初等数论
第二章 同余

中山大学 计算机学院

平方乘(Square-and-Multiply)算法

- 实现RSA密码的关键一步是实现模n的指数运算。
- 模指数运算通常采用Square-and-Multiply平方乘算法。

算法 $(\mathrm{Square-and-Multiply}(x, c, n))$

$z \leftarrow 1$
for $i = l - 1$ **downto** 0
  $z \leftarrow z^2 \bmod n$
  if $c_i = 1$ **then** $z \leftarrow z \cdot x \bmod n$
endfor
return z

- 算法中$c_i = 0, 1$是c的二进制表示系数，即

$$c = \sum_{i=0}^{l-1} c_i 2^i,$$

而l是整数c的二进制长度。

平方乘(Square-and-Multiply)算法举例

例 (计算$9726^{3533} \pmod{11413}$)

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	$11025^2 \times 9726 = 5761$

由此可得$9726^{3533} \pmod{11413} = 5761$。

理解平方乘(Square-and-Multiply)算法

- 在平方乘算法中，我们考虑每一次循环完成之后，z的值，然后可以发现，在完成l次循环之后，z的值等于$x^c \bmod n$。

理解平方乘(Square-and-Multiply)算法

- 在平方乘算法中，我们考虑每一次循环完成之后，z的值，然后可以发现，在完成l次循环之后，z的值等于$x^c \bmod n$。

1. $i = l - 1$，第1次循环，$z = x^{c_{l-1}} \bmod n$；

理解平方乘(Square-and-Multiply)算法

- 在平方乘算法中，我们考虑每一次循环完成之后，z的值，然后可以发现，在完成l次循环之后，z的值等于$x^c \bmod n$。

1. $i = l-1$，第1次循环，$z = x^{c_{l-1}} \bmod n$;
2. $i = l-2$，第2次循环，$z = (x^{c_{l-1}})^2 x^{c_{l-2}} = x^{c_{l-1}2+c_{l-2}} \bmod n$;

- 在平方乘算法中，我们考虑每一次循环完成之后，z的值，然后可以发现，在完成l次循环之后，z的值等于$x^c \bmod n$。

1. $i = l-1$，第1次循环，$z = x^{c_{l-1}} \bmod n$;
2. $i = l-2$，第2次循环，$z = (x^{c_{l-1}})^2 x^{c_{l-2}} = x^{c_{l-1}2+c_{l-2}} \bmod n$;
3. $i = l-3$，第3次循环，
   $z = (x^{2c_{l-1}+c_2})^2 x^{c_{l-3}} = x^{c_{l-1}2^2+c_{l-2}2+c_{l-3}} \bmod n$;

理解平方乘(Square-and-Multiply)算法

- 在平方乘算法中，我们考虑每一次循环完成之后，z的值，然后可以发现，在完成l次循环之后，z的值等于$x^c \bmod n$。

1. $i = l-1$，第1次循环，$z = x^{c_{l-1}} \bmod n$；
2. $i = l-2$，第2次循环，$z = (x^{c_{l-1}})^2 x^{c_{l-2}} = x^{c_{l-1}2+c_{l-2}} \bmod n$；
3. $i = l-3$，第3次循环，
   $z = (x^{2c_{l-1}+c_2})^2 x^{c_{l-3}} = x^{c_{l-1}2^2+c_{l-2}2+c_{l-3}} \bmod n$；
4. $\vdots$
5. $i = k$，第$l-k$次循环，$z = x^{c_{l-1}2^{l-k-1}+c_{l-2}2^{l-k-2}+\cdots+c_{l-k}} \bmod n$；

理解平方乘(Square-and-Multiply)算法

- 在平方乘算法中，我们考虑每一次循环完成之后，z的值，然后可以发现，在完成l次循环之后，z的值等于$x^c \bmod n$。

1. $i = l-1$，第1次循环，$z = x^{c_{l-1}} \bmod n$;
2. $i = l-2$，第2次循环，$z = (x^{c_{l-1}})^2 x^{c_{l-2}} = x^{c_{l-1}2+c_{l-2}} \bmod n$;
3. $i = l-3$，第3次循环，
   $z = (x^{2c_{l-1}+c_2})^2 x^{c_{l-3}} = x^{c_{l-1}2^2+c_{l-2}2+c_{l-3}} \bmod n$;
4. $\vdots$
5. $i = k$，第$l-k$次循环，$z = x^{c_{l-1}2^{l-k-1}+c_{l-2}2^{l-k-2}+\cdots+c_{l-k}} \bmod n$;
6. $\vdots$
7. $i = 0$，第l次循环，$z = x^{c_{l-1}2^{l-1}+c_{l-2}2^{l-2}+\cdots+c_0} \bmod n$。

理解平方乘(Square-and-Multiply)算法(续)

- 平方乘算法可以结合下面的等式去理解，它从层层嵌套的括号最里面一步步地计算到括号的最外面，每跳出一层括号计算一次平方。

$$
\begin{aligned}
&x^{c_{l-1}\cdot 2^{l-1}+c_{l-1}\cdot 2^{l-2}+\cdots+c_1\cdot 2+c_0}\\
&\quad = x^{2(c_{l-1}\cdot 2^{l-2}+c_{l-1}\cdot 2^{l-3}+\cdots+c_2\cdot 2+c_1)+c_0}\\
&\quad = x^{2(2(c_{l-1}\cdot 2^{l-3}+c_{l-1}\cdot 2^{l-4}+\cdots+c_3\cdot 2+c_2)+c_1)+c_0}\\
&\quad = x^{2(2(2(c_{l-1}\cdot 2^{l-4}+c_{l-1}\cdot 2^{l-5}+\cdots+c_4\cdot 2+c_3)+c_2)+c_1)+c_0}\\
&\quad = x^{2(2(2(2(c_{l-1}\cdot 2^{l-5}+c_{l-1}\cdot 2^{l-6}+\cdots+c_5\cdot 2+c_4)+c_3)+c_2)+c_1)+c_0}\\
&\quad \vdots\\
&\quad = x^{2(2(2(2\cdots(2(c_{l-1})+c_{l-2})+c_{l-3})+\cdots+c_5)+c_4)+c_3)+c_2)+c_1)+c_0}
\end{aligned}
$$

5. 第二章小结

1. 模m同余相等与整数相等的相似性.
2. $\left.\begin{array}{c} ad \equiv bd \bmod m \\ (d,m)=1 \end{array}\right\} \Longrightarrow a \equiv b \bmod m.$
3. $\left.\begin{array}{c} a \equiv b \bmod m_1 \\ a \equiv b \bmod m_2 \end{array}\right\} \Longrightarrow a \equiv b \bmod [m_1, m_2]$
4. 完全(简化)剩余系的写法.
5. 整数a与正整数m互素，则当x取遍模m的简化(完全)剩余系，相应的数ax也构成模m的简化(完全)剩余系.
6. 设m_1与m_2互素，如果x_1取遍模m_1的简化(完全)剩余系，x_2取遍模m_2的简化(完全)剩余系，则$m_2x_1 + m_1x_2$取遍模m_1m_2简化(完全)剩余系.
7. Wilson定理及其证明思想.
8. $(m,n)=1 \Longrightarrow \varphi(mn)=\varphi(m)\varphi(n)$，且$\varphi(p^\alpha)=p^\alpha - p^{\alpha-1}$.
9. Euler定理：如果m是正整数，且整数a与m互素，则$a^{\varphi(m)} \equiv 1 \bmod m$.
10. Fermat小定理：如果p是素数，a是整数，则$a^p \equiv a \bmod p$.
11. 平方乘算法（模重复平方计算法）.