xctf_Crypto_onetimepad

原创

M3ng@L 于 2022-01-07 12:51:19 发布 327 收藏

分类专栏： CTF比赛复现 文章标签： python 密码学

本文链接：https://blog.csdn.net/qq_51999772/article/details/122361981

XCTF_onetimepad

虽然叫做onetimepad，但实际上不是严格的一次一密加密 Vernam加密法_百度百科

题目描述

```python
#!/usr/bin/env python
# coding=utf-8

from os import urandom

def process(m, k):
    tmp = m ^ k
    res = 0
    for i in bin(tmp)[2:]:
        res = res << 1;
        if (int(i)):
            res = res ^ tmp
        if (res >> 256):
            res = res ^ P
    return res

def keygen(seed):
    key = str2num(urandom(32))
    while True:
        yield key
        key = process(key, seed)

def str2num(s):
    return int(s.encode('hex'), 16)

P = 0x1000000000000000000000000000000000000000000000000000000000000425L

true_secret = open('flag.txt').read()[:32]
assert len(true_secret) == 32
print 'flag{%s}' % true_secret
fake_secret1 = "I_am_not_a_secret_so_you_know_me"
fake_secret2 = "feeddeadbeefcafefeeddeadbeefcafe"
secret = str2num(urandom(32))

generator = keygen(secret)
ctxt1 = hex(str2num(true_secret) ^ generator.next())[2:-1]
ctxt2 = hex(str2num(fake_secret1) ^ generator.next())[2:-1]
ctxt3 = hex(str2num(fake_secret2) ^ generator.next())[2:-1]
f = open('ciphertext', 'w')
f.write(ctxt1+'\n')
f.write(ctxt2+'\n')
f.write(ctxt3+'\n')
f.close()
```

程序分析

主程序主要实现了

```
ctxt1 = flag ^ key1
ctxt2 = fake_secret1 ^ key2
ctxt3 = fake_secret2 ^ key3
```

且我们已知key2，key3，ctxt1，ctxt2，ctxt3

显然我们需要`求key1`

那么key1,2,3是怎么生成的呢

keygen()函数套用了process()函数生成了generator

所以key是通过`process()函数`生成的

分析process()函数

攻防世界-密码学-onetimepad - _Mind - 博客园 (cnblogs.com)

~~真的这篇paper数学推导分析的太好了，这里我直接引用ta的了，但是ta最后是使用sagemath直接在域中进行求解，我这里用python来实现~~

另外要理解这个有限域$GF(2^8)$还可以参照这篇paper：(11条消息) 有限域GF(2^8)的四则运算及拉格朗日插值_luotuo44的专栏-CSDN博客_gf域的运算

关系到有限域$GF(2^8)$的乘法运算

看完这两篇文章我们可以知道process()函数返回的是在$GF(2^8)$下的$(m+k)$

由于群的性质：有限群中任何一个元素a的$n-1$次方（n等于群的元素个数）等于1

所以我们`再调用process()函数`来将$(m+k)$进行`升次`，目的是达到$(m+k)$

> 同时这里注意process函数有两个传参，我们将第二个传参置0，这样每次调用函数相当于$((m+k)+0)$
> 且每次升次只会相当于再平方，所以我们重复调用255次

那么最后$(m+k)^{255}=m+1$

注意这里的加号实际上是`异或`，所以我们可以从key3，key2推导出`secret`，也就是seed

推导一下

key3 = (key2 ^ secret)

得到secret = key3 ^ key2

key2 = (key1 ^ secret)

得到key1 = key2 ^ secret

那么flag = key1 ^ ctxt1

Python实现

```python
from Crypto.Util.number import *
import gmpy2

def process(m, k):
    tmp = m ^ k
    res = 0
    for i in bin(tmp)[2:]:
        res = res << 1
        if (int(i)):
            res = res ^ tmp
        if (res >> 256):
            res = res ^ P
    return res

P = 0x10000000000000000000000000000000000000000000000000000000000000425
c1 = 0xaf3fcc28377e7e983355096fd4f635856df82bbab61d2c50892d9ee5d913a07f
c2 = 0x630eb4dce274d29a16f86940f2f35253477665949170ed9e8c9e828794b5543c
c3 = 0xe913db07cbe4f433c7cdeaac549757d23651ebdccf69d7fbdfd5dc2829334d1b
m2 = b'I_am_not_a_secret_so_you_know_me'
m3 = b'feeddeadbeefcafefeeddeadbeefcafe'

m2 = bytes_to_long(m2)
m3 = bytes_to_long(m3)
k3 = m3 ^ c3
k2 = m2 ^ c2
temp_k = k3
for _ in range(255):
    temp_k = process(temp_k,0)
secret = temp_k ^ k2
temp_k = k2
for _ in range(255):
    temp_k = process(temp_k,0)
k1 = temp_k ^ secret
m = c1 ^ k1
print(long_to_bytes(m))
```