指数对的 k阶自适应窗口表示算法

李学俊[1,2], 胡 磊[2]
(1.西安电子科技大学计算机网络与信息安全教育部重点实验室, 陕西西安 710071;
2.中科院研究生院信息安全国家重点实验室, 北京 100049)

摘 要: 给出了一种新的计算指数对 $g^a h^b$ 的 Straus-Shamir类算法, 该算法基于整数对的一个新表示, 即 k阶自适应窗口表示 (k-AWE). 证明了 k-AWE的平均联合 Hamming密度为 3/(3k+1), 与同类算法相比, 本文算法更为有效. 明确分析了在 512到 2048比特密钥长度的密码学应用中, 窗口宽度的最佳取值为 k=3.

关键词: 指数对; k阶自适应窗口表示; Straus-Shamirs算法
中图分类号: TP309.7 文献标识码: A 文章编号: 0372-2112 (2006) 08-1513-4

Adaptive *k*-Ary Window Expansion Algorithm for Pairs of Exponentiations

LI Xue-jun[1,2], HU Lei[2]
(1.Key Laboratory of Computer Networks and Information Security, Ministry of Education, Xidian University, Xi'an, Shaanxi 710071, China; 2.State Key Laboratory of Information Security, Graduate School of the Chinese Academy of Sciences, Beijing 100049, China)

Abstract Based on a new expansion for pairs of integers called adaptive k-ary window expansion (k-AWE), a new Straus-Shamir-like method for computing $g^a h^b$ is proposed. The average joint Hamming weight of the k-AWE is 3/(3k+1). Comparing with other methods, it is shown that the method can be on-line implemented more efficiently. At the same time, the optimal value for k is also precisely analyzed. Our result shows that k=3 is a best choice for cryptographic application of usual 512~2048 bits key size.

Key words pairs of exponentiations; adaptive k-ary window expansion (k-AWE); Straus-Shamir algorithm

1 引言

公钥密码系统的实现中[1], 模指数 $g^a \pmod p$运算是最费时的运算之一. 常用的指数运算算法是二进制算法(或称平方-乘积算法)[2]. 另外, 还有许多提高指数运算效率的技巧, 基本思想是基于不同的数字集或进制[3], 获得指数 a的不同表示, 其中表示中的非零值尽可能少, 从而使得算法中的乘法运算尽可能少, 比如 m进制算法[4]、NAF(Non-Adjacent Form)算法[5]、自适应 m进制分割算法[6,7]等等.

有些公钥密码系统, 特别是大多数数字签名(除了RSA)的验证过程[8], 通常需要计算两个指数的乘积 $g^a h^b \pmod p$——指数对. 一个直接的计算方法是先分别计算两个指数 g^a 和 h^b, 然后将结果相乘. Shamir[8]认为可以将整数对 a和 b的二进制表示同时引入二进制算法, 使得两个指数运算合并而为一个, 从而大大提高指数对的计算效率. 该算法的最初思想源于 Straus[9], 我们称之为 Straus—Shamir算法. 在某些密码系统中, 比如椭圆曲线密码系统, 逆运算可以非常有效地计算, 对于这类应用, Solinas提出整数对的 JSF(Joint Sparse Form)表示[10]. 在所有取自数字集{0, ±1}的整数对联合表示中, JSF表示具有最小联合 Hamming密度. 基于 JSF表示的 Sraus-Shamir类算法计算 $g^a h^b$, 有时比 Sraus-Shamir算法更有效.

本文给出一组新的 Straus-Shamir类算法, 基于整数对的 k阶自适应窗口表示 k-AWR(Adaptive k-Ary Window Expansion). k-AWR表示是从左到右实现, 引入 Straus-Shamir算法后, 可以实时实现. k-AWR表示取自数字集 $\{0, 1, \cdots, 2^k-1\}$, 不涉及逆运算. Straus-Shamir类算法的效率取决于整数对的平均联合 Hamming密度, 本文详细分析了 k-AWR表示, 其长度不会长于相应二进制表示的长度, 平均联合 Hamming密度为 3/(3k+1), 在同类算法中具有明显的速度优势. k-AWR算法以窗口宽度 k为参数, 不过 k值不固定, 是自适应的, 我们的结果显示, 对于 512到 2048比特密钥长度的典型密码学应用, 最佳选择是 k=3.

收稿日期: 2005-03-30; 修回日期: 2006-04-10
基金项目: 国家自然科学基金 (No. 60373041)

2 相关算法

2.1 二进制算法

二进制算法[1]是计算指数 g^a 的最基本算法,其中 g 为乘法群 F_p^* 的元素, a 通常为一大整数,比如长 1024比特,二进制表示为 $a=\langle a_{L-1}, a_{L-2}, \cdots, a_1, a_0\rangle$, $a_i\in\{0,1\}$. 算法从左到右扫描所有 a_i,每一比特需要一个平方运算,而每一非零比特又需要一个乘法运算,如果二进制表示的长度为 L,二进制算法需要 L 个平方和平均 L/2个乘法运算. 当计算指数对 g^ah^b 时,其中 g 和 h 为乘法群 F_p^* 中元素, a 和 b 为 L 比特的大整数,最直接的方法是分别计算 g^a 和 h^b,然后相乘. 然而,这样总共需要 2L 个平方和平均 L 个乘法运算.

2.2 Straus-Shamir算法

为加速计算 g^ah^b, Straus-Shamir算法将 a 和 b 的二进制表示同时引入二进制算法,从而两个指数的乘积运算合并而为一个,比如计算 $g^{37}h^{20}$,具体过程如下:

37= 〈 1 0 0 1 0 1 〉

20= 〈 0 1 0 1 0 0 〉

1 g^2 g^4h^2 g^8h^4 $g^{18}h^{10}$ $g^{36}h^{20}$

×g g $g^{37}h^{20}$

×h g^2h

×gh g^9h^5

其中预计算并存储 gh,当 a 和 b 的二进制表示在相同位置上都为 1时,可以乘以一个 gh,而不必分别乘以 g 和 h. 一对整数的二进制表示在同一位置上都为 1的概率为 1/4,所以 Straus-Shamir算法需要 L 个平方和平均 3L/4个乘法运算. 整数对的联合二进制表示的非零列数即为 Straus-Shamir算法中的乘法运算数,称为整数对的联合 Hamming密度.

2.3 基于 JSF 表示的 Straus-Shamir类算法

某些密码系统,比如椭圆曲线密码系统,逆运算可以非常有效地计算,因此可以将 a和 b表示为符号二进制形式. 一个显而易见的方法是用 NAF表示[5]作为符号二进制表示,引入 Straus-Shamir算法后需要 L 个平方和平均 5L/9个乘法运算. 该算法虽然有效,但不是最优. 为了最优化整数对的联合符号二进制表示, 2001年, Solinas提出了 JSF表示[10],并证明 JSF表示在整数对的所有取自数字集 $\{0,\pm1\}$的联合表示中,具有最小的平均联合 Hamming密度,为 L/2,引入 Straus-Shamir算法后需要 L 个平方和平均 L/2个乘法运算. Straus-Shamir算法总是从左到右扫描整数对表示的每一列,而整数对的 JSF 表示却需要从右到左实现,两个算法不能同时实现,需要 2L 的空间存储 JSF表示后再引入 Straus-Shamir算法.

3 指数对的 *k* 阶自适应窗口表示 (*k*-AWE)算法

3.1 符号表示

k-AWE算法包括预计算和主要算法两个过程. 参数 k 表示预计算过程中指数的最长比特,限定了预计算的窗口宽度. a 和 b 的联合二进制表示定义如下

$$a=\langle a_{L-1}, a_{L-2}, \cdots, a_1, a_0\rangle$$

$$b=\langle b_{L-1}, b_{L-2}, \cdots, b_1, b_0\rangle$$

其中 $a_i, b_i\in\{0,1\}$. a[i..j]表示整数 a 从第 i位到第 j位的比特串,其中 $i, j\in\{L-1, L-2, \cdots, 1, 0\}$, $i\geqslant j$,即 $a[i..j]=\langle a_i, a_{i-1}, \cdots, a_{j+1}, a_j\rangle$.

3.2 预计算

预计算是在主要算法之前计算并存储一组指数对 g^uh^v,其中 u, $v\in\{0,1,\cdots,2^k-1\}$,但如果 $u=0,2,\cdots,2^k-2$,我们规定相应存储的 $v\neq0,2,\cdots,2^k-2$. 预计算的结果存储在表 P[u,v],其中 P[i,j]存放 g^ih^j. 如果参数 k 确定,需要预计算的指数对的个数为 $(2^k)^2\times(3/4)-2$. 比如,当 k=3时,预计算数为 46,预计算表如表 1.

表 1

P[u,v]	v=0	v=1	v=2	v=3	v=4	v=5	v=6	v=7
u=0	/	/	/	h^3	/	h^5	/	h^7
u=1	/	gh	gh^2	gh^3	gh^4	gh^5	gh^6	gh^7
u=2	/	g^2h	/	g^2h^3	/	g^2h^5	/	g^2h^7
u=3	g^3	g^3h	g^3h^2	g^3h^3	g^3h^4	g^3h^5	g^3h^6	g^3h^7
u=4	/	g^4h	/	g^4h^3	/	g^4h^5	/	g^4h^7
u=5	g^5	g^5h	g^5h^2	g^5h^3	g^5h^4	g^5h^5	g^5h^6	g^5h^7
u=6	/	g^6h	/	g^6h^3	/	g^6h^5	/	g^6h^7
u=7	g^7	g^7h	g^7h^2	g^7h^3	g^7h^4	g^7h^5	g^7h^6	g^7h^7

如果按照如下顺序预计算,共需乘法群 F_p^* 中 52个模乘运算:

$h \to gh \to gh^2 \to gh^3 \to gh^4 \to gh^5 \to gh^6 \to gh^7$

$\downarrow$ $\downarrow$

$h^2 \to g^2 \to g^2h \to g^2h^3 \to g^2h^5 \to g^2h^7$

$\downarrow$ $\downarrow$

$h^3 \to g^3 \to g^3h \to g^3h^2 \to g^3h^3 \to g^3h^4 \to g^3h^5 \to g^3h^6 \to g^3h^7$

$\downarrow$ $\downarrow$

$h^4 \to g^4 \to g^4h \to g^4h^3 \to g^4h^5 \to g^4h^7$

$\downarrow$ $\downarrow$

$h^5 \to g^5 \to g^5h \to g^5h^2 \to g^5h^3 \to g^5h^4 \to g^5h^5 \to g^5h^6 \to g^5h^7$

$\downarrow$ $\downarrow$

$h^6 \to g^6 \to g^6h \to g^6h^3 \to g^6h^5 \to g^6h^7$

$\downarrow$ $\downarrow$

$h^7 \to g^7 \to g^7h \to g^7h^2 \to g^7h^3 \to g^7h^4 \to g^7h^5 \to g^7h^6 \to g^7h^7$

更一般地,如果 k 确定,需要乘法群 F_p^* 中模乘运算数为 $3\times2^{2k-2}+2^k-4$.

3.3 主要算法

新算法扫描 a 和 b的联合二进制表示,从最高有效位 a_{L-1}, b_{L-1}开始.

Algorithm 指数对的 k 阶自适应窗口表示的 Straus-Shamir类算法

输入: 乘法群 F_p^* 中的两个元素 g 和 h,大整数对 a 和

b的联合二进制表示

输出: 乘法群 F_p^* 中元素 $x = g^a h^b$

(1) $i \leftarrow L-1$; $x \leftarrow 1$;

(2) While $i \geqslant 0$ and $a_i = 0$ and $b_i = 0$执行 do $i \leftarrow i-1$;

(3) if $i < 0$ then返回 x;

(4) while $i \geqslant 0$执行

a if $i \geqslant k-1$ then $j = i-k+1$ else $j = 0$;

b while $j \leqslant i$ and $a_j = 0$ and $b_j = 0$执行 $j = j+1$;

c $u \leftarrow$比特串 $a[i..j]$的值;

d $v \leftarrow$比特串 $b[i..j]$的值;

e while $i \geqslant j$执行 { $i \leftarrow i-1$; $x \leftarrow x^2 \pmod p$; }

f $x \leftarrow x \cdot P[u, v] \pmod p$;

g while $i \geqslant 0$ and $a_i = 0$ and $b_i = 0$执行

{ $i \leftarrow i-1$; $x \leftarrow x^2 \pmod p$; }

(5) 返回 x;

算法从左到右扫描 a 和 b 的每一比特, 将联合二进制表示转换成 k-AWE表示, 同时, 实现 Straus-Shamir类算法, 所以该算法是实时的. 另外, 整数对 a 和 b 的 k-AWE表示不会比它们相应的联合二进制表示的长度更长.

4 k-AWE表示的联合 Hamming密度

为了确定新算法的运算量, 必须确定整数对 k-AWE表示的长度和平均联合 Hamming密度. 长度决定了主要算法所需平方运算数, 对于给定的L, a 和 b 的 k-AWE表示的长度取值于区间 $[L, L-k+1]$, 如果L 很大而 k 很小, 比如 L = 1024且 k = 3, 所需平方运算数将趋于L.

密度决定了主要算法所需乘法运算数. 假设主要算法所需平均乘法运算数为 $N_{k,L}$, k 为大于 1的任意整数, 则 $N_{k,L}/L$ 表示 a 和 b 的 k-AWE表示的平均联合 Hamming密度.

引理 1　$N_{k,L} = \begin{cases} (2^L - 1)/2^L, & 0 \leqslant L \leqslant k \\ N_{k,L-1}/4 + 3(1 + N_{k,L-k})/4 & L > k \end{cases}$

证明 1　当 L = 0时, 显然 $N_{k,0} = 0$ 当 $0 < L \leqslant k$ 时, a_i 和 b_i 都为 0的平均概率为 $1/2^L$, 这时需要的乘法运算数为 0. a_i 和 b_i 至少一个为 1的平均概率为 $1-1/2^L$, 由于预计算表 P [u, v], 这时需要一个乘法, 所以有 $N_{k,L} = \frac{1}{2^L} \times 0 + \frac{2^L - 1}{2^L} \times 1 = \frac{2^L - 1}{2^L}$.

当 $L > k$ 时, $a_{L-1} = 0$且 $b_{L-1} = 0$的概率为 1/4这时所需乘法运算数是 $N_{k,L-1}$. a_{L-1}及 b_{L-1}至少一个为 1的概率是 3/4这时所需乘法运算数包括两个部分: 一部分考虑子串 $g^{a[L-1..L-k]} h^{b[L-1..L-k]}$, 由于预计算表 P [u, v], 需要一个乘法运算. 另一部分考虑子串 $g^{a[L-k-1..0]} h^{b[L-k-1..0]}$, 需要乘法运算数为 $N_{k,L-k}$, 因此得到递归关系 $N_{k,L} = N_{k,L-1}/4 + 3(1 + N_{k,L-k})/4$

引理 2　$\lim_{L\to\infty} \frac{N_{k,L}}{L} = \frac{3}{3k+1}$

证明　设 $\Delta_{k,L} = N_{k,L} - 3L/(3k+1)$, 当 $0 \leqslant L \leqslant k$ 时, 根据引理 1可得 $|\Delta_{k,L}| = \left| \frac{2^L - 1}{2^L} - \frac{3L}{3k+1} \right|$, 显然有 $|\Delta_{k,L}| < 1$. 当 $L > k$ 时, 假设任意 $L < n$ 都有 $|\Delta_{k,L}| < 1$ 那么当 L = n 时, 根据引理 1可得

$$|\Delta_{k,L}| = \left| \frac{1}{4} N_{k,L-1} + \frac{3}{4} + \frac{3}{4} N_{k,L-k} - \frac{3L}{3k+1} \right|$$

$$= \left| \frac{1}{4} \left(\frac{3(L-1)}{3k+1} + \Delta_{k,L-1} \right) + \frac{3}{4} + \frac{3}{4} \left(\frac{3(L-k)}{3k+1} + \Delta_{k,L-k} \right) - \frac{3L}{3k+1} \right|$$

$$= \frac{1}{4} |\Delta_{k,L-1} + 3\Delta_{k,L-k}|$$

显然有 $|\Delta_{k,L}| < 1$. 从而可以得到 $\lim_{L\to\infty} \frac{\Delta_{k,L}}{L} = 0$, 于是

$$\lim_{L\to\infty} \frac{N_{k,L}}{L} = \lim_{L\to\infty} \left(\frac{3}{3k+1} + \frac{\Delta_{k,L}}{L} \right) = \frac{3}{3k+1} + \lim_{L\to\infty} \frac{\Delta_{k,L}}{L} = \frac{3}{3k+1}.$$

定理　对于任意整数 $k > 1$, 整数对的 k-AWE表示的联合 Hamming密度为 $3/(3k+1)$.

定理的证明可以直接由引理 1和引理 2获得. 因此, 本文算法的主要算法需要 L 个平方和平均 $3L/(3k+1)$ 个乘法运算.

5 实现

5.1 一个整数对 k-AWE表示的实例

例如, 假设整数对 a = 53和 b = 102, 则 NAF表示为

53= (0, 1, 0, -1, 0, 1, 0, 1)

102= (1, 0, -1, 0, 1, 0, -1, 0)

联合 Hamming密度为 8. Sollinas的 JSF表示为

53= (0, 1, 0, 0, -1, 0, -1, -1)

102= (1, 0, 0, -1, -1, 0, -1, 0)

联合 Hamming密度为 6. k-AWE(k = 3)表示为

53= (0, 0, 3, 0, 0, 0, 5)

102= (0, 0, 6, 0, 0, 0, 6)

联合 Hamming密度为 2.

5.2 窗口宽度 k 的最佳选择

综上所述, 基于 k-AWE表示的 Straus-Shamir类算法计算指数对, 需要的乘法运算总数为 $f(k) = 3 \times 2^{2k-2} + 2^k - 4 + 3L/(3k+1)$. 如果 L 确定, 窗口宽度 k 值, 与预计算乘法数成正比, 而与主要算法的乘法数成反比. 对于确定的 L, 通过极小化函数 f(k)可以获得窗口宽度 k 的最佳选择.

f(k)的导函数为 $f'(k) = \ln 2 \times (6 \times 2^{2k-2} + 2^k) - 9L/(3k+1)^2$, 非线性方程 $f'(k) = 0$的解即为 k 的最佳选择, 利用割线迭代公式, 假设两个初始值 $k_0 = 1$ 和 $k_1 = 2$, L = 1024, 六次迭代后获得 k = 3. 13≈ 3. 该结果也可如表 2验证. 对于某些确定的 L, 表 2列出预计算、主要算法以及总的乘法运算数.

从表 2中可知, 对于 512到 2048比特密钥长度的典型密码学应用, 最好取窗口宽度 k = 3. 下面将本文算法和同类算法相比较, 包括四种不同的整数对表示, 引入 Straus-

Shamir算法主要需要平方和乘法两种运算,其中平方运算数都是趋于 L,而需要的乘法运算数如表 3.

表 2

窗口宽度 k		k = 1	k = 2	k = 3	k = 4	k = 5	k = 6
预计算的乘法数		1	12	52	204	796	3132
L = 512	主要乘法数	384	219	153	118	96	80
	总乘法数	385	231	205*	322	892	3212
L = 1024	主要乘法数	768	438	307	236	192	161
	总乘法数	769	450	359*	440	988	3293
L = 1536	主要乘法数	1152	658	460	354	288	242
	总乘法数	1153	670	512*	558	1084	3374
L = 2048	主要乘法数	1536	877	614	472	384	323
	总乘法数	1537	889	666*	676	1180	3455

表 3

表示方法	联合二进制	NAF	JSF	k-AWE
L = 512	384	284	256	205(k = 3)
L = 1024	768	568	512	359(k = 3)
L = 1536	1152	853	768	512(k = 3)
L = 2048	1536	1137	1024	666(k = 3)

6 结论

Shamir[8]提出利用 Straus-Shamir算法加速指数对的计算. Solinas[10]提出整数对的 JSF表示,进一步提高了 Straus-Shamir算法的运算效率.本文在总结前人工作的基础上,提出了整数对的一种新的 k-AWE表示,引入 Straus-Shamir算法后,在同类算法中具有明显的速度优势.由于有些乘法群 F_p^* 上的公钥密码体制,特别是大多数数字签名(除了 RSA)的验证过程,需要指数对的运算,所以本文的研究是非常必要的.

参考文献:

[1] A J Menezes, et al. Handbook of Applied Cryptography [M]. New York, USA: CRC Press, 1997.

[2] D E Knuth. The art of Computer Programming, Volume 2 Seminumerical Algorithms, Third edition[M]. San Francisco, USA: Addison-Wesley, 1997.

[3] B Phillips, N Burgess. Minimal weight digit set conversions [J]. IEEE Transactions on Computers, 2004, 53(6): 666-677.

[4] D Gordon. A survey of fast exponentiation methods[J]. Journal of Algorithms, 1998, 27(1): 129-146.

[5] S Arno, F S Wheeler. Signed digit representations of minimal Hamming weight[J]. IEEE Transactions on Computers, 1993, 42(8): 1007-1010.

[6] L C K Hui, K Y Lam. Fast square-and-multiply exponentiation for RSA[J]. Electronics Letters, 1994, 30(17): 1396-1397.

[7] K Y Lam, L C K Hui. Efficiency of SS(I) square-and-multiply exponentiation algorithms[J]. Electronics Letters, 1994, 30(25): 2115-2116.

[8] T ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms[J]. IEEE Transactions on Information Theory, 1985, IT-31(4): 469-472.

[9] E G Straus. Addition chains of vectors (problem 5125) [J]. American Mathematical Monthly, 1964, 70: 806-808.

[10] J A Solinas. Low-Weight Binary Representations for Pairs of Integers[DB/OL]. http://www.cacr.math.uwaterloo.ca/, CORR 2001-41, University of Waterloo, 2001.

作者简介:

李学俊 女,1969年生,中科院研究生院信息安全国家重点实验室博士后,现为西安电子科技大学教师,研究兴趣:椭圆曲线密码算法、密码协议分析等. E-mail: aluckydd@mail.xidian.edu.cn

胡 磊 男,1967年生,中科院研究生院信息安全国家重点实验室教授、博导,研究兴趣:椭圆曲线公钥密码、密码序列等.