

第9章 公钥密码学与RSA

1 什么是公钥密码体制

- 公钥密码又称为双钥密码和非对称密码，是1976年由Diffie和Hellman在其“密码学新方向”一文中提出的，见划时代的文献：

W.Diffie and M.E.Hellman, New Directrions in Cryptography, IEEE
Transaction on Information Theory, V.IT-22.No.6, Nov 1976, PP.644-654

单向陷门函数是满足下列条件的函数 f ：

- (1)给定 x ，计算 $y=f(x)$ 是容易的；
 - (2)给定 y ，计算 x 使 $y=f(x)$ 是困难的。
- (所谓计算 $x=f^{-1}(Y)$ 困难是指计算上相当复杂;若经过数年的计算得出正确结果就无实际意义了。)

(3)存在 δ ，使得利用 δ 对给定的函数值 y ，计算 x 使 $y=f(x)$ 是容易的。

注：1*. 仅满足(1)、(2)两条的函数称为单向函数；若对第(3)条也满足,称该函数为带 δ 陷门信息的单向函数,也称为陷门单向函数。

2*. 当用陷门单向函数 f 作为加密函数时，可将 f 公开，这相当于公开加密密钥。此时加密密钥便称为公开钥，记为 P_k 。 f 函数的设计者将 δ 保密，用作解密密钥，此时 δ 称为秘密钥匙，记为 S_k 。由于加密函数是公开的，任何人都可以将信息 x 加密成 $y=f(x)$ ，然后送给函数的设计者（当然可以通过不安全信道传送）；由于设计者拥有 S_k ，他自然可以解出 $x=f^{-1}(y)$ 。

3*. 单向陷门函数的第(2)条性质表明窃听者由截获的密文 $y=f(x)$ 推测 x 是计算上不可行的。

2.Diffie-Hellman密钥交换算法

Diffie和Hellman在其里程碑意义的文章中，虽然给出了密码的思想，但是没有给出真正意义上的公钥密码实例，也既没能找出一个真正带陷门的单向函数。然而，他们给出单向函数的实例，并且基于此提出Diffie-Hellman密钥交换算法。这个算法是基于有限域中计算离散对数的困难性问题之上的：设 F 为有限域，有 $g \in F$ ，使 F 的乘法群 $F^* = F \setminus \{0\} = \langle g \rangle$ 。并且对任意正整数 x ，计算 g^x 是容易的；但是已知 g 和 y 求 x 使 $y = g^x$ ，是计算上几乎不可能的。这个问题被称为有限域 F 上的离散对数问题。公钥密码学中使用最广泛的有限域为素域 F_p 。

对Diffie-Hellman密钥交换协议描述: Alice和Bob协商好一个大素数 p ，和大周期的整数 g ， $1 < g < p$ ， g 最好是 F_p 中的本原元，即 $F_p^* = \langle g \rangle$ 。 p 和 g 可为网络上的所有用户共享。

当Alice和Bob要进行保密通信时，他们可以按如下步骤来做：

(1) Alice选取大的随机数 x ，并且计算

$$X = g^x \pmod{P}$$

(2) Bob选取大的随机数 x' ，并计算 $X' = g^{x'} \pmod{P}$

(3) Alice将 X 传送给Bob；Bob将 X' 传送给Alice。

(4) Alice计算 $K = (X')^x \pmod{P}$; Bob计算 $K' = (X)^{x'} \pmod{P}$, 易见， $K = K' = g^{xx'} \pmod{P}$ 。

由(4)知，Alice和Bob已获得了相同的秘密值 K 。双方以 K 作为加解密钥以传统对称密钥算法进行保密通信。

注：Diffie-Hellman密钥交换算法拥有美国和加拿大的专利。

3 RSA公钥算法

RSA公钥算法是由Rivest,Shamir和Adleman在1978年提出来的（见Communitations of the ACM. Vol.21.No.2. Feb. 1978, PP.120-126)该算法的数学基础是初等数论中的Euler（欧拉)定理，并建立在分解大整数因子的困难性之上。

将 $Z/(n)$ 表示为 Z_n ，其中 $n=pq$; p,q 为素数且相异。若 $Z_n^* \equiv \{g \in Z_n \mid (g,n)=1\}$ ，易见 Z_n^* 为 $\varphi(n)$ 阶的乘法群，且有 $g^{\varphi(n)} \equiv 1 \pmod{n}$ ，而 $\varphi(n)=(p-1)(q-1)$ 。

RSA密码体制描述如下：

首先，明文空间 P =密文空间 $C=Z_n$.加解密过程为：

(1).密钥的生成

选择 p,q 为互异素数，计算 $n=p \times q$, $\varphi(n)=(p-1)(q-1)$, 选整数 e 使 $(\varphi(n),e)=1, 1 < e < \varphi(n)$), 计算 d , 使 $d=e^{-1} \pmod{\varphi(n)}$), 公钥 $P_k=\{e,n\}$; 私钥 $S_k=\{d,p,q\}$ 。

注意, 当 $0 < M < n$ 时, 有:

$M^{k\varphi(n)+1} \equiv M \pmod{n}$, 而 $ed \equiv 1 \pmod{\varphi(n)}$, 易见

$$(M^e)^d \equiv M \pmod{n}$$

(2). 加密 (用 e, n):

明文: $M < n$ 密文: $C = M^e \pmod{n}$.

(3). 解密 (用 d, p, q)

若 密文为 C , 则明文 $M = C^d \pmod{n}$

注: 1*, 加密和解密是一对互逆运算。

2*, 对于 $0 < M < n$ 时, 若 $(M, n) \neq 1$, 则 M 为 p 或 q 的整数倍,
假设 $M = cp$, 由 $(cp, q) = 1$ 有

$$M^{\varphi(q)} \equiv 1 \pmod{q} \implies M^{k\varphi(q)\varphi(p)} \equiv 1 \pmod{q}$$

有 $M^{k\varphi(q)\varphi(p)} = 1 + tq$ 对其两边同乘 $M = cp$ 有

有 $M^{k\varphi(n)+1} = M + tcpq = M + tcn$ 于是

有 $M^{k\varphi(n)+1} \equiv M \pmod{n}$

例子：若Bob选择了 $p=101$ 和 $q=113$ ，那么， $n=11413$ ，
 $\varphi(n)=100 \times 112 = 11200$ ；然而 $11200 = 2^6 \times 5^2 \times 7$ ，一个正整数 e 能用作加密指数，当且仅当 e 不能被 2, 5, 7 所整除（事实上，若Bob不会分解 $\varphi(n)$ ，也可试验一些数和 $\varphi(n)$ 作辗转相除法来求得 e ，使 $(e, \varphi(n))=1$ ）。假设Bob成功选择了 $e=3533$ ，那么继续用辗转相除法可以求得他的解密密钥 $d = e^{-1} \equiv 6597 \pmod{11200}$ 。

Bob在一个目录中公开 $n=11413$ 和 $e=3533$ ，现假设Alice想发送明文 9726 给Bob，她计算密文：

$$9726^{3533} \pmod{11413} = 5761$$

并且在一个信道上发送密文5761。当Bob接收到密文5761时，他用他的秘密解密指数（私钥） $d=6597$ 进行解密： $5761^{6597} \pmod{11413}=9726$

注： RSA的安全性是基于加密函数 $e_k(x)=x^e \pmod{n}$ 是一个单向函数，所以对那些没有陷门信息的人来说,求逆计算不可行。而Bob掌握 $\phi(n)=(p-1)(q-1)$,从而用欧氏算法可以解出解密私钥 d .

4 RSA密码体制的实现

实现的步骤如下： Bob为实现者

(1) Bob寻找出两个大素数 p 和 q

(2) Bob计算出 $n=pq$ 和 $\phi(n)=(p-1)(q-1)$.

(3) Bob选择一个随机数 $e(0 < e < \phi(n))$ ，满足 $(e, \phi(n)) = 1$

(4) Bob使用辗转相除法计算 $d=e^{-1}(\text{mod } \phi(n))$

(5) Bob在目录中公开 n 和 e 作为她的公开钥。

密码分析者攻击RSA体制的关键点在于如何分解 n 。若分解成功使 $n=pq$ ，则可以算出 $\phi(n) = (p-1)(q-1)$ ，然后由公开的 e ，解出秘密的 d 。（猜想：攻破RSA与分解 n 是多项式等价的。然而，这个猜想至今没有给出可信的证明！！！！）

于是要求：若使RSA安全， p 与 q 必为足够大的素数，使分析者没有办法在多项式时间内将 n 分解出来。建议选择 p 和 q 大约是100位的十进制素数。模 n 的长度要求至少是512比特。EDI攻击标准使用的RSA算法中规定 n 的长度为512至1024比特位之间，但必须是128的倍数。国际数字签名标准ISO/IEC 9796中规定 n 的长度位512比特位。

为了抵抗现有的整数分解算法，对RSA模 n 的素因子 p 和 q 还有如下要求：

- (1) $|p-q|$ 很大，通常 p 和 q 的长度相同；
- (2) $p-1$ 和 $q-1$ 分别含有大素因子 p_1 和 q_1
- (3) p_1-1 和 q_1-1 分别含有大素因子 p_2 和 q_2
- (4) $p+1$ 和 $q+1$ 分别含有大素因子 p_3 和 q_3

为了提高加密速度，通常取 e 为特定的小整数，如EDI国际标准中规定 $e=2^{16}+1$ ，ISO/IEC9796中甚至允许取 $e=3$ 。这时加密速度一般比解密速度快10倍以上。

下面研究加解密算术运算，这个运算主要是模 n 的求幂运算。著名的“平方-和-乘法”方法将计算 $x^c(\bmod n)$ 的模乘法的数目缩小到至多为 $2l$ ，这里的 l 是指数 c 的二进制表示比特数。若设 n 以二进制形式表示有 k 比特，即 $k=\lceil \log_2 n \rceil + 1$ 。由 $l \leq k$ ，这样 $x_c(\bmod n)$ 能在 $O(k^3)$ 时间内完成。
(注意，不难看到，乘法能在 $O(k^2)$ 时间内完成。)

平方-和一乘法算法：

指数 c 以二进制形式表示为：

$$c = \sum_{i=0}^{t-1} c_i 2^i = c_0 + c_1 2 + \dots + c_{t-1} 2^{t-1} \quad c_i \in \{0, 1\}$$

$$X^c = x^{c_0} \times (x^2)^{c_1} \times \dots \times (x^{2^{t-1}})^{c_{t-1}}$$

预计算: $x^2 = xx$

$$x^4 = x^{2^2} = x^2 x^2$$

.

.

.

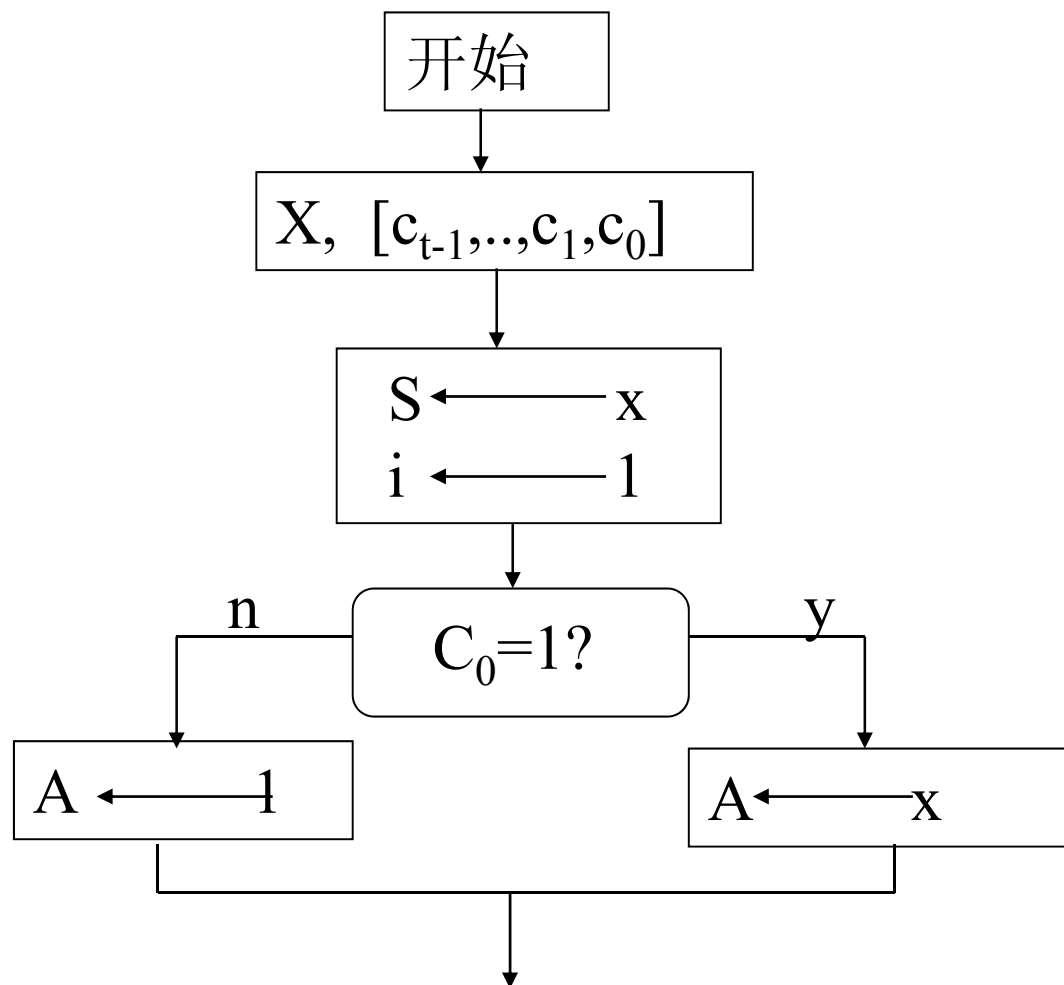
$$x^{2^{t-1}} = x^{2^{t-2}} * x^{2^{t-2}}$$

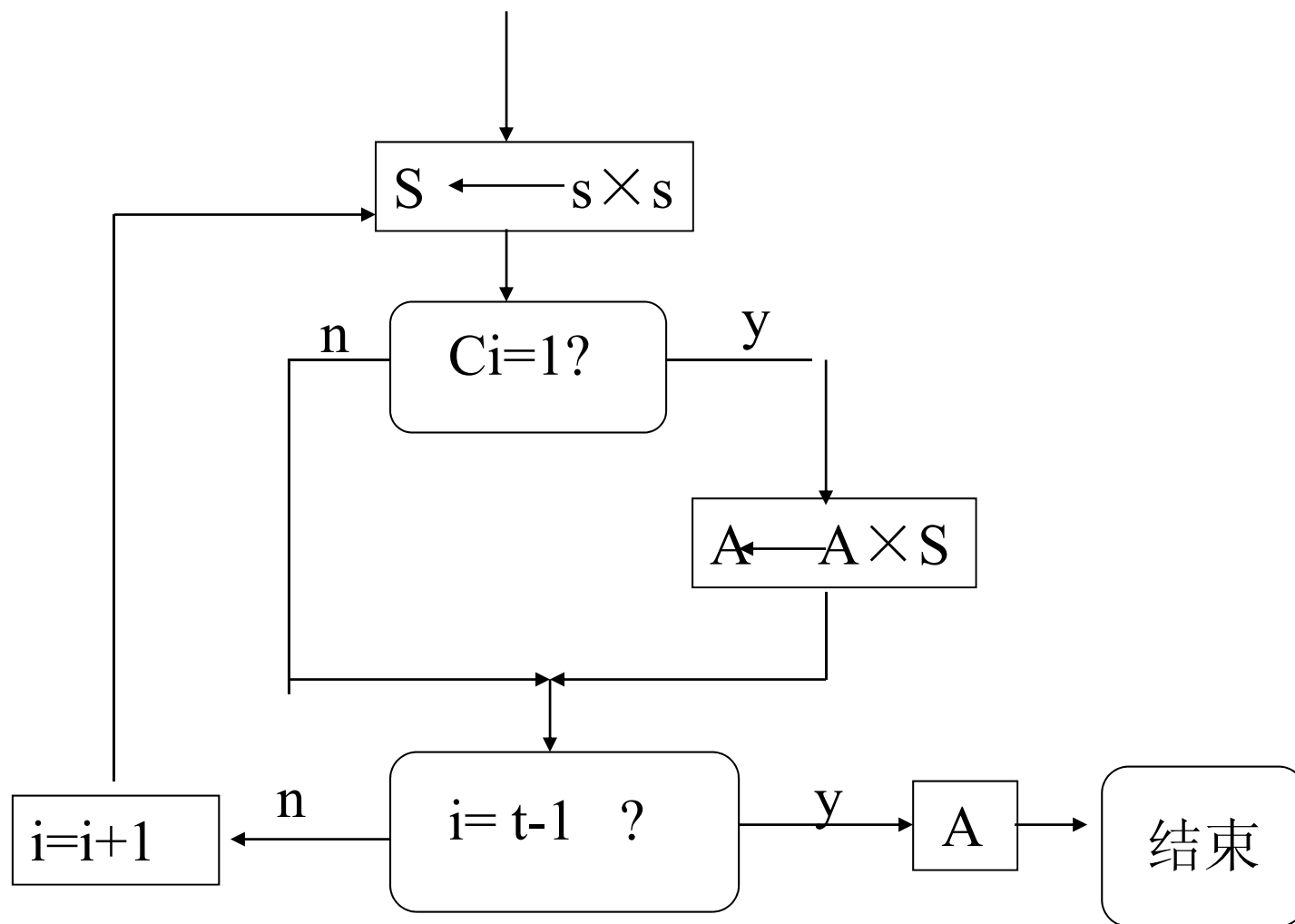
t-1次乘法

x^c 计算: 把那些 $c_i=1$ 对应的 x^{2^i} 全部乘在一起, 便得 x^c 。至多用了t-1次乘法。请参考书上的177页, 给出计算

$x^c \pmod n$ 算法程序:

$$A = x^c \quad c = c_0 + c_1 2 + \dots + c_{t-1} 2^{t-1} = [c_{t-1}, \dots, c_1, c_0]_2$$





5 RSA签名方案

签名的基本概念

传统签名（手写签名）的特征：

- (1) 一个签名是被签文件的物理部分；
- (2) 验证物理部分进行比较而达到确认的目的。(易伪造)
- (3) 不容易忠实地“copy”!!!

定义：(数字签名方案) 一个签名方案是有签署算法与验证算法两部分构成。可由五元关系组 (P, A, K, S, V) 来刻化：

- (1) P 是由一切可能消息(messages)所构成的有限集合；
- (2) A 是一切可能的签名的有限集合；
- (3) K 为有限密钥空间，是一些可能密钥的有限集合；
- (4) 任意 $k \in K$ ，有签署算法 $\text{Sig}_k \in S$ 且有对应的验证算法 $\text{Ver}_k \in V$, 对每一个

$\text{Sig}_k: P \rightarrow A$ 和 $\text{Ver}_k: P \times A \rightarrow \{\text{真}, \text{假}\}$
 满足条件: 任意 $x \in P, y \in A$. 有签名方案的一个签名:

$$\text{Ver}(x, y) = \begin{cases} \text{真} & \text{若 } y = \text{sig}(x) \\ \text{假} & \text{若 } y \neq \text{Sig}(x) \end{cases}$$

注: 1*. 任意 $k \in K$, 函数 Sig_k 和 Ver_k 都为多项式时间函数。

2*. Ver_k 为公开的函数, 而 Sig_k 为秘密函数。

3*. 如果坏人 (如 Oscar) 要伪造 Bob 的对 x 的签名, 在计算上是不可能的。也即, 给定 x , 仅有 Bob 能计算出签名 y 使得 $\text{Ver}_k(x, y) = \text{真}$ 。

4*. 一个签名方案不能是无条件安全的, 有足够的时间使 Oscar 总能伪造 Bob 的签名。

RSA 签名: $n = pq, P = A = \mathbb{Z}_n$, 定义密钥集合
 $K = \{(n, e, p, q, d) \mid n = pq, de \equiv 1 \pmod{\phi(n)}\}$

注意： n 和 e 为公钥； p,q,d 为保密的（私钥）。对 $x \in P$,
Bob要对 x 签名，取 $k \in K$ 。 $\text{Sig}_k(x) \equiv x^d \pmod{n} \equiv y \pmod{n}$
于是

$$\text{Ver}_k(x,y)=\text{真} \iff x \equiv y^e \pmod{n}$$

（注意： e,n 公开；可公开验证签名 (x,y) 对错！！也即是否否为Bob的签署）

- 注：** 1*.任何一个人都可对Bob的一个签署 y ,利用Bob的公钥来计算 $x=?e_k(y)$ ，来验证Bob对消息 x 是否真的签名。
- 2*.签名消息的加密传递问题：假设Alice想把签了名的消息加密送给Bob，她按下述方式进行：对明文 x ，Alice计算对 x 的签名， $y=\text{Sig}_{\text{Alice}}(x)$ ，然后用Bob的公开加密函数 e_{Bob} ，计算出 $Z=e_{\text{Bob}}(x,y)$,Alice 将 Z 传给Bob，Bob收到 Z 后，第一步解密：

$$d_{\text{Bob}}(Z)=d_{\text{Bob}}e_{\text{Bob}}(x,y)=(x,y)$$

然后用Alice公钥检验

$$\text{Ver}_{\text{Alice}}(x,y)=\text{真}$$

问题：若Alice首先对消息 x 进行加密，然后再签名，结果如何呢？ $Y=\text{Sig}_{\text{Alice}}(e_{\text{Bob}}(x))$ ，注意 $z=e_{\text{Bob}}(x)$ 。

Alice 将 (z,y) 传给Bob，Bob先将 z 解密，获取 x ；然后用 $\text{Ver}_{\text{Alice}}$ 检验关于 x 的加密签名 y 。这个方法的一个潜在问题是，如果Oscar获得了这对 (z,y) ，他能用自己的签名来替代Alice的签名

$$y'=\text{Sig}_{\text{Oscar}}(e_{\text{Bob}}(x)) , z=z' =e_{\text{Bob}}(x)$$

(注意：Oscar能签名密文 $e_{\text{Bob}}(x)$ ，甚至他不知明文 x 也能做。Oscar传送 (z,y') 给Bob,Bob可能推断明文 x 来自Oscar。所以，至今人们还是推荐先签名后加密。)

6.ElGamal方案

ElGamal公钥密码体制是基于求解离散对数困难问题的。
设 p 至少是150位的十进制素数， $p-1$ 有大素因子。 Z_p 为有域，
若 α 为 Z_p 中的本原元，有 $Z_p^* = \langle \alpha \rangle$ 。若取 $\beta \in Z_p^* = Z_p \setminus \{0\}$ ，
如何算得一个唯一得整数 a ，（要求， $0 \leq a \leq p-2$ ），满足

$$\alpha^a = \beta \pmod{p}$$

将 a 记为 $a = \log_{\alpha} \beta$

一般来说，求解 a 在计算上是难处理的。

Z_p^* 中的Elgamal公钥体制的描述：设明文空间为 $P = Z_p^*$ ，密文空间为 $C = Z_p^* \times Z_p^*$ ，定义密钥空间 $K = \{(p, \alpha, a, \beta) \mid \beta = \alpha^a \pmod{p}\}$

公开钥为： p, α, β

秘密钥（私钥）： a

Alice 取一个秘密随机数 $k \in \mathbb{Z}_{p-1}$ ，对明文 x 加密

$$e_k(x, k) = (y_1, y_2)$$

其中， $y_1 = \alpha^k \pmod{p}, y_2 = x\beta^k \pmod{p}$

Bob 解密，

$$d_k(y_1, y_2) = y_2 (y_1^\alpha)^{-1} \pmod{p}$$

注：1*. 容易验证 $y_2 (y_1^\alpha)^{-1} = x(\alpha^a)^k (\alpha^{ka})^{-1} = x$!!

2*. 利用 ElGamal 加密算法可给出基于此的签名方案：

Bob 要对明文 x 进行签名，他首先取一个秘密随机数 k 作为签名

$$\text{Sig}_k(x, k) = (\gamma, \delta)$$

其中 $\gamma = \alpha^k \pmod{p}, \delta = (x - a\gamma)k^{-1} \pmod{p-1}$

对 $x, \gamma \in \mathbb{Z}_p^*$ 和 $\delta \in \mathbb{Z}_{p-1}$ ，定义 $\text{Ver}_k(x, \gamma, \delta) = \text{真}$ ，它等价于

$$\beta^\gamma \gamma^\delta = \alpha^x \pmod{p}$$

要说明的是，如果正确地构造了这个签名，那么验证将是成功的，因为

$$\beta^r \gamma^\delta = \alpha^{a\gamma + k\delta} \pmod{p} = \alpha^{a\gamma + k\delta} \pmod{p}$$

由上面知道， $\delta = (x - a\gamma)k^{-1} \pmod{p-1}$ 可以推出

$$k\delta = x - a\gamma \pmod{p-1} \text{ 有 } a\gamma + k\delta \equiv x \pmod{p-1}$$

所以 $\beta^r \gamma^\delta = \alpha^x \pmod{p}$

该签名方案已经被美国**NIST**(国家标准技术研究所) 确定为签名标准（1985）。

有关**RSA**方面的内容，请访问网址：

www.RSAsecurity.com