

- week_2

- Crypto

- RSA大冒险2

- 1——维纳攻击
 - 2——观察题目，发现p和q的差在 $\text{pow}(2,256)$ 左右，相差不大，用yafu分解，得到p, q
 - 且从题目得知 $\text{gcd}(e, \phi) = 2$ ，故令 $m1 = \text{pow}(m, 2)$, $e1 = e // 2$, $d1 = \text{invert}(e1, \phi)$ ，最后对m1开方即可
 - 3——p高位泄露，但是泄露位数不足以使用copper_smith算法，故爆破，查阅资料得知，small_roots中的第三个参数epsilon决定了格基的维数，维数越大，所需泄露位数越少，经过测试发现，在epsilon取0.01时，仅需泄露263位即可，故爆破最后一位，得解

- ezDH

- 直接用sagemath的discrete_log即可求出A_secret和B_secret，进而求得shared_secret，即为ECC加密中的私钥，在使用sagemath计算 $c \cdot \text{shared_secret} \cdot P1$ 即可（当时做完之后忘记写wp了，以至于拖了一段时间，exp也丢了，只能讲讲思路）

- ezBlock

- 差分攻击，观察题目可知这是4轮加密，故先制作1, 2, 3轮的差分分布表
 -

```

文件  编辑  查看

[4096, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 320, 608, 192, 256, 192, 160, 272, 144, 416, 304, 128, 272, 352, 288, 192],
[0, 640, 696, 136, 152, 232, 176, 208, 88, 520, 416, 64, 160, 96, 264, 248],
[0, 240, 216, 552, 232, 296, 160, 240, 376, 120, 144, 272, 288, 240, 392, 328],
[0, 224, 184, 328, 216, 376, 352, 208, 296, 328, 304, 320, 208, 256, 328, 168],
[0, 304, 344, 232, 408, 216, 192, 368, 328, 280, 320, 368, 208, 128, 168, 232],
[0, 240, 304, 272, 384, 224, 192, 368, 304, 288, 304, 336, 256, 144, 208, 272],
[0, 208, 208, 208, 208, 384, 432, 256, 320, 352, 320, 432, 144, 192, 272, 160],
[0, 128, 128, 224, 304, 272, 384, 336, 384, 144, 192, 464, 288, 336, 224, 288],
[0, 288, 272, 128, 272, 304, 352, 304, 224, 384, 352, 272, 256, 240, 192, 256],
[0, 256, 232, 200, 312, 312, 288, 304, 232, 376, 336, 224, 240, 256, 264, 264],
[0, 160, 136, 392, 344, 296, 272, 288, 392, 184, 240, 416, 240, 256, 264, 216],
[0, 304, 280, 392, 200, 296, 288, 144, 248, 248, 240, 208, 224, 464, 296, 264],
[0, 208, 248, 296, 296, 184, 240, 320, 248, 152, 208, 240, 416, 304, 264, 472],
[0, 272, 144, 336, 240, 224, 272, 288, 224, 176, 224, 160, 560, 288, 240, 448],
[0, 304, 96, 208, 272, 288, 336, 192, 288, 128, 192, 192, 336, 544, 432, 288]

```

```

main.py x  S盒_test.py x  ezBlock.py x
Project
29  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
30  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
31  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
32  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
33  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
34  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
35  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
36  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
37  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
38  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
39  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
40  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
41  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
42  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
43  ]
44  for i in range(16):
45      for i1 in range(16):
46          k1 = table[i][i1]
47          for i2 in range(16):
48              k2 = table[i1][i2]
49              for i3 in range(16):
50                  k3 = table[i2][i3]
51                  tableout[i][i2] += k1*k2*i3
52  for i in range(16):
53      print(f"{tableout[i]},")

```

- 再根据此表得到第四轮前各个差分的概率，并根据其统计第四轮备选key的概率，并得出可能的key

```

for r in range(0,16,4):
    key_list = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    x = combinations(1, 2)
    for i in x:
        xor0 = ((m[i[0]] >> r) & 0xf) ^ ((m[i[1]] >> r) & 0xf)
        c1 = (c[i[0]] >> r) & 0xf
        c2 = (c[i[1]] >> r) & 0xf
        for inxor in range(16):
            p = table[xor0][inxor]
            for t in range(16):
                if (r_substitute(t ^ c1) & 0xf) ^ (r_substitute(t ^ c2) & 0xf) == inxor:
                    key_list[t] += p
    print(key_list)
    M = key_list[0]
    key = 0
    for i in range(16):
        if key_list[i] > M:
            M = key_list[i]
            key = i
    print(hex(key))

```

- 类似的依次解出前几轮的备选key，可以得出大致的flag

- 另，可以通过观察明文序列和第一轮密文的异或结果的离散程度，来大致确定错误的轮密钥的位置（大概吧）

- **Web**

- **Gopher_Shop**

- 考察整数溢出，尝试在输入苹果购买数量时运用uint的上界溢出来实现攻击，尝试uint64的上界
18446744073709551615，无果，后得知golang中的溢出只能存在于运算中，故尝试
 $1844674407370955162 * 10 = 18446744073709551620$ ，即
 $18446744073709551620 - 18446744073709551615 = 5$ ，可以购买，随机卖出，再买入flag即可

以上内容整理于 [幕布文档](#)