攻防世界crypto进阶区onetimepad

原创

M10++ 于 2021-08-23 09:53:45 发布 381 收藏

分类专栏： ctf 文章标签： unctf 网络安全 信息安全 安全 密码学

本文链接：https://blog.csdn.net/M10_2339656216/article/details/119862416

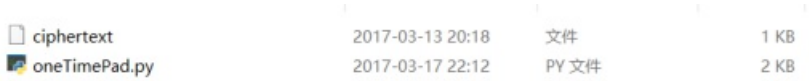
文章目录

content

攻防世界 crypto 高手进阶区 4分题（动态分数） onetimepad 的

writeup

下载题目—解压zip文件
得到：一个密文文件和一段py



分别用notepad++打开

```
af3fcc28377e7e983355096fd4f635856df82bbab61d2c50892d9ee5d913a07f
630eb4dce274d29a16f86940f2f35253477665949170ed9e8c9e828794b5543c
e913db07cbe4f433c7cdeaac549757d23651ebdccf69d7fbdfd5dc2829334d1b
```

```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 = 0x10000000000000000000000000000000000000000000000000000000000000425L

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()
```


这里的关键是process(m,k)只是把m⊕k 赋值给了GF(2^256), 为了反平方，我们可以简单地平方255次
脚本（方法一）：

```
c1 = 0xaf3fcc28377e7e983355096fd4f635856df82bbab61d2c50892d9ee5d913a07f
c2 = 0x630eb4dce274d29a16f86940f2f35253477665949170ed9e8c9e828794b5543c
c3 = 0xe913db07cbe4f433c7cdeaac549757d23651ebdccf69d7fbdfd5dc2829334d1b

k2 = c2 ^ str2num(fake_secret1)
k3 = c3 ^ str2num(fake_secret2)

kt = k3
for i in xrange(255):
    kt = process(kt, 0)

seed = kt ^ k2
print "SEED", seed
assert process(k2, seed) == k3

kt = k2
for i in xrange(255):
    kt = process(kt, 0)

k1 = kt ^ seed
print "K1", seed
assert process(k1, seed) == k2

m = k1 ^ c1
print `hex(m)[2:-1].decode("hex")`
```

脚本（方法二）：

```
from Crypto.Util.number import bytes_to_long,long_to_bytes

K.<x>=GF(2L**256,modulus=x^256+x^10+x^5+x^2+1)

def polify(N):
    bN=list(bin(N)[2:])
    bN.reverse()
    return K(bN)

def unpolify(Poly):
    bN=Poly.polynomial().list()
    bN.reverse()
    return long(''.join([str(it) for it in bN]),2)

def solve():
    cip1=polify(0xaf3fcc28377e7e983355096fd4f635856df82bbab61d2c50892d9ee5d913a07f)
    cip2=polify(0x630eb4dce274d29a16f86940f2f35253477665949170ed9e8c9e828794b5543c)
    cip3=polify(0xe913db07cbe4f433c7cdeaac549757d23651ebdccf69d7fbdfd5dc2829334d1b)
    msg2=polify(bytes_to_long('I_am_not_a_secret_so_you_know_me'))
    msg3=polify(bytes_to_long('feeddeadbeefcafefeeddeadbeefcafe'))
    secret=cip2+msg2+(cip3+msg3).sqrt()
    key1=(cip2+msg2).sqrt()+secret
    msg1=cip1+key1
    return long_to_bytes(unpolify(msg1))

if __name__=='__main__':
    print 'flag{'+solve()+'}'
```

脚本（方法三）：

```
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)

src1 = 0xaf3fcc28377e7e983355096fd4f635856df82bbab61d2c50892d9ee5d913a07f
src2 = 0x630eb4dce274d29a16f86940f2f35253477665949170ed9e8c9e828794b5543c
src3 = 0xe913db07cbe4f433c7cdeaac549757d23651ebdccf69d7fbdfd5dc2829334d1b

fake_secret1 = "I_am_not_a_secret_so_you_know_me"
fake_secret2 = "feeddeadbeefcafefeeddeadbeefcafe"

k2 = src2 ^ str2num(fake_secret1)
k3 = src3 ^ str2num(fake_secret2)

kt = k3
for i in range(255):
    kt = process(kt, 0)

seed = kt ^ k2
print "SEED:", seed
assert process(k2, seed) == k3

kt = k2
for i in range(255):
    kt = process(kt, 0)

k1 = kt ^ seed
print "K1:", k1
assert process(k1, seed) == k2

m = k1 ^ src1
print hex(m)[2:-1].decode("hex")
```

summary

双手奉上大佬blog：
https://www.cnblogs.com/coming1890/p/13607557.html
（很好的解题过程及分析）