Homework Week2

20199102-Sun Qilong

March 6, 2020

Problem 1

对 AES128 加密算法,Padding 格式为末尾 b 个字节 b, 0≤ b ≤ 16, 写出 Padding Oracle 攻击算法并编程实验验证.

1. (建议) 阅读论文 (超星平台资料目录下载): Serge Vaudenay,” Security Flaws Induced by CBC Padding-Applications to SSL,IPSEC, WTL-”
2. 分析 Padding Oracle 攻击 (RC5, 密文长度为 l 个分组) 的时间复杂度 (使用的加解密运算的次数).
3. 借助网络找到 AES128 的加解密代码 (合作完成).
4. 编写用 Padding Oracle 方法攻击 AES128(Padding 格式：最后一个分组一定含 Padding 数据并以 b 个 b 结尾,1 ≤ b ≤ 16)
5. 验证攻击程序的正确性和时间消耗 (可两人相互验证对方程序).

answer

1) 通过阅读论文基本掌握 CBC 模式下通过 PKCS#7 进行 padding 的 padding-oracle 攻击方式
2) 考虑长度为 n、采用 PKCS#7 进行 padding、AES-CBC 模式生成的密文，padding 的长度 b 应满足 $(0 \leq b \leq 16)$ 且期望为 8，则非 padding 部分的长度期望为 $n-8$。对于非 padding 部分的密文的每一位需要遍历 $[0, 255]$ 的空间进行暴力破解。故，平均复杂度可以表示为 $256(n-8)+8$，时间复杂度为 $O(256n)$
3) 使用 python 的 Crypto.Cipher 包完成 AES 加解密运算。

```python
from Crypto.Cipher import AES
from binascii import a2b_hex, b2a_hex
import time


def add_to_16_by_number(text):
    add = 16 - (len(text.encode('utf-8')) % 16)
    text = text.encode('utf-8')
    text = text + add.to_bytes(length=1, byteorder='big', signed=False) * add
    return text


# 加密函数
def encrypt(iv, text):
    key = '9999999999999999'.encode('utf-8')
    mode = AES.MODE_CBC
    text = add_to_16_by_number(text)
    cryptos = AES.new(key, mode, iv)
    cipher_text = cryptos.encrypt(text)
    return cipher_text


# 解密后，去掉补足的空格用strip() 去掉
def decrypt(iv, text):
```

```python
    key = '9999999999999999'.encode('utf-8')
    mode = AES.MODE_CBC
    cryptos = AES.new(key, mode, iv)
    plain_text = cryptos.decrypt(text)
    return plain_text


def check(iv, string):
    text = decrypt(iv, string)
    number = text[-1]
    if number > 16 or number == 0:
    return text, False

    flag = True
    for i in range(number * -1, -1):
    if text[i] != number:
    flag = False
    return text, flag
```

4) padding-oracle 攻击的代码如下

```python
import CBC
import time

def findB(iv, c1, c2):
    for i in range(16):
        c1_change = c1[:i] + (c1[i] ^ 1).to_bytes(length=1, byteorder='big', signed=False) + c1[i + 1:]
        C = c1_change + c2
        _, ans = CBC.check(iv, C)
        if not ans:
            return 16 - i
    return 0


def calculatec2(iv, c1, c2, b):
    m2 = bytes()
    for i in range(15 - b, -1, -1):
        c1_end = bytes()
        for k in range(i + 1, 16):
            c1_end = c1_end + (c1[k] ^ (16 - i) ^ (15 - i)).to_bytes(length=1, byteorder='big', signed=False)
            for j in range(256):
                c1_change = c1[:i] + j.to_bytes(length=1, byteorder='big', signed=False) + c1_end
                if CBC.check(iv, c1_change + c2)[1]:
                m2 = (j ^ (16 - i) ^ c1[i]).to_bytes(length=1, byteorder='big', signed=False) + m2
                c1 = c1_change
```

```python
                pass
    return m2


def calculatec1(iv, front, c1):
    iv_end = bytes()
    ans = bytes()
    for i in range(256):
        iv_change = front[:-1] + i.to_bytes(length=1, byteorder='big', signed=False)
        if CBC.check(iv, iv_change + c1)[1]:
            iv_end = (0x01 ^ i ^ front[-1]).to_bytes(length=1, byteorder='big', signed=False)
            ans = calculatec2(iv_change, iv_change, c1, 1)
            break
    return ans + iv_end


def paddingOracle(iv, C):
    C = iv + C
    eposion = len(C) / 16
    ans = bytes()
    for i in range(int(eposion) - 1):
        c1 = C[-32:-16]
        c2 = C[-16:]
        C = C[:-16]
        if i == 0:
            b = findB(iv, c1, c2)
            if b != 16:
                ans = calculatec2(iv, c1, c2, b)
            else:
                ans = bytes()
        else:
            ans = calculatec1(iv, c1, c2) + ans

    return ans


if __name__ == '__main__':
    iv = b'1234567887654321'
    M = 'test for padding-oracle. 20199102 Sun Qilong.' * 5
    C = CBC.encrypt(iv, M)
    print("len:", int(len(C) / 16), "blocks")

    time_start = time.time()
    print(paddingOracle(iv, C))
    time_end = time.time()
    print('spend time:', time_end - time_start, 's')
```

5) 为验证正确性和时间消耗，我们选用如下的字符串作为明文进行加密

test for padding-oracle. 20199102 Sun Qilong.test for padding-oracle. 20199102 Sun Qilong.test for padding-oracle. 20199102 Sun Qilong.test for padding-oracle. 20199102 Sun Qilong.test for padding-oracle. 20199102 Sun Qilong.

攻击结果如下图所示：攻击长度 15block(240bit)，时间 1.213s(intel core i7-8650U@1.9GHz 8GB DDR4-3733)

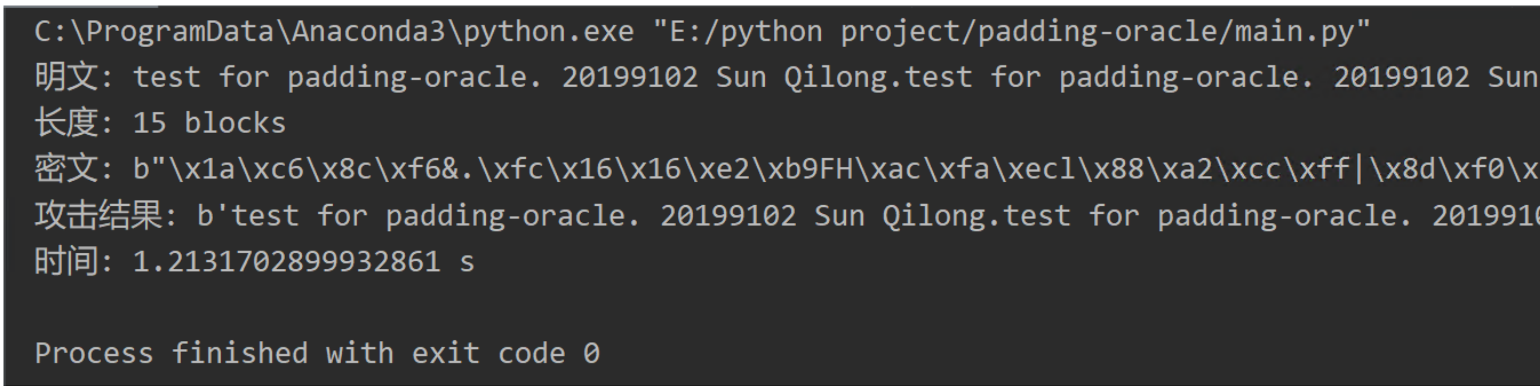

Figure 1: 程序运行结果

Problem 2

设 $\varepsilon = \varepsilon_1\varepsilon_2\ldots\varepsilon_{n+1}$ 是 *i.i.d.*, 定义 $\chi(\varepsilon_i) = \varepsilon_i \otimes \varepsilon_{i+1}$, 即 $\varepsilon_i = 1$ 表示在第 i 位置，后面的比特发生反转 (与 ε_i 不一样). 设 $\xi = \sum_{i=1}^{n} \chi(\varepsilon_i)$

1. ξ 是什么分布?

2. $\varepsilon =$ 11001001000011111101101010100010001000010110100011000010001101 00110001001100011001100010100010111000, 计算观察值和 P 值 (借助 erfc 函数的程序)。

3. 取 $\alpha = 0.01$, 拒绝还是接受 “ε 是 *i.i.d.* 样本”?

answer

1) ξ 服从参数为 $(n, \frac{1}{2})$ 的二项分布，证明如下：
   1. $\chi(\varepsilon_i)$ 为 0，1 构成的序列，且两者互斥。
   2. $p(\chi(\varepsilon_i) = 0) = p(\chi(\varepsilon_i) = 1) = \frac{1}{2}$
   3. $\chi(\varepsilon_i)$n 次结果相互独立

2) 计算观察值和 P 值结果与代码如下，

```
#include<cstdio>
#include<cmath>
#include<cstring>

using namespace std;

```

```c

double calerfc(char *str){
    int len = strlen(str);
    int ones = 0;
    for(int i = 0; i < len; i++)
        ones += str[i] - '0';//(str[i] != str[i + 1]);
    double ans = (ones - (len * 1.0/ 2)) / (sqrt(len*1.0 / 4))
        ;
    printf("obs:%lf\n", ans);
    ans = erfc(ans / sqrt(2));


    return ans;
}

int main (void){
    char str[] = "
        11001001000011111101101010100010001000010110100"\
    "01100001000110100110001001100011001100010100010111000";
    printf("P:%lf", calerfc(str));
}
```

```
"E:\C++ project\random test\cmake-build-debug\random_test.exe"
obs:1.600000
P:0.109599
Process finished with exit code 0
```

Figure 2: 程序运行结果

3) $(P = 0.110) > (\alpha = 0.01)$ 故接受 “ε 是 $i.i.d.$ 样本”

problem 3

设 $\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}dt$ 是标准正态分布的分布函数，定义余差函数 $erfc(z) = \dfrac{2}{\sqrt{\pi}}\int_z^\infty e^{-u^2}du$. 证明 $\mathbb{P}(|x| > |c|) = erfc(\frac{|c|}{\sqrt{2}})$

answer

$$P(|x| > |c|) = 2 * \frac{1}{\sqrt{2\pi}}\int_{|c|}^{\infty} e^{-\frac{t^2}{2}}dt$$

令 t = $\sqrt{2}\mu$, 则 dt = $\sqrt{2}d\mu$, 带入上式积分

$$P(|x| > |c|) = \frac{2}{\sqrt{\pi}}\int_{\frac{|c|}{\sqrt{2}}}^{\infty} e^{-\mu^2}d\mu = erfc(\frac{|c|}{\sqrt{2}})$$