно писать AES-128 с нуля. Мы сделаем AES-128 из подручных средств. На Python функция дешифрования будут выглядеть примерно так:

```
def aes128_decrypt(block, key):
    if len(block) != 16:
        return None

    cipher = AES.new(key, AES.MODE_ECB)
    return cipher.decrypt(block)
```

http://cryptopals.com/sets/1/challenges/7

Решение:

```
def ECB(text=bytes, key=bytes):
    blocks = [text[i:i + len(key)] for i in range(0, len(text), len(key))]
    text = b''
    for bl in blocks:
        text += aes128_decrypt(bl, key)
    pkcs = text[len(text) - 1]
    if text[len(text) - pkcs:] == bytes([pkcs]) * pkcs:
        text = text[:len(text) - pkcs]
    return text


def task1():
    f = open('1.txt', 'r')
    text = f.read()
    print(ECB(base64.b64decode(text), 'YELLOW SUBMARINE'.encode()).decode('ascii'))
```

I'm back and I'm ringin' the bell

A rockin' on the mike while the fly girls yell

In ecstasy in the back of me

Well that's my DJ Deshay cuttin' all them Z's

Hittin' hard and the girlies goin' crazy

Vanilla's on the mike, man I'm not lazy.

…

Задание 2
В файле 2.txt находится несколько шифротекстов. Один из них был зашифрован в режиме ECB. Найдите его.

Помните, в чем основная проблема режима ECB? Одинаковые 16 байт открытого текста дают одинаковые 16 байт шифротекста.

http://cryptopals.com/sets/1/challenges/8

Решение:

```
def task2():
    f = open('2.txt', 'r')
    for num, text in enumerate(f.readlines()):
        blocks = [text[i:i + 16 * 2] for i in range(0, len(text), 16 * 2)]
        [print(num, i, blocks.count(bl)) for i, bl in enumerate(blocks) if blocks.count(bl) != 1]
```

```
132 1 4
132 3 4
132 5 4
132 7 4
```

Искомый шифротекст находится в 132 строке, в нем блоки 1,3,5,7 совпадают.

Задание 3
Реализуйте PKCS#7 паддинг, который будет дополнять блок до заданной длины. У вас должна получиться функция `pkcs7_padding(block, target_length)`.

Для примера, `pkcs7_padding(“YELLOW SUBMARINE”, 20)` вернет `YELLOW SUBMARINE\x04\x04\x04\x04`.

http://cryptopals.com/sets/2/challenges/9

Решение:

```
def pkcs7_padding(block=bytes, target_length=int):
    dif = target_length - len(block)
    for x in range(0, dif):
        block += bytes([dif])
    return block


def task3():
    print(pkcs7_padding('YELLOW SUBMARINE'.encode(), 20))
```

```
b'YELLOW SUBMARINE\x04\x04\x04\x04'
```

Задание 4
Реализуйте функции шифрования и дешифрования AES-128 в режиме CBC (используйте код из задания 1). Дешифруйте файл 4.txt с помощью ключа `YELLOW SUBMARINE` и вектора инициализации, состоящего из нулей `\x00\x00\x00...`.

http://cryptopals.com/sets/2/challenges/10

Решение:

```
def CBC_enc(plaintext=bytes, iv=bytes, key=bytes):
    ct = b''
    for i in range(0, len(plaintext), len(key)):
        bl = plaintext[i:i + len(key)]
        bl = bytes([(b ^ i) for b, i in zip(bytes(bl), bytes(iv))])
        iv = aes128_decrypt(bl, key)
        ct += iv
    return ct


def CBC_dec(ciphertext=bytes, iv=bytes, key=bytes):
    pt = b''
    for i in range(0, len(ciphertext), len(key)):
        bl = ciphertext[i:i + len(key)]
        bl_cp = aes128_decrypt(bl, key)
        pt += bytes([(b ^ i) for b, i in zip(bytes(bl_cp), bytes(iv))])
        iv = bl
    pkcs = pt[len(pt) - 1]
    if pt[len(pt) - pkcs:] == bytes([pkcs]) * pkcs:
        pt = pt[:len(pt) - pkcs]
    return pt.decode()


def task4():
    f = open('4.txt', 'r')
    key = 'YELLOW SUBMARINE'
    text = 'we are all in the'.encode()
    print(CBC_enc(pkcs7_padding(text, (len(text) // len(key) + 1) * len(key)), b'\x00' * len(key),
key.encode()))
    print(CBC_dec(base64.b64decode(f.read()), b'\x00' * len(key), key.encode()))
```

I'm back and I'm ringin' the bell
A rockin' on the mike while the fly girls yell
In ecstasy in the back of me
Well that's my DJ Deshay cuttin' all them Z's
Hittin' hard and the girlies goin' crazy
Vanilla's on the mike, man I'm not lazy.

Задание 5
К этому моменту у вас должны быть готовы ECB и CBC режимы AES.

Напишите функцию, которая генерирует случайный ключ (16 байт из /dev/urandom).

Напишите функцию, которая берет случайный ключ и шифрует с его помощью открытый текст. Функция будет выглядеть как `encryption_oracle(your-input)` и возвращать шифротекст.

Также функция-оракул должна присоединять 5-10 (число выбирается случайно) рандомных байт перед открытым текстом и 5-10 рандомных байт после открытого текста.

Пусть функция-оракул в половине случаев шифрует в режиме ECB, а в другой половине случаев в режиме CBC (режим выбирается случайным образом).

Вам нужно написать программу, которая примет на вход шифротекст и будет способна определить какой из режимов шифрования был использован (ECB или CBC). Примечание: вы можете подавать на вход функции-оракула открытый текст произвольной длины.

http://cryptopals.com/sets/2/challenges/11

Решение:

```
def task5():
    def encryption_oracle(plaintext):
        def generate_key(length):
            return bytes([random.randint(0, 255) for _ in range(length)])

        BLOCK_SIZE = 16
        plaintext = generate_key(random.randint(5, 10)) + plaintext + generate_key(random.randint(5,
10))
        plaintext = pkcs7_padding(plaintext, (len(plaintext) // BLOCK_SIZE + 1) * BLOCK_SIZE)
        if random.randint(1, 2) == 1:
            return (ECB(plaintext, generate_key(BLOCK_SIZE)))
        else:
            return (CBC_enc(plaintext, b'\x00' * BLOCK_SIZE, generate_key(BLOCK_SIZE)))

    res_len = [len(encryption_oracle(('A').encode())) for i in range(10)]
    key_len = math.gcd(min(res_len), max(res_len))
    for k in range(100):
        flag = False
        res = encryption_oracle(('A' * 50).encode())
        for j in range(0, len(res), key_len):
            for l in range(j + key_len, len(res), key_len):
                if res[j:j + key_len] == res[l:l + key_len]:
                    flag = True
                    break
            if flag: break
        if flag:
            print('ECB')
        else:
            print('CBC')
```

ECB:
b']\xbdL\xc8N\xa9\x96\x8f\x98\xd8D\x16\x10&\x88\xda\xc5f\xbfD\x8ezjk\x05n\xa0O8\x1fn\xe2\xc5f\xbf
D\x8ezjk\x05n\xa0O8\x1fn\xe2\x17\r\x89\x8ev\x85\x94X0\x0c%\x8f\xb7\x9a`*'

CBC:
b'\xc9%\x9b;&\x04\r\xc0Z\xfc\x8fL\x10\xdb\xff,\xf4\xd6\x9d\xc8S\xae\x10\xda\x0b\x84\xa93\xe4.^\xde,\
x9d\xd5\x00\t\x1e\x87[T\xe8\xde\x9c\x0e^Z"!\x99)\x96\x06\x07\\\x80\x99w\nptl\xd3\x8d\x06S\x1aO\xd
4\xc4\xb5\xe3\x97\x19\xc3${\\\x87u'

…

Задание 6
Модифицируйте функцию `encryption_oracle` из задания 5 так, чтобы она шифровала только в режиме ECB на случайном ключе, который остается одинаковым в пределах запуска программы (например, сделайте глобальную переменную KEY и берите значение из os.urandom).

Функция будет добавлять к открытому тексту base64-декодированное значение (это нужно сделать до шифрования):

```
Um9sbGluJyBpbiBteSA1LjAKV2l0aCBteSByYWctdG9wIGRvd24gc28gbXkg

aGFpciBjYW4gYmxvdwpUaGUgZ2lybGllcyBvbiBzdGFuZGJ5IHdhdmluZyBq

dXN0IHRvIHNheSBoaQpEaWQgeW91IHN0b3A/IE5vLCBJIGp1c3QgZHJvdmUg

YnkK
```

Внимание! Не декодируйте это значение. Суть задания в том, что вы не знаете что внутри base64.

В итоге ваша функция будет возвращать значение:

```
AES-128-ECB(ваша-строка || неизвестная-строка, случайный-ключ)
```

В такой схеме вы можете восстановить содержимое неизвестной строки, сделав несколько запросов к функции-оракулу! Алгоритм выглядит примерно так:

Шаг 1. Узнайте размер блока (вы уже его знаете, но все равно выполните этот шаг). Для этого подавайте на вход строки из одинаковых байт, каждый раз добавляя по одному байте: “A”, “AA”, “AAA” и так далее. Подумайте о том, в какой момент вы сможете точно определить длину блока.

Шаг 2. Поймите, что функция использует ECB режим шифрования. Вам это уже известно, но все равно выполните этот шаг.

Шаг 3. Создайте блок данных, длина которого в точности на единицу меньше длины блока (например, если длина блока 8, то блок данных будет “AAAAAAA”). Задайтесь вопросом: что функция шифрования поставит на позицию последнего байта?

Шаг 4. Подавайте на вход функции-оракула все возможные значения последнего байта (“AAAAAAAA”, “AAAAAAAB”, “AAAAAAAC” и так далее). Запомните первый блок каждого получившегося шифротекста.

Шаг 5. Возьмите блок шифротекста из шага 3 и найдите его в списке из шага 4. Теперь вы знаете первый байт неизвестной строки.

Шаг 6. Повторите алгоритм для второго и последующих байт.

http://cryptopals.com/sets/2/challenges/12

Решение:

```
def task6():
    def generate_key(length):
        return bytes([random.randint(0, 255) for _ in range(length)])

    def encryption_oracle(plaintext):
        b64str =
b'Um9sbGluJyBpbiBteSA1LjAKV2l0aCBteSByYWctdG9wIGRvd24gc28gbXkgaGFpciBjYW4gYmxvdwpUaGUgZ2lybGllcyBvbiBzdGFuZGJ5IHdhdmluZyBqdXN0IHRvIHNheSBoaQpEaWQgeW91IHN0b3A/IE5vLCBJIGp1c3QgZHJvdmUgYnkK'
        plaintext = (plaintext + b64str)
        return (ECB(pkcs7_padding(plaintext, (len(plaintext) // BLOCK_SIZE + 1) * BLOCK_SIZE), key))

    BLOCK_SIZE = 16
    key = generate_key(BLOCK_SIZE)
```

```
    key_len = len(encryption_oracle('A'.encode()))
    i = 2
    while True:
        k = len(encryption_oracle(('A' * i).encode()))
        if k != key_len:
            key_len = k - key_len
            break
        i += 1

    text, k = '', 0
    while True:
        try:
            for j in range(0, key_len):
                r = [
                    encryption_oracle(('A' * (key_len - 1 - j) + text + chr(i)).encode())[k * key_len:(k + 1) *
key_len]
                    for i in range(0, 127)]
                text += chr(
                    r.index(encryption_oracle(('A' * (key_len - j - 1)).encode())[k * key_len:(k + 1) * key_len]))
        except Exception as err:
            print(base64.b64decode(text).decode('ascii'))
            break
        k += 1
```

Rollin' in my 5.0

With my rag-top down so my hair can blow

The girlies on standby waving just to say hi

Did you stop? No, I just drove by