CTF Crypto for beginners

Focus on RSA

NeSE Sikesibian

2024/1

Contents

密码学概述

什么是CTF Crypto?

什么是密码学?

密码学/CTF Crypto常用工具

数学基础

整除与同余

数论定理

二次剩余

经典公钥密码算法RSA

RSA算法简介

RSA经典分析方法

密码学概述

CTF Crypto 方向

CTF竞赛中的Crypto方向是涉及密码学相关挑战和任务的部分。包括密码分析、对称加密、非对称加密、数字签名和认证以及编码与编码理论等方面的问题。参赛者需要通过解密密文、破解算法、验证数字签名等方式来应对密码学挑战。参与Crypto方向的CTF竞赛需要对密码学的原理和常见算法有一定了解，并具备数学和逻辑思维能力。通过解决这些密码学问题，参赛者可以提高对密码学知识的理解，同时也锻炼密码学相关问题的解题能力。

密码学

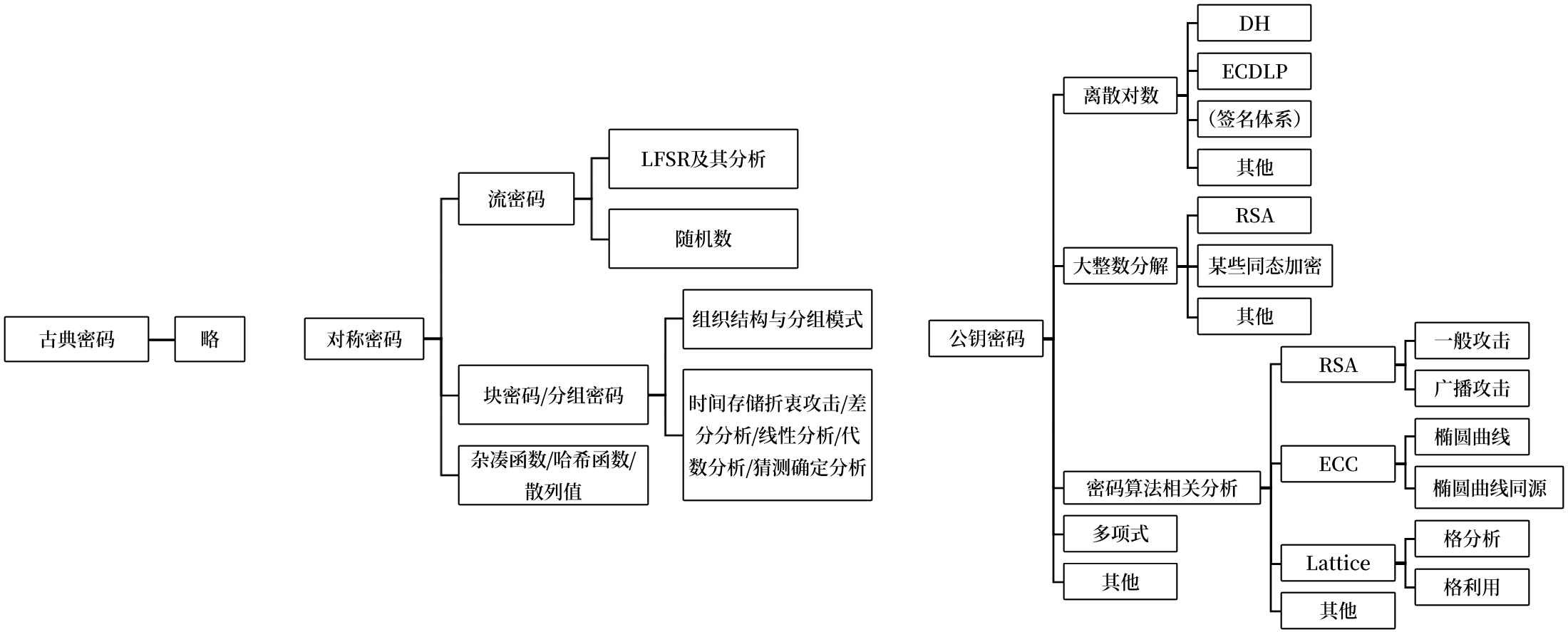
密码学是研究信息安全和数据保护的科学。它涉及设计、分析和应用加密算法以及相关的通信协议和系统。密码学的主要目标是确保在不安全的环境中传递敏感信息时，能够保证信息的机密性、完整性和身份验证。

其主要包括两个核心方面：加密和解密。加密是将原始数据转化为无法理解或解读的形式，以保证数据在传输或存储过程中不被未经授权的人所访问或修改。解密则是将加密后的数据恢复成可读的原始格式。密码一般分为对称加密和非对称加密两种基本类型。对称加密使用同一个密钥进行加密和解密，速度较快，但需要安全地共享密钥。非对称加密使用公钥和私钥进行加密和解密，相对更安全，但速度较慢。

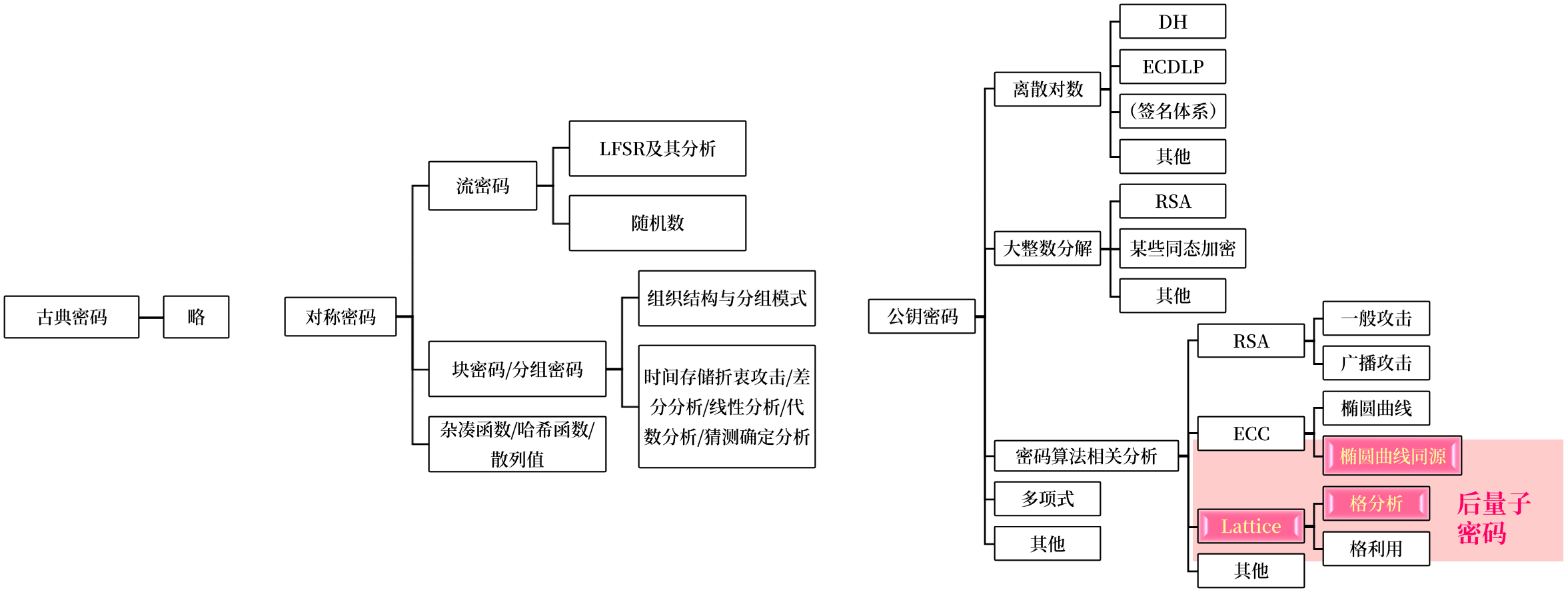
除了加密和解密，密码学还涉及数字签名、哈希函数、密钥交换协议等其他关键概念和方法。数字签名用于验证数据的完整性和真实性，哈希函数用于生成固定长度的摘要以验证数据完整性，密钥交换协议用于安全地共享密钥等。

密码学在现代通信和信息系统中起着重要的作用，应用于网络安全、电子支付、数据传输以及保护个人隐私等领域。它是确保数据安全性和保护隐私的基础，并且随着技术的发展和需求的增加，密码学也在不断演化和创新。

CTF涉及的主要密码学方向



CTF涉及的主要密码学方向



Some Useful Tools

Python3 脚本语言，我们常常使用这个语言来编写Crypto漏洞利用脚本。建议安装的库：Crypto库，gmpy2库，Sympy库，z3库。

Sagemath开源数学软件系统，基于python语言构建，支持很多高等数学运算。

数学基础

CTF Crypto 需要的数学基础

- 代数（线性代数+近世代数……）
- 初等数论
- ……

初等数论

整除与同余

数论定理

二次剩余初步

$a =$ 整除? $qb + r$ 带余除法 ⟶ 同余理论

$a =$ 整除? $p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ 素因子分解

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

欧几里得性质

$a =$ 整除?

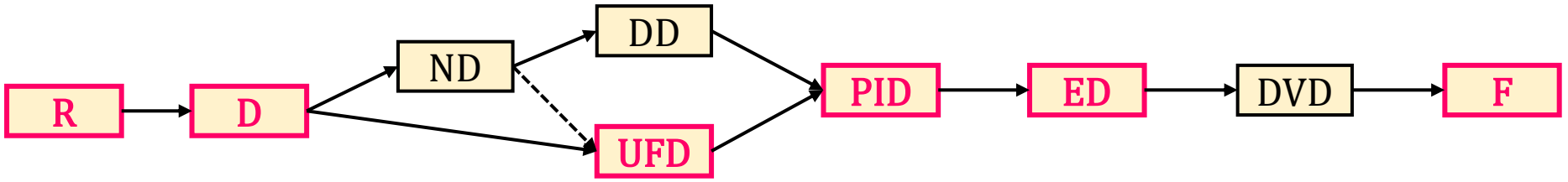
$qb + r$ 带余除法 ⟶ 同余理论

整环

$p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ 素因子分解

唯一因子分解

素元，不可约元



注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

欧几里得性质

$a =$ 整除? $qb + r$ 带余除法 ⟶ 同余理论

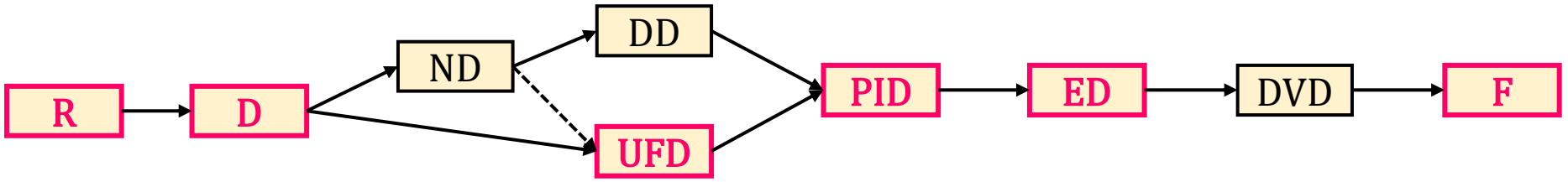
辗转相除法与裴蜀定理

$p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ 素因子分解

整环

唯一因子分解

素元，不可约元



注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

辗转相除法与裴蜀定理

欧几里得算法就是算法之祖，是算法界的活化石——高德纳

裴蜀定理

设 $g=\gcd(a,b)$，则存在整数 u,v 使得 $ua+vb=g$

扩展欧几里得算法

`extgcd(a, b)` 返回满足 `ua+vb=(a, b)` 的 `u, v`

`extgcd(b, a-qb)` 返回满足 `u'b+v'(a-qb)=(b, a-qb)` 的 `u', v'`

$$ua+vb=(a,b)=(b,a-qb)=v'a+(u'-qv')b$$

练习：分别用递归和递推方法实现扩展欧几里得算法
（思考初始条件终止条件是什么）

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

辗转相除法与裴蜀定理

整除与同余

数论定理

二次剩余初步

$$a_i = q_i b_i + r_i$$

$$a_{i+1} = q_{i+1} b_{i+1} + r_{i+1}$$

(a_i, b_i)

↓

(a_{i+1}, b_{i+1})

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

辗转相除法与裴蜀定理

练习：实现矩阵形式的扩展欧几里得算法
（思考初始初始向量和终止向量是什么）

$$a_i = q_i b_i + r_i$$

$$a_{i+1} = q_{i+1} b_{i+1} + r_{i+1}$$

(a_i, b_i)

↓

(a_{i+1}, b_{i+1})

$$\begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \begin{pmatrix} a_i \\ b_i \end{pmatrix} = \begin{pmatrix} b_i \\ r_i \end{pmatrix} = \begin{pmatrix} a_{i+1} \\ b_{i+1} \end{pmatrix}$$

$$\begin{pmatrix} a_i \\ b_i \end{pmatrix} = \begin{pmatrix} q_i & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_{i+1} \\ b_{i+1} \end{pmatrix}$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

辗转相除法与裴蜀定理

扩展欧几里得算法的应用

- 求最小公倍数
- 求线性序列周期
- 求特殊方程解的个数
- 求模逆
- …

拓展点

- 欧式环、UFD上的 gcd 算法，有什么区别?
- 二进制改进
- Lehmer 改进
- 多项式 gcd（Half-GCD）
- 近似 gcd 问题
- …

注：考虑到大家已经学习过离散数学或线性代数 I，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

欧拉定理 若整数 a, n 满足 $\gcd(a, n) = 1$，那么 $a^{\varphi(n)} \equiv 1 \pmod{n}$

环 R

↓ 单位（即可逆元）

单位群 G ⟶（所有元素左乘 $a \in G$）**陪集 $a \cdot G = G$**

$$\prod_{g \in G} g = \prod_{g \in G} ag = a^{|G|} \prod_{g \in G} g$$

$$1 = a^{|G|}$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

练习：$n = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$的欧拉函数是什么？

练习：$n = p \cdot q$的欧拉函数是什么？

整除与同余

欧拉定理

若整数 a, n 满足 $\gcd(a, n) = 1$，那么 $a^{\varphi(n)} \equiv 1 \pmod{n}$

数论定理

$|\mathbb{Z}_n| = n$ 环 R

单位（即可逆元）

$|\mathbb{Z}_n^\times| = \varphi(n)$ 单位群 G ⟶ 陪集 $a \cdot G = G$

所有元素左乘 $a \in G$

二次剩余初步

$$\prod_{g \in G} g = \prod_{g \in G} ag = a^{|G|} \prod_{g \in G} g$$

$$1 = a^{|G|}$$

$$1 = a^{\varphi(n)}$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

费马小定理：若整数 a，素数 p 满足 $\gcd(a,p)=1$，那么 $a^{p-1}\equiv 1 \pmod p$

↑

欧拉定理：若整数 a,n 满足 $\gcd(a,n)=1$，那么 $a^{\varphi(n)}\equiv 1 \pmod n$

$|\mathbb{F}_p| = p$ 环 R

↓ 单位（即可逆元）

$|\mathbb{F}_p^*| = p-1$ 单位群 G ⟶ 陪集 $a \cdot G = G$

所有元素左乘 $a \in G$

$$\prod_{g\in G} g = \prod_{g\in G} ag = a^{|G|}\prod_{g\in G} g$$

$$1 = a^{|G|}$$

$$1 = a^{p-1}$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

练习：实现快速模幂算法（或称重复平方法）

费马小定理

若整数 a，素数 p 满足 $\gcd(a,p)=1$，那么 $a^{p-1}\equiv 1 \pmod p$

有整数 a，素数 p，那么 $a^p \equiv a \pmod p$

↑

欧拉定理

若整数 a,n 满足 $\gcd(a,n)=1$，那么 $a^{\varphi(n)}\equiv 1 \pmod n$

$|\mathbb{F}_p| = p$ 环 R

↓ 单位（即可逆元）

$|\mathbb{F}_p^*| = p-1$ 单位群 G ⟶ 陪集 $a \cdot G = G$

所有元素左乘 $a \in G$

$$\prod_{g\in G} g = \prod_{g\in G} ag = a^{|G|}\prod_{g\in G} g$$

$$1 = a^{|G|}$$

$$1 = a^{p-1}$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

费马小定理

欧拉定理

卡米歇尔定理

练习：实现快速模幂算法（或称重复平方法）

若整数 a，素数 p 满足 $\gcd(a,p)=1$，那么 $a^{p-1}\equiv 1 \pmod p$

有整数 a，素数 p，那么 $a^p \equiv a \pmod p$

若整数 a,n 满足 $\gcd(a,n)=1$，那么 $a^{\varphi(n)}\equiv 1 \pmod n$

若整数 a,n 满足 $\gcd(a,n)=1$，那么 $a^{\lambda(n)}\equiv 1 \pmod n$

其中 $\lambda(n)=\operatorname{lcm}(\operatorname{ord}(g\in G))$

拓展点

- 素性检验
- 同余方程开方
- 积性函数
- 欧拉恒等式
- 求逆元
- …

$p\equiv 3 \pmod 4$：

$$a^p \equiv a \pmod p \Rightarrow a^{p+1}\equiv a^2 \pmod p \Rightarrow \left(\pm a^{\frac{p+1}{4}}\right)^2 \equiv a \pmod p$$

$p\equiv 1 \pmod 4$：

Tonelli-Shanks 算法

模素数开方算法：

- AMM算法

模素数幂开方方法：

- AMM算法 + Hensel提升

注：考虑到大家已经学习过离散数学或线性代数 I，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

费马小定理

若整数 a，素数 p 满足 $\gcd(a,p)=1$，那么 $a^{p-1}\equiv 1 \pmod p$

威尔逊定理

素数 $p \Leftrightarrow (p-1)!\equiv -1 \pmod p$，$1<p\in\mathbb{N}$

- 方程 $x^{p-1}-1\equiv 0 \pmod p$
- $1\cdots p-1$恰好为这个方程的所有根
- 由韦达定理即得

有志青年可以尝试使用 Sylow 第三定理 进行证明☺

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

中国剩余定理

设正整数$m_1 \cdots m_k$两两互素，那么对于下述同余方程组

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \cdots \\ x \equiv a_k \pmod{m_k} \end{cases}$$

在模 $M = m_1 \cdots m_k$ 意义下有下述唯一整数解：

$$x \equiv a_1 M_1 M_1' + \cdots + a_k M_k M_k' \pmod{M}$$

其中 $M_i = \frac{M}{m_i}$，$M_i' = M_i^{-1} \pmod{m_i}$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

中国剩余定理

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \cdots \\ x \equiv a_k \pmod{m_k} \end{cases}$$

在模 $M = m_1 \cdots m_k$ 意义下有下述唯一整数解：

$$x \equiv a_1 M_1 M_1' + \cdots + a_k M_k M_k' \pmod{M}$$

其中 $M_i = \frac{M}{m_i}$，$M_i' = M_i^{-1} \pmod{m_i}$

$$\begin{bmatrix} x \\ a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} = a_1 \begin{bmatrix} M_1 M_1' \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \cdots + a_k \begin{bmatrix} M_k M_k' \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} + k \begin{bmatrix} M \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

中国剩余定理

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \cdots \\ x \equiv a_k \pmod{m_k} \end{cases}$$

在模 $M = m_1 \cdots m_k$ 意义下有下述唯一整数解：

$$x \equiv a_1 M_1 M_1' + \cdots + a_k M_k M_k' \pmod{M}$$

其中 $M_i = \frac{M}{m_i}$，$M_i' = M_i^{-1} \pmod{m_i}$

代数版本：幺环R，设$I_1, \cdots I_n$为R的理想，且当$i \neq j$时满足$I_i + I_j = R$，那么就存在下述环同构：

$$R/(I_1 \cap \cdots \cap I_n) \cong \prod_{i=1}^{n} (R/I_i)$$

拓展：Frobenius同态

练习：实现中国剩余定理解同余方程

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

二次剩余

设整数 $m>1$，$\gcd(m,a)=1$，若整数 a 满足 $x^2\equiv a \pmod m$ 可解，那么就把 a 称为对于模 m 的二次剩余，否则称为 m 的二次非剩余。

模p二次剩余

设素数 $p>2$，那么在模 p 的完全剩余系中，恰好有 $\frac{1}{2}(p-1)$ 个二次剩余，又恰好有 $\frac{1}{2}(p-1)$ 个二次非剩余。

勒让德符号

设素数 $p>2$，且 $p\nmid n$，那么引入如下记号：

$$\left(\frac{n}{p}\right)=\begin{cases}1, & n\text{为}p\text{的二次剩余}\\ 0, & n\text{为}p\text{的倍数}\\ -1, & n\text{为}p\text{的二次非剩余}\end{cases}$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

二次剩余

设整数 $m>1$，$\gcd(m,a)=1$，若整数 a 满足 $x^2\equiv a \pmod m$ 可解，那么就把 a 称为对于模 m 的二次剩余，否则称为 m 的二次非剩余。

模p二次剩余

设素数 $p>2$，那么在模 p 的完全剩余系中，恰好有 $\frac{1}{2}(p-1)$ 个二次剩余，又恰好有 $\frac{1}{2}(p-1)$ 个二次非剩余。

勒让德符号

设素数 $p>2$，且 $p\nmid n$，那么引入如下记号：

$$\left(\frac{n}{p}\right)=\begin{cases}1, & n\text{为}p\text{的二次剩余}\\ 0, & n\text{为}p\text{的倍数}\\ -1, & n\text{为}p\text{的二次非剩余}\end{cases}$$

欧拉判据

$$\left(\frac{n}{p}\right)\equiv n^{\frac{p-1}{2}}\ (mod\ p)$$

练习：实现勒让德符号

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

数论定理

二次剩余初步

二次剩余性质

下面暂时不考虑符号等于0的情况

对合性： $\frac{1}{\left(\frac{n}{p}\right)} = \left(\frac{n}{p}\right)$

积性： $\left(\frac{m}{p}\right)\left(\frac{n}{p}\right) = \left(\frac{mn}{p}\right)$

周期性： $\left(\frac{kp+m}{p}\right) = \left(\frac{m}{p}\right)$

对称性：

- 二次剩余 × 二次剩余 = 二次剩余
- 二次剩余 × 二次非剩余 = 二次非剩余
- 二次非剩余 × 二次非剩余 = 二次剩余

完全剩余系	=	完全剩余系
0	都乘上一个二次非剩余a →	0
+		+
二次剩余		二次非剩余
+		+
二次非剩余		二次剩余

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

二次互反律

两个不等的奇素数 p, q 一定满足：

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}}$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

二次互反律

两个不等的奇素数 p, q 一定满足：

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}}$$

雅克比符号

在允许素因子重复的情况下设：

$$m = \prod_{i=1}^{t} p_i$$

那么可以定义对于互素的正整数 m, n，雅克比符号为：

$$\left(\frac{n}{m}\right) = \prod_{i=1}^{t}\left(\frac{n}{p_i}\right)$$

类似的结论

两个互素的正奇数 m, n 一定满足：

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}$$

思考：这也是二次互反律吗？
（拓展：高斯和，雅克比和）

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

如何在未知分解的情况下求解雅克比符号

二次剩余刻画了二次同余方程的可解性

两个互素的正奇数 m, n 一定满足：

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

如何在未知分解的情况下求解雅克比符号

二次剩余刻画了二次同余方程的可解性

两个互素的正奇数 m,n 一定满足：

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right)=(-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}$$

即：

$$\left(\frac{m}{n}\right)=(-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}\left(\frac{n}{m}\right)$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

如何在未知分解的情况下求解雅克比符号

二次剩余刻画了二次同余方程的可解性

两个互素的正奇数 m,n 一定满足：

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right)=(-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}$$

即：

$$\left(\frac{m}{n}\right)=(-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}\left(\frac{n}{m}\right)$$

$$\left(\frac{12345}{54323}\right)=(-1)^{\frac{1}{4}\times 12344\times 54322}\left(\frac{54323}{12345}\right)=\left(\frac{4\times 12345+4943}{12345}\right)$$

$$=\left(\frac{4943}{12345}\right)=\left(\frac{2459}{4943}\right)=-\left(\frac{25}{2459}\right)=-\left(\frac{9}{25}\right)=-\left(\frac{7}{9}\right)=-\left(\frac{2}{7}\right)$$

$$=-1$$

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

如何在未知分解的情况下求解雅克比符号

二次剩余刻画了二次同余方程的可解性

两个互素的正奇数 m,n 一定满足：

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right)=(-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}$$

即：

$$\left(\frac{m}{n}\right)=(-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}\left(\frac{n}{m}\right)$$

出现了，辗转相除法！

$$\left(\frac{12345}{54323}\right)=(-1)^{\frac{1}{4}\times 12344\times 54322}\left(\frac{54323}{12345}\right)=\left(\frac{4\times 12345+4943}{12345}\right)$$

$$=\left(\frac{4943}{12345}\right)=\left(\frac{2459}{4943}\right)=-\left(\frac{25}{2459}\right)=-\left(\frac{9}{25}\right)=-\left(\frac{7}{9}\right)=-\left(\frac{2}{7}\right)$$

$$=-1$$

练习：实现雅克比符号的计算
（注意前面提到的定理并不完全，
注意考虑偶素数和负数的情况）

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

初等数论

整除与同余

- 辗转相除法与裴蜀定理
- 同余的各种性质
- …

数论定理

- 欧拉函数的计算
- 欧拉定理、费马小定理（求逆，开方）
- 中国剩余定理的计算与理解

二次剩余初步

- 二次剩余的对称性
- 雅可比符号的计算
- …

注：考虑到大家已经学习过离散数学或线性代数Ⅰ，这一部分初等数论仅作为简单回顾

经典公钥密码算法RSA

What is RSA?

公钥密码算法

- 密钥对生成算法 gen()
- 加密算法 Enc()
- 解密算法 Dec()

What is RSA?

公钥密码算法

RSA

密钥对生成算法 gen()

- 选取两个不相等**大素数** p 和 q，计算**公钥-模数** $n = p \cdot q$。
- 顺便计算 n 的**欧拉函数** $\varphi(n) = (p-1)(q-1)$。
- 选取正整数 e 作为**公钥-加密指数**。一般要求 $1 < e < \varphi(n)$ 以及 $\gcd(e, \varphi(n)) = 1$。
- 获取**私钥-解密指数** $d = e^{-1} (\mathrm{mod}\ \varphi(n))$。

加密算法 Enc()

- 对明文进行预处理，一般是**执行编码操作**
- 计算 $c = m^e \ (\mathrm{mod}\ n)$，得到的 c 作为**密文**

解密算法 Dec()

- 计算 $m = c^d \ (\mathrm{mod}\ n)$，得到的 m 解码作为**明文**

$$\because \boldsymbol{d \equiv e^{-1} \ (mod\ \varphi(n))}$$

$$\therefore \exists \boldsymbol{k} \in \mathbb{Z}, \boldsymbol{ed = k \cdot \varphi(n) + 1}$$

$$\therefore \boldsymbol{c^d \equiv (m^e)^d \equiv m^{ed}}$$

$$\therefore \boldsymbol{c^d \equiv m^{k\varphi(n)+1} \equiv \left(m^{\varphi(n)}\right)^k \cdot m \equiv m \ (mod\ n)}$$

What is RSA?

公钥密码算法

RSA

密钥对生成算法 gen()

- 选取两个不相等**大素数** p 和 q，计算**公钥-模数** $n = p \cdot q$。
- 顺便计算 n 的**欧拉函数** $\varphi(n) = (p-1)(q-1)$。
- 选取正整数 e 作为**公钥-加密指数**。一般要求 $1 < e < \varphi(n)$ 以及 $\gcd(e, \varphi(n)) = 1$。
- 获取**私钥-解密指数** $d = e^{-1} (\mathrm{mod}\ \varphi(n))$。

加密算法 Enc()

- 对明文进行预处理，一般是**执行编码操作**
- 计算 $c = m^e \ (\mathrm{mod}\ n)$，得到的 c 作为**密文**

解密算法 Dec()

- 计算 $m = c^d \ (\mathrm{mod}\ n)$，得到的 m 解码作为**明文**

练习：下面这个RSA算法正确性的证明有什么不完整的地方？

$$\because \boldsymbol{d \equiv e^{-1} \ (mod\ \varphi(n))}$$

$$\therefore \exists \boldsymbol{k} \in \mathbb{Z}, \boldsymbol{ed = k \cdot \varphi(n) + 1}$$

$$\therefore \boldsymbol{c^d \equiv (m^e)^d \equiv m^{ed}}$$

$$\therefore \boldsymbol{c^d \equiv m^{k\varphi(n)+1} \equiv \left(m^{\varphi(n)}\right)^k \cdot m \equiv m \ (mod\ n)}$$

What is RSA?

公钥密码算法

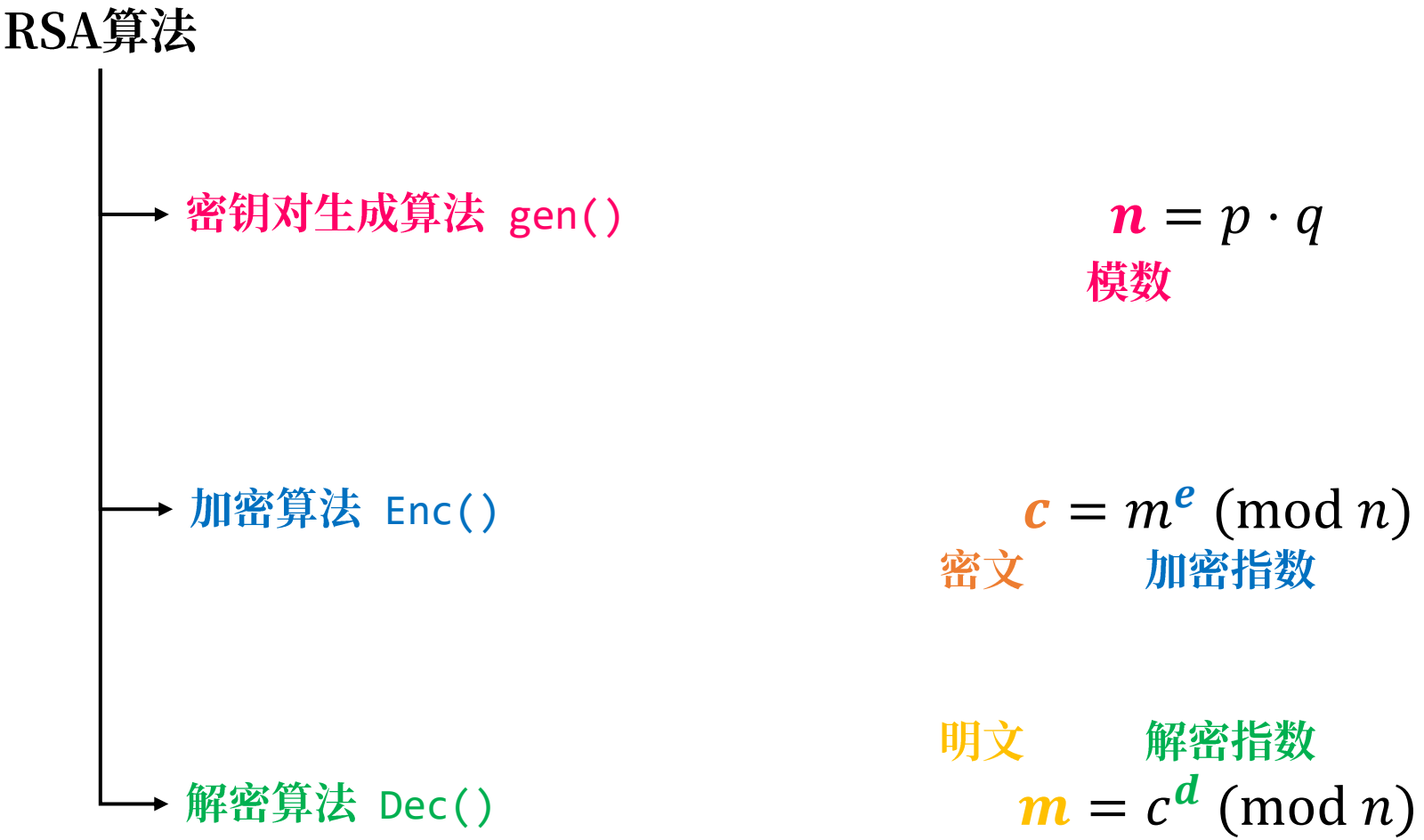
- 密钥对生成算法 gen()
- 加密算法 Enc()
- 解密算法 Dec()

RSA例程

```
from Crypto.Util.number import*
class my_RSA:
    def gen_n(BITS):
        p = getPrime(BITS)
        q = getPrime(BITS)
        n = p*q
        return n,p,q

    def encrypt(plaintext, e, n):
        m = bytes_to_long(plaintext)
        c = pow(m, e, n)
        return c

    def decrypt(p, q, e, n, c):
        phi = (p - 1) * (q - 1)
        d = inverse(e, phi)
        m = pow(c, d, n)
        pt = long_to_bytes(m)
        return pt

plaintext=b"test_RSA"
n, p, q=my_RSA.gen_n(32)
e = 0x10001
print("public key = ", (n, e))
print("c = ", my_RSA.encrypt(plaintext, e, n))
print("Your plaintext is: ", my_RSA.decrypt(p, q, e, n, my_RSA.encrypt(plaintext, e, n)))
```

- 选取两个不相等**大素数** p 和 q，计算**公钥-模数** $n = p \cdot q$。
- 对明文进行预处理，一般是**执行编码操作**
- 计算 $c = m^e \pmod{n}$，得到的 c 作为**密文**
- 计算 n 的**欧拉函数** $\varphi(n) = (p-1)(q-1)$。
- 获取**私钥-解密指数** $d = e^{-1} \pmod{\varphi(n)}$。
- 计算 $m = c^d \pmod{n}$，得到的 m 解码作为**明文**
- 选取正整数 e 作为**公钥-加密指数**。一般要求 $1 < e < \varphi(n)$ 以及 $\gcd(e, \varphi(n)) = 1$。

What is RSA?


RSA算法
密钥对生成算法 gen()
$n = p \cdot q$
模数
加密算法 Enc()
$c = m^e \pmod{n}$
密文
加密指数
明文
解密指数
解密算法 Dec()
$m = c^d \pmod{n}$

What is RSA?

RSA算法

- 密钥对生成算法 gen()

$$n = p \cdot q$$

模数

模数的素因子分解方法

（Fermat，光滑数，比特信息泄露，指数分解模数，线性特征素因子，CopperSmith）

- 加密算法 Enc()

选择密文攻击

任意密文解密

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

密文　　加密指数

低加密指数

Rabin

加密指数与欧拉函数不互素

- 解密算法 Dec()

选择明文攻击

明文　　解密指数

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

*d泄露

*dp泄露

*连分数方法（过时但是重要）

*CopperSmith方法系列

What is RSA?

RSA算法

- 密钥对生成算法 gen()
  - $n_1 = p_1 \cdot q_1$ 模数 1
  - $n_2 = p_2 \cdot q_2$
- 加密算法 Enc()
  - $c_1 = m_1^{e_1} \pmod{n_1}$ 密文 加密指数
  - $c_2 = m_2^{e_2} \pmod{n_2}$
- 解密算法 Dec()
  - $m_1 = c_1^{d_1} \pmod{n_1}$ 明文 解密指数
  - $m_2 = c_2^{d_2} \pmod{n_2}$

What is RSA?

RSA算法

模数的素因子分解方法
（Fermat，光滑数，比特信息泄露，指数分解模数，线性特征素因子，CopperSmith）

密钥对生成算法 gen()

$n_1 = p_1 \cdot q_1$ 模数 1

$n_2 = p_2 \cdot q_2$

广播攻击
模不互素，共模攻击，多项式

加密算法 Enc()

$c_1 = m_1^{e_1} \pmod{n_1}$

密文 加密指数

$c_2 = m_2^{e_2} \pmod{n_2}$

选择密文攻击
任意密文解密

低加密指数
Rabin
加密指数与欧拉函数不互素

明文 解密指数

解密算法 Dec()

$m_1 = c_1^{d_1} \pmod{n_1}$

$m_2 = c_2^{d_2} \pmod{n_2}$

选择明文攻击

*d泄露
*dp泄露
*连分数方法（过时但是重要）
*CopperSmith方法系列

直接模数分解

- **分解大数的网站：**factordb.com 它可以分解一些小的大数，它的数据库里也存有一些已经被分解出来的大素数
- 其他**分解大数的网站：**https://www.alpertron.com.ar/ECM.HTM、https://www.wolframalpha.com/
- **分解大数的强力工具yafu：**yafu（windows系统下）具体的一些安装操作可参考CSDN上的介绍
- Cado-nfs：一个数域筛算法的实现，可以用于分解大整数和解离散对数
- sagemath上的一些函数也可以帮助我们分解大数，如 factor 等等

直接模数分解

- **分解大数的网站：**factordb.com 它可以分解一些小的大数，它的数据库里也存有一些已经被分解出来的大素数
- 其他**分解大数的网站：**https://www.alpertron.com.ar/ECM.HTM、https://www.wolframalpha.com/
- **分解大数的强力工具yafu：**yafu（windows系统下）具体的一些安装操作可参考CSDN上的介绍
- Cado-nfs：一个数域筛算法的实现，可以用于分解大整数和解离散对数
- sagemath上的一些函数也可以帮助我们分解大数，如 factor 等等

- **yafu的应用**

```
factor(12772956926542139933)
```

- **sagemath的应用**

```
factor(12772956926542139933)
```

- **python的应用**

```
from sympy.ntheory import factorint
n = 12772956926542139933
print(factorint(n))
'''
{3513657493: 1, 3635231081: 1}
'''
```

注：yafu会把结果储存在同目录的文件factor.log里

```
Starting factorization of 12772956926542139933
using pretesting plan: normal
no tune info: using qs/gnfs crossover of 95 digits
**************************
Fermat method found factors:
prp10 = 3635231081
prp10 = 3513657493
Total factoring time = 0.0000 seconds
```

我们可以看到yafu利用费马方法获得了两个素因子3635231081和3513657493

练习：如果已知模数是一个素数或者是一个素数的幂（这常常是算法上好探测的），那么应该如何解密？

直接模数分解——费马分解法

因子分解算法之祖，几乎所有整数环上的因子分解算法都是由费马分解法启发而来。——沃兹基硕德

$$n = p \cdot q \longrightarrow n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

直接模数分解——费马分解法

因子分解算法之祖，几乎所有整数环上的因子分解算法都是由费马分解法启发而来。——沃兹基硕德

$$n = p \cdot q \longrightarrow n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

$$\left(\frac{p-q}{2}\right)^2 = \left(\frac{p+q}{2}\right)^2 - n$$

很小，可以对 $p+q$ 进行枚举

$$n + \left(\frac{p-q}{2}\right)^2 = \left(\frac{p+q}{2}\right)^2$$

很小，可以对 $p-q$ 进行枚举

直接模数分解——费马分解法

因子分解算法之祖，几乎所有整数环上的因子分解算法都是由费马分解法启发而来。——沃兹基硕德

$$n = p \cdot q \longrightarrow n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

$$\left(\frac{p-q}{2}\right)^2 = \left(\frac{p+q}{2}\right)^2 - n$$

很小，可以对 $p+q$ 进行枚举

我们选择这种方法 ✓

$$n + \left(\frac{p-q}{2}\right)^2 = \left(\frac{p+q}{2}\right)^2$$

很小，可以对 $p-q$ 进行枚举

直接模数分解——费马分解法

因子分解算法之祖，几乎所有整数环上的因子分解算法都是由费马分解法启发而来。——沃兹基硕德

$$n = p \cdot q \longrightarrow n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

$$\left(\frac{p-q}{2}\right)^2 = \left(\frac{p+q}{2}\right)^2 - n$$

也很小　　很小，可以进行枚举

方法： 从 $x = \lceil \sqrt{n} \rceil$ 开始逐渐增大 x，考察 $x^2 - n$ 是否为一个平方数即可。

练习：为什么不选择上一页右边那种方法？
（提示：从微分的角度进行考虑，同时可以留意等式两边的数量级）

直接模数分解——费马分解法

因子分解算法之祖，几乎所有整数环上的因子分解算法都是由费马分解法启发而来。——沃兹基硕德

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

平方同余式

方法： 从 $x = \lceil \sqrt{n} \rceil$ 开始逐渐增大 x，考察 $x^2 - n$ 是否为一个平方数即可。

连分数法	根据 $\sqrt{n}$ 的简单连分数展开来寻找同余式 $x^2 \equiv y^2 \pmod{n}$
二次筛法	令 $f(x) = (x + \sqrt{n})^2 - n$，寻找光滑的一系列 x_i 使得 $\prod f(x_i)$ 为平方数
数域筛法	构造数域，找环上的平方元，然后根据同态建立平方同余关系

直接模数分解——特殊分解法：光滑数分解

光滑数

B-光滑数指对正整数 B，如果自然数 N 的素因子都不大于 B，那么就称 N 是一个 B-光滑数。

直接模数分解——特殊分解法：光滑数分解

光滑数

B-光滑数指对正整数 B，如果自然数 N 的素因子都不大于 B，那么就称 N 是一个 B-光滑数。

$p-1$ 光滑

$p+1$ 光滑

直接模数分解——特殊分解法：光滑数分解

光滑数

B-光滑数指对正整数 B，如果自然数 N 的素因子都不大于 B，那么就称 N 是一个 B-光滑数。

$p-1$ 光滑　　**Pollard's $p-1$ 算法**　　条件：$p-1$ 的素因子幂的次数足够低

```
import gmpy2
def pollard_pm1(n, ubound):
    a = 2
    B = 2
    while B < ubound:
        a = pow(a, B, n)
        p = gmpy2.gcd(a - 1, n)
        if (p != 1) and (p != n):
            q = n // p
            return p, q
        B += 1
```

$p+1$ 光滑

$$a\ (p \nmid a)$$

$$\downarrow \text{费马小定理}$$

$$a^{t(p-1)} \equiv 1\ (mod\ p) \longrightarrow a^{t(p-1)} - 1 = k \cdot p$$

$$\downarrow$$

$$p \mid \gcd(a^{t(p-1)} - 1, n)$$

$$\downarrow p-1 \text{ 是 } B\text{-光滑的}$$

$$p \mid \gcd(a^{B!} - 1, n)$$

$$p = \gcd(a^{B!} - 1, n)\ ?$$

直接模数分解——特殊分解法：光滑数分解

光滑数

B-光滑数指对正整数 B，如果自然数 N 的素因子都不大于 B，那么就称 N 是一个 B-光滑数。

$p-1$ **光滑**	**Pollard's $p-1$ 算法**	条件：$p-1$ 的素因子幂的次数足够低
$p+1$ **光滑**	**William's $p+1$ 算法**	条件：$p+1$ 的素因子幂的次数足够低

思路：将费马小定理推广到二次域上去，利用卢卡斯序列进行构造

局部 $p-1$ 光滑

局部 $p+1$ 光滑

条件：$p\pm1$ 含有较大素因子，但其余部分光滑

Pollard's p − 1 algorithm - Wikipedia

直接模数分解——特殊分解法：比特信息泄露

LSB 搜索

折半查找

直接模数分解方法大赏

Continued fraction (CFRAC)	连分数方法
Dixon's	随机平方方法
Lenstra elliptic curve (ECM)	利用椭圆曲线上的光滑阶点
Pollard's rho	迭代多项式产生的随机数
Quadratic sieve (QS)	二次筛法
General number field sieve (GNFS)	一般数域筛法，即构造可以构造平方同余式的多项式的根
Special number field sieve (SNFS)	特殊数域筛法，改进和变体的一般数域筛法，适用于特定形式的整数，即 $t^e - s$
Fermat's	费马分解法
Shor's	基于量子计算，能在多项式时间内解决，但目前暂时无法在经典计算机上高效解决

现代整数分解算法的框架　　**选择因子基** → **收集光滑关系** → **解线性方程组** → **求解公因子**

低加密指数

非广播情形

- 模数太大，明文过小

$$m = \sqrt[e]{c + k \cdot n}$$

- 转化为多项式

$$f(x) = (m_1 x + m_0)^e \ (mod\ n)$$

CopperSmith?

$e|\varphi(n)$ 怎么办？

注：e 不能太大，最多20bit左右

$$\begin{cases} f(x) = (m_1 x + m_0)^e \pmod p \\ f(x) = (m_1 x + m_0)^e \pmod q \end{cases} \xrightarrow{\text{AMM}} \begin{cases} x_p \pmod p \\ x_q \pmod q \end{cases} \xrightarrow{\text{CRT}} x \pmod n$$

$$m = \sqrt[e]{c + k \cdot n} \qquad n = n_1 \cdots n_e$$

$$f_1(x) = (m_1 x + m_0)^e \ (mod\ n)$$
$$f_2(x) = (m_1' x + m_0')^{e'} \ (mod\ n)$$

低加密指数

- 模数太大，明文过小 $m = \sqrt[e]{c + k \cdot n}$ $n = n_1 \cdots n_e$
- 广播情形
  - 转化为多项式

$$f_1(x) = (m_1 x + m_0)^e \pmod{n}$$
$$f_2(x) = (m_1' x + m_0')^{e'} \pmod{n}$$

共指数，不共模

模不互素

$$n_1 = p \cdot q$$
$$n_2 = p \cdot r$$

Håstad 攻击

$$\begin{cases} c_1 = m^3 \pmod{n_1} \\ c_2 = m^3 \pmod{n_2} \\ c_3 = m^3 \pmod{n_3} \end{cases} \xrightarrow{\text{CRT}} m^3 \equiv c \pmod{n_1 n_2 n_3} \longrightarrow m = \sqrt[e]{c + k \cdot n}$$

$m < n_i \Rightarrow m^3 < n_1 n_2 n_3$

共指数，不共模　模不互素，Håstad 攻击

模数太大，明文过小　$m = \sqrt[e]{c + k \cdot n}$　$n = n_1 \cdots n_e$

低加密指数

广播情形

→ **转化为多项式**

$$f_1(x) = (m_1 x + m_0)^e \pmod{n}$$
$$f_2(x) = (m_1' x + m_0')^{e'} \pmod{n}$$

共指数，共模

Franklin-Reiter攻击：多项式 gcd

$$\begin{cases} f_1(x) = m^e \pmod{n} \\ f_2(x) = \big(p(m)\big)^e \pmod{n} \end{cases}$$

相关消息　$m_2 = p(m_1)$

p 为多项式

不共指数，共模

→ 共明文　裴蜀定理

→ 相关消息　多项式 gcd

共指数，不共模　模不互素，Håstad 攻击

模数太大，明文过小　$m = \sqrt[e]{c + k \cdot n}$　$n = n_1 \cdots n_e$

低加密指数

广播情形

→ **转化为多项式**

$$f_1(x) = (m_1 x + m_0)^e \pmod{n}$$
$$f_2(x) = (m_1' x + m_0')^{e'} \pmod{n}$$

共指数，共模

Franklin-Reiter攻击：多项式 gcd

$$\begin{cases} f_1(x) = m^e & \pmod{n} \\ f_2(x) = \big(p(m)\big)^e & \pmod{n} \end{cases}$$

相关消息　$m_2 = p(m_1)$

p 为多项式

不共指数，共模

- 共明文 → 裴蜀定理
- 相关消息 → 多项式 gcd

应用情景：LCG线性同余产生器与RSA结合

共指数，不共模　模不互素，Håstad 攻击

模数太大，明文过小　$m = \sqrt[e]{c + k \cdot n}$　$n = n_1 \cdots n_e$

低加密指数

广播情形

转化为多项式

$$f_1(x) = (m_1 x + m_0)^e \ (mod\ n)$$
$$f_2(x) = (m_1' x + m_0')^{e'} \ (mod\ n)$$

不共指数，不共模

SMUPE 攻击

$$\begin{cases} f_1(x) = m^{e_1} + \cdots (\mathrm{mod}\ n_1) \\ f_2(x) = m^{e_2} + \cdots (\mathrm{mod}\ n_2) \\ \vdots \\ f_k(x) = m^{e_k} + \cdots (\mathrm{mod}\ n_k) \end{cases} \xrightarrow{\text{多项式CRT}} F(x) = m^e + \cdots (\mathrm{mod}\ n_1 \cdots n_k)$$

CopperSmith 解小根

要求 $\sum_i \frac{1}{e_i} \geq 1$

练习：补充完成下表对于 RSA 广播攻击的总结

明文	（加密）指数	模数（解密指数）	可能可以使用的攻击方法
共	共	共	退化为非广播情形
共	共	不共	
共	不共	共	
共	不共	不共	
不共	共	共	
不共	共	不共	
不共	不共	共	
不共	不共	不共	

CopperSmith 定理

对于正整数 n，$f \in \mathbb{Z}[x]$为一个一元 d 次多项式，那么有算法可以帮助我们在多项式时间内，利用 n 和 f 的表达式，有效找到 $f(x_0) \equiv 0 \ (mod\ n)$ 的所有满足 $|x_0| < X$ 的整数解。其中 $X = N^{\frac{1}{d}-\varepsilon}$，$\varepsilon \geq 0$。

相关重要结论

设$f(x) = \sum_{i=0}^{d} a_i x^i$ 为一个 d 次多项式，定义其L2范数为$\|f(x)\|_2 = \sqrt{\sum_{i=0}^{d} a_i^2}$，此时设 $|x_0| < B$，

小根x_0

$f(x_0) \equiv 0 \ (\mathrm{mod}\ n)$，如果满足下述条件：

$$\|f(Bx)\|_2 = \sqrt{\sum_{i=0}^{d} (a_i B^i)^2} \leq \frac{n}{\sqrt{d+1}}$$

只要多项式范数和零点足够小。。。

那么一定成立 $f(x_0) = 0, x_0 \in \mathbb{Z}$。

$$\frac{n}{\sqrt{d+1}} \geq \sqrt{\sum_{i=0}^{d} (a_i B^i)^2} > \sqrt{\sum_{i=0}^{d} (a_i x_0^i)^2} \geq \frac{\left|\sum_{i=0}^{d} a_i x^i\right|}{\sqrt{d+1}} = \frac{|f(x_0)|}{\sqrt{d+1}}$$

CopperSmith 定理

对于正整数 n，$f \in \mathbb{Z}[x]$为一个一元 d 次多项式，那么有算法可以帮助我们在多项式时间内，利用 n 和 f 的表达式，有效找到 $f(x_0) \equiv 0 \ (mod\ n)$ 的所有满足 $|x_0| < X$ 的整数解。其中 $X = N^{\frac{1}{d}-\varepsilon}$，$\varepsilon \geq 0$。

使用 CopperSmith 及相关定理 CopperSmith 方法开启了现代密码学长达十余年的利用格进行密码分析的研究。

概述

如果我们预先知道了一组多项式 f_i，恰好满足 $f_i(x_0) \equiv 0 \ (\mathrm{mod}\ N)$，那么就可以把这一组多项式视为（或整合为）一组基底，而这一系列多项式的线性组合所构成的各个多项式都会以 x_0 为零点。

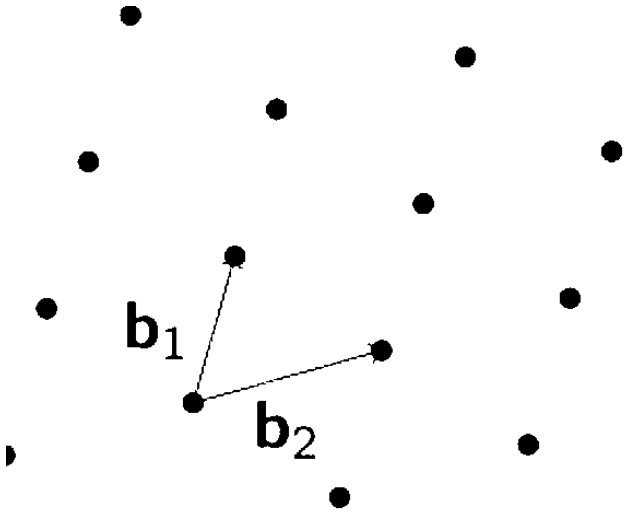
问题：如何找到一个向量空间中短的向量？

格基规约算法！

使用 CopperSmith 及相关定理 CopperSmith 方法开启了现代密码学长达十余年的利用格进行密码分析的研究。

如果我们预先知道了一组多项式 f_i，恰好满足 $f_i(x_0) \equiv 0 \pmod{N}$，那么就可以把这一组多项式视为（或整合为）一组基底，而这一系列多项式的线性组合所构成的各个多项式都会以 x_0 为零点。

格（有限秩自由 $\mathbb{Z}$ 模）：整系数向量空间 $\mathcal{L} = B \cdot \mathbb{Z}^n$



$$\mathcal{L} = (b_1 \quad b_2) \cdot \mathbb{Z}^2$$

格基规约

给定某一个格上的一组（线性无关）向量，求解这个格上长度较短的向量的过程

经典算法：LLL算法

```
sage: mat1 = random_matrix(ZZ, 5, 5)
sage: matLLL = mat1.LLL()
sage: matLLL
[-1  1  0 -1 -1]
[ 1 -2  2 -1 -1]
[ 1  2  3  2  0]
[-4 -2 -1  2 -1]
[ 4  0 -2  1 -5]
sage: mat1
[ -1   0   2  -3  -3]
[  0 -11   0   1   0]
[  0   2   5  -1  -3]
[ -1   1   0  -1  -1]
[ -1   2   0   3  -8]
```

注意：要将矩阵转换到ZZ（即整数环）上（使用 change_ring 这一方法）

使用 CopperSmith 及相关定理

CopperSmith 方法开启了现代密码学长达十余年的利用格进行密码分析的研究。

如果我们预先知道了一组多项式 f_i，恰好满足 $f_i(x_0) \equiv 0 \pmod N$，那么就可以把这一组多项式视为（或整合为）一组基底，而这一系列多项式的线性组合所构成的各个多项式都会以 x_0 为零点。

CopperSmith 定理推论

设 n 有某个未知因子 b，且 $b < N^{\beta}$，那么对于一元 d 次多项式 f，我们可以在 $\log Nd$ 的时间内找到满足 $f(x_0) \equiv 0 \pmod b$ 且 $x < n^{\frac{\beta^2}{d}}$ 的小根 x_0。

CopperSmith 能够作用于不确定的有限整环上

因子低位泄露：知道某个素因子 p 的低位

$$f(x) = 2^{\gamma}x + p_l \equiv 0 \ (mod\ p)$$

$$\downarrow$$

$$nf(x) = n(2^{\gamma}x + p_l) = p^2q \equiv 0 \ (mod\ n)$$

$$\downarrow$$

$$n^i f^j(x) = p^{i+j}q^i \equiv 0 \ (mod\ n)$$

对这一系列多项式构成的向量空间做格基规约

使用 CopperSmith 及相关定理

CopperSmith 方法开启了现代密码学长达十余年的利用格进行密码分析的研究。

如果我们预先知道了一组多项式 f_i，恰好满足 $f_i(x_0) \equiv 0 \pmod N$，那么就可以把这一组多项式视为（或整合为）一组基底，而这一系列多项式的线性组合所构成的各个多项式都会以 x_0 为零点。

CopperSmith 定理推论

设 n 有某个未知因子 b，且 $b < N^\beta$，那么对于一元 d 次多项式 f，我们可以在 $\log Nd$ 的时间内找到满足 $f(x_0) \equiv 0 \pmod b$ 且 $x < n^{\frac{\beta^2}{d}}$ 的小根 x_0。

因子低位泄露：知道某个素因子 p 的低位

$$f(x) = 2^\gamma x + p_l \equiv 0 \ (mod\ p)$$

Sagemath中有包装好的方法

```
#sagemath
n = # 模数
p_l = # 素因子低位
p_bits = 1024 # 素因子p的总位数
p_l_bits = p_l.nbits()
PR.<x> = PolynomialRing(Zmod(n))
f = 2 ^ p_l_bits * x + p_l
f = f.monic()  # 多项式首一化！十分重要的一步！！！
roots = f.small_roots(X = 2 ^ (p_bits - p_l_bits), beta = 0.4)  # 寻找小于 2^(p_bits-p_l_bits) 且大于等于 n^0.4 的因子
if roots:
    p = gcd(2 ^ p_l_bits * int(roots[0]) + p0, n)
    print(f"n = {n}\np = {p}\nq = {n//p} ")
```

使用 CopperSmith 及相关定理

CopperSmith 方法开启了现代密码学长达十余年的利用格进行密码分析的研究。

如果我们预先知道了一组多项式 f_i，恰好满足 $f_i(x_0) \equiv 0 \pmod N$，那么就可以把这一组多项式视为（或整合为）一组基底，而这一系列多项式的线性组合所构成的各个多项式都会以 x_0 为零点。

CopperSmith 定理推论

设 n 有某个未知因子 b，且 $b < N^\beta$，那么对于一元 d 次多项式 f，我们可以在 $\log Nd$ 的时间内找到满足 $f(x_0) \equiv 0 \pmod b$ 且 $x < n^{\frac{\beta^2}{d}}$ 的小根 x_0。

因子低位泄露：知道某个素因子 p 的低位

$$f(x) = 2^\gamma x + p_l \equiv 0 \ (mod\ p)$$

因子高位泄露：知道某个素因子 p 的高位

练习：留作练习

局部明文泄露：仅明文的一小部分不知道

练习：留作练习

解密指数低位泄露：知道解密指数的低位

α 一般至少为 512

$$f(x) = kx^2 + (ed_0 - kn - k)x + kn \equiv 0 \pmod{2^\alpha}$$

练习：推导上述多项式

使用 CopperSmith 及相关定理 CopperSmith 方法开启了现代密码学长达十余年的利用格进行密码分析的研究。

如果我们预先知道了一组多项式 f_i，恰好满足 $f_i(x_0) \equiv 0 \pmod N$，那么就可以把这一组多项式视为（或整合为）一组基底，而这一系列多项式的线性组合所构成的各个多项式都会以 x_0 为零点。

多元CopperSmith

Boneh-Durfee 攻击 —— 低解密指数（同时就会有高加密指数） $d < n^{0.292}$

脚本：*mimoo/RSA-and-LLL-attacks: attacking RSA via lattice reductions (LLL) (github.com)*

$$ed = k\varphi(n) + 1 = k(n - (p+q) + 1) + 1$$

$$\downarrow$$

$$-k(n - (p+q) + 1) \equiv 1 \ (mod\ e)$$

$$\downarrow \quad x = -k, y = -(p+q), A = n + 1$$

$$f(x, y) = x \cdot (A + y) \equiv 1 \ (mod\ e)$$

$$\dots$$

拓展点

- $(p^2-1)(q^2-1)$：二阶矩阵RSA，详见这篇论文开头的总结 *A new attack on some RSA variants*（该论文后面的证明有点小问题，可以先不管）
- 低解密指数攻击
- 解密指数泄露分解模数
- 维纳攻击与扩展维纳攻击

d_p 泄露 背景：在中国剩余定理加速RSA加密的过程中，往往会计算 $d_p = d \pmod{p-1}$

$$e \cdot d_p \equiv e \cdot d \equiv 1 \pmod{p-1}$$

$$\downarrow$$

$$e \cdot d_p - 1 = k \cdot (p-1)$$

随便选取一个 a

$$a^{e \cdot d_p} \pmod{n} = a^{k(p-1)+1} \pmod{n}$$

$$a^{e \cdot d_p} \pmod{p} = a^{k(p-1)+1} \pmod{p} = a$$

$$p \mid a^{e \cdot d_p} - a$$

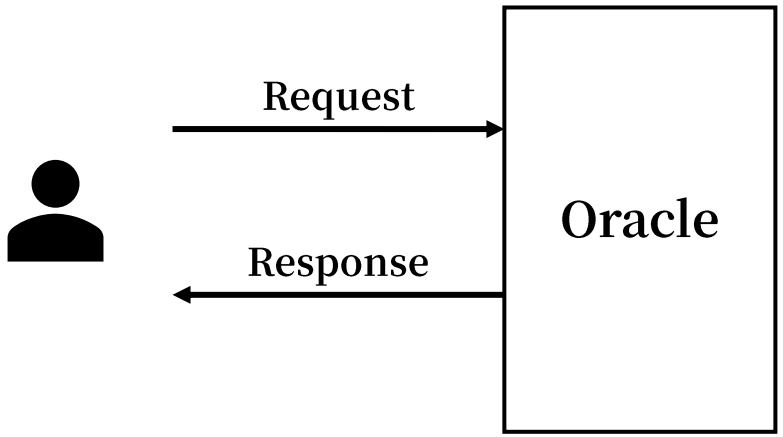
$$\downarrow$$ 极大概率

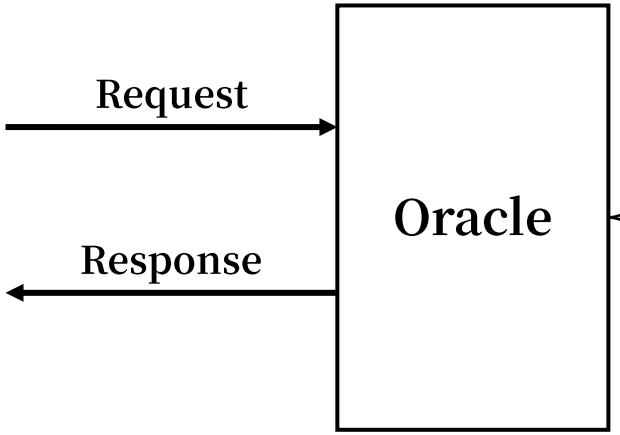
$$p = \gcd(a^{e \cdot d_p} - a, n)$$

拓展点

- 局部 d_p 泄露
- 已知 d_p 高位
- d_p 过小

Attacks on RSA





加密Oracle　　选择明文攻击

$$c_1 = 2^e \pmod n$$
$$c_2 = 4^e \pmod n \longrightarrow n \mid c_1^2 - c_2$$
$$c_3 = 8^e \pmod n \qquad n \mid c_1^3 - c_3$$

常用方法：特殊值

解密Oracle　　选择密文攻击

- 任意密文解密　　送入 x，返回 $x \cdot m$
- Parity Oracle 与 Byte Oracle

矩阵RSA

多项式RSA

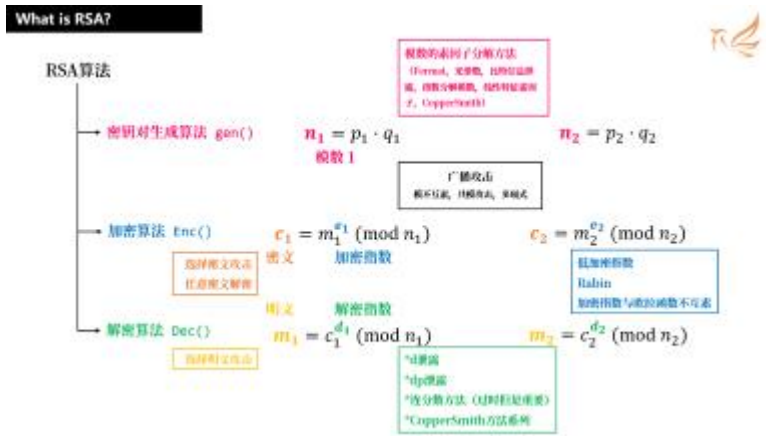
高斯整环（解决平方和 $n = p^2 + q^2$）

ECC-RSA

…

Q & A


Elementary Number Theory
初等数论
整除与同余
数论定理
二次剩余初步


What is RSA?
RSA算法
密钥对生成算法 gen()
加密算法 Enc()
解密算法 Dec()
模数 1
广播攻击
加密指数
解密指数