第5章 RSA密码体制和整数因子分解

杨礼珍

提交作业Email：yanglizhen_exe@163.com
课件下载Email：yanglizhen_course@163.com，密码：tongjics

同济大学计算机科学与技术系, 2017

Outline

本章作业

课本习题：5.10、5.14、5.34
思考题：课本习题5.12（可以使用mathlab计算）、5.15、5.30
学习要点：

- 公钥体制的思想
- RSA基本原理
- RSA的实现算法（略，转移到信息安全数学基础）
- RSA的安全性分析：对因子分解算法的掌握不做要求

5.1 公钥密码体制简介

目前为止，我们学习了以下的密码体制：

- 古典密码体制：对称加密，为复杂的加密体制打下基础
- 对称加密
  - 流密码
  - 分组密码
- Hash函数：提供数据完整性保护

5.1 公钥密码体制简介

我们所学习的密码体制都只有一个密钥，特点是：

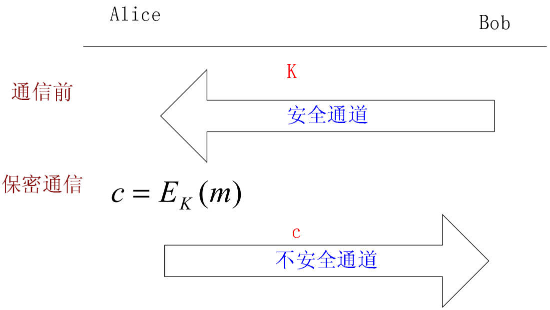
- 加密密钥和解密密钥一致，或者相似(因此称为对称密码)
- 加解密钥都必须保密(因此称为私钥密码)，加密方和解密方都需要知道秘密密钥。

它们的局限性：

1. 密钥分配问题
2. 密钥管理困难
3. 无法解决不可否认性

5.1 公钥密码体制简介
私钥密码体制的局限性

- 密钥分配问题： 通信双方要进行加密通信，需要通过秘密的安全信道协商加密密钥，而这种安全信道可能很难实现。



$$c = E_K(m)$$

$$m = D_K(c)$$

5.1 公钥密码体制简介
私钥密码体制的局限性

Alice　　　　Bob

问题：如果Alice和Bob互不相识，如何安全交换密钥？

保密通信 $c = E_K(m)$

c

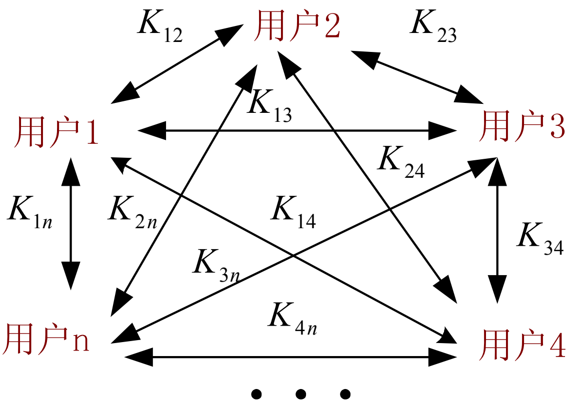
不安全通道

$m = D_K(c)$

5.1 公钥密码体制简介

私钥密码体制的局限性

- 密钥管理困难：在有多个用户的网络中，任何两个用户之间都需要有共享的秘密密钥，如果用户数是n，那么每个人需要保存$n-1$个秘密密钥，共需要$\frac{n(n-1)}{2}$个密钥!



5.1 公钥密码体制简介

- 无法解决不可否认性（即数据的发送方无法否认他发送的数据），因为接收方和发送方都拥有秘密密钥，无从判断数据由谁生成。

Bob　　　　　　　　　　Alice

消息m是一封恐吓信

$c = E_K(m)$　　c

$m = D_K(c)$

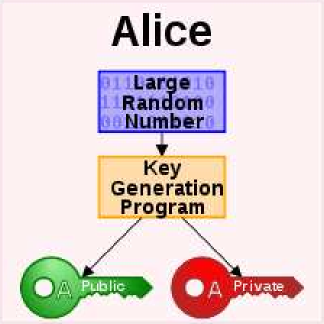
Alice向法庭出示Bob发给她的恐吓信，

但法庭无法识别是Bob发送的，还是Alice捏造的

5.1 公钥密码学简介

公钥密码体制解决了以上问题！它有以下特点：

- 加密密钥*pk*是公开的（也称为公开密钥，公钥），解密密钥*sk* 是保密的（也称为私钥），只有所有者知道。
- 由公开密钥*pk*计算私钥*sk*是是困难的。



5.1 公钥密码学简介

公钥密码体制的思想：Alice有秘密文件要发送给Bob

Alice

Bob

5.1 公钥密码学简介

公钥密码体制的思想：Bob把若干个相同的锁（对应于公钥）给Alice，他拥有一把打开锁的秘密钥匙（对应于私钥）

Alice

Bob

5.1 公钥密码学简介

公钥密码体制的思想：Alice把秘密文件放进箱子里，然后用Bob的锁关上箱子(对应于加密)

Alice

Bob

5.1 公钥密码学简介

公钥密码体制的思想：Bob收到Alice发来的箱子后，用钥匙打开锁(对应于解密)

Alice

Bob

5.1 公钥密码学简介

公钥密码体制的思想：Bob现在可以查看秘密文件了！Alice可以使用剩下的锁来继续传送秘密文件，但在公钥体制中，完全没有加密次数限制！

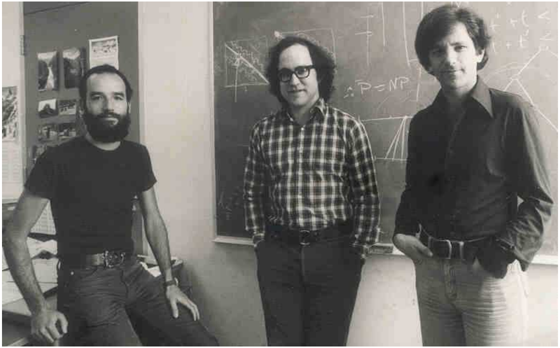
Alice

Bob

5.1 公钥密码学简介

公钥密码体制的问世

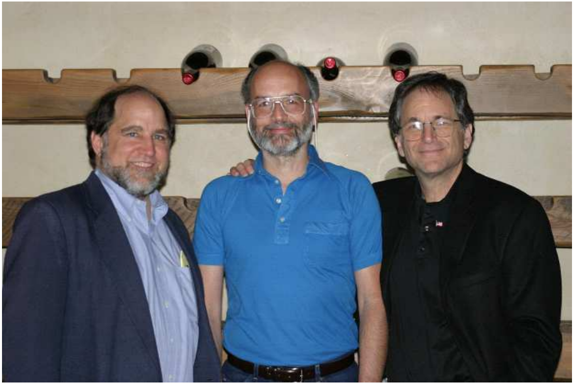
- 1976年Diffie和Hellman在IEEE Info. Theory发表的文章提出来公钥密码体制的思想。
- 1977年Rivest、Shamir和Adleman提出了第一个公钥密码体制—RSA密码体制。



Adi Shamir, Ronald Rivest, and Len Adleman publish RSA at MIT in 1977.

5.1 公钥密码学简介

- 2002年RSA的三位作者获得Turing奖。



2002: Rivest, Shamir, and Adleman receive ACM Turing Award.

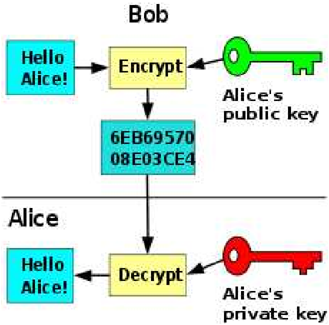
5.1 公钥密码学简介

1997: 1997年12月由英国政府通信总部（GCHQ）正式解密的文档揭示，在GCHQ工作期间， 1973年lifford Cocks在一篇非公开文献中提出了RSA，比正式公开文献提早4年发现。

5.1 公钥密码学简介
公钥密码的应用

公钥密码的应用：

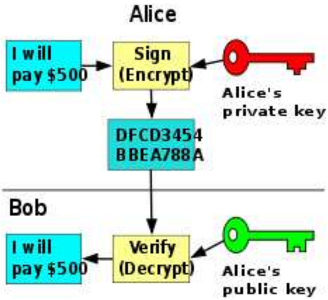
- 加密：Bob用Alice的公钥加密信息，发送给Alice，Alice用她的私钥解密。



5.1 公钥密码学简介

公钥密码的应用

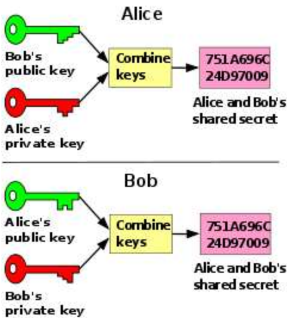
- 数字签名：Alice用她私钥签名支付信息，发送给Bob，Bob用Alice的公钥验证Alice对支付信息的签名。



5.1 公钥密码学简介

公钥密码的应用

- 生成共享密钥：在Diffie-Hellman密钥分配方案中，Alice和Bob可以使用对方的公钥和自己的私钥计算他们的共享秘密密钥，这个共享密钥用于通信中的对称加密。



如何构造安全有效的公钥密码：

- 有效性：加密运算和解密运算能够快速实现。
- 安全性：由公钥pk推导出私钥sk是计算困难的

$$\text{公钥}pk \not\Longrightarrow \text{私钥}sk$$

相当于已知公钥pk及加密函数e_{pk}推导出它的逆运算d_{sk}是困难的。

- 拥有私钥则很容易解密

具有以上性质的函数称为**陷门单向函数**：

1. 单向函数：容易计算但求逆困难的函数。
2. 具有特定的陷门(即私钥)知识后容易求其逆。

单向函数例子：

- $f(p,q)=pq$，其中p,q是两个大素数，大整数的因子分解目前没有有效的算法。
- 安全的Hash函数是单向函数。

单向限门函数例子：

Example

假定$n=pq$，其中p和q为素数，b为整数，那么

$$f(x)=x^b \pmod{n}$$

是单向函数。如果知道n的因子分解，那么可以容易求它的逆：

$$f^{-1}(x)=x^a \pmod{n}$$

其中

$$ab\equiv 1 \pmod{(p-1)(q-1)}$$

5.1 公钥密码学简介

公钥密码的构造及分类

- 单向限门函数通常由数学困难问题构造得到。
- 根据安全性所依赖的困难问题，公钥密码体制主要分为两类
  - 基于分解大整数的困难性：RSA体制及其变体。
  - 基于离散对数问题的困难性：ElGamal密码体制及其变体（如椭圆曲线密码体制）。
  - 其他类型的公钥密码体制有：基于格问题的密码体制、基于背包问题的密码体制、有限自动机密码体制。

5.1 公钥密码学简介

公钥密码的缺点

公钥密码体制的问题

- 算法少，被公认安全的实用算法更少。
- 速度慢，无法用于大量数据加密。公钥密码一般和对称密码结合使用：
  - 发送者Bob：使用对称密钥k加密数据，用Alice的公开密钥pk加密对称密钥k。
  - 接受者Alice：使用她的私钥sk解密对称密钥k，再使用k解密加密数据。

5.1 公钥密码学简介

内容安排

- 5.2节：介绍相关的数论知识
- 5.3-5.5节：介绍RSA密码体制及其快速实现。重点掌握
- 5.6-5.7节：介绍RSA的攻击算法

5.2 更多的数论知识

与本章相关的数论知识：

- 欧几里得（Euclidean）算法：计算gcd(a, b)
- 扩展Euclidean算法：计算$a^{-1} \bmod b$
- 中国剩余定理：求解模数两两互素的同余方程组，用于RSA的解密的快速算法（见P.180 5.13）以及攻击算法中。
- Fetmat小定理：RSA、Fetmat素性测试、Miller-Rabin素性测试的原理
- Lagrange定理：Fetmat小定理在乘法群上的推广

5.2 更多的数论知识
欧几里得（Euclidean）算法

Euclidean算法(辗转相除法)：求$\gcd(a,b)$
基本原理：如果$a=qb+r$，那么$\gcd(a,b)=\gcd(b,r)$。
基本过程：首先令$r_0=a, r_1=b$，然后执行如下除法过程：

$$
\begin{array}{lcll}
r_0 & = & q_1 r_1 + r_2, & 0 \le r_2 < r_1 \\
r_1 & = & q_2 r_2 + r_3, & 0 \le r_3 < r_2 \\
\vdots & & \vdots & \vdots \\
r_{m-2} & = & q_{m-1} r_{m-1} + r_m, & 0 \le r_m < r_{m-1} \\
r_{m-1} & = & q_m r_m &
\end{array}
$$

那么$\gcd(a,b)=\gcd(r_1,r_2)=\ldots=\gcd(r_{m-2},r_{m-1})=r_m$。

Example

为了求gcd$(99, 35)$，执行如下除法过程：

$$
\begin{aligned}
99 &= 2 \times 35 + 29 & \Rightarrow (99, 35) = (35, 29) \\
35 &= 1 \times 29 + 6 & \Rightarrow (35, 29) = (29, 6) \\
29 &= 4 \times 6 + 5 & \Rightarrow (29, 6) = (6, 5) \\
6 &= 1 \times 5 + 1 & \Rightarrow (6, 5) = 1
\end{aligned}
$$

我们把等式从后依次代入：

$$
\begin{aligned}
1 &= 6 - 1 \times 5 \\
&= 6 - 1 \times (29 - 4 \times 6) \\
&= 5 \times 6 - 1 \times 29 \\
&= 5 \times (35 - 1 \times 29) - 1 \times 29 \\
&= 5 \times 35 - 6 \times 29 \\
&= 5 \times 35 - 6 \times (99 - 2 \times 35) \\
&= 17 \times 35 - 6 \times 99 \quad \Longrightarrow 35^{-1} \bmod 99 = 17
\end{aligned}
$$

5.2 更多的数论知识

扩展Euclidean算法

一般的：

- 对任意整数a, b，存在整数s, t满足$sa + tb = \gcd(a, b)$

  证明思路： 由Euclidean算法可见：

$$\begin{aligned}\gcd(a, b) &= r_m = r_{m-2} - q_{m-1} r_{m-1} \\ &= \text{往后依次代入} \ldots = sa + tb\end{aligned}$$

- 如果$\gcd(a, b) = 1$，那么有$sa + tb = 1$，即

$$t \equiv b^{-1} \bmod a$$

- 如何计算s, t?

步骤 j	除法过程	计算s_j, t_j满足 $r_j = s_j a + t_j b$
0	$r_0 = a$	$s_0 = 1, t_0 = 0$
1	$r_1 = b$	$s_1 = 0, t_1 = 1$
2	$r_0 = q_1 r_1 + r_2 \Rightarrow r_2 = r_0 - q_1 r_1$	$s_2 = -q_2$ $t_2 = 1 + q_1$
3	$r_1 = q_2 r_2 + r_3 \Rightarrow r_3 = r_1 - q_2 r_2$	s_3, t_3?
$\vdots$	$\vdots$	$\vdots$
j	$r_{j-2} = q_{j-1} r_{j-1} + r_j \Rightarrow r_j = r_{j-2} - q_{j-1} r_{j-1}$	s_j, t_j?
$\vdots$	$\vdots$	$\vdots$
m	$r_{m-2} = q_{m-1} r_{m-1} + r_m$ $r_{m-1} = q_m r_m$	s_m, t_m?
	$\gcd(a, b) = r_m = s_m a + t_m b$	

下面推导s_j, t_j的递归关系(注意：颜色相同的部分是等价的)：

$$\begin{aligned}
r_j &= r_{j-2} - q_{j-1} r_{j-1} (\text{欧几里德算法}) \\
&= (s_{j-2}a + t_{j-2}b) - q_{j-1}(s_{j-1}a + t_{j-1}b) (\text{代入} r_{j-2}, r_{j-1} \text{的表达式}) \\
&= (s_{j-2} - q_{j-1}s_{j-1})a + (t_{j-2} - q_{j-1}t_{j-1})b \\
&= s_j a + t_j b
\end{aligned}$$

由此得到s_j, t_j的递归关系（此为课本定理5.1的结论。注意，课本中t_j的公式有印刷错误）：

$$\begin{aligned}
s_j &= s_{j-2} - q_{j-1}s_{j-1} \\
t_j &= t_{j-2} - q_{j-1}t_{j-1}
\end{aligned}$$

注：算法中蓝色部分用于求s, t，去掉蓝色部分即为欧几里得算法。

算法5.2 Extendend Euclidean Algorithm(a, b)

$a_0 \leftarrow a, b_0 \leftarrow b$

$t_0 \leftarrow 0, t \leftarrow t, s_0 \leftarrow 1, s \leftarrow 0$

$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor, r \leftarrow a_0 - qb_0$

while $r > 0$

$$\text{do} \begin{cases} temp \leftarrow t_0 - qt, t_0 \leftarrow t, t \leftarrow temp & (\text{计算}t_j\text{并更新}) \\ temp \leftarrow s_0 - qs, s_0 \leftarrow s, s \leftarrow temp & (\text{计算}s_j\text{并更新}) \\ a_0 \leftarrow b_0, b_0 \leftarrow r, q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor, r \leftarrow a_0 - qb_0 & (\text{计算}q_{j+1}, r_{j+2}\text{并更新}) \end{cases}$$

$r \leftarrow b_0$

return (r, s, t)

comment:

- $r = gcd(a, b)$且$sa + tb = r$.
- 若gcd$(a, b) = 1$，则有$t = b^{-1} \bmod a, s = a^{-1} \bmod b$。

5.2 更多的数论知识

扩展Euclidean算法

例5.1：计算$28^{-1} \bmod 75$。根据扩展Euclidean算法如下计算：

i	r_i	q_i	s_i	t_i
0	75		1	0
1	28	2	0	1
2	19	1	1	-2
3	9	2	-1	3
4	1	9	3	-8

那么有$3 \times 75 - 8 \times 28 = 1$，即

$$28^{-1} \bmod 75 = -8 \bmod 75 = 67$$

5.2 更多的数论知识

扩展Euclidean算法

扩展Euclidean算法的改进：

- 改进1：$b^{-1} \bmod a = t \bmod a$，而序列$t_0, t_1, \ldots$与序列$s_0, s_1, \ldots$无关，因此可以把序列$s_0, s_1, \ldots$的有关计算删掉。
- 改进2：每次主循环求t时都进行模a运算，结果不变，而且因为减少了t的长度而减少了主循环运算的计算时间。即：把

$$temp \leftarrow t_0 - qt$$

改成

$$temp \leftarrow t_0 - qt \bmod a$$

改进算法参考课本中的算法5.3

如何利用p, q加快RSA解密运算？

如何由

$$\begin{aligned} x_1 &= y^a \bmod p \\ x_2 &= y^a \bmod q \end{aligned}$$

计算$x = y^a \bmod n$？因为计算$y^a \bmod p$和$y^a \bmod q$要比$y^a \bmod n$快的多，如果对上述问题存在有效算法，就可以加快RSA的解密运算。

解决以上问题的是中国剩余定理。中国剩余定理给出了同余方程组的有效解。

中国剩余定理

假定$m_1, \ldots, m_r$为两两互素的正整数，又假定$a_1, \ldots, a_r$为整数，那么同余方程组

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ \vdots & \quad \vdots \\ x &\equiv a_r \pmod{m_r} \end{aligned}$$

有模$M = m_1 \times m_2 \times \ldots \times m_r$的唯一解，且由下式给出

$$x = \sum_{i=1}^{r} a_i M_i \gamma_i \bmod M$$

其中

$$M_i = \frac{M}{m_i}, \gamma_i = M_i^{-1} \bmod m_i, 1 \leq i \leq r.$$

证明：解的正确性可直接验证。我们看到：

$$M_i \equiv \frac{M}{m_i} \equiv \frac{m_1 m_2 \dots m_r}{m_i} \pmod{m_j} = \begin{cases} 0, j \neq i \\ 1, j = i \end{cases}$$

因此

$$x = \sum_{i=1}^{r} a_i M_i \gamma_i \bmod M \equiv a_j \pmod{m_j}$$

解唯一性证明（反证法）：如果解不唯一，即存在$x \not\equiv y \bmod M$同时满足同余方程组，那么有

$$x - y \not\equiv 0 \bmod M$$

那么存在s, t且$0 < t < M$满足

$$x - y = sM + t = sm_1 m_2 \dots m_r + t$$

因为$m_1, m_2, \dots, m_r$互素，必然存在m_j使得$t \nmid m_j$，也就是

$$x - y = sm_1 \dots m_j \dots m_r + t \equiv t \not\equiv 0 \bmod m_j$$

另一方面$x - y \equiv a_j - a_j \equiv 0 \bmod m_j$, 矛盾！因此解是唯一的。

Example

例5.3 假定$r = 3, m_1 = 7, m_2 = 11, m_3 = 13$，同余方程组

$$x \equiv 5 \pmod 7, x \equiv 3 \pmod{11}, x \equiv 10 \pmod{13}$$

存在模$M = 7 \times 11 \times 13 = 1001$的唯一解。解如下计算：

$$M_1 = 143, M_2 = 91, M_3 = 77$$

$$\gamma_1 = M_1^{-1} \bmod m_1 = 5, \gamma_2 = 4, \gamma_3 = 12$$

因此

$$x = (5 \times 143 \times 5 + 3 \times 91 \times 3 + 10 \times 77 \times 12) \pmod{1001} = 894$$

其它数论结论

- 整数g模M的阶定义为使得$g^m \equiv 1 \bmod M$最小的正整数。
- 对正整数m，定义

$$\mathbb{Z}_m = \{0, 1, \ldots, m-1\}, \mathbb{Z}_m^* = \{x \in \mathbb{Z}_m | \gcd(x, m) = 1\}.$$

例：$\mathbb{Z}_9^* = \{1, 2, 4, 5, 7, 8\}$

- 正整数M的欧拉函数$\phi(M)$定义为小于M且与M互素的正整数的个数。即

$$\phi(m) = |\mathbb{Z}_m^*|$$

如果M的素因子分解如下：

$$M = p_1^{r_1} \ldots p_s^{r_s},$$

其中$p_1, \ldots, p_s$为素数，$r_1, \ldots, r_s > 0$。那么

$$\phi(M) = \frac{M(p_1 - 1) \ldots (p_s - 1)}{p_1 \ldots p_s}$$

特别当$n = pq$，p, q为素数，那么

$$\phi(n) = (p-1)(q-1) \qquad \phi(p) = p - 1$$

其它数论结论

Theorem

(Fermat's Little Theorem，费马小定理) 假定p是一个素数，那么对整数b有

$$b^p \equiv b \pmod p$$

如果$b \not\equiv 0 \bmod p$，则

$$b^{p-1} \equiv 1 \pmod p$$

Theorem

(欧拉定理) 对整数$\gcd(b, n) = 1$，

$$b^{\phi(n)} \equiv 1 \pmod n$$

其它数论结论

群的定义及基本性质(主要在第6章的ElGamal密码体制用到)

定义：群$(G,\cdot)$，其中G为元素集合，$\cdot$是定义在G上的二元运算，且满足

封闭性： 对$\forall a,b\in G$有$a\cdot b\in G$

结合律： 对$\forall a,b,c\in G$有$(a\cdot b)\cdot c=a\cdot(b\cdot c)$。

存在单位元： $\exists e\in G$，使得对$\forall a\in G$有$e\cdot a=a\cdot e=a$，称e为G的单位元。

所有元素可逆： 对$\forall a\in G$，$\exists b\in G$满足$a\cdot b=b\cdot a=e$，则称b为a的逆元，写为$b=a^{-1}$。

备注：

- $a\cdot b$一般简写为ab。
- 当运算符号表示成$\cdot$时，单位元通常写为1；当运算符号表示成$+$时，单位元通常写为0。
- 如果运算符号可从上下文判断出来，用G表示群。

其它数论结论

如果群$(G,\cdot)$满足交换律则称为**交换群**（或**阿贝尔群**，当运算符为乘法$\cdot$时，又称为**乘法群**）。

交换律： 对$\forall a,b\in G$有$ab=ba$。

如果G为有限集合，则称为**有限群**。

定理： $(\mathbb{Z}_m^*,\cdot)$是交换群，单位元为1，其中$\cdot$表示$\bmod m$乘法运算。

例: $\mathbb{Z}_9^*=\{1,2,4,5,7,8\}$是群，那么对每个元素都存在逆元。因为

$$\begin{aligned}
1\times 1 &\equiv 1 \pmod 9\\
2\times 5 &\equiv 1 \pmod 9\\
4\times 7 &\equiv 1 \pmod 9\\
8\times 8 &\equiv 1 \pmod 9
\end{aligned}$$

那么$1^{-1}=1$，$2^{-1}=5$，$4^{-1}=7$，$8^{-1}=8$。

其它数论结论

定理(群的基本性质): 设G是群。

- **(i)** **消去律**成立：如果$xa = xb$或$ax = bx$，则$a = b$。
- **(ii)** **单位元唯一**：e是G中满足对一切$x \in G$有$ex = x = xe$的唯一元素。
- **(iii)** 对每个$x \in G$，其**逆元唯一**：只有一个元素$x' \in G$满足$xx' = e = x'x$。
- **(iV)** 对一切$x \in G$，$(x^{-1})^{-1} = x$。

Theorem

(Lagrange) 假定G是一个阶为n的乘法群(也称为交换群、阿贝尔群)，且$g \in G$。那么g的阶整除n。

注释：

- G的阶n定义为G的元素的个数
- g的阶(order)定义为使得$g^m = 1$的最小的正整数m

循环群(cyclic group)：存在一个元素$\alpha \in G$其阶为$|G|$，则称G为循环群，α称为生成元（也称为原根、本原元）

定理：如果p是一个素数，那么群$\mathbb{Z}_p^*$是一个循环群。如果$\alpha \in \mathbb{Z}_p^*$是其本原元，当且仅当

$$\{\alpha^i : 0 \le i \le p-2\} = \mathbb{Z}_p^*$$

定理：G为群，$g \in G$的阶为n。那么$g^m (m \ge 1)$的阶为$\frac{n}{\gcd(n,m)}$。

例：设$p = 13$。通过计算2的连续幂次，可以验证2是模数13的本原元：

$$\begin{aligned}
&2^1 \equiv 2, 2^2 \equiv 4, 2^3 \equiv 8, 2^4 \equiv 3, 2^5 \equiv 6, 2^6 \equiv 12, \\
&2^7 \equiv 9, 2^8 \equiv 9, 2^9 \equiv 5, 2^{10} \equiv 10, 2^{11} \equiv 7, 2^{12} \equiv 1 \pmod{13}
\end{aligned}$$

其它数论结论

定理： 令p为素数。$f(x)$是Z_p上的多项式，设其次数为n。则$f(x) = 0$在Z_p中至多有n个根。

- 二次剩余、Legendre符号、Jacobi符号定义及运算规则，见Solovay-Strassen素性测试。

5.3 RSA密码体制

- RSA以它的三位发明人名字命名：Rivest, Shamir & Adleman
- 使用最广泛的公钥加密算法
- 1977年提出
- 可用于数据加密和数字签名。
- 基于大整数分解的困难性。

5.3 RSA密码体制

RSA密码体制

公开密钥(n, b) 加密指数b	$n = pq$ b与$(p-1)(q-1)$互素
私钥(a, p, q) 解密指数a	p, q是素数 $a = b^{-1} \bmod (p-1)(q-1)$
加密$e_K(x)$ 解密$d_K(y)$	$y = x^b \bmod n$ $x = y^a \bmod n$

验证加密和解密是逆运算：

$\phi(n)$定义为欧拉函数，即$\mathbb{Z}_n$与n互素的元素个数。那么

$$\phi(n) = (p-1)(q-1)$$

且

$$ab \equiv 1 \pmod{\phi(n)}$$

所以存在某个整数$t \geq 1$有

$$ab = t\phi(n) + 1$$

对所有与n互素的整数x有：

$$\begin{aligned} d_K(y) &\equiv (x^a)^b \pmod{n} \\ &\equiv x^{t\phi(n)+1} \pmod{n} \\ &\equiv (x^{\phi(n)})^t x \pmod{n} \\ &\equiv 1^t x \pmod{n} \\ &\equiv x \pmod{n} \end{aligned}$$

上面等式中利用了数论结论：$x^{\phi(n)} \equiv 1 \pmod{n}$

Example

例5.5 Bob生成密钥参数：

1. 选取$p = 101, q = 113$
2. 计算$n = pq = 11413$,
   $\phi(n) = (p-1)(q-1) = 11200 = 2^6 5^2 7$
3. 选取加密指数$b = 3533$
4. 计算解密指数$a = b^{-1} \bmod 11200 = 6597$

则他的公钥是$(n = 11413, b = 3533)$，
私钥是$(p = 101, q = 113, a = 6597)$。
Alice 使用Bob公开的公钥加密消息9726：

$$e_K(9726) = 9726^{3533} \bmod 11413 = 5761$$

然后把密文通过信道发送给Bob，Bob使用解密指数计算出明文

$$d_K(y) = 5761^{6597} \bmod 11413 = 9726$$

RSA的实现算法：

算法5.4 RSA参数生成算法

1. 生成两个随机大素数p, q。（使用素性测试算法）
2. $n \leftarrow pq$，且$\phi(n) \leftarrow (p-1)(q-1)$
3. 选择一个随机数$b(1 < b < \phi(n))$，使得$\gcd(b, \phi(n)) = 1$（使用Euclidean算法判断）
4. $a \leftarrow b^{-1} \bmod \phi(n)$（使用扩展Euclidean算法计算）
5. 公钥为(n, b)，私钥为(p, q, a)

实现RSA需要涉及以下运算：

运算	算法	计算复杂度(非最优上界)
比较运算：x>y,x=y,x<y		
$x+y$		$O(k)$
$x-y$		$O(k)$
$x\cdot y$		$O((\log(kl))^2)$
$\lfloor x/y \rfloor$		$O(l(k-l))$
$x \bmod n$		
$y^c \bmod n$	平方-乘算法	$O(k^2\log c)$
$gcd(x,y)$	Euclidean算法	$O((\log(k))^2)$
$y^{-1} \bmod n$	扩展Euclidean算法	$O((\log(k))^2)$
生成随机素数	素性测试算法	

其中假定x,n,y的二进制长度分别为k,k,l，且$k\geq l$。

如何快速实现指数运算？

Example

计算：3^9

方法一： 直接计算，需要8次乘法运算。

方法二： 注意到：

$$\begin{aligned}
3^9 &= 3(3^4)^2 \\
&= 3((3^2)^2)^2 && \text{第1次乘法} \\
&= 3(9^2)^2 && \text{第2次乘法} \\
&= 3\cdot 81^2 && \text{第3次乘法} \\
&= 3\cdot 6561 && \text{第4次乘法} \\
&= 19683
\end{aligned}$$

算法原理：

假定指数c的二进制表示为$c=(c_{l-1},\ldots,c_1,c_0)$，那么

$$
\begin{aligned}
c &= c_{l-1}2^{l-1}+c_{l-2}2^{l-2}+\ldots+2c_1+c_0\\
&= (2c_{l-1}+c_{l-2})2^{l-2}+c_{l-3}2^{l-3}+\ldots+2c_1+c_0\\
&= (2(2c_{l-1}+c_{l-2})+c_{l-3})2^{l-3}+c_{l-4}2^{l-4}+\ldots+2c_1+c_0\\
&= 2(\ldots 2(2(2c_{l-1}+c_{l-2})+c_{l-3})+\ldots)+c_1)+c_0
\end{aligned}
$$

因此有：

$$
\begin{aligned}
x^c &= x^{2(\ldots 2(2(2c_{l-1}+c_{l-2})+c_{l-3})+\ldots)+c_1)+c_0}\\
&= (\ldots(((x^{c_{l-1}})^2x^{c_{l-2}})^2x^{c_{l-3}})\ldots)^2x^{c_0}\\
&= (\ldots(((1^2\cdot x^{c_{l-1}})^2x^{c_{l-2}})^2x^{c_{l-3}})\ldots)^2x^{c_0}
\end{aligned}
$$

根据：$x^c = (\ldots(((1^2 \cdot x^{c_{l-1}})^2 x^{c_{l-2}})^2 x^{c_{l-3}})\ldots)^2 x^{c_0}$

平方-乘算法(x, c, n)

计算：$z = x^c \bmod n$。
假定c的二进制表示为$c = \sum_{i=0}^{l-1} c_i 2^i, c_i \in \{0, 1\}$。
$z \leftarrow 1$(初始化)
for $i \leftarrow l - 1$ downto 0

$$\text{do} \begin{cases} z \leftarrow z^2 \bmod n & (\ldots)^2 \\ \text{if } \; c_i = 1 & \\ \text{then } \; z \leftarrow (z \times x) \bmod n & (\ldots \cdot x^{c_i}) \end{cases}$$

return (z)

计算复杂度分析：如果n的二进制表示有k位

- 乘法运算的复杂度为$O(k^2)$
- 乘法运算次数为$O(l) = O(\log_2(c))$

计算复杂度为$O((\log c) \times k^2)$。

Example

例5.5（续）$n = 11413$，公开的解密指数为$b = 3533$，Alice利用平方-乘算法计算$9726^{3533} \bmod 11413$ 来加密明文9726：

i	b_i	z
11	1	$1^2 \times 9726 = 9726$
10	1	$9726^2 \times 9726 = 2659$
9	0	$2659^2 = 5634$
8	1	$5634^2 \times 9726 = 9167$
7	1	$9167^2 \times 9726 = 4958$
6	1	$4958^2 \times 9726 = 7783$
5	0	$7783^2 = 6298$
4	0	$6298^2 = 4629$
3	1	$4629^2 \times 9726 = 10185$
2	1	$10185^2 \times 9726 = 105$
1	0	$105^2 = 11025$
0	1	$11025^2 \times 9726 = 5761$

计算复杂度分析：

假定x的二进制表示位数为k，y的二进制表示位数为l，那么：

计算	复杂度
$x+y$	$O(k)$
$x-y$	$O(k)$
xy	$O(kl)$
$\lfloor x/y \rfloor$	$O(l(k-l))$
$\gcd(x,y)$	$O(k^2)$

假定n的二进制表示位数为k，$0 \le m_1, m_2 \le n-1$，那么

计算	复杂度
$(m_1+m_2) \bmod n$	$O(k)$
$(m_1-m_2) \bmod n$	$O(k)$
$(m_1 m_2) \bmod n$	$O(k^2)$
$(m_1)^{-1} \bmod n$	$O(k^2)$
$(m_1)^c \bmod n$	$O((\log c) \times k^2)$

素性测试

生成随机素数的方法：

1. 生成随机整数n
2. 辨别n是否为素数，有两种辨别方法：

确定性算法 概率1确定x是否为素数，2002年三位印度计算机科学家发现了第一个多项式时间的算法，称为AKS素性测试，计算复杂度为$O(\log^{12}(n))$

随机算法 如果x通过某些素数判定准则，则x可能为素数，如果不通过则x肯定为合数。

如：Solovay-Strassen素性测试、Miller-Rabin素性测试、Fermat素性测试、Lucas素性测试。

素性测试

成功概率分析

- 在$1 \sim N$之间随机选取一个数，其为素数的概率$\approx 1/\ln N$。
- 512比特的随机整数为素数的概率大约为$1/\ln 2^{512} \approx 1/355$。
- 在RSA中，大素数p, q选取为512比特的素数，是可以在随机选取的355 个数中以高概率找到一个素数的。

素性测试
Fermat素性测试

- 回顾Fermat小定理：如果p为素数，则对$a \neq 0 \bmod p$有$a^{p-1} \equiv 1 \bmod p$。
- 反之，如果对整数n，整数$0 < a < n$有$a^{n-1} \not\equiv 1 \bmod n$，则$n$为合数。

根据以上性质构造素性测试算法：

Fermat素性测试

输入：　整数n，测试次数k

反复运行k次：　随机选取整数$a \in \{1, \ldots, n-1\}$

如果$a^{n-1} \not\equiv 1 \bmod n$则$n$为合数，否则$n$可能为素数。

素性测试

Fermat素性测试

- 通过增加测试次数k来提高测试准确性。
- 计算复杂度：$O(\log(n)^3)$
- 缺陷：存在合数n，

$$\text{对所有}0 < a < n, \gcd(a, n) = 1\text{有}a^{n-1} \equiv 1 \bmod n$$

  这样的合数称为Carmichael数，如561是一个Carmichael数。

  - Carmichael 数非常稀少，而且距离很远。对小于10^{16}的整数，存在246,683个Carmichael数，279,238,341,033,925个素数。
- 应用：PGP加密软件使用Fermat素性测试，在PGP中，通过测试的数为Carmichael数的概率小于$\frac{1}{10^{50}}$。

素性测试
基本概念

- 判定问题：只回答“是(yes)”或者“否(No)”的问题。
- 随机算法：使用了随机数的算法。
- 判定问题的一个偏是(yes-biased)Monte-Carlo算法：算法给出“是”的回答总是正确的，给出“否”的回答也许不正确。如果对应该为“是”的输入至多以ε的概率给出“否”的答案则说该算法具有错误概率ε。
- 判定问题的一个偏否(no-biased)Monte-Carlo算法：算法给出“否”的回答总是正确的，给出“是”的回答也许不正确。果对应该为“否”的输入至多以ε的概率给出“是”的答案则说该算法具有错误概率ε。

- Miller-Rabin测试，也称为强伪素数测试，是对Fermat测试的改进。
- 原理：$a^{p-1} \equiv 1 \bmod p$等价于$a^{(p-1)/2} \equiv \pm 1 \bmod p$，因为对素数$p$，$x^2 \equiv 1 \bmod p$有2个解$\pm 1$（对合数解的数量多于2或无解）
- 分析：如果$p-1 = 2^k m$，其中m是一个奇数。

$$
\begin{array}{lrcl}
 & a^{2^k m} \equiv 1 \bmod p & \Leftrightarrow & a^{2^{k-1} m} \equiv \pm 1 \bmod p \\
\text{如果} & a^{2^{k-1} m} \equiv 1 \bmod p & \Leftrightarrow & a^{2^{k-2} m} \equiv \pm 1 \bmod p \\
\text{如果} & a^{2^{k-2} m} \equiv 1 \bmod p & \Leftrightarrow & a^{2^{k-3} m} \equiv \pm 1 \bmod p \\
 & \vdots & & \\
\text{如果} & a^{2m} \equiv 1 \bmod p & \Leftrightarrow & a^{m} \equiv \pm 1 \bmod p
\end{array}
$$

- 结论：如果$p-1 = 2^k m$是素数，序列$a^m, a^{2m}, \ldots, a^{2^{k-1}m}, a^{2^k m} \bmod p$形如

$$(1, 1, \ldots, 1)\text{或者}(*, \ldots, *, -1, 1, \ldots, 1)$$

根据上面的素数性质，推导出Miller-Rabin素性测试，前面的讨论知道，对于合数问题是一个偏是的Monte Carlo算法。

算法5.7 Miller-Rabin(n)

把$n-1$写成$n-1=2^k m$，其中m是一个奇数

随机选取整数a，使得$1 \le a \le n-1$

$b \leftarrow a^m \bmod n$ （从a^m开始检查）

If $b \equiv 1 \pmod{n}$ （这时形为$(1,1,\ldots,1)$）

    then return ("n is prime")

for $i \leftarrow 0$ **to** $k-1$

do

- **if** $b \equiv -1 \pmod{n}$ （这时形为$(* * * -1,1,\ldots,1)$）
- **then return** ("n is prime")
- **else** $b \leftarrow b^2 \bmod n$

return ("n is composite")

错误概率分析

- 如果n是奇合数，则至多有$(n-1)/4$个$a \in \{1, \ldots, n-1\}$让n通过Miller-Rabin测试。
- 这说明奇合数只有至多$1/4$的概率通过一次Miller-Rabin测试。
- 奇合数通过k次Miller-Rabin测试的概率至多为$1/4^k$。

定义两个随机变量：

a：一个特定长度的随机奇整数n是合数

b：算法连续回答了m次“n是一个素数”

分析：

- 错误概率为$Pr[a|b]$，待求
- $Pr[b|a] \le \frac{1}{4^m}$ (即合数通过m次素性测试的概率)
- $Pr[\bar{a}] \approx \frac{2}{\ln n}$，即奇整数$n$为素数的概率 (根据素数性质得到)
- $Pr[a] \approx 1 - \frac{2}{\ln n}$
- $Pr[b|\bar{a}] = 1$

演算

$$
\begin{aligned}
Pr[a|b] &= \frac{Pr[b|a]Pr[a]}{Pr[b]} && \text{(贝叶斯公式)} \\
&= \frac{Pr[b|a]Pr[a]}{Pr[b|a]Pr[a] + Pr[b|\bar{a}]Pr[\bar{a}]} && \text{(全概率公式)} \\
&\approx \frac{Pr[b|a](\ln n - 1)}{Pr[b|a](\ln n - 1) + 2} && \text{(代入前面的估计式并约简)} \\
&\leq \frac{4^{-m}(\ln n - 1)}{4^{-m}(\ln n - 1) + 2} && (\text{代入} Pr[b|a] \leq 4^{-m}) \\
&= \frac{\ln n - 2}{\ln n - 2 + 2^{2m+1}}
\end{aligned}
$$

m	4^{-m}	错误概率的界
1	0.250	0.978
5	0.977×10^{-3}	0.147
10	0.954×10^{-6}	0.168×10^{-3}
50	0.789×10^{-30}	0.139×10^{-27}

素性测试
Solovay-Strassen测试（略）

- Solovay-Strassen测试没有Miller-Rabin测试效率高（运行一次算法，奇合数通过Miller-Rabin测试的概率至多为1/4，通过Solovay-Strassen测试至多为1/2）。
- Solovay-Strassen测试有关的定义：

二次剩余

假设p是奇素数，那么：

a定义为模p的二次剩余 ： $a \not\equiv 0 \pmod{p}$且剩余方程$y^2 \equiv a \pmod{p}$有解。

a定义为模p的二次非剩余 ： $a \not\equiv 0 \pmod{p}$且剩余方程$y^2 \equiv a \pmod{p}$无解。

素性测试
Solovay-Strassen测试

Solovay-Strassen测试有关的定义：

定义5.3(Legendre符号)

假设p是奇素数，对任一整数a，定义legendre符号$\left(\frac{a}{p}\right)$如下：

$$\left(\frac{a}{p}\right)=\begin{cases}0 & a\equiv 0 \pmod p\\ 1 & a\text{是一个模}p\text{二次剩余}\\ -1 & a\text{是一个模}p\text{二次非剩余}\end{cases}$$

可证明

$$\left(\frac{a}{p}\right)\equiv a^{(p-1)/2} \pmod p$$

素性测试
Solovay-Strassen测试

Solovay-Strassen测试有关的定义：
把奇素数的Legendre符号推广到奇数上：

定义5.4(Jacobi符号)

假设n是正奇数，且n的素因子分解如下：

$$n = \prod_{i=1}^{k} p_i^{e_i}$$

对整数a，定义Jacobi符号$\left(\frac{a}{n}\right)$如下：

$$\left(\frac{a}{n}\right) \equiv \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{e_i}$$

素性测试

Solovay-Strassen测试

Solovay-Strassen测试的原理

- 如果n是奇素数，那么$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n}$成立。
- 如果n是奇合数，那么$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n}$成立的概率至多为$1/2$，同余方程成立的$a$称为对于基底$n$的Euler伪素数。

算法5.6 Solovay-Strassen算法(n)

随机选取整数a，使得$1 \leq a \leq n-1$
计算$x \leftarrow \left(\frac{a}{n}\right)$
如果$x = 0$ 那么返回(“n是合数”)
否则计算$y \leftarrow a^{(n-1)/2} \pmod{n}$，如果$x \equiv y \pmod{n}$，那么返回(“$n$是素数”)，否则返回(“$n$是合数”)

素性测试

Solovay-Strassen算法的有效性需回答以下2个问题：

- 如何有效的计算$\left(\frac{a}{n}\right)$：根据p.149的性质1-4
- 测试多少次才能以高概率确定一个奇数为素数？错误概率是多少？

素性测试

有效的计算$\left(\frac{a}{n}\right)$的4条依据：

1.如果n是一个正奇数，且$m_1 \equiv m_2 \pmod{n}$，那么

$$\left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right)$$

2.如果n是一个正奇数，那么：

$$\left(\frac{2}{n}\right) = \begin{cases} 1 & 1 \equiv \pm 1 \pmod{8} \\ 1 & -1 \equiv \pm 3 \pmod{8} \end{cases}$$

3.如果n是一个正奇数，那么：

$$\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right)\left(\frac{m_2}{n}\right)$$

特别地，如果$m = 2^k t$且t为一个奇数，那么：

$$\left(\frac{m}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{t}{n}\right)$$

素性测试

4.如果m,n正奇数，那么：

$$\left(\frac{m}{n}\right)=\begin{cases}-\left(\frac{n}{m}\right) & m\equiv n\equiv 3 \pmod 4\\ \left(\frac{n}{m}\right) & \text{其他情况}\end{cases}$$

例5.8：计算$\left(\frac{7411}{9283}\right)$
计算$\left(\frac{m}{n}\right)$的一般过程：

- 把m写成如下形式$m=2^k t$且t是奇数，根据性质2,3把问题归结为计算$\left(\frac{t}{n}\right)$
- 如果$t<n$，那么根据性质4归结为计算$\left(\frac{n}{t}\right)$，否则根据性质1归结为计算$\left(\frac{s}{n}\right)$，其中$s=t \bmod n$
- 继续以上过程直到得到最终结果

所需要的时间复杂度为$O((\log(n))^2)$。

素性测试

运行m次Solovay-Strassen算法，错误概率分析：
定义随机变量：

- a：一个特定长度的随机奇整数n是一个合数
- b：算法连续回答了m次“n是一个素数”

分析如下：

- 输入为合数时运行1次算法回答是素数的概率$\leq \frac{1}{2}$(见习题5.22)
- 输入为合数时运行m次算法都回答是素数的概率$= Pr[b|a] \leq 2^{-m}$
- 算法运行m次都回答n是素数时，n是合数的概率$= Pr[a|b]$

$$Pr[a|b] \leq \frac{\ln n - 2}{\ln n - 2 + 2^{m+1}}$$

m	2^{-m}	错误概率的界
10	$.977 \times 10^{-3}$	$.147$
50	$.888 \times 10^{-15}$	$.157 \times 10^{-12}$
100	$.789 \times 10^{-30}$	$.139 \times 10^{-27}$

5.6 分解因子算法

RSA的分析方法：

- 分解模数n
  - 二次筛法
  - 椭圆曲线分解算法
  - 数域筛法
  - 其它算法：Pollard的ρ方法和$\rho-1$算法，William的$p+1$算法，连分式算法，试除法等。
- 计算$\phi(n)$
- 计算解密指数
- 其它攻击

5.6 分解因子算法
试除法

分析：假设n是奇合数，那么n有一个素因子$p \le \lfloor\sqrt{n}\rfloor$

- 对每一正整数$a \le \lfloor\sqrt{n}\rfloor$，如果$a$整除$n$，则停止，并输入$a$作为$n$的因子。
- 计算复杂度为$O(\sqrt{n})$，只适合$n < 10^{12}$。

原理：

- 假设p是n的一个素因子，且$p-1$的每个素数幂因子$q \leq B$。那么$(p-1)|B!$
- 因此$a \equiv 2^{B!} \equiv 2^{p-1} \equiv 1 \pmod{p}$ (费马小定理)
- 最后有$p|(a-1) \Rightarrow$ 很可能有$\gcd(a-1, n) = p$

算法5.8 Pollard $\rho - 1$ factoring Algorithm

输入：(n, B) (n是待分解整数，B是指定的“界”)
$a \leftarrow 2$
for $j \leftarrow 2$ to B (计算$a \leftarrow 2^{B!} \pmod{n}$)
  do $a \leftarrow a^j \pmod{n}$
$d \leftarrow \gcd(a-1, n)$
if $1 < d < n$ then return (d)
else return (" *faiture*")

5.6 分解因子算法
Pollard $\rho - 1$算法

例5.9

$n = 15770708441$。如果取$B = 180$，根据算法5.8计算得到$a = 11620221425$，$d = (a - 1, n) = 135979$，因此d是n的一个因子。

- $\rho - 1$算法的计算复杂度：$OB \log B(\log n)^2 + (\log n)^3)$
- 成功的关键：对n的某个素因子，对每一个素数幂$q|p - 1$都有$q \leq B$，即要求n有一个素因子p使得$p - 1$只有小的素因子。
- RSA的模数n避免$\rho - 1$因子分解的一般方法：选择两个大素数p_1, q_1，使得$p = 2p_1 + 1, q = 2q_1 + 1$（这种形式的素数称为安全素数）

Pollard的ρ方法(不做要求)

思路：假设f是整系数多项式，x_1是正整数，考虑序列$x_1, x_2, \ldots$，其中

$$x_j = f(x_{j-1}) \bmod n \quad j \geq 2$$

对n的一个素因子p，必然存在i, j有(用反证法易得)

$$x_i \equiv x_j \pmod p \Rightarrow f(x_i) \equiv f(x_j) \pmod p$$

因为$x_{i+1} = f(x_i) \bmod n, x_{j+1} = f(x_j) \bmod n$，那么：

$$\begin{aligned}
x_{i+1} \bmod p &= (f(x_i) \bmod n) \bmod p = f(x_i) \bmod p \\
x_{j+1} \bmod p &= f(x_j) \bmod p \\
&\Rightarrow x_{i+1} \equiv x_{j+1} \pmod p \\
&\Rightarrow x_{i+2} \equiv x_{j+2} \pmod p \\
&\Rightarrow x_{i+3} \equiv x_{j+3} \pmod p \\
&\vdots
\end{aligned}$$

5.6 分解因子算法

那么对所有$i' = lk$，其中$l = j - i$有

$$x_{i'} = x_{lk} \equiv x_{lk+l} \equiv x_{lk+2l} \equiv \ldots \equiv x_{2lk} = x_{2i'} \pmod p$$

因此

$$p | (x_{i'} - x_{2i'}, n)$$

根据$x_{i+1} = f(x_i)$生成两个序列

$$\begin{array}{llllll} & x_1, & x_2, & x_3, & \ldots, & x_i & \ldots \\ & x_2, & x_4, & x_6, & \ldots, & x_{2i} & \ldots \\ \text{计算：} & x_1 - x_2 & x_2 - x_4 & x_3 - x_6 & \ldots & x_i - x_{2i} & \ldots \end{array}$$

根据以上的讨论，必存在$x_i - x_{2i}$有$(x_i - x_{2i}, n) \neq 1$

5.6 分解因子算法

Pollard的ρ方法

算法5.9 Pollard ρ Factoring Algorithm (n, x_1)

输入：整数系数多项式$f(x)$，n, x_1

$x \leftarrow x_1$

$x' \leftarrow f(x) \bmod n$

$p \leftarrow gcd(x - x', n)$

while $p = 1$

$$
do \begin{cases}
\text{注：在第}i\text{次反复中，} x = x_i, x' = x_{2i} & \\
x \leftarrow f(x) \bmod n & (\text{计算}x_i) \\
x' \leftarrow f(x') \bmod n & (\text{计算}x_{2i-1}) \\
x' \leftarrow f(x') \bmod n & (\text{计算}x_{2i}) \\
p \leftarrow gcd(x - x', n) &
\end{cases}
$$

if $p = n$ then return ("failure")

else return (p)

$n = 7171 = 71 \times 101$，$f(x) = x^2 + 1$，$x_1 = 1$，那么序列x_i前21个数为

1	2	5	26	677	6557	4105
6347	4903	2218	219	4936	4210	4560
4872	375	4377	4389	2016	5471	88

上面的值，模71后的结果如下

1	2	5	26	38	25	58
28	4	17	6	27	21	16
44	20	46	58	28	4	17

上面表中第一个碰撞为：

$$x_7 \bmod 71 = x_{18} \bmod 71 = 58$$

因此$l = 18 - 7 = 11$，那么

$$(x_{11} - x_{22}, n) \neq 1$$

5.6 分解因子算法
Dixon的随机平方算法

基本原理：

- 如果有$x \not\equiv \pm y \pmod{n}$且$x^2 \equiv y^2 \pmod{n}$，则

$$n|(x-y)(x+y)$$

那么$gcd(x+y,n)$和$gcd(x-y,n)$都是n的不为1和n的因子。

寻找x, y满足$x^2 \equiv y^2 \pmod{n}$的思路：
（1）选取因子基$\mathfrak{B} = \{p_1, \ldots, p_b\}$（如取为最小的$b$个素数）。
（2）选取整数$z_1, z_2, \ldots, z_c$，假定有（若没有如下形式则继续选取）：

$$\left\{\begin{array}{lll} z_1^2 & \equiv & p_1^{\alpha_{11}} \times p_2^{\alpha_{21}} \times \ldots \times p_b^{\alpha_{b1}} (\bmod n) \\ z_2^2 & \equiv & p_1^{\alpha_{12}} \times p_2^{\alpha_{22}} \times \ldots \times p_b^{\alpha_{b2}} (\bmod n) \\ & \vdots & \\ z_c^2 & \equiv & p_1^{\alpha_{1c}} \times p_2^{\alpha_{2c}} \times \ldots \times p_b^{\alpha_{bc}} (\bmod n) \end{array}\right. \tag{1}$$

(3)对$x_i \in \{0, 1\}$有

$$z_1^{2x_1} z_2^{2x_2} \ldots z_c^{2x_c} \equiv p_1^{\alpha_{11}x_1 + \alpha_{12}x_2 + \ldots + \alpha_{1c}x_c} \times \ldots \times p_b^{\alpha_{b1}x_1 + \alpha_{b2}x_2 + \ldots + \alpha_{bc}x_c} \pmod{n} \tag{2}$$

若有

$$\left\{\begin{array}{llll} \alpha_{11}x_1 + \alpha_{12}x_2 + \ldots + \alpha_{1c}x_c & \equiv & 0 & \pmod 2 \\ & \vdots & & \\ \alpha_{b1}x_1 + \alpha_{b2}x_2 + \ldots + \alpha_{bc}x_c & \equiv & 0 & \pmod 2 \end{array}\right. \tag{3}$$

则同余式(2)两边均为平方数。当$c > b$时，同余方程(3)很可能有解，判断是否有解及求解可用Gaussian消除法。

算法优化：

- 取$z=j+\sqrt{kn}, j=0,1,2\dots, k=1,2\dots$时，$z^2 \bmod n$较小，此时能在因子基$\mathfrak{B}$中分解的概率较高。
- 假定$n\approx 2^r, m\approx 2^{\sqrt{r \operatorname{lb} r}}$，$|\mathfrak{B}|=b\approx \frac{m}{\ln m}$时，有优化的运行时间：

$$O(e^{1+o(1)\sqrt{\ln n \ln\ln n}})$$

二次筛法、椭圆曲线算法和数域筛法的渐进运行时间：

二次筛法	$O(e^{(1+o(1))\sqrt{\ln n \ln\ln n}})$
椭圆曲线算法	$O(e^{(1+o(1))\sqrt{2\ln p \ln\ln p}})$
数域筛法	$O(e^{(1.92+O(1)(\ln n)^{1/3}(\ln\ln n)^{2/3})})$

5.6 分解因子算法

因子分解的里程碑事件

- 1983年，用二次筛法成功分解了一个69位的10进制数。
- 1986年，Lenstra和Manasse利用二次筛法成功分解了一个106位的十进制整数。
- 1994年4月Atkins、Graff、Lenstra和Leyland使用二次筛法分解了一个称作RSA-129的129位的10进制整数。
- RSA-130在1996年被分解，RSA-140在1999年2月被分解，RSA-155在1999年被分解。

5.7 对RSA的其它攻击

计算$\phi(n)$

- 如果知道$\phi(n)$，那么可由加密指数b计算出解密指数$a = b^{-1} \bmod \phi(n)$
- 计算$\phi(n)$并不比分解n困难。如果知道$n, \phi(n)$，那么可以通过求解如下方程来分解n：

$$pq = n$$

$$\phi(n) = (p-1)(q-1) \Longrightarrow p + q = -n + \phi(n) - 1$$

等价于求解如下方程

$$p^2 - (n - \phi(n) + 1)p + n = 0$$

5.7.1 计算$\phi(n)$

例 5.13

假定$n = 84\,773\,093$，且敌手已经得到$\phi(n) = 84\,754\,668$，那么通过求解如下方程计算得到两个根9539和88 887

$$p^2 - 18\,426p + 84\,773\,093 = 0$$

5.7.2 解密指数

- 如果解密指数a已知，那么n可以通过一个随机算法在多项式时间内分解。即计算a并不比分解n容易。
- 由解密指数a分解n的**原理**(根据中国剩余定理)：

$$x^2 \equiv 1 \pmod{n = pq} \Leftrightarrow \begin{cases} x^2 \equiv 1 \pmod{p} \\ x^2 \equiv 1 \pmod{q} \end{cases}$$

  因此$x^2 \equiv 1 \pmod{n = pq}$如果有根存在，那么存在4个根，其中2个为$\pm 1 \bmod n$，称为平凡根，如果$x \neq \pm 1 \pmod{n}$，因为$(x-1)(x+1) = kn$那么$\gcd(x+1, n) \neq 1$.
- 算法关键：找到$x^2 \equiv 1 \pmod{n}$且$x \neq \pm 1 \pmod{n}$

5.7.2 解密指数

- 分解算法思路：
  假设$ab \equiv 1 \pmod{\phi(n)}$,$ab - 1 = 2^s r$，$r$为奇数。如果$(w, n) = 1$，由欧拉公式得到

$$w^{ab} \equiv w \pmod{n}$$

或者

$$w^{2^s r} \equiv 1 \pmod{n} \Rightarrow w^{2^{s-1} r} \bmod n = \begin{cases} 1\text{：继续开平方} \\ -1\text{：失败} \\ \text{其它值：可成功分解} \end{cases}$$

即序列$(w^r, w^{2r}, w^{2^2 r}, \ldots, w^{2^s r})$形为以下三种类型：

$$\begin{cases} (1, 1, \ldots, 1) & \text{：无法分解}\ n \\ (*, *, \ldots, -1, 1, \ldots, 1) & \text{：无法分解}\ n \\ (*, *, \ldots, x = w^{2^t r} \bmod n \neq \pm 1, 1, \ldots, 1) & \text{：成功}, \gcd(x+1, n) \neq 1 \end{cases}$$

5.7.2 解密指数

- 根据以上思路得到课本算法5.10(P.159)，其运行一次的成功概率至少为$1/2$。

计算复杂度中的相关术语：

定义5.5（图灵规约）： 假定G和H为问题。一个从G到H的图灵规约(Turing reduction)是一个具有如下性质的算法SolveG：

1. 假定存在某一算法SolveH求解问题H；
2. SolveG可以调用SolveH并使用它的任一输出值，但SolveG不能对SolveH执行的实际运算做任何限定（也就是说，把SolveH看成“黑盒子”，称为谕示器(oracle)）；
3. 假定SolveH的运行时间是$O(1)$时，SolveG是一个多项式时间算法；
4. SolveG正确地求解问题G。

如果存在一个从G到H的图灵规约，则记为$G \propto_T H$。

已证：

- 计算解密指数a及$\phi(n) \propto_T$分解n
- 分解$n \propto_T$计算a
- 分解$n \propto_T$计算$\phi(n)$

计算复杂度中的相关术语：

计算容易： 如果该问题存在多项式时间算法

计算困难： 如果该问题不存在多项式时间算法

G问题难度≤H问题： 如果$G \propto_T H$。这时如果H计算容易，G也计算容易。但G计算容易，H未必计算容易。

G问题难度=H问题： G问题难度≥H问题并且G问题难度≤H问题。因此：

- 分解n的难度=计算a的难度=计算$\phi(n)$的难度

5.7.3 Wiener的低解密指数攻击

M.Wiener利用连分式原理，如果RSA的解密指数a满足如下条件：

$$3a < n^{1/4}, q < p < 2p$$

那么可以有效计算出解密指数a。

5.9 RSA的语义安全性

敌手攻击的目的：

- 完全破解（total break）：敌手找到Bob的秘密密钥
- 部分攻破（partial break）：敌手能以某一不可忽略的（non-negligible）概率解密以前没有见过的密文。
- 密文识别（distinguishable of ciphertext）：敌手能够以超过1/2的概率识别两个给定明文对应的密文，或者识别出给定明文的密文和随机串。

语义安全（semantic security）：敌手不能在多项式时间内识别出密文。

5.9 RSA的语义安全性
5.9.1 与明文比特相关的部分信息

RSA密文“泄露”出去的信息：给定密文y，因为$y = x^b \bmod n$，且b为奇数（因为$\gcd(b, \phi(n)) = 1$），因此Jacobi符号

$$\left(\frac{y}{n}\right) = \left(\frac{x}{n}\right)^b = \left(\frac{x}{n}\right)$$

所以，无需密钥就可以从密文有效计算$\left(\frac{x}{n}\right)$。
因此，RSA不是语义安全的。

5.9 RSA的语义安全性
5.9.1 与明文比特相关的部分信息

考虑2个部分信息泄露问题：给定$y = e_K(x)$

- 计算
$$parity(y) = \begin{cases} 0 & \text{x为偶数} \\ 1 & \text{x为奇数} \end{cases}$$
即$parity(y)$表示x的二进制表示的最低位
- 计算
$$half(y) = \begin{cases} 0 & \text{当}0 \le x < n/2 \\ 1 & \text{当}n/2 \le x < n \end{cases}$$

下面证明：计算RSA密文、计算$parity(y)$和计算$half(y)$的难度一样。

5.9 RSA的语义安全性
5.9.1 与明文比特相关的部分信息

如果能有效计算$half(y)$，则能有效计算密文：
在$\mathbb{Z}_n$中RSA有乘法性质：$e_K(x_1)e_K(x_2) = e_K(x_1x_2)$。所以

$$y2^{ib} \bmod n = ye_K(2^i) \bmod n = e_K(x2^i \bmod n)$$

因此

$$\begin{aligned}
& half(y2^{ib} \bmod n) \\
&= half(e_K(x2^i \bmod n)) \\
&= \begin{cases} 0 & \text{当} 0 \le x2^i \bmod n < n/2 \Leftrightarrow \frac{kn}{2^i} \le x < \frac{n/2+kn}{2^i} \\ 1 & \text{当} n/2 \le x2^i \bmod n < n \Leftrightarrow \frac{n/2+kn}{2^i} \le x < \frac{(k+1)n}{2^i} \end{cases}
\end{aligned}$$

其中$k = 0, 1, \ldots, 2^i - 1$。取$i = 0, 1, \ldots, \lfloor lbn \rfloor$，就可由上式唯一确定出$x$的值。

5.9 RSA的语义安全性
5.9.1 与明文比特相关的部分信息

计算$parity(y)$多项式时间等价于计算$half(y)$：

$$\begin{aligned} half(y) &= parity((y \times e_K(2)) \bmod n) \\ parity(y) &= half((y \times e_K(2^{-1})) \bmod n) \end{aligned}$$