密码学 入门手册

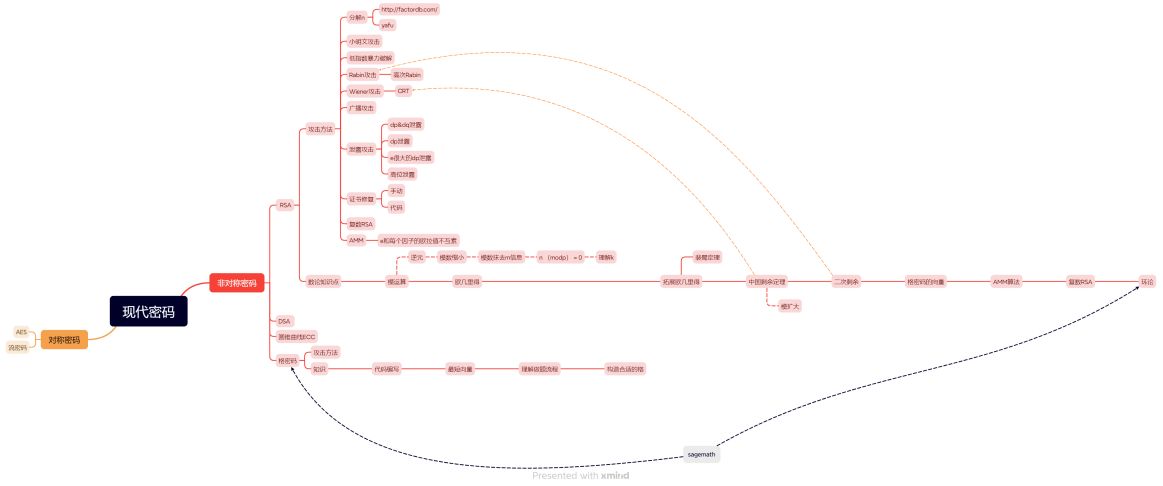
一、密码学是什么

人话：一种用来研究如何加密来保护信息，如何解密获取信息的学问。

二、为什么学密码

可以了解 开创并实现现代密码学的RSA算法、未来将抵御量子计算机的格密码等的“后量子密码技术”，以及学到在特殊情况下我们用现代计算机来破解特定的RSA现代密码和后量子密码技术的攻击手段。

密码有清晰的路线学习



干脆的做题反馈（就跟数学题一样，会就是会，不会就是不会）

或者说没有理由，当你入门密码学的时候，就会发现做密码题也会上瘾！

三、需要的工具

1. 电脑
2. 软件：`yafu`，`python`，笔记软件
3. 草稿纸和笔（随电脑携带的，经常需要演算关系式）
4. 网站 factordb.com尝试分解n 、 探索商城 | NSSCTF 、 RSA入门 | DexterJie'Blog （大佬的博客，还可以听听歌）等等

四、学习路线

编程：python

1. 配置好python编译器和集成开发环境（`pycharm`或者`vscode`），`print("hello python world!")`
2. 理解变量和变量类型，变量类型转换 `a = b'65537'  a = int(a)`
3. 运算符号，赋值= ， 加减乘除 +-*/ ，整除 // ，比较大小 < > <= >= ， 等于==，取模 %

4. 基本语法
5. 循环语句，布尔类型（真和假，1和0，True和False）
6. python列表，数组
7. pip install 安装`pycyptodome`和`gmpy2`包，模块导入
8. 函数的使用，`pow（a,b,c）`相当于 (a ** b) % c
9. python各种进制的格式头（二进制0b，十六进制0x等等）

密码学

1. RSA公钥加密算法（推荐探索商城 | NSSCTF，挺系统的）下面我也会简单介绍RSA公钥加密算法。

数学：

1. 理解模运算
2. 逆元计算的运用（暂时不要求理解）
3. 理解海纳百川的k和无所不能的公因数

五、开始学习！

（一）掌握一定语言基础

首先建议学python，密码题经常出现很大的数字和较难的运算，靠手算很难得到答案，所以需要一门计算机语言来帮我们计算。而python是符合直觉的语言，对于新手而言没有过多的条条框框，所以推荐从python入门编程语言，了解一下知识点就行。掌握下面的除画红线的知识点。

讲讲我个人学习python的经历：我一开始就买了本python的入门书籍开始啃，啃到后面for循环就开始做题了，一开始也不会做题，就看别人的writeup（行话解析：看看别人的思路，抄作业，但这是个好的行为，大伙开始都是看过来的），别人的wp（writeup的缩写）一般都会有python代码，就跟着打在自己的电脑上（建议不要直接复制粘贴），遇到不会的就上网搜索，一边学密码学，一边练编程技术。

（二）了解`rsa`算法（公钥加密算法）

"毫不夸张地说，只要有计算机网络的地方，就有RSA算法。" ——阮一峰

1.知识点

- 元素

  RSA算法里面一共有 8个元素。

```
p : 一个素数。
q : 另一个素数
n : 由p 和 q 组成的 ， 一般为 n = p x q
e : 一个被称为“公钥因子”的数。
d : 一个被称为“私钥因子”的数。下面的公式会具体展现e和d的转化。
phi : 一个与n有关，由p，q计算而得到的数。具体就是n的欧拉函数 φ(n) 。
m : 明文，就是要保护的信息。
c : 密文，就是要破解的信息。下面的公式会具体展现明文和密文的转化。
```

- 公式

  RSA算法中一共由如下公式

  1.加密公式

  $$c = m^e \pmod{n}$$

  2.解密公式

  $$m = c^d \pmod{n}$$

  3.e和d的关系式

  $$d = e^{-1} \pmod{phi}$$

  4.p和q是素数的情况下，n的欧拉函数phi

  $$phi = (p-1)*(q-1)$$

  5.n与p,q的关系式

  $$n = p*q$$

  6.大小关系

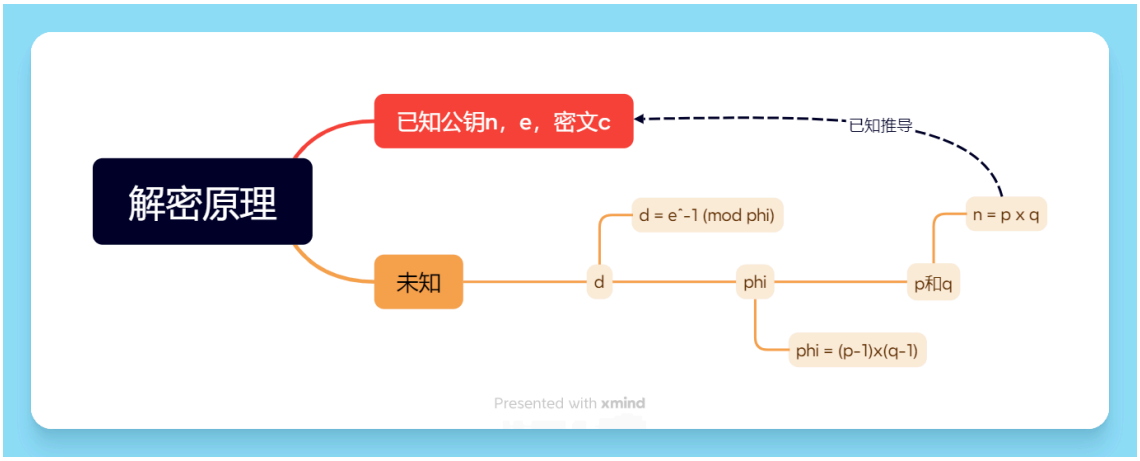
$$m^e < n$$
$$n \approx phi$$

> 入门后可以尝试推导一下上面的公式，入门前只需要懂得如何在python中实现上面公式即可。

- 原理
  - 解密原理

一般RSA算法加密会公开公钥n和e，信息传输难免会泄露密文c，所以我们可以得到的一共有三个元素n，e，c。从上面的解密公式可以知道我们需要密钥因子d，而计算d需要e和phi，e已知，求未知的phi又需要p和q，有关p和q的式子只有一个$n = p * q$

因此解密的关键是在得到p和q



  - 难破解的原理

RSA算法开创了现代密码并实现了非对称加密，其中难破解的原理基于大整数分解。

计算n = p*q简单，但由n分解出p，q

手算 12 的因子，很快可以列出 [1,2,3,4,6,12]，但一旦上升到很大的整数，例如

```
6969872410035233098344189258766624225446081814953480897731644163180991292913719910322241873463164232700368119465476508174863062276659958418657253738005689
```

就根本拆不开成两个数字。

  - 我们解题的原理

上面说到，n很难分解，那我们是靠什么来破解的呢?

答：靠一个个特殊的情况，有时候出题人会给出泄露的d，或者p和q之间的关系让我们数学推导等等特殊情况。针对这一个个特殊情况，我们也有相应的攻击方法（如下）。

小明文攻击

低指数暴力破解

Rabin攻击与中国剩余定理的初始应用

Wiener攻击（未亲自推导）

(一)

(二)

(三)

广播攻击

低加密指数

不互素的CRT

拓展欧拉定理

Rabin进阶

dp&dq泄露

p-1光滑

AMM

RSA证书

手动处理

代码处理

变种RSA——SchmidtSamoa

复数与rsa

- 数学基础

  RSA中涉及到我们之前没有或者说不太重视的一些数学知识点，接下来我们一起来学习一下。

  - 取模

上面公式四个有三个都出现了一个英文单词 mod ， 这可不是游戏中打mod的意思， 在数学中，这被称为 “模” ，数学符号是这个`%`，相信大家小时候都学过余，这个就是模。12除5余2的余。接下来讲一种新的观点，12在 $\mod 5$ 的世界中就是2 ， 这个世界的数学范围是[0 , 5) 。

- 模运算

  模运算和平时的运算基本上没有太大区别，只是在模运算中，我们人为的划定了一个集合[0 , n) ，其中出现的数字都不会超过这个集合。模运算也可以实现等式左右两边交换，例如e和d的关系式也可以写成 $e * d = 1 \pmod n$ 。

  而运算符号加减乘除的除就有些变化了。上面的e^{-1} 可以写成 $\frac{1}{e}$ ，也就是1除于e，但在模运算中不能直接除，

  例如 $\frac{a+1}{b-1} \pmod n$ 要先求b的逆元 $(b-1)^{-1}$，再将这逆元$(b-1)^{-1}$与$(a+1)$相乘。

  在上面第三个公式 e和d的关系式中有d^{-1},这涉及到模运算中求逆元，这点展开会涉及到其他数学知识，这里先放着不谈。只要记住在模运算中，没有除法，只有求逆元然后再相乘。具体的代码实现会在下面讲述。

- 海纳百川的k和无所不能的公因数

  在上面的取模和模运算中，涉及到大量的 $\mod n$ ，如何转化成可以手算推理的东西呢?

  答：$k * n$ 其中k是任意整数，例如 $12 \pmod 5 = 12 - 2 * 5$ 其中的2就是k啦。

  我们之后会遇到一部分的数学推导题，出题人会给出有关p和q的关系式。草稿纸上推理处理完关系式后，经常能得到一个有关p或者q的式子$a = b \pmod p$，此时我们只需要将这个式子转化一下 $a = b + k_1 * p$ 即 $k_1 * p = a - b$ 求 `a-b` 和 `n` 的公因数就可以得到p了，因为$n = p * q$ 也可以看成 $n = k_2 * p$，而k_1一般不等于k_2 所以就可以得到p。具体情况请见羊城杯 2021Bigrsa 共享素数。

- 计算机基础
  - 把 “我喜欢密码学” 转化为`整数`——**编码**

这看起来有点不可思议，文字和数字怎么联系在一起，但计算机上就可以，计算机底层是一堆二进制，只有0和1 。要想在屏幕上展现文字，需要”编码“，编码将数字按照规定的表转换成对应的文字，常见的编码有ASCII表。

二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形
0010 0000	32	20	(space)	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(	0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29	)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k
0010 1100	44	2C	,	0100 1100	76	4C	L	0110 1100	108	6C	l
0010 1101	45	2D	–	0100 1101	77	4D	M	0110 1101	109	6D	m
0010 1110	46	2E	.	0100 1110	78	4E	N	0110 1110	110	6E	n
0010 1111	47	2F	/	0100 1111	79	4F	O	0110 1111	111	6F	o
0011 0000	48	30	0	0101 0000	80	50	P	0111 0000	112	70	p
0011 0001	49	31	1	0101 0001	81	51	Q	0111 0001	113	71	q
0011 0010	50	32	2	0101 0010	82	52	R	0111 0010	114	72	r
0011 0011	51	33	3	0101 0011	83	53	S	0111 0011	115	73	s
0011 0100	52	34	4	0101 0100	84	54	T	0111 0100	116	74	t
0011 0101	53	35	5	0101 0101	85	55	U	0111 0101	117	75	u
0011 0110	54	36	6	0101 0110	86	56	V	0111 0110	118	76	v
0011 0111	55	37	7	0101 0111	87	57	W	0111 0111	119	77	w
0011 1000	56	38	8	0101 1000	88	58	X	0111 1000	120	78	x
0011 1001	57	39	9	0101 1001	89	59	Y	0111 1001	121	79	y
0011 1010	58	3A	:	0101 1010	90	5A	Z	0111 1010	122	7A	z
0011 1011	59	3B	;	0101 1011	91	5B	[	0111 1011	123	7B	{
0011 1100	60	3C	<	0101 1100	92	5C	\	0111 1100	124	7C	|
0011 1101	61	3D	=	0101 1101	93	5D	]	0111 1101	125	7D	}
0011 1110	62	3E	>	0101 1110	94	5E	^	0111 1110	126	7E	~
0011 1111	63	3F	?	0101 1111	95	5F	_				

有了编码，我们就能把想保护传输的信息转换成整数，再通过RSA算法加密计算得到密文c。

$$c = m^e \pmod{n}$$

2.代码实现

- 加密脚本（不需要掌握熟背，只用知道每一行是什么意思就行）

```
# 导入Crypto.Util.number模块中的所有函数
from Crypto.Util.number import *
# 导入gmpy2模块，用于高性能的数学运算
import gmpy2
# 从秘密模块导入flag，通常这是CTF挑战中的隐藏信息
from 秘密 import flag

# 这是给出了flag的样式，并不是真正的flag，但已经提供了一些提示，有些题目会根据flag头来破解
```

```
flag = b"saga131{******}"

# 生成两个256位的质数p和q
p = getPrime(256)
q = getPrime(256)

# 计算n，即p和q的乘积，用于RSA算法的模数
n = p*q
# 定义公钥指数e，通常为65537
e = 65537
# 将flag转换为长整数
m = bytes_to_long(flag)

# 使用gmpy2模块的powmod函数进行模幂运算，加密消息得到密文c
c = gmpy2.powmod(m, e, n)

# 打印模数n
print(f'n = {n}')
# 打印公钥指数e
print(f'e = {e}')
# 打印加密后的密文c
print(f'c = {c}')
# 打印质数p（通常在CTF中不会直接给出）
print(f'p = {p}')
# 打印质数q（通常在CTF中不会直接给出）
print(f'q = {q}')
"""
# 以下是打印的结果，这些值通常在CTF挑战中给出
n =
402494157468012450231636398154705109803267753152845716685967026186172831308128263
566402389053403458655684549432349768392381391573923446647239626132060048 3
e = 65537
c =
226967182640114431119923862488626190608050511354278604627242247124377735518111678
279381846350389469161980779137969837090666693533616114290831130137310320
p = 626583158329096604786858721118702336860354970630735587389802252143513861989 39
q = 642363510920625159459987294971535321400678618360881952422579762174992524606 97
"""
```

- 解密脚本

```
# 导入Crypto.Util.number模块中的所有函数，用于处理数字和字节之间的转换等
from Crypto.Util.number import *

"""输入部分"""
# 已知的模数n，用于RSA加密和解密
n =
402494157468012450231636398154705109803267753152845716685967026186172831308128263
566402389053403458655684549432349768392381391573923446647239626132060048 3
# 已知的公钥指数e，通常为65537
e = 65537
# 已知的密文c，需要被解密
c =
226967182640114431119923862488626190608050511354278604627242247124377735518111678
279381846350389469161980779137969837090666693533616114290831130137310320
```

```
# 已知的质数p，用于计算私钥
p = 626583158329096604786858721118702336860354970630735587389802252143513861989390
# 已知的质数q，用于计算私钥
q = 642363510920625159459987294971535321400678618360881952422579762174992524606970

"""处理部分"""
# 计算欧拉函数ϕ(n)，用于RSA算法中的私钥计算
phi = (p-1)*(q-1)
# 计算私钥指数d，即 e在模ϕ(n)的逆元
d = inverse(e,phi)
# 使用私钥指数d解密密文c，得到明文m     具体就是 m = c**d (modn)
m = pow(c,d,n)
# 将解密后的长整数m转换回字符串，得到原始的flag信息
flag = long_to_bytes(m)

"""输出部分"""
# 打印解密后的flag信息
print(flag)
```

六、以后的路

入门RSA算法后，将跟着NSSCTF的工坊课程自主继续深造RSA，在这基础上一起学习AES对称加密、圆锥曲线算法、LCG流密码、格密码和DSA密码等等。

七、写在最后的话

密码路上道阻且长，希望兄弟们能坚持下去，必定能见到密码学的彩虹！