现代密码学

第五章：公钥密码

赵臻

zzhen@xidian.edu.cn

2023.04.05

复杂性的度量

- 随着算法输入长度 n 的增加，复杂度也会增加，**复杂性的度量**只需要反映出他们**随着输入长度的增加而增加的长度**，我们**将复杂性定义为关于n的函数f(n),** 使用渐近分析方法，常用符号如下

- 1. **大O小o**（给出了**f(n)的渐近上界**，最常用的比较函数增长程度的符号）

假设 f, g : N →R+ 是两个函数，则

（1）**f(n)=O(g(n))，** 如果 $\overline{\lim_{n\to\infty}}\frac{f(n)}{g(n)}\le c$，即∃正实数c 和正整数$n_0$, 使得当$n>n_0$时，有$f(n)\le cg(n)$。（f(n)的增长程度至多与g(n)的相同）

（2）**f(n)=o(g(n))**, 如果 $\lim_{n\to\infty}\frac{f(n)}{g(n)}=0$ ，即∀正实数c, ∃正整数n_0, 使得当$n>n_0$时，有$f(n)<cg(n)$。（f(n)的增长程度严格小于g(n)的）

复杂性的度量

举例

➢ 1. 大O小o（最常用的比较函数增长程度的符号）

- 多项式

假设 f(n) 为 k 次多项式，$f(n)=O(n^k)$, 且 $f(n)=o(n^{k+1})$

如果$f(n)=3n^8+2n^2+9$, 则$f(n)\leq 4n^8=O(n^8)$。即，大O符号内常系数可以忽略。

- 对数

因为 $\lim_{n\to\infty}\frac{logn}{n}=0$, 所以log n=o(n).

因为只要k>0, c>0，就有 $\lim_{n\to\infty}\frac{(logn)^k}{n^c}=0$, 有log n=o($n^c$),即对数增长程度严格小于多项式的增长程度。

- 指数

因为只要k>0, c>1，就有 $\lim_{n\to\infty}\frac{n^k}{c^n}=0$, 所以$n^k$=o($c^n$)

即，多项式的增长程度严格小于指数的增长程度。

复杂性的度量

- 随着输入长度n的增加，复杂度也会增加，**复杂性的度量**只需要反映出他们**随着输入长度的增加而增加的长度**，使用渐近分析方法，常用符号如下

- 2. 大Ω小ω（**f(n)的渐近下界**）

  假设 f, g : N →R+ 是两个函数，则

  （1）**f(n)=Ω(g(n))**，如果∃正实数c 和正整数n_0，使得当$n>n_0$时，有 $0 \le cg(n) \le f(n)$ 。（f(n)的增长程度至少与g(n)的相同）

  （2）**f(n)=ω(g(n)),** 即∀正实数c, ∃正整数n_0, 使得当$n>n_0$时，有$0 \le cg(n) \le f(n)$ 。（f(n)的增长程度严格大于g(n)的）

可忽略/不可忽略

- 可忽略（Negligible）和不可忽略（Non-negligible）
  - 假设一个方案（问题）由安全参数 λ 构造（生成），令 ε（λ）函数是攻破该方案或解决该问题的概率/优势。如果
  - $\varepsilon(\lambda) = \frac{1}{O(2^{\lambda})}$, 则与 λ 相关的$\varepsilon(\lambda)$是**可忽略的**，即随着 λ 的增加，$\varepsilon(\lambda)$快速趋近于0；
  - $\varepsilon(\lambda) = \frac{1}{O(\lambda^{n})}$, 则与 λ 相关的$\varepsilon(\lambda)$是**不可忽略的**。

主要用来描述概率和优势
是可忽略的/不可忽略的

可证明安全的公钥密码方案

- **方案在安全模型下的安全定义：**

  如果任何PPT敌手在此安全模型中胜利的优势为$\epsilon(\lambda)$，其中$\epsilon(\lambda)$是可忽略的，则我们说**方案在此安全模型下是安全的**。

  ↓

  怎么证明“任何PPT敌手在此安全模型中胜利的优势为可忽略的$\epsilon(\lambda)$”？

  ↓

  利用计算复杂性理论里的“**归约（reduction）**”的思想

可证明安全的公钥密码方案

- 计算复杂性理论中的归约（reduction）：

反证法：

安全假设：计算问题A是困难的。

如果计算问题B是简单的，我们能够证明A也是简单的。

该结果与前提条件矛盾，说明假设错误 “B是简单的”错误。

因此计算问题B也是困难的。

问题B的困难性被归约到安全假设（问题A困难）=

问题B的困难性被归约到问题A的困难性假设=

利用解决问题B的算法去解决问题A

可证明安全的公钥密码方案

- 归约：

问题B的困难性被归约到问题A的困难性假设

如果计算问题B是简单的，我们能够证明A也是简单的。

即：

- 解决问题 B 可以被转换为解决问题 A
- 可以将问题 A 的例子转换为问题 B 的例子
- 可以将问题 B 的解转换为问题 A 的解
- 问题 A 不困难于问题 B

可证明安全的公钥密码方案

举例

- 归约：

计算问题 A: $x_A = (g, g^a, g^b)$, 求解 $y_A = g^{ab}$（CDH问题，已知困难问题）

计算问题 B: $x_B = (g, g^c, g^d)$, 求解 $y_B = g^{c^2+cd}$

可证明安全的公钥密码方案

举例

- 归约：

计算问题 A: $x_A = (g, g^a, g^b)$, 求解 $y_A = g^{ab}$（CDH问题，已知困难问题）

计算问题 B: $x_B = (g, g^c, g^d)$, 求解 $y_B = g^{c^2+cd}$

证明：如果 B 是简单的，则我们可以证明 A 也是简单的。

假设 c=a, d=b-a, 计算 $g^c = g^a, g^d = g^{b-a} = \frac{g^b}{g^a}$.

给定 $x_B = (g, g^c, g^d) = \left(g, g^a, \frac{g^b}{g^a}\right)$, 可计算得到 $y_B = g^{c^2+cd} =$

$g^{a^2+a(b-a)} = g^{ab}$. 根据反证法思想，由于A是困难的，B也是困难的。

可证明安全的公钥密码方案

- 归约练习：

计算问题 A: $x_A = (g, g^a, g^b)$, 求解 $y_A = g^{ab}$

（CDH问题，已知困难问题）

计算问题 B: $x_B = (g, g^c, g^d)$, 求解 $y_B = g^{c^2-d^2}$

练习：证明问题B的困难性。

可证明安全的公钥密码方案

- 公钥密码方案安全证明中的安全归约（Security Reduction）：

  利用计算复杂性理论里的“归约（reduction）”的思想

反证法：

安全假设：一个计算问题P是困难的。

如果密码方案 S 在安全模型 M 下是不安全的，我们能够证明P是简单的（解决该困难问题）。

该结果与前提条件矛盾，说明假设错误 “S在安全模型M下是不安全的”错误。

因此密码方案 S 在安全模型M下是安全的。

安全模型M下的方案S的安全性被归约到问题P的困难性假设

可证明安全的公钥密码方案

归约	把**问题B的困难性**	归约到	问题A的困难性假设
安全归约	把**安全模型M下的方案S的安全性**	归约到	问题P的困难性假设

- 计算复杂性理论中的**归约（reduction）**：

将问题 B的困难性归约到问题A的困难性假设

↓

证明中，问题 A 的实例 x_A 可以用于构建问题 B 的实例 x_B

↓

解决问题 B 可以被转换为解决问题 A

- 方案证明中的**安全归约（Security reduction）**：

将安全模型 M 中的方案 S归约到问题P的困难性假设

↓

证明中，问题 P 的实例 x_A 可以用于“模拟”安全模型 M 中的方案 S

↓

攻破方案 S 在安全模型 M 下的安全性可以被转换为解决问题P

可证明安全的公钥密码方案

➢ 一个密码方案的安全归约

如果**密码方案 S 在某安全模型下是不安全的，**我们能够证明P是简单的（解决该困难问题）。

- 假设存在概率多项式时间（PPT）敌手可以在某安全模型下攻破密码方案S，则我们可以构造一个模拟者（Simulator）解决困难问题P。
  - **模拟（Simulation）**：利用困难问题P的实例x_P模拟方案S（被敌手在该安全模型M下攻击的方案S）。
  - **求解（Solution）**：模拟者通过敌手对模拟方案 S 的攻击，求解实例的解y_P
  - **分析（Analysis）**：分析如果假设成立，正确求解问题P的实例x_P的解的优势不可忽略。

离散对数问题前置知识

- **群：** 定义：设G为某种元素组成的一个非空集合，若在G内定义一个称为乘法的二元运算“·”，满足以下条件：

（1） （封闭性）$\forall a, b \in G$，有$a \cdot b \in G$;

（2） （结合性）$\forall a, b, c \in G$，有$a \cdot (b \cdot c) =$ （$a \cdot b$）$\cdot c$;

（3） （单位元）对G中有一个元素e，对G中任一元素g，有

$$e \cdot g = g \cdot e = g$$

元素e称为单位元;

（4） （逆元）对G中任一元素g都存在G中的一个元素g'，使得

$$g \cdot g' = g' \cdot g = e$$

g称为可逆元，g'称为g的逆元，记作g^{-1};

则称G关于乘法运算“·”形成一个群，记作（G，·），通常在不混淆的情况下省略“·”，用G来表示一个群，$a \cdot b$也简记为ab。

离散对数问题前置知识

- **群：**
  - $(Z,+)$，$(Q,+)$，$(R,+)$，$(C,+)$
  - $(Q^*,\cdot)$，$(R^*,\cdot)$，$(C^*,\cdot)$

举例

- m为正整数，$Z_m=\{0,1,2,\cdots,m-1\}$对于如下定义的加法：$a\oplus b=(a+b)\,(mod\ m)$，$Z_m$关于“$\oplus$”构成一个加法交换群$(Z_m,\oplus)$，或者$(Z_m,+)$。
- ${Z_m}^*=Z_m\backslash\{0\}$对于如下定义的乘法：$a\otimes b=ab\,(mod\ m)$，当$m=p$时，${Z_p}^*$关于“$\otimes$”构成一个乘法交换群$({Z_p}^*,\otimes)$，或者$({Z_p}^*,\cdot)$。

离散对数问题前置知识

➢ 循环群

定义： 设G为群，$a \in G$，则使$a^n = e$成立的最小正整数**n称为元素a的阶**，记为$ord(a)$。若一个群G的每一个元素都是某一固定元素a的幂，即$G = \{a^n | n \in Z\}$，则称G为**循环群**，也称G是由元素a生成的，记为$G = \langle a \rangle$，**a称为G的一个生成元。**

例6.2.1　8阶加法群$(Z_8, +)$，$Z_8 = \{0,1,2,3,4,5,6,7\}$。

解　$\overbrace{1 + \cdots + 1}^{8} = 8 \, mod \, 8 = 0$

$$\overbrace{5 + \cdots + 5}^{8} = 40 \, mod \, 8 = 0$$

$$\overbrace{3 + \cdots + 3}^{8} = 24 \, mod \, 8 = 0$$

$$\overbrace{7 + \cdots + 7}^{8} = 56 \, mod \, 8 = 0$$

1,3,5,7均为8阶群$(Z_8, +)$中的8阶元，即为生成元，记$Z_8 = \langle 1 \rangle = \langle 3 \rangle = \langle 5 \rangle = \langle 7 \rangle$

离散对数问题前置知识

➢ 循环群

□ 定理：素数阶群均为循环群，且该群中任意非单位元均为生成元。

证明　　设p为素数，$|G| = p$。任取$a \in G$，且$a \neq e$。于是

$$|\langle a\rangle| \,|\, |G| \Leftrightarrow |\langle a\rangle| \,|\, \mathrm{p}$$

又由$a \neq e$，得$|\langle a\rangle| \neq 1$，从而只能有$|\langle a\rangle| = p$，因此$G = \langle a\rangle$为一个循环群。

椭圆曲线离散对数问题前置知识

➢ 椭圆曲线

- 定义：椭圆曲线是指由Weierstrass方程确定的曲线。

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

令 $x = X/Z, y = Y/Z.$

$$y^2 + a_1xy + a_2y = x^3 + a_2x^2 + a_4x + a_6.$$

➢ 公钥密码学中主要用到两类椭圆曲线：

- 域 Fp (p>3)上的椭圆曲线，表示为

$$y^2 = x^3 + ax + b(\mathrm{mod}\, p)，\text{其中} a, b \in F_p,$$

- 域 F_{2^m}(m≥1) 上的椭圆曲线，表示为

$$y^2 + xy = x^3 + ax^2 + b，\ a, b \in F_{2^m}，\ \text{且} b \neq 0$$

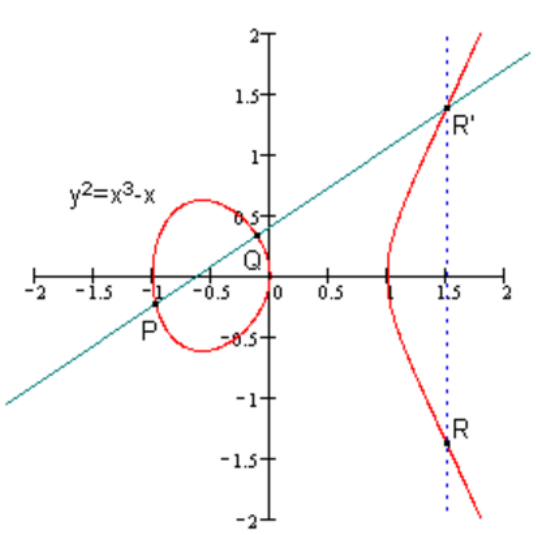
椭圆曲线离散对数问题前置知识

- 椭圆曲线

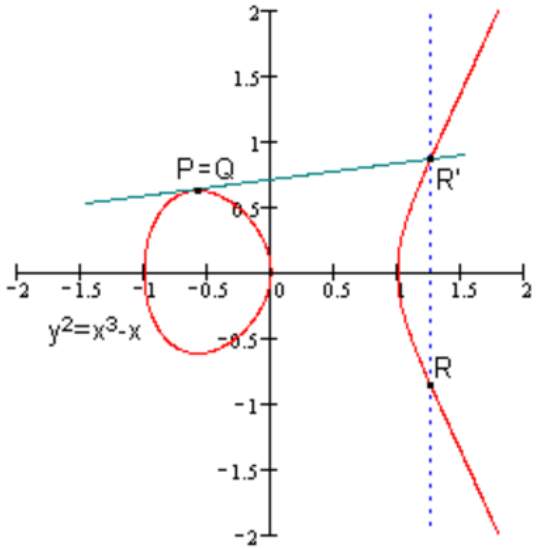
域 Fp (p>3)上的椭圆曲线，表示为

$$y^2 = x^3 + ax + b(\mathrm{mod}\, p)，\text{其中} a, b \in F_p，$$

选取其中特殊曲线 $y^2 = x^3 - x$ 为例，椭圆曲线上的加法运算如下。



$P \neq Q$



$P = Q$

椭圆曲线离散对数问题前置知识

- 椭圆曲线

下面是中国国家密码管理局关于国密SM2椭圆曲线公钥密码算法推荐曲线参数，推荐使用素数域256位椭圆曲线，椭圆曲线方程：$y^2 = x^3 + ax + b(\mathrm{mod}\, p)$. 其中$G(G_x, G_y)$是基点，$n$是基点$G$的阶，即$nG = O$. 曲线参数如下：

p=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF

a=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC

b=28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93

n=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123

G_x=32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7

G_y=BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

椭圆曲线离散对数问题前置知识

- 双线性对（基于椭圆曲线群）
  - 定义：令$G_1 =< P >$以及$G_2 =< Q >$是椭圆曲线上阶为 p 的加法循环群， G_T是阶为 p 的乘法循环群。双线性对映射$e: G_1 \times G_2 \to G_T$ 满足以下性质。

- 双线性： $\forall R \in G_1, e(P+R,Q) = e(P,Q)e(R,Q);$

  $\forall S \in G_2, e(P,Q+S) = e(P,Q)e(P,S)$
- 非退化性 $\forall R \in G_1$, 如果 $e(R,S)=1$, $S=\mathcal{O}_2$;

  $\forall S \in G_2$, 如果 $e(R,S)=1$, $R=\mathcal{O}_1$
- 可计算性： $\forall R \in G_1, S \in G_2$, $e(R,S)$ 可计算。

椭圆曲线离散对数问题前置知识

- **双线性对（基于椭圆曲线群）现在的写法**

  - 定义：令G_1、G_2、G_T是阶为 p 的乘法循环群。双线性对映射e: $G_1 \times G_2 \to G_T$ 满足以下性质。

  - 双线性：$\forall g_1 \in G_1, g_2 \in G_2, r, s \in Z_l, e(g_1^r, g_2^s) = e(g_1, g_2)^{rs}$
  - 非退化性：$\forall g_1 \neq e_1, g_2 \neq e_2, e(g_1, g_2) \neq e_T$，其中$e_1, e_2, e_T$分别是群$G_1$、$G_2$、$G_T$的单位元。
  - 可计算性：$\forall g_1 \in G_1, g_2 \in G_2, e(g_1, g_2)$ 可计算。

  - 我们将$\mathrm{PG} = (G, G_T, g, p, e)$称为一个双线性群。

椭圆曲线离散对数问题前置知识

- 双线性对（基于椭圆曲线群）
  - Type-1: e: $G_1 \times G_1 \rightarrow G_T$,对称双线性对
  - Type-2: e: $G_1 \times G_2 \rightarrow G_T$，但存在计算同构$\phi: G_2 \rightarrow G_1$
  - Type-3: e: $G_1 \times G_2 \rightarrow G_T$，且$G_1$、$G_2$ 之间不存在计算同构

Type	Hash to G_2	Short G_1	Homomorphism	Poly time generation
1(small char)	✓	×	✓	×
1(large char)	✓	×	✓	✓
2	×	✓	✓	✓
3	✓	✓	×	✓

单向函数举例

离散对数DL。给定一大素数p（比如，p在2^{1024}数量级），$p-1$含另一大素数因子。称$\log_2 p$为素数p的长度。$\{1, 2, \ldots, p-1\}$关于$mod\ p$乘法构成了一乘群Z_p^*，它是一个$p-1$阶循环群。

设一个生成元为整数g，$1 < g < p-1$。

- 设一个整数x，$1 < x < p-1$。
- 设y满足$y = g^x \bmod p$。

单向函数举例

已知x，g，p， 求$y=g^x \bmod p$容易。

这是因为，采用折半相乘，只需要不超过$2\log_2 p$次的mod p乘法运算。

（实际上只需要不超过$2\log_2 x$次的mod p乘法运算。如$x=15=1111_2$，

$g^{15} \bmod p=(((g)^2 g)^2 g)^2 g \bmod p$，

要用6次mod p乘法 ）

单向函数举例

若已知y，g，p，求x满足$y=g^x \bmod p$，称为求解离散对数问题。记为$x=\log_g y \bmod p$。

求解离散对数问题的“最笨的方法”当然就是穷举，对每一个$x \in \{0, 1, 2, \ldots, p-1\}$检验是否$y=g^x \bmod p$。穷举求解法的运算次数约为$(p-1)/2$。许多求解离散对数问题的算法比穷举快得多，比如Shanks算法，Pohlig-Hellman算法等。最快求解法的运算次数约为数量级

$$O(\exp(\sqrt{(\ln p)(\ln\ln p)}))$$

单向函数举例

这个计算量称为亚指数计算量。这是什么概念呢？我们知道p的长度是$\log_2 p$。看以下的不等式。

当$\log_2 p \approx 1024$时，亚指数计算量不小于2^{100}数量级。至少在在当前的计算水平之下是不能实现的。

$$\exp(\sqrt{(\ln p)(\ln\ln p)}) << \exp(\sqrt{(\ln p)(\ln p)})$$

$$= p = 2^{\log_2 p};$$

$$\exp(\sqrt{(\ln p)(\ln\ln p)}) >> \exp(\sqrt{(\ln\ln p)(\ln\ln p)})$$

$$= \ln p \sim \log_2 p.$$

单向函数举例-群论困难问题

- 计算性困难问题

  $\boldsymbol{G} =< \boldsymbol{g} >$，阶为$\boldsymbol{p}$

  - **Discrete Logarithm (DL) 问题**

    给定： $g, g^a \in G$， 求解： a

  - **Computational Diffie-Hellman (CDH) 问题**

    给定： $g, g^a, g^b \in G$， 求解： g^{ab}

  - **q-Strong Diffie-Hellman (q-SDH) 问题**

    给定： $g, g^a, g^{a^2}, \cdots, g^{a^q} \in G$， 求解： $(s, g^{\frac{1}{a+s}}) \in Zp \times G$

  - **q-Strong Diffie-Hellman Inversion (q-SDHI) 问题**

    给定： $g, g^a, g^{a^2}, \cdots, g^{a^q} \in G$， 求解： $g^{\frac{1}{a}}$

单向函数举例-群论困难问题

- 计算性困难问题

$e: G \times G \to G_T$, $G =< g >$， G, G_T阶为p

- **Computational Bilinear Diffie-Hellman (CBDH) 问题**

  给定： $g, g^a, g^b, g^c \in G$， 求解： $\mathrm{e}(\mathrm{g},\mathrm{g})^{abc}$

- **q-Bilinear Diffie-Hellman Inversion (q-BDHI) 问题**

  给定： $g, g^a, g^{a^2}, \cdots, g^{a^q} \in G$， 求解： $\mathrm{e}(\mathrm{g},\mathrm{g})^{\frac{1}{a}}$

- **q-Bilinear Diffie-Hellman (q-BDH) 问题**

  给定： $g, g^a, g^{a^2}, \cdots, g^{a^q}, g^{a^{q+2}}, g^{a^{q+3}}, \cdots, g^{a^{2q}}, h \in G$， 求解：

$\mathrm{e}(\mathrm{g},\mathrm{g})^{a^{q+1}}$

单向函数举例-群论困难问题

- 判定性困难问题

$G =< g >$，阶为p

- **Decisional Diffie-Hellman (DDH) 问题**

  给定：$g, g^a, g^b, Z \in G$， 判定：Z= g^{ab}?

- **Variant Decisional Diffie-Hellman (Variant DDH) 问题**

  给定：$g, g^a, g^b, g^{ac}, Z \in G$， 判定：Z= g^{bc}?

思考：DDH 问题在双线性群上是否成立？

单向函数举例-群论困难问题

- 判定性困难问题
  - **Decisional Diffie-Hellman (DDH) 问题**

    给定： $g, g^a, g^b, Z \in G$， 判定：Z= g^{ab}?

思考：DDH 问题在双线性群上是否成立？

- Type-1: e: $G_1 \times G_1 \to G_T$,对称双线性对

$$e(g^a, g^b) = e(g, g)^{ab}$$

- Type-2: e: $G_1 \times G_2 \to G_T$，但存在计算同构$\phi: G_2 \to G_1$

  若 G = G_1，DDH问题成立

  若 G = G_2，$e(\phi(g^a), g^b) = e(\phi(g)^a, g) = e(\phi(g), g)^{ab}$

- Type-3: e: $G_1 \times G_2 \to G_T$，且G_1、G_2 之间不存在计算同构

  DDH问题成立

单向函数举例-群论困难问题

- 判定性困难问题
  - $e: G \times G \to G_T$, $G =< g >$，循环群G, G_T阶为p
  - **Decisional Bilinear Diffie-Hellman (DBDH) 问题**

    给定：$g, g^a, g^b, g^C, Z \in G$， 判定： Z=e(g, g) ^{abc}?
  - **Decisional Linear (DLIN) 问题**

    给定：$g, g^a, g^b, g^{ac_1}, g^{bc_2}, Z \in G$， 判定：Z= $g^{c_1+c_2}$?
  - **q-DABDHE 问题**

    给定：$g, g^a, g^{a^2}, \cdots, g^{a^q}, h, h^{a^{q+2}}, Z \in G$， 判定： e(g, h)$^{a^{q+1}}$

单向函数举例-群论困难问题

- 判定性困难问题
  - $e: G \times G \to G_T$, $G =< g >$，循环群G, G_T阶为p
  - **Decisional (f, g, F)-GDDHE 问题**

    给定：$g, g^a, g^{a^2}, \cdots, g^{a^{n-1}}, g^{af(a)}, g^{baf(a)} \in G$

    $\mathrm{h}, h^a, h^{a^2}, \cdots, h^{a^{2k}}, \quad , h^{bg(a)} \in G$

    $Z \in G_T$,

    $f(x), g(x)$分为是n, k次的互素多项式

    判定：Z= $e(g,h)^{bf(a)}$？

思考：上述判定性困难问题在type-2、3上是否困难？

单向函数举例

大整数分解*FAC*。设有二大素数*p*和*q*。设*n*=*pq*。

若已知*p*和*q*，求*n*=*pq*只需一次乘法。

但若已知*n*，求*p*和*q*满足*n*=*pq*，则称为大整数分解问题。迄今为止，已知的各种算法的渐近运行时间约为：

试除法：*n*/2。

二次筛(QS)：$O(\exp\sqrt{\ln n \ln\ln n})$

椭圆曲线(EC)：$O(\exp\sqrt{2\ln p \ln\ln p})$

数域筛(NFS)：$O(\exp(1.92(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}))$

单向函数举例

背包问题。已知向量

$$A=(a_1, a_2, \ldots, a_N), \quad a_i\text{为正整数},$$

称其为**背包向量**，称每个a_i为物品重量。给定向量

$$\boldsymbol{x}=(x_1, x_2, \ldots, x_N), \quad x_i \in \{0, 1\},$$

求和式（称为背包重量）

$$S= a_1x_1+ a_2x_2+\ldots +a_Nx_N$$

容易，只需要不超过$N-1$次加法。但已知A和S，求$\boldsymbol{x}$则非常困难，称其为背包问题，又称作**子集和**(Subset-Sum)问题。一般只能用穷举搜索法，有2^N种可能。N大时，相当困难。

单向函数举例

背包问题的特例：超递增背包问题。将物品重量从小到大排列：a_1，a_2，a_3，…，a_N。称该背包问题为超递增背包问题，如果：

$a_1<a_2$；

$a_1+a_2<a_3$；

$a_1+a_2+a_3<a_4$；

…

$a_1+a_2+a_3+\ldots+a_{N-1}<a_N$。

（超递增背包问题是容易解决的。）

单向函数举例

定理 设超递增背包重量为S。如果k满足$a_k<S<a_{k+1}$，则a_k是背包中的最大物品重量。

定理的证明

首先，背包中没有大于a_k的物品重量。

其次，背包中确有等于a_k的物品重量。

证明完毕。

注意到，寻找k满足$a_k<S<a_{k+1}$只需要对比N次。

单向函数举例

超递增背包问题的解决方法

解决方法是可行的。设背包重量S，步骤如下。

（1）穷举：找k满足$a_k<S<a_{k+1}$。（这说明背包中的最大物品重量是a_k）

（2）记忆：存储这个k。

（3）卸载：如果$S>0$，则令$S:=S-a_k$，返回（1）。如果$w=0$，则到（4）。

（4）输出前面存储的所有的k，停止。

单向函数举例

格的最小向量问题(SVP)。

若干个*N*维向量组成的集合，如果满足

“集合中任何若干个向量的整数线性组合仍是集合中的一个向量。”

则该集合称为一个格。

关于格有以下的性质和概念。

- 如果格中存在这样的几个向量，满足①它们（实数）线性无关；②格中的任何其它向量都能唯一地表示为这几个向量的整数线性组合。则这几个向量构成的向量组称为基。
- 基中的向量的个数称为格的维数。
- 格的维数总是不超过*N*。

单向函数举例

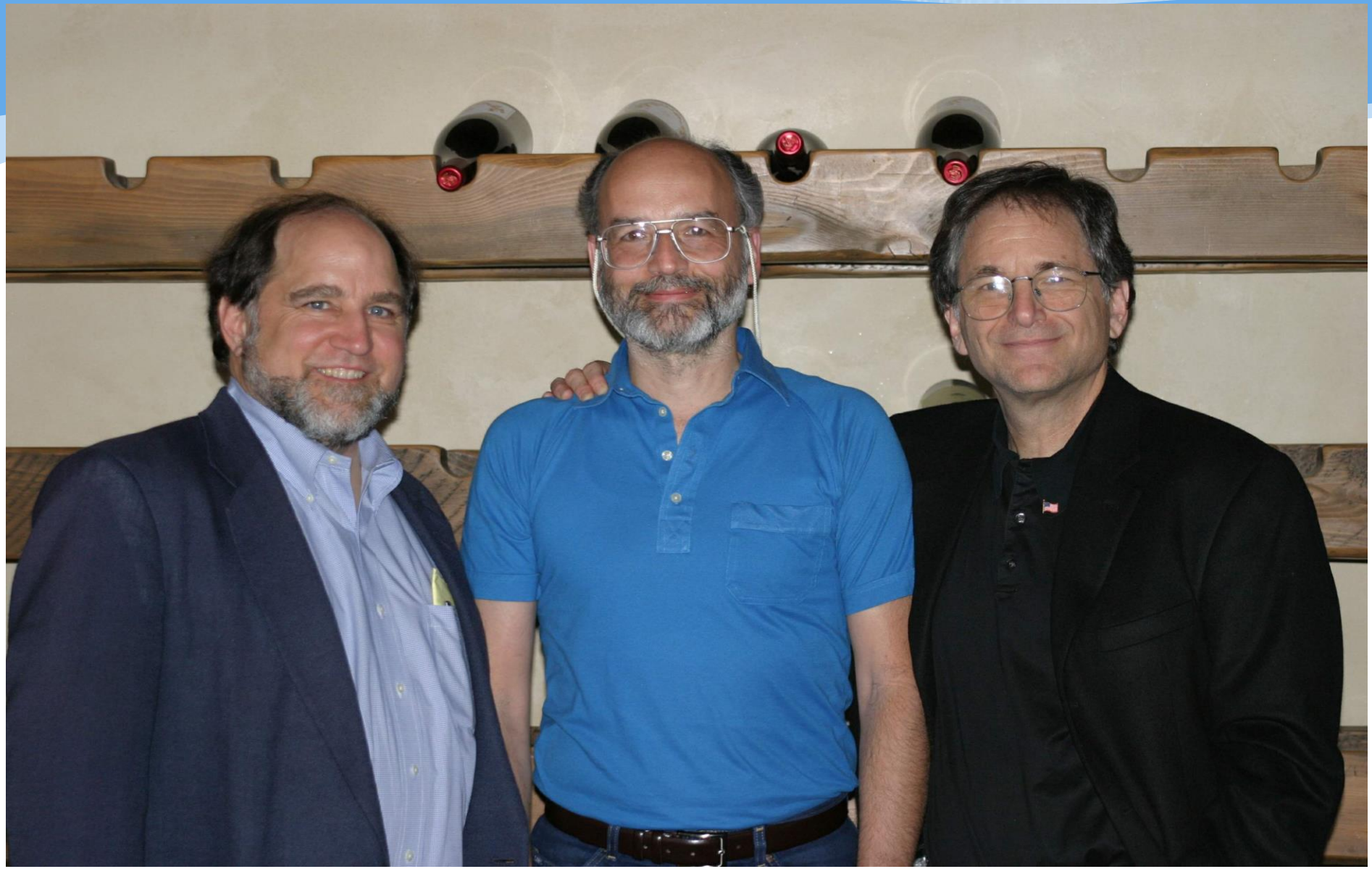
给定一个格的一组基。寻找格中的“尺寸最小”的向量（即模最小的向量），称为格的最小向量问题（shortest vector problem；SVP）。又称为格归约。

实际上，格归约的传统算法为LLL算法，以后又有各种改进的算法。当格的维数比较大时（比如，维数大于200），当前的所有格归约算法都不是有效算法。

公钥密码体制RSA

- **RSA公钥算法**由 MIT 的**Rivest**, **Shamir**和**Adleman**在I978年提出来的
- 是被最广泛接受并实现的通用公钥密码算法，已成为公钥密码的国际标准
- 该算法的数学基础是初等数论中的欧拉定理，其**安全性基于大整数因子分解的困难性**

Adi Shamir Ron Rivest Len Adleman



密码相关图灵奖获得者

1995年	曼纽尔·布卢姆	Manuel Blum	计算复杂度理论，及其在密码学和程序校验上的应用
2000年	姚期智	Andrew Chi-Chih Yao	计算理论，包括伪随机数生成，密码学与通信复杂度
2002年	罗纳德·李维斯特	Ronald L. Rivest	公钥密码学（RSA加密算法）
	阿迪·萨莫尔	Adi Shamir	
	伦纳德·阿德曼	Leonard M. Adleman	
2002年	罗纳德·李维斯特	Ronald L. Rivest	公钥密码学（RSA加密算法）
	阿迪·萨莫尔	Adi Shamir	
	伦纳德·阿德曼	Leonard M. Adleman	

RSA算法描述

1. 密钥的生成

1. 选择两个大素数 p和q,（$p \neq q$，需要保密）

2. 计算$n = p \times q$， $\varphi(n) = (p-1) \times (q-1)$

3. 选择整数 e 使 $(\varphi(n), e) = 1$， $1 < e < \varphi(n)$

4. 计算d，使$d = e^{-1} mod\, \varphi(n)$,

得到：公钥为$\{e, n\}$； 私钥为$\{d\}$

2. 加密(用e，n)： 明文$M < n$， 密文$C = M^e (mod\, n)$.

3. 解密(用d，n)： 密文C， 明文$M = C^d (mod\, n)$

验证： $C^d \, mod\, n = (M^e)^d \, mod\, n = M^{ed} \, mod\, n \equiv M \, mod\, n$

(由Euler定理推论)

例

设$p=3, q=5$, 则

$$n=p*q=3*5=15,\ \varphi(n)=(p-1)(q-1)=2*4=8$$

取$e=3$, 则$d=3$, 因为$ed \equiv 1\ mod\ \varphi(n) \equiv 1\ mod\ 8$

加密:

假设明文$m=7$, 则密文$c=m^e=7^3 \equiv 13(mod\ 15)$

解密:

$$m=c^d=13^3 \equiv 7(mod\ 15)$$

加密:

假设明文$m=9$, (9,15不互素), 则密文

$$c=m^e=9^3 \equiv 9(mod\ 15)$$

解密: $m=c^d=9^3 \equiv 9(mod\ 15)$

一些疑问

1. 如果每个人都需要一个不同的素数，难道素数不会被用光吗？
   - 不会：**小于n的素数的总数约为$n/(\ln n)$（素数定理）**

   例：小于10^{100} 的素数个数$x = 10^{100}/ln^{10^{100}}$

$$ln^{10^{100}} < log_2 10^{100} = 100 * log_2 10 \quad < 100 * 4 = 400$$

$$x = 10^{100}/ln^{10^{100}} > 10^{100}/400$$

2. 是否会有两个人偶然地选择了同样的素数呢？
   - **很少发生**：　从M个素数中选出q个素数，至少有两个素数相同的概率至少为$1/2$，此时$\boldsymbol{q = 1.17M^{1/2}}$
3. **产生素数困难吗？**

素数的生成

- 最自然的方法：先随机生成一个适当大小的奇数n，再检测其素性；如果不是，则选取后继的随机数直到找到通过检验的素数为止
  - 通过检测$\leq \sqrt{n}$的素数能否整除n来检测（**效率太低，指数时间算法**）
  - **概率素性检测**：即这个检验只是确定一个给定的整数可能是素数。虽然缺乏完全的确定性，但这些检验可以按需做到**使得概率尽可能地接近1**

- 一般的，选取一个素数的过程如下：

1. 随机选一个奇数n(如使用**伪随机数产生器**)

2. 用某种概率性算法（如Miller-Rabin算法）对n进行一次素性检验，如果n没有通过检验，转到步骤1

3. 重复步骤2足够多次，如果n都通过了检测，则认为n为素数

RSA的实现

❖如何快速的计算$a^m \ (mod\ n)$?

将m表示为二进制形式$b_k b_{k-1} \dots b_0$，即：

$$m = b_k 2^k + b_{k-1} 2^{k-1} + \cdots + b_1 2 + b_0$$

如：$19 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$,

所以：

$$a^{19} = ((((\ a^1)^2\ a^0)^2\ a^0)^2\ a^1)^2\ a^1$$

注意：所有的乘法或乘方运算之后都有一个模运算

RSA的安全性

- 为什么要保密p, q?

  已知p, q，就可以算出Euler函数$\varphi(n) = (p-1)(q-1)$，也就可以利用欧几里得除法，根据$ed \equiv 1 \ (mod \ \varphi(n))$, 由$e$求出$d$

- **RSA的安全性依赖于大数分解问题**
  - 目前，还未能从数学上证明由c和e计算出m一定需要分解n
  - 然而，如果新方法能使密码分析者推算出d，它也就成为大数分解的一个新方法

对RSA的攻击

- 针对n分解的攻击
- 循环攻击
- 同模攻击
- 选择密文攻击
- 低加密指数攻击
- 时间攻击

针对n分解的攻击

- **试除法：**

  完全尝试所有小于$\sqrt{n}$的素数。根据素数理论，尝试的次数上限为

$$2\sqrt{n}/\log\sqrt{n}$$

- **因子分解法：**
  - $p \pm 1$法
  - 连分数法
  - 二次筛法
  - 椭圆曲线筛法
  - 数域筛法 $(O(e^{(lnN)^{\frac{1}{3}}(lnlnN)^{\frac{2}{3}}(C+o(1))}))$

循环攻击

设m是明文，(N,e)是公钥，密文$c=m^e \bmod N$

- 攻击者得到密文c后，对密文c依次进行如下变换：

$$c_1=c^e \bmod N;\quad c_2=c_1^e \bmod N;$$

……

$$c_k=(c_{k-1})^e \bmod N;\quad c_{k+1}=c_k^e \bmod N$$

- $k+1$步迭代之后，如果$\boldsymbol{c_{k+1}=c}$。由$c=m^e \bmod N$可知$\boldsymbol{c_k=m}$

循环攻击实例

- 设RSA公钥体制的参数为

$$p=383, q=563, N=pq=215629, e=49, d=56957$$

则加密和解密变换分别为:

$$c=m^{49} \mod 215629$$

$$m=c^{56957} \mod 215629$$

如果明文消息为 $m=123456$，则对应的密文是:

$$123456^{49}=1603 \mod 215629$$

- 密码分析者利用公钥(215629,49)和密文1603进行循环攻击得到：

$$c_1 = 180661; \quad c_2 = 109265; \quad c_3 = 131172;$$

……

$$c_9 = 123456; \quad c_{10} = 1603;$$

- 攻击者经过**10次变换**就得到了 $c_{10} = c$，则

$$m = c_9 = 123456$$

- **为什么会这样哪?**

定义 设$m>1$是整数，a是与m互素的正整数，则使得

$$a^e \equiv 1(\mathrm{mod}\ \ m)$$

成立的最小正整数e叫做a对模m的指数. 记作$ord_m(a)$.

如果a对模m的指数是$\varphi(m)$，则a叫做模m的原根

定理 设$\boldsymbol{m>1}$，$\boldsymbol{(a,m)=1}$. 则整数$\boldsymbol{d}$使得

$\boldsymbol{a^d \equiv 1(mod\ m)}$的充分必要条件是 $\boldsymbol{ord_m(a)|d}$

推论 设$\boldsymbol{m>1}$，$\boldsymbol{(a,m)=1}$. 则 $\boldsymbol{ord_m(a)|\varphi(m)}$

根据推论，整数$\boldsymbol{a}$模$\boldsymbol{m}$的指数$\boldsymbol{ord_m(a)}$是$\boldsymbol{\varphi(m)}$的因数，所以可在$\boldsymbol{\varphi(m)}$的因数中求$\boldsymbol{ord_m(a)}$

- 设m在模N下的指数为$ord_N(m)$，由 $m^{e^k} = m \mod N$

  所以 $m^{e^k-1} = 1 \bmod N$，则 $ord_N(m) \mid e^k - 1$，即

$$e^k = 1 \bmod \ \mathrm{ord}_N(m)$$

k取满足上式的最小值，即为 e模$ord_N(m)$的指数

则要避免循环攻击，则k 要大，而 k 与 $\varphi(ord_N(m))$ 的因子有关，所以

I）需要$ord_N(m)$要大，且$\varphi(ord_N(m))$要有大的素因子

II）又$ord_N(m)$是$\varphi(N)=\varphi(p)\varphi(q)=(p-1)(q-1)$的因子，所以需要$p-1$和$q-1$都有大的因子

回到例子

- 当$N = 215629$时，$\varphi(N) = (383-1) \times (563-1) = 2^2 \times 191 \times 281$
- 此时

$$49^{10} = 1 \mod \varphi(N)$$

- 49的**指数**仅为**10**，所以只需要10次运算就可破解系统

同模攻击

- 假设m是明文，两用户的公钥分别是e_1和e_2，且$(e_1, e_2) = 1$，共同的模数N，两个密文分别为：

$$c_1 \equiv m^{e_1} \mod N$$
$$c_2 \equiv m^{e_2} \mod N$$

- 攻击者知道N，e_1，e_2，c_1和c_2，可如下恢复明文m
  - $(e_1, e_2) = 1$，由欧几里德算法可找出r, s满足$re_1 + se_2 = 1$。假定r是负数，那么

$$(c_1^{-1})^{-r} \cdot c_2^{s} = m^{re_1 + se_2} \equiv m \mod N$$

- 无需秘密密钥d，就可以得到明文m(**RSA的共模攻击**)

在使用RSA公钥密码的通信中，不同用户的密钥不能有相同的模值

选择密文攻击

RSA模幂运算的性质：

$$E_K(ab) = E_K(a)E_K(b)$$

1. 破译密文
   - 用户A公钥为(e,n)，攻击者监听到发给A的密文$c = m^e \bmod n$
   - 攻击者随机选取一个$r < n$，计算$y = r^e \bmod n$，
   $$t = yc \pmod n$$
   - 攻击者发送t给A，要求A对消息t签名(A用私钥签名)
   - A把签名返回给攻击者，攻击者就得到了$s = t^d \bmod n$。

   攻击者计算：
   $$r^{-1}s \pmod n = y^{-d} \times t^d \pmod n$$
   $$= y^{-d} \times y^d \times c^d \pmod n = c^d \bmod n = m,$$
   于是攻击者获得了明文m

2. 骗取签名

- 攻击者有非法消息m_i要获得A的签名，但m_i直接给A不能被接受，于是攻击者选择一个r，算得$y=r^e \pmod n$，再计算$m=y*m_i \pmod n$，将m给A签名，获得$s=m^d \pmod n$，计算

$$sr^{-1}=y^d {m_i}^d r^{-1} \pmod n$$

$$=r\times r^{-1}\times {m_i}^d \pmod n={m_i}^{d(} \bmod n),$$

于是攻击者得到了m_i的签名

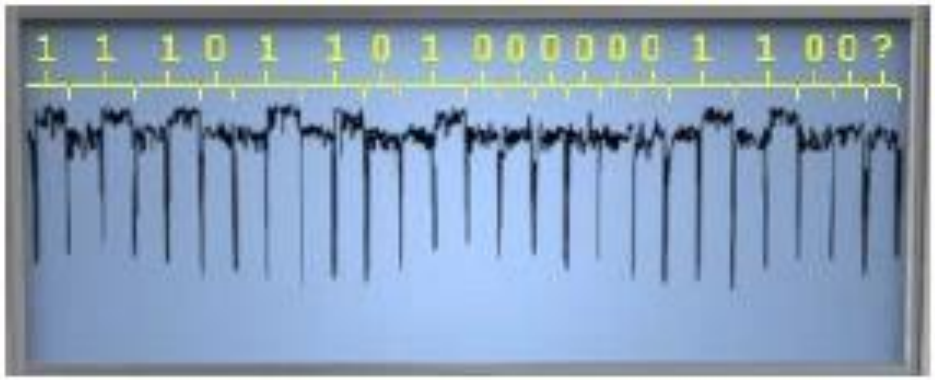
∴不要给陌生人提交的随机文件签名

低加密指数攻击

- **小的公钥**可加快加密的速度，但过小的公钥易受到攻击
  - 如果3个用户都使用3作为公钥，对同一个明文 m 加密，则$c_1 = m^3 \ (mod\ n_1)$，$c_2 = m^3 \ (mod\ n_2)$，$c_3 = m^3 \ (mod\ n_3)$，$\gcd(n_1, n_2, n_3) = 1$，且$m < n_1$，$m < n_2$，$m < n_3$
  - 由**中国剩余定理**可从c_1，c_2，c_3计算出c，且$c = m^3 \ mod\ (n_1 n_2 n_3)$，显然$m^3 < n_1 n_2 n_3$，所以$m = c^{1/3}$
  - Hastad证明如果采用不同的模n，相同的公钥e，则对$e(e+1)/2$个线性相关的消息加密，系统就会受到威胁。所以一般选取16位以上素数(速度快且可防止攻击)
  - 对短消息，**用随机数填充**以保证$m^e \ (mod\ n) \neq m^e$，从而杜绝低加密指数攻击

时间攻击

- **时间攻击**主要针对RSA核心运算是非常耗时的模乘，只要能够精确监视RSA的解密过程，就能估算出私有密钥d
  - **模幂**是**按位**计算的，每次迭代执行一次模平方，并且如果当前位是1，则还需要进行一次模乘



  - **实际操作很困难**。如果加密前对数据做**盲化**处理，使得加密时间具有随机性，最后进行**去盲**，即可抵抗定时攻击，但增加了数据处理步骤

- 综上所述，使用RSA体制时必须注意以下问题：
  1. 选择素数p和q时，应使其欧拉函数$\varphi(p)$和$\varphi(q)$的最小公倍数尽可能大($\varphi(p)$和$\varphi(q)$ **有大的素因子**)。最小公倍数越大，幂剩余函数的周期就越长—**避免循环攻击**
  2. 密钥中的各项参数应选得足够大—**避免穷举攻击**
  3. 在同一个通信网络中，不同的用户不应该使用共同的模数—**避免同模攻击**

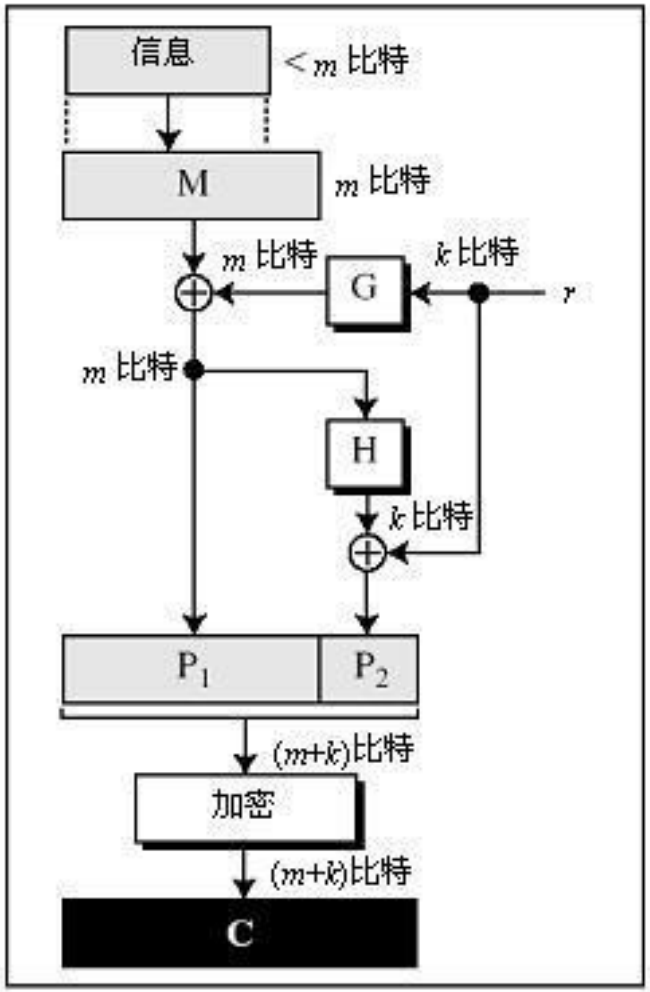
RSA-OAEP：最佳非对称加密填充

- **RSA-OAEP**：对消息编码的一种方法，由M. Bellare和 P. Rogaway提出
  - 首先对消息进行**填充**，然后用RSA进行加密
  - 该方法是**可证明安全**的

- **加密操作**由3步组成：
  - 长度检查
  - EME-OAEP编码
  - RSA加密

M：填充信息　　P：明文($P_1 \| P_2$)　　G：公共函数(k-bit to m-bit)

r：一次性 随机数　　C：密文　　H：公共函数(m-bit to k-bit)



发送者