PlaidCTF 2019 CRYPTO Writeup

aliyuuu / 2019-04-22 09:50:00 / 浏览数 3528 安全技术 CTF 顶(0) 踩(0)

PlaidCTF 2019 CRYPTO Writeup

 上周末参加了一下PlaidCTF 2019国际比赛，详情可见CTF TIME，下面主要总结分享一下两道密码题的解题思路。

1.R u SAd?

1.1 分析

首先查看题目文件，共有三个文件rusad(python■■■■)■flag.enc(■■■■■■)■key.sad.pub(■■■■■■■■)。主要分析rusad加密代码，这是一个RSA加解密程序，

```
class Key:
   PRIVATE_INFO = ['P', 'Q', 'D', 'DmP1', 'DmQ1']
   def __init__(self, **kwargs):
       for k, v in kwargs.items():
           setattr(self, k, v)
       assert self.bits % 8 == 0

   def ispub(self):
       return all(not hasattr(self, key) for key in self.PRIVATE_INFO)

   def ispriv(self):
       return all(hasattr(self, key) for key in self.PRIVATE_INFO)

   def pub(self):
       p = deepcopy(self)
       for key in self.PRIVATE_INFO:
           if hasattr(p, key):
               delattr(p, key)
       return p

   def priv(self):
       raise NotImplementedError()
def genkey(bits):
   assert bits % 2 == 0
   while True:
       p = genprime(bits // 2)
       q = genprime(bits // 2)
       e = 65537
       d, _, g = egcd(e, (p-1) * (q-1))
       if g != 1: continue
       iQmP, iPmQ, _ = egcd(q, p)
       return Key(
           N=p*q, P=p, Q=q, E=e, D=d%((p-1)*(q-1)), DmP1=d%(p-1), DmQ1=d%(q-1),
           iQmP=iQmP%p, iPmQ=iPmQ%q, bits=bits,
       )
def encrypt(key, data):
   data = bytes2num(pad(data, key.bits))
   assert 0 <= data and data < key.N
   data = pow(data, key.E, key.N)
   return num2bytes(data, key.bits // 8)

def decrypt(key, data):
   assert key.ispriv() and len(data) * 8 == key.bits
   data = bytes2num(data)
   assert 0 <= data and data < key.N
   v1 = pow(data, key.DmP1, key.P)
   v2 = pow(data, key.DmQ1, key.Q)
   data = (v2 * key.P * key.iPmQ + v1 * key.Q * key.iQmP) % key.N
   return unpad(num2bytes(data, key.bits // 8))
```

1.2 思路

 注意到公钥文件key.sad.pub中包含的参数有N■E,iQmP,iPmQ，其中iQmP, iPmQ, _ = egcd(q, p);iQmP=iQmP%p, iPmQ=iPmQ%q

```
iQmP, iPmQ, _ = egcd(q, p)#■iQmp*q+iPmQ*p=1,■■■■■■■■■■■■■■■
iQmP = iQmP%p              #iQmP = iQmP + p * i(i=0■1)
iPmQ = iPmQ%q              #iPmQ = iPmQ + Q * i(i=0■1)
```

最终得到iQmp*q+iPmQ*p=p*q+1=N+1，结合p*q=N可以得到两个方程，p、q两个未知数两个方程，直接求解即可：

```
b = N+1
a = gmpy2.iroot(b*b-4*iQmP*iPmQ*N, 2)[0]
p1 = (b+a)/(2*iQmP)
# p2 = (b-a)/(2*iQmP)
# q1 = (b+a)/(2*iPmQ)
q2 = (b-a)/(2*iPmQ)
# if (p1*q2==N):
#   print("success")
```

1.3 EXP

求解得到P和Q之后，正常计算欧拉函数phi(N)，计算私钥D。由于加密进行了填充，直接用D解密存在问题，将参数代入到Key中，然后调用decrypt■■解密即可，将代码

```
f = argparse.FileType('rb')("key.sad.pub")
key = pickle.load(f)
key.Q = 250046722278554099953861756633361886851776385412866660564418308476181008081986681673078142362244298852952411401946336
key.P = 316590778098857066994823618304777175728370817796776264358299033749215812408491800631085520192740218260927812872185686
if(key.P*key.Q==key.N):
   print("success")
e = 65537
d, _, g = egcd(e, (key.P-1) * (key.Q-1))
key.DmP1=d%(key.P-1)
key.DmQ1=d%(key.Q-1)
key.bits=4096
with open('flag.enc', 'rb') as f:
   cipher = f.read()
# print(len(cipher)*8)
m = decrypt(key,cipher)
print(m)
```

2.Horst

2.1 分析

题目给了一个python加密程序和一段密文，同样首先分析加密程序。程序首先定义了一个Permutation■，数据类型是列表list，主要关注的函数为■■■■mul和■■■■inv

```
import os
import random
from hashlib import sha1

M = 3
N = 64
n = 2

class Permutation:
   def __init__(self, L):
       self.n = len(L)
       self.L = L
       assert all(i in L for i in range(self.n))

   def __mul__(self, other):
       assert self.n == other.n
       return Permutation([other.L[self.L[i]] for i in range(self.n)])

   def __eq__(self, other):
       return self.L == other.L

   def inv(self):
       return Permutation([self.L.index(i) for i in range(self.n)])
```

```
   def cycles(self):
       elts = list(range(self.n))
       cycles = []
       while len(elts) > 0:
           cur = []
           i = elts[0]
           while i not in cur:
               cur.append(i)
               elts.remove(i)
               i = self.L[i]
           cycles.append(cur)
       return cycles

   def __getitem__(self, i):
       return self.L[i]

   def __str__(self):
       return "".join("({})".format(" ".join(str(e) for e in c)) for c in self.cycles())

   def __repr__(self):
       return "Permutation({})".format(self.L)


def random_permutation(n):
   random.seed(os.urandom(100))
   L = list(range(n))
   for i in range(n-1):
       j = random.randint(i, n-1)
       L[i], L[j] = L[j], L[i]
   return Permutation(L)

for i in range(100):
   x = random_permutation(N)
   assert x * x.inv() == Permutation(list(range(N)))

def encrypt(m, k):
   x, y = m
   for i in range(M):
       x, y = (y, x * k.inv() * y * k)
   return x, y

def decrypt(c, k):
   x, y = c
   for i in range(M):
       x, y = (y * k.inv() * x.inv() * k, x)
   return x, y

if __name__ == "__main__":
   k = random_permutation(N)
   print "The flag is: PCTF{%s}" % sha1(str(k)).hexdigest()
   pairs = []
   for i in range(n):
       pt = random_permutation(N), random_permutation(N)
       ct = encrypt(pt, k)
       assert pt == decrypt(ct, k)
       pairs.append((pt,ct))
   with open("data.txt", "w") as f:
       f.write(str(pairs))
```

2.2 思路

 分析加密函数，得知加密过程是进行三轮运算`x, y = (y, x * k.inv() * y * k)`，对于这里定义的list操作，`[0,1,2,3,4,...,63]`■■■■■■，其中`k.inv()*k`=■■■可以将`k.inv()`理解为`k^-1`。经过测试这里的list操作满足乘法结合律`(a*b)*c=a*(b*c)`，不满足乘法交换律`a*b!=b*a`。

```
## ■■■
x1 = y
y1 = x*k^-1*y*k
## ■■■
x2 = y1 = x*k^-1*y*k
```

```
y2 = x1*k^-1*y1*k = y*k^-1*x*k^-1*y*k*k
## ■■■
x3 = y2 = y*k^-1*x*k^-1*y*k*k
y3 = x2*k^-1*y2*k = (x*k^-1*y*k)*k^-1*(y*k^-1*x*k^-1*y*k*k)*k
##
#■■: y3 = x*k^-1*y*x3*k (x,y■■pt,x3,y3■■ct)
##  k*x^-1*y3 = y*x3*k
```

分析加密流程可以得到k*A=B*k;A=x^-1*y3;B=y*x3，题目提供两组明密文对，因此可以计算得到两组A、B数据

计算得到：

```
A1=[56, 31, 46, 28, 5, 52, 12, 14, 10, 34, 22, 47, 6, 0, 39, 17, 32, 38, 13, 40, 4, 18, 15, 55, 50, 24, 9, 45, 59, 41, 23, 43,
B1=[48, 19, 58, 24, 20, 47, 31, 53, 59, 23, 1, 5, 42, 37, 33, 55, 2, 29, 12, 27, 8, 11, 56, 9, 44, 63, 14, 25, 10, 49, 61, 60,
A2=[40, 16, 48, 27, 18, 7, 55, 10, 9, 13, 5, 31, 57, 14, 35, 45, 60, 23, 41, 15, 63, 30, 39, 8, 33, 43, 59, 44, 6, 50, 1, 25,
B2=[3, 54, 28, 2, 44, 59, 27, 31, 50, 4, 35, 36, 21, 0, 8, 19, 38, 20, 14, 25, 16, 61, 26, 10, 57, 39, 55, 60, 33, 29, 52, 22,
```

再次分析乘法操作：[other.L[self.L[i]] for i in range(self.n)]

假设k=[k0,k1,k2,k3,k4,...,k63],k*A=B*k等价于A[ki]=k[B[i]]; i=0,1,2,3,...,63。

现在考虑找到满足B[i]=i的数据，在B2中找到B2[29]=29因此i=29代入上诉式子得到：A2[k29]=k[29]，同样在A2中搜索A[i]=[i]的数据得到A2[58]=58，因此k29

```
for i in range(64):
   if(i==B2.index(i)):
       print i
   if(i==A2.index(i)):
       print i
#29
#58
#[Finished in 0.2s]
```

2.3 EXP

在得到一组数据k29=58，就能根据A[ki]=k[B[i]]; i=0,1,2,3,...,63式子将其余数据推算出来。

例如：在B1和A1中找到相关的i，j，使得B1[i]=29;A1[j]=58，则A1[ki]=k29=58，就有k[i]=j

```
K = dict.fromkeys(A1, -1)
k = []
i = 29
value = 58
while i not in k:
   k.append(i)
   K[i] = value
   value = A1.index(value)
   i = B1.index(i)
print len(k)
print k
print K
```

通过上诉代码可先得到8组数据，再代入A2、B2中搜索即可得到所有数据：

```
K = {0: 59, 1: 2, 2: 50, 3: 29, 4: 55, 5: 15, 6: 43, 7: 30, 8: 27, 9: 6, 10: 57, 11: 22, 12: 7, 13: 26, 14: 3, 15: 35, 16: 24,
# ■■■K■■■■k■■■■
k = Permutation([59, 2, 50, 29, 55, 15, 43, 30, 27, 6, 57, 22, 7, 26, 3, 35, 24, 40, 53, 46, 49, 10, 16, 12, 41, 47, 60, 11, 5
print "The flag is: PCTF{%s}" % sha1(str(k)).hexdigest()
```

3.总结

这次的两个python加解密题还是比较简单的，主要考察的基础数学运算。RuSAd只要理解了egcd函数的原理就能直接解题，Horst相对麻烦一下，首先需要理解题目定义的

上一篇：Nginx配置不当可能导致的安全问题 下一篇：西湖论剑初赛easyCpp探究

1. 0 条回复

- 动动手指，沙发就是你的了！

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板