Birkenwald 密码学入门

Birkenwald 密码学入门

前言

与平时所说的“账号密码”不同，这里提到的密码学研究的是“加密”，而常说的“账号密码”对应的说法应该 是“口令”（password）。一个明文经过密钥加密后，就会变为一段“让人看不懂”的密文，而经过密钥解 密后，就会变回有意义的明文。然而，有些人没有密钥也想获取密文中的部分或者全部内容，甚至是直 接利用密文去达到自己的要求，这就是CTFer在比赛中需要达成的目的。在比赛中，通常会把加密算法 公开出来，然后再将密文等信息发送给参赛者，参赛者需要分析加密算法的缺点，以找到利用密文的方法。

在密码学中还有许多有意思的问题，例如，如何在没有事先商定好密钥的情况下，在一个公开的网络中 完成信息的秘密传输？如何确认这个信息是由朋友本人传输的而不是他人伪造？是否存在理论上不可攻 破的密码体系 诸多问题，将在以后的学习中得到解答。

这里所简单描述的都是现代密码学，有现代当然也有古典密码了，不过古典密码学更多的是一些替换编码，对于现代密码所需要的知识相关性不太大，在CTF中古典密码被划分到Misc中也是情有可原的。

编程语言

目前大一新生大多都接触到了C 语言或者C++语言，但是C 语言或者C++对于密码学的学习而言并帮助并不是很大，想要用C 语言处理密码学里的大数运算是很困难。相比之下Python 就方便很多了。事实上不管你是学习哪个方向的，Python 都是非常重要的且必须学的。

学习Python 的话首推廖雪峰的在线教程：Python 2.7 教程，Python 3.x 教程(。)切记学习编程语言千万不能浮躁，不能说“我觉得这里的代码我看懂了”，“我觉得应该没有bug” 这样想当然而不去动手操练。

编写Python 的IDE 的话，推荐VScode、sublime、pycharm，这三者都是免费使用的

（后两者社区版）。在配置环境等等的时候可能会遇到很多坑，在解决问题的过程你也许就会同时学到很多计算机的基础知识，如果出现什么错误的话，先要动手去百度或者谷歌，实在解决不了的话再来群里问学长学姐们。

数论

密码学里所用到数学方面的知识并不是萌新们认为的高中的函数求导、解析几何这种数学，而是和离散数学有关的一个数学分支：数论

数论主要研究的是整数之间性质，学习初等数论的话可以看这本书《密码编码学与网络安全——原理与实践》中和数论有关的章节，主要搞懂：同余、模运算性质、逆元、欧拉定理、中国剩余定理这几个概念、公式的推导和证明就行了，然后可以用上面学的python 写几个demo 验证这几个数学公式是否正确。

当然，数论是一门非常深奥的学科，这里说的知识只是一点皮毛而已，如果要深入学习的话，还可以了解一下sagemath，这是一个非常强大的数学工具，集成了许多python 和数学有关的库，可以让我们非常简单的处理一些抽象的数学运算，但是不推荐萌新学习，因为学习的资料太少。

密码学

接着就可以学习一点简单的密码学知识了，首先是古典密码学，古典密码学就是古代的时候常常用来加密信息的方式，现代已经很少使用，因为它们都不够安全，可以被现代计算机很容易的暴力破解。

古典密码学上面没有必要花费太多时间，了解一下常见的就行了，比如凯撒密码，维吉尼亚密码、栅栏密码。很多古典密码都是有在线加解密网站的，这里推荐两篇入门的CTF 中Crypto（密码类）入门必看、密码学（Crypto）一些在线解密网站。在遇到类似题目的时候再根据题目描述去查阅就行了。

其中还涉及到了一些编码的知识，注意区别一下编码和密码。编码是每个CTF 选手都必须掌握的。比如如何将需要加密的英文信息编码成参与数学运算的数字。

然后可以了解一下最著名最常考的RSA 加密方式：李永乐老师 11 分钟讲RSA 加密算法光看这个视频肯定是不够的，结合视频上讲的和学习到的数论知识，推导一下RSA 算法的正确性，搞懂公钥、私钥是什么，以及是怎么计算的，用的是什么算法，最后再思考如何用代码实现，虽然这可能有点难度。

然后就可以了解一下RSA 的攻击手法了，现在常考的 RSA 的攻击手法都基本上来源于斯坦福大学应用密码学和计算机安全教授Dan Boneh 发表的论文：

Twenty Years of Attacks on the RSA Cryptosystem

尽管国内有翻译，但还是鼓励萌新们读英文文档，搞懂攻击的原理。可以对应类似的题目去试试怎么解密出明文。再然后的话就可以结合实际的情况，看看书，看看bugku,buu 的题目，更进一步的学习啦， 永远欢迎热爱学习热爱技术的同学。最后再强调一点，不论是学习什么，都要记笔记！记笔记！记笔记！（markdown、markdown、markdown!!!）

例题讲解

```
import libnum
import gmpy2
p=libnum.generate_prime(1024)
q=libnum.generate_prime(1024)
n=p*q
e=65537
phi_n=(p-1)*(q-1)
d=gmpy2.invert(e,phi_n)
print(gmpy2.invert(3,77200))
```

```
flag='s3c{807048823036c2e5f646a7d16b37f998}'
m=libnum.s2n(flag)
c=pow(m,e,n)
print(m)
print(pow(m,phi_n,n))
print ("n=",n)
print ("e=",e)
print ("c=",c)
print("d=",d)
```

首先这是最简单的rsa，给了私钥d，那么可以直接调用公式来解决

也就是

$m = c^d mod n$

也就是

```
m=pow(c,d,n)
```

给了共私钥什么都可以出来了，当然这第一题也就是理解一下rsa的常规解题

我们再来看看这种情况

```
import libnum
import gmpy2
p=libnum.generate_prime(1024)
flag='s3c{807048823036c2e5f646a7d16b37f998}'
q=gmpy2.next_prime(p)
n=p*q
e=6537
m=libnum.s2n(flag)
c=pow(m,e,n)
print('n=',n)
print('e=',e)
print('c=',c)
```

我们要先知道gmpy2.next_prime的意思，它的意思在这就是取p下一个素数，那么这两个数将会非常接近，我们可以对其开方，然后在其上下几千的范围和n相余，来找到正确的解，当然还有一种方法叫做费马分解

```
# _*_ coding:utf-8 _*_
```

```python
import math


def fermat(n):
    a = math.ceil(math.sqrt(n))
    # b的平方
    b2 = a * a - n
    b = round(math.sqrt(b2))
    while b * b != b2:
        a += 1
        b2 = a * a - n
        b = round(math.sqrt(b2))
    print(a, b, n)
    return a - b, a + b
```

其实也就是找到两个数

$$p = \frac{a+b}{2}$$
$$q = \frac{a-b}{2}$$

需要注意的是这对于奇合数才有用，因为偶数的话，分解出来的两个数如果是一奇一偶，则相加除于2不是一个整数。单次分解的复杂度是 O(∣p−q∣)，在实际应用中费马质数分解的效率其实不高，因为实际上两个质因数相近（ ∣p−q∣比较小）的情况是比较少见的

当然还有很多攻击类型，例如dp泄露等等，可以结合https://lazzzaro.github.io/2020/05/06/crypto-RSA/（la佬blog）来学习，我接下来主要对nss上的中难题目进行讲解

作者：@Birkenwald