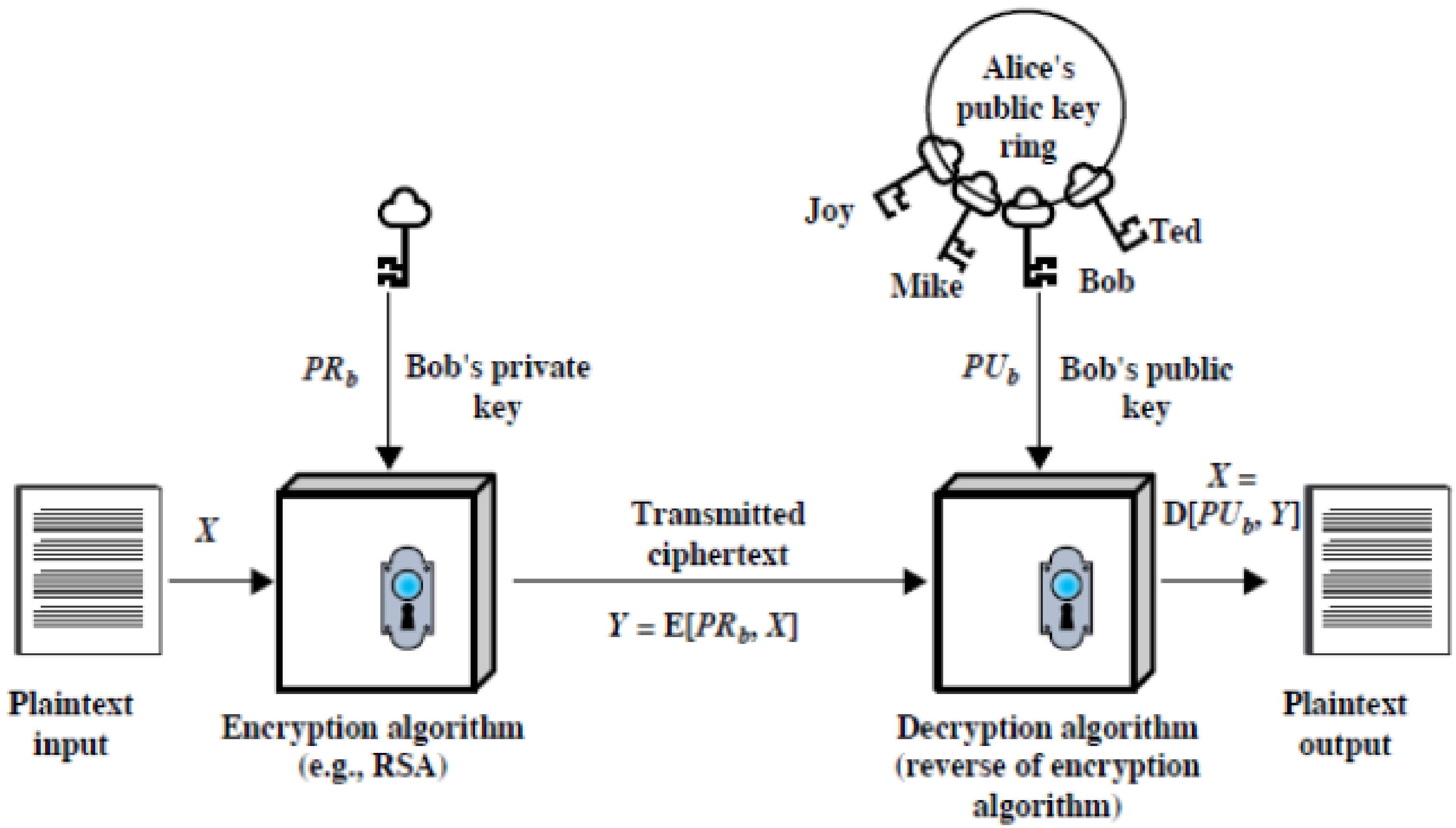
第5章 RSA密码体制和整数因子分解

苏 明

概览

- **5．1** 公钥密码学简介
- **5．2** 更多的数论知识
- **5．3 RSA**密码体制
- **5．4** 素性检测
- **5．5** 模**n**的平方根
- **5．6** 分解因子算法
- **5．7** 对**RSA**的其他攻击
- **5．8 Rabin**密码体制
- **5．9 RSA**的语义安全性

公钥体制


Alice's
public key
ring
Joy
Mike
Bob
Ted
PR_b
Bob's private
key
PU_b
Bob's public
key
X
Transmitted
ciphertext
$Y = E[PR_b, X]$
$X =$
$D[PU_b, Y]$
Plaintext
input
Encryption algorithm
(e.g., RSA)
Decryption algorithm
(reverse of encryption
algorithm)
Plaintext
output

RSA公钥加密算法

- RSA是分组密码，对于某个n,它的明文和密文定义在Z_n上。
- 对于明文块M和密文块C,加密和解密有如下的形式：

$$C = M^e \bmod n$$

$$M = C^d \bmod n = M^{ed} \bmod n$$

RSA公钥加密算法

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

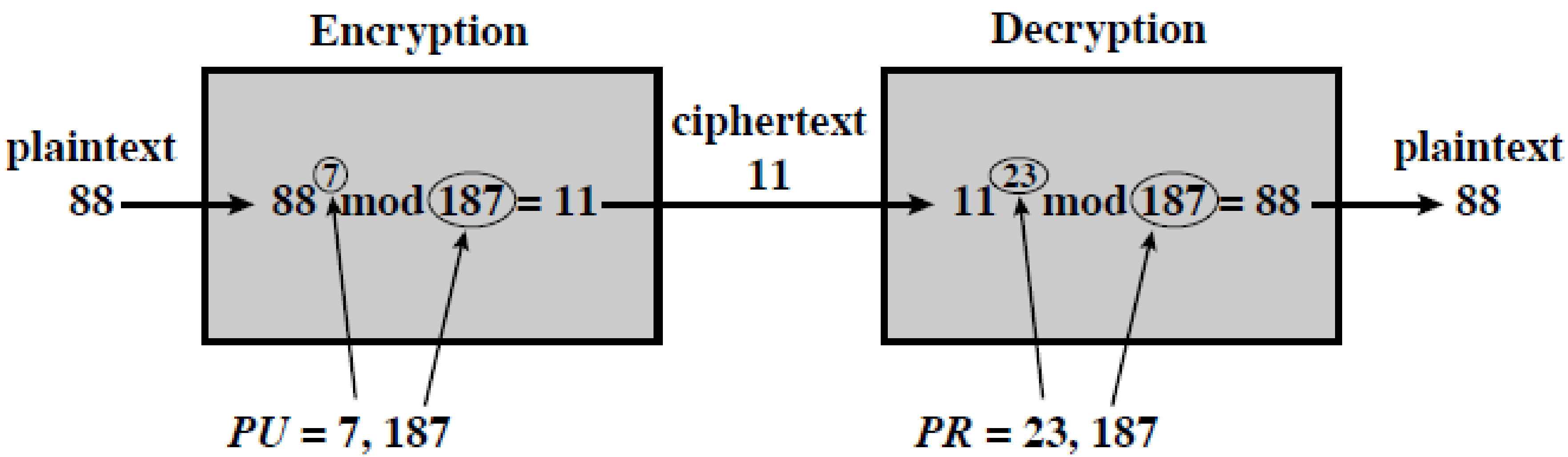
RSA公钥加密算法

Encryption	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption	
Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

RSA公钥加密算法

- RSA算法实例



5．1 公钥密码学简介

- 单向函数：一个函数容易计算但难于求逆

- 陷门(Trapdoor)

  包含容易求出*逆函数*的**秘密**信息

- 陷门单向函数(Trapdoor one-way function)

  单向函数&具有陷门容易求逆

5．1 公钥密码学简介

- RSA 中的陷门？

$$f(x) = x^b \bmod n$$

5．2 更多的数论知识

- 辗转相除法(Euclidean Algorithm)

If a>b, and a=b*q+r (r: remainder)

(a,b)=(b,r)

5．2 更多的数论知识

- 辗转相除法(Euclidean Algorithm)

算法 Euclidean Algorithm(a, b)

$r_0 \leftarrow a$

$r_1 \leftarrow b$

$m \leftarrow 1$

while $r_m \neq 0$

$$\textbf{do}\begin{cases} q_m \leftarrow \left\lfloor \dfrac{r_{m-1}}{r_m} \right\rfloor \\ r_{m+1} \leftarrow r_{m-1} - q_m r_m \\ m \leftarrow m+1 \end{cases}$$

$m \leftarrow m-1$

return$(q_1, \cdots, q_m; r_m)$

comment: $r_m = \gcd(a, b)$

5．2 更多的数论知识

扩展Euclidean算法

- Input a, b
- Output: r=gcd(a,b)

  & sa+tb=r

Algorithm EXTENDED EUCLIDEAN ALGORITHM(a,b)

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$s_0 \leftarrow 1$
$s \leftarrow 0$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$

$$\textbf{do} \begin{cases} temp \leftarrow t_0 - qt \\ t_0 \leftarrow t \\ t \leftarrow temp \\ temp \leftarrow s_0 - qs \\ s_0 \leftarrow s \\ s \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor \\ r \leftarrow a_0 - qb_0 \end{cases}$$

$r \leftarrow b_0$
return (r,s,t)
comment: $r = \gcd(a,b)$ and $sa + tb = r$

5．2 更多的数论知识

- Multiplicative Inverse(a,b) $b^{-1} mod\ a$

Algorithm MULTIPLICATIVE INVERSE(a, b)

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$

$$
\textbf{do} \begin{cases} temp \leftarrow (t_0 - qt) \bmod a \\ t_0 \leftarrow t \\ t \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor \\ r \leftarrow a_0 - qb_0 \end{cases}
$$

if $b_0 \neq 1$
  then b has no inverse modulo a
  else return (t)

5．2 更多的数论知识

中国剩余定理

- 韩信点兵

兵不满一万，每5人一列、9人一列、13人一列、17人一列都剩3人，则兵有多少?

定理 假定 $m_1,\cdots,m_r$ 为两两互素的正整数，又假定 $a_1,\cdots,a_r$ 为整数。那么同余方程组 $x \equiv a_i (\bmod m_i)(1 \leqslant i \leqslant r)$ 有模 $M = m_1 \times \cdots \times m_r$ 的唯一解，此解由下式给出：

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

其中 $M_i = M / m_i$，且 $y_i = M_i^{-1} \bmod m_i$，$1 \leqslant i \leqslant r$。

5．2 更多的数论知识

- The **order** of *G* is the number of elements in G.

- The **order** of *an element* $g \in G$ is defined to be the smallest positive integer m such that $g^m = 1$.

5．2 更多的数论知识

(Fermat) *Suppose p is prime and $b \in \mathbb{Z}_p$. Then $b^p \equiv b \pmod{p}$.*

Proof：

5．2 更多的数论知识

THEOREM 6.4 (Lagrange) *Suppose G is a multiplicative group of order n, and $g \in G$. Then the order of g divides n.*

COROLLARY *If $b \in \mathbb{Z}_n{}^*$, then $b^{\phi(n)} \equiv 1 \pmod{n}$.*

5．2 更多的数论知识

一个元素α具有模p的阶等于$p-1$，称为一个模p的本原元素。

$\beta=\alpha^i$的阶为多少？

$$\frac{p-1}{\gcd(p-1,i)}$$

5. 3 RSA密码体制

密码体制 RSA 密码体制

设 $n=pq$，其中 p 和 q 为素数。设 $\mathcal{P}=\mathcal{C}=\mathbb{Z}_n$，且定义

$$\mathcal{K}=\{(n,p,q,a,b):ab\equiv 1(\bmod \phi(n))\}$$

对于 $K=(n,p,q,a,b)$，定义

$$e_K(x)=x^b \bmod n$$

和

$$d_K(y)=y^a \bmod n$$

$(x,y\in\mathbb{Z}_n)$。值 n 和 b 组成了公钥，且值 p,q 和 a 组成了私钥。

5. 3 RSA密码体制

- RSA实现

基本运算的复杂度

x 和 y 分别是 k 位和 ℓ 位二进制表示的正整数；即 $k=\lfloor \mathrm{lb}x \rfloor+1$，$\ell=\lfloor \mathrm{lb}y \rfloor+1$。 $k \geqslant \ell$

- 计算 $x+y$ 的时间复杂度为 $O(k)$
- 计算 $x-y$ 的时间复杂度为 $O(k)$
- 计算 xy 的时间复杂度为 $O(k\ell)$
- 计算 $\lfloor x/y \rfloor$ 的时间复杂度为 $O(l(k-l))$； $O(k\ell)$ 是一个弱估计
- 计算 $\gcd(x,y)$ 的时间复杂度为 $O(k^3)$

Bit operations

5. 3 RSA密码体制

- RSA实现

Z_n中基础运算复杂度　n 为一个 k 比特整数

- 计算 $(m_1+m_2) \bmod n$ 的时间复杂度为 $O(k)$
- 计算 $(m_1-m_2) \bmod n$ 的时间复杂度为 $O(k)$
- 计算 $(m_1 m_2) \bmod n$ 的时间复杂度为 $O(k^2)$
- 计算 $(m_1)^{-1} \bmod n$ 的时间复杂度为 $O(k^3)$
- 计算 $(m_1)^c \bmod n$ 的时间复杂度为 $O((\log c)\times k^2)$

5. 3 RSA密码体制

- $x^c \bmod n$

$$c=\sum_{i=0}^{\ell-1} c_i 2^i \qquad c_i=0\text{或}1，\ 0\leqslant i\leqslant \ell-1。$$

算法 Square-and-Multiply(x, c, n)

$z \leftarrow 1$

for $i \leftarrow \ell-1$ **downto** 0

$$\textbf{do}\begin{cases} z \leftarrow z^2 \bmod n \\ \textbf{if}\quad c_i = 1 \\ \quad\textbf{then}\quad z \leftarrow (z \times x) \bmod n \end{cases}$$

return(z)

$\ell \leq \text{模乘次数} \leq 2\ell$

5.4 素性检测

素数、质数

除了1和该数自身外，无法被其他自然数整除的数

- 定理：存在无限多个素数

- **素数在密码学中有着广泛的应用**

结构化的数学；安全信任的基础

素数定理

素数计算函数п(n)：不大于n的素数之数量

素数定理表示，п(n)的可由下列公式近似给出：

$$\pi(n) \approx \frac{n}{\ln n}$$

素性测试算法

- 如何判定素数p？

(RSA1024：需要512位的素数p, q)

直接试除法 O ($\sqrt{p}$)

非多项式时间算法

素性测试算法-AKS

Agrawal, Manindra; Kayal, Neeraj; Saxena, Nitin (2004). **"PRIMES is in P"**

1. Check if n is a perfect power: if $n = a^b$ for integers $a > 1$ and $b > 1$, output *composite*.
2. Find the smallest r such that $\mathrm{ord}_r(n) > (\log_2 n)^2$. (if r and n are not coprime, then skip this r)
3. For all $2 \le a \le \min(r, n-1)$, check that a does not divide n: If $a|n$ for some $2 \le a \le \min(r, n-1)$, output *composite*.
4. If $n \le r$, output *prime*.
5. For $a = 1$ to $\left\lfloor \sqrt{\varphi(r)} \log_2(n) \right\rfloor$ do

   if $(X+a)^n \neq X^n + a \pmod{X^r - 1, n}$, output *composite*;
6. Output *prime*.

素性测试算法

- 为什么要用概率算法？
- Miller-Rabin算法理论依据？
- 会不会有隐患？

素性测试算法

- 费尔马小定理:

如果b是一个正整数，p是一个素数，并且gcd(b,p)=1,那么 $b^{p-1} \equiv 1 \pmod{p}$

素性测试算法

我们说n是一个基为b的probable prime 如果 $b^{n-1} \equiv 1 \pmod{n}$

如果n是一个合数，那么称其为基为b的pseudoprime.

例子：341=11×13，

但是$2^{341-1} \equiv 1\ (mod\ 341)$

素性测试算法

- Base-2 Fermat pseudoprimality test (Chinese test)

[1] Initialize the values $i \geq 3$ and $j > i$. Set $n \leftarrow i$.

[2] If $2^n \pmod{n} = 2$, then n is a *base-2 probable prime*, else n is composite.

[3] $n \leftarrow n + 1$. If $n \leq j$ goto [2], else goto [4].

[4] Terminate the execution of the algorithm.

素性测试算法-Strong Pseudoprimality test

- **定理：** 设p是一个**素数**。那么 $x^2 \equiv 1 \pmod p$ 当仅当 $x \equiv \pm 1 \pmod p$.

性质：如果存在非$\pm 1 \pmod n$ 的1的平方根$\pmod n$，那么n是合数。

素性测试算法-Strong Pseudoprimality test

- 设$n = 1 + 2^j d$ 为素数,其中d为奇数，那么 ***b-序列***

$$\{b^d,\ b^{2d},\ b^{4d},\ b^{8d},\ \cdots,\ b^{2^{j-1}d},\ b^{2^j d}\} \mod n$$

具有如下的形式：

$$(1,\ 1,\ \cdots,\ 1,\quad 1,\ 1,\ \cdots,\ 1),$$
$$(?,\ ?,\ \cdots,\ ?,\ -1,\ 1,\ \cdots,\ 1),$$

- 反过来，如果呈现

$$(?,\ \cdots,\ ?,\ 1,\ 1,\ \cdots,\ 1),$$
$$(?,\ \cdots,\ ?,\ ?,\ ?,\ \cdots,\ -1),$$
$$(?,\ \cdots,\ ?,\ ?,\ ?,\ \cdots,\ ?),$$

那么n肯定是一个合数。

素性测试算法-Strong Pseudoprimality test

- Miller-Rabin测试

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 $1 \leqslant a \leqslant n-1$

$b \leftarrow a^m \bmod n$

if $b \equiv 1 \pmod n$

　then return ("n is prime")

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

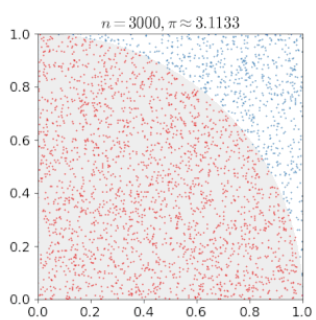
do $\begin{cases} \textbf{if } b \equiv -1 \pmod n \\ \quad \textbf{then return}(\text{“}n \text{ is prime”}) \\ \quad \textbf{else } b \leftarrow b^2 \bmod n \end{cases}$

return("n is composite")

*Monte Carlo 算法

Monte Carlo method

- 也称*统计模拟*方法，是1940年代中期由于科学技术的发展和电子计算机的发明，而提出的一种以概率统计理论为指导的数值计算方法。是指使用随机数(或更常见的伪随机数)来解决很多计算问题的方法



使用蒙特卡罗方法估算π值. 放置30000个随机点后,π的估算值与真实值相差0.07%.

Monte Carlo 算法

- 判定问题偏是（yes-biased）

  是 *回答总是正确的*

- 一个偏是的MonteCarlo算法具有错误概率ϵ， 如果“是”的实例 至多以ϵ的概率给出一个不正确的回答‘否’

- 判定问题偏否（no-biased）

素性测试算法

定理 5.11 Miller-Rabin 算法对于合数问题是一个偏是的 Monte Carlo 算法。

Miller-Rabin测试

Miller-Rabin算法是一个合数问题偏是的**Monte Carlo算法**。

- 如果Miller-Rabin算法指出n是合数，那么n一定是合数。

  反证法…

Miller-Rabin测试

Theorem: 如果n是一个奇合数, 那么至多有(n-1)/4的基数b ($1 \le b < n$)会通过Miller-Rabin测试。

Proof.

Miller-Rabin测试

- 大致估计 (k: number of rounds)

k	$1/4^k$
10	$< 10^{-6}$
25	$< 10^{-15}$
30	$< 10^{-18}$
50	$< 10^{-30}$
100	$< 10^{-60}$

素性测试算法

- 为什么要用概率算法？ -O((logp)^3)
- Miller-Rabin算法理论依据-每轮通过几率<1/4
- 会不会有隐患？ -出错率低于机器出错概率

5.5 模n的平方根

- Legendre记号
- 求解方程$y^2 = a(mod\ p)$
- 二次剩余(quadratic residue)
- 二次非剩余(quadratic non-residue)

5.5 模n的平方根

判别问题

- 给定一个奇素数p和一个整数a, a是一个模p二次剩余吗？

- 应用领域：椭圆曲线公钥体制……

5.5 模n的平方根

定理 设 p 为一个奇素数，a 为一个正整数。那么 a 是一个模 p 二次剩余，当且仅当

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

Proof：

5.5 模n的平方根

问题 5.2 Quadratic Residues

实例：一个奇素数 p 和一个整数 a。

问题：a 是一个模 p 二次剩余吗？

- 计算复杂度？

$$O((\log p)^3)$$

5.5 模n的平方根

- Legendre记号

假定 p 是一个奇素数。对任何整数 a，定义 Legendre 符号 $\left(\frac{a}{p}\right)$ 如下：

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

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

5.5 模n的平方根

- Jacobi记号

定义 假定 n 是一个奇正整数，且 n 的素数幂因子分解为

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

设 a 为一个整数。那么 Jacobi 符号 $\left(\frac{a}{n}\right)$ 定义为：

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

5.5 模n的平方根

- Jacobi记号

- 如果n是一个合数且满足$\left(\frac{a}{n}\right)=1$

是否a是一个平方剩余数（在Z_n上）？

5.5 模n的平方根

- 二次互反律（ **law of quadratic reciprocity** ）

如果兩個數都是正奇數，那麼二次互反律對雅可比符號也成立：

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{(m-1)(n-1)/4}.$$

5.5 模n的平方根

- 如果n是素数，那么$y^2 = a(mod\ n)$要么有两个解（$\left(\frac{a}{n}\right) = 1$），要么没有解（$\left(\frac{a}{n}\right) = -1$）

一般的：

定理 假定 p 为一个奇素数，e 为一个正整数，且 $\gcd(a, p) = 1$。那么同余方程 $y^2 \equiv a(\bmod\ p^e)$ 当 $\left(\frac{a}{p}\right) = -1$ 时没有解，当 $\left(\frac{a}{p}\right) = 1$ 时有两个解（模 p^e）。

Pf.

5.5 模n的平方根

定理 假定 $n>1$ 是一个奇数，且有如下分解

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

其中 p_i 为不同的素数，且 e_i 为正整数。进一步假定 $\gcd(a,n)=1$。那么同余方程 $y^2\equiv a \pmod n$ 当 $\left(\dfrac{a}{p_i}\right)=1$ 对于所有的 $i\in\{1,\cdots,\ell\}$ 成立时有 2^{ℓ} 个模 n 的解，其他情形下没有解。

Pf.

Solovay-Strassen算法

定理 假定p是一个奇素数，那么

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

算法 Solovay-Strassen (n)

随机选取整数a，使得$1 \leqslant a \leqslant n-1$

$x \leftarrow \left(\frac{a}{n}\right)$

if $x = 0$

then return ("n is composite")

$y \leftarrow a^{(n-1)/2} \pmod{n}$

if $x \equiv y \pmod{n}$

then return ("n is prime")

else return ("n is composite")

Solovay-Strassen算法

- 时间复杂度？

$$O((\log n)^3)$$

Solovay-Strassen算法

Theorem：Solovay-Strassen算法是一个偏是的Monte Carlo算法， 并至多具有1/2的错误概率.

Proof.

Solovay-Strassen算法

- 算法运行了m次，n是一个素数的概率？
- 直观的估计 $1-2^{-m}$

Solovay-Strassen算法

- 严格的估计：校正为 $\frac{\ln n-2}{\ln n-2+2^{m+1}}$ $O(\frac{1}{2^m})$

m	2^{-m}	错误概率的界
1	0.500	0.989
2	0.250	0.978
5	0.312×10^{-1}	0.847
10	0.977×10^{-3}	0.147
20	0.954×10^{-6}	0.168×10^{-3}
30	0.931×10^{-9}	0.164×10^{-6}
50	0.888×10^{-15}	0.157×10^{-12}
100	0.789×10^{-30}	0.139×10^{-27}

Solovay-Strassen 算法的错误概率

5.6 分解因子算法

- Pollard p-1算法 （优雅而简洁）

Algorithm POLLARD $p-1$ FACTORING ALGORITHM(n, B)

$a \leftarrow 2$
for $j \leftarrow 2$ **to** B
  do $a \leftarrow a^j \bmod n$
$d \leftarrow \gcd(a-1, n)$
if $1 < d < n$
  then return (d)
  else return ("failure")

5.6 分解因子算法

- Pollard p-1算法-Explanation:

每一个素数幂 $q \mid (p-1)$， 有 $q \leqslant B$ 。

$$(p-1) \mid B!$$

我们有

$$a \equiv 2^{B!} \pmod{n}$$

$$a \equiv 2^{B!} \pmod{p}$$

于是有

$$a \equiv 1 \pmod{p}$$

$d = \gcd(a-1, n)$ 是 n 的一个非平凡因子(除非 $a=1$)

5.6 分解因子算法

- 时间复杂度分析？

$$O(B\log B(\log n)^2+(\log n)^3)$$

- 缺陷：*B small*

5.6 分解因子算法

- Example

例 5.9 假定 $n=15\ 770\ 708\ 441$。如果选取 $B=180$ 应用算法 5.8，可以发现 $a=11\ 620\ 221\ 425$，d 计算的结果为 135 979。事实上，n 的完全素因子分解为

$$15\ 770\ 708\ 441=135\ 979\times 115\ 979$$

在这个例中，分解成功是因为 135 978 仅有“小”的素因子：

$$135\ 978=2\times 3\times 131\times 173$$

因此，通过选取 $B\geqslant 173$，会有 $135\ 978 \mid B!$，正是我们所需要的。

5.7 对RSA的其他攻击

$$n = pq$$

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

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

求解$\phi(n)$~分解n

5.7 对RSA的其他攻击

Las Vegas型随机算法

- 不一定给出答案的随机算法：可能失败而终止；但一旦返回一个答案，就是正确的

一个对于给定的值 a, b 和 n 作为输入，以至少 1/2 的概率分解 n 的 Las Vegas 算法。

如果算法运行 m 次，那么 n 被分解的概率至少为 $1-1/2^m$

5.7 对RSA的其他攻击

- 可以利用模n=pq 的*非平凡平方根*x来分解n

- Explanation:
- By gcd(x+1,n), gcd(x-1,n)

5.7 对RSA的其他攻击

算法 RSA-FACTOR(n, a, b)

Comment: 假定 $ab \equiv 1(\bmod \phi(n))$

记 $ab-1=2^s r$， r 为奇数

随机选择 w 使得 $1 \leqslant w \leqslant n-1$

$x \leftarrow \gcd(w, n)$

if $1 < x < n$

  then return (x)

Comment: x 是 n 的一个因子

$v \leftarrow w^r \bmod n$

if $v \equiv 1(\bmod n)$

  then return ("failure")

while $v \not\equiv 1(\bmod n)$

  do $\begin{cases} v_0 \leftarrow v \\ v \leftarrow v^2 \bmod n \end{cases}$

if $v_0 \equiv -1(\bmod n)$

  then **return** ("failure")

  else $\begin{cases} x \leftarrow \gcd(v_0+1, n) \\ \text{return}(x) \end{cases}$

Comment: x 是 n 的一个因子

可以证明：该算法的成功概率至少为1/2

5.7 对RSA的其他攻击

- 2009年12月12日，编号为RSA-768（768 bits, 232 digits）数也被成功分解
- 普遍认为用户应尽快升级到2048-bit或以上
- NIST建议的RSA密钥长度为至少2048位

5.8 Rabin密码体制

- 可证明安全

密码体制 Rabin 密码体制

设 $n=pq$，其中 p 和 q 为素数，且 $p,q\equiv 3(\bmod 4)$。设 $\mathcal{P}=\mathcal{C}=\mathbb{Z}_n^*$，且定义

$$\mathcal{K}=\{(n,p,q)\}$$

对 $K=(n,p,q)$，定义

$$e_K(x)=x^2 \bmod n$$

和

$$d_K(y)=\sqrt{y} \bmod n$$

n 为公钥，p 和 q 为私钥。

5.8 Rabin密码体制

$$x^2 \equiv y \pmod{n}$$

$$\Updownarrow$$

$$z^2 \equiv y \pmod{p} \qquad z^2 \equiv y \pmod{q}$$

$$\begin{aligned}(\pm y^{(p+1)/4})^2 &\equiv y^{(p+1)/2} \pmod{p} \\ &\equiv y^{(p-1)/2} y \pmod{p} \\ &\equiv y \pmod{p}\end{aligned} \quad \Longrightarrow \quad y \text{ 模 } p \text{ 的两个平方根为 } \pm y^{(p+1)/4} \bmod p$$

- 根据CRT求4个平方根

5.8 Rabin密码体制

可证明安全性

- 规约(Turing reduction)

Definition Suppose that **G** and **H** are problems. A ***Turing reduction*** from **G** to **H** is an algorithm SOLVEG with the following properties:

1. SOLVEG assumes the existence of an arbitrary algorithm SOLVEH that solves the problem **H**.
2. SOLVEG can call the algorithm SOLVEH and make use of any values it outputs, but SOLVEG cannot make any assumption about the actual computations performed by SOLVEH (in other words, SOLVEH is an oracle that is treated as a "black box").
3. SOLVEG is a polynomial-time algorithm, when each call to the oracle is regarded as taking $O(1)$ time. (Note that the complexity of SOLVEG takes into account all the computations that are done "outside" the oracle.)
4. SOLVEG correctly solves the problem **G**.

If there is a Turing reduction from **G** to **H**, we denote this by writing $\mathbf{G} \propto_T \mathbf{H}$.

5.8 Rabin密码体制

Algorithm RABIN ORACLE FACTORING(n)

external RABIN DECRYPT
choose a random integer $r \in \mathbb{Z}_n{}^*$
$y \leftarrow r^2 \bmod n$
$x \leftarrow$ RABIN DECRYPT(y)
if $x \equiv \pm r \pmod{n}$
  then return ("failure")

$$\textbf{else} \begin{cases} p \leftarrow \gcd(x+r, n) \\ q \leftarrow n/p \\ \textbf{return} \ (\text{“}n = p \times q\text{”}) \end{cases}$$

5.8 Rabin密码体制

- 我们可以证明： （成功概率为1/2）

$$\textbf{Factoring} \propto_T \textbf{Rabin decryption},$$

Integer Factorization：The problem is clearly in class NP, but it is generally suspected that it is not *NP-complete*, though this has not been proven.

5.9 RSA的语义安全性

- 构造一个公钥密码体制使得敌手不能（在多项式时间内）识别密文—语义安全
- 信息泄漏--RSA

$$y = x^b \bmod n$$

$\gcd(b, \phi(n)) = 1$，必然是 b 为奇数

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

因此任何人可以计算$(\frac{x}{n})$

5.9 RSA的语义安全性

- 其他泄露：

1. 给定 $y = e_K(x)$，计算 parity(y)，其中 parity(y) 表示 x 的二进制表示的最低位数[即当 x 为偶数时 parity(y) = 0；x 为奇数时 parity(y) = 1]。
2. 给定 $y = e_K(x)$，计算 half(y)，其中当 $0 \leqslant x < n/2$ 时 half(y) = 0；当 $n/2 < x \leqslant n-1$ 时 half(y) = 1。

RSA Decryption $\propto_T$ Half

5.9 RSA的语义安全性

Algorithm ORACLE RSA DECRYPTION(n, b, y)

计算复杂度？

external HALF

$k \leftarrow \lfloor \log_2 n \rfloor$

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

do $\begin{cases} h_i \leftarrow \text{HALF}(n, b, y) \\ y \leftarrow (y \times 2^b) \bmod n \end{cases}$

$lo \leftarrow 0$

$hi \leftarrow n$

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

do $\begin{cases} mid \leftarrow (hi + lo)/2 \\ \textbf{if } h_i = 1 \\ \quad \textbf{then } lo \leftarrow mid \\ \quad \textbf{else } hi \leftarrow mid \end{cases}$

return $(\lfloor hi \rfloor)$

$O((\log n)^3) + O(\log n) \times \text{half}$ 的复杂度

5.9 RSA的语义安全性

乘法同态性 $e_K(x_1)e_K(x_2) = e_K(x_1 x_2)$

$$h_i = \mathrm{half}(y \times (e_K(2))^i) = \mathrm{half}(e_K(x \times 2^i))$$

$$\mathrm{half}(e_K(x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{2}\right)$$

$$\mathrm{half}(e_K(2x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{4}\right) \cup \left[\frac{n}{2}, \frac{3n}{4}\right)$$

$$\mathrm{half}(e_K(4x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{8}\right) \cup \left[\frac{n}{4}, \frac{3n}{8}\right) \cup \left[\frac{n}{2}, \frac{5n}{8}\right) \cup \left[\frac{3n}{4}, \frac{7n}{8}\right)$$

5.9 RSA的语义安全性

- 可以利用**二分查找**来找到x

假定 $n=1457, b=779$，且我们有一个密文 $y=722$。然后假定，利用谕示器 half，得到下面的 h_i 值:

i	0	1	2	3	4	5	6	7	8	9	10
h_i	1	0	1	0	1	1	1	1	1	0	0

i	lo	mid	hi
0	0.00	728.50	1457.00
1	728.50	1092.75	1457.00
2	728.50	910.62	1092.75
3	910.62	1001.69	1092.75
4	910.62	956.16	1001.69
5	956.16	978.92	1001.69
6	978.92	990.30	1001.69
7	990.30	996.00	1001.69
8	996.00	998.84	1001.69
9	998.84	1000.26	1001.69
10	998.84	999.55	1000.26
	998.84	999.55	999.55

5.9 RSA的语义安全性

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

- 计算 parity(y) $\equiv_p$ 计算half(y)

可以相信：计算parity(y)、计算half(y)是困难的