基于伪随机数生成器的标量乘改进算法①

李 辉, 刘中华, 易军凯

(北京化工大学 信息科学与技术学院, 北京 100029)

摘 要: 标量乘算法是椭圆曲线密码体制中最基本、最耗时的算法, 包含点加和倍点两种运算. 传统的改进方法通过改造标量表示形式减少非零元位数来降低标量乘中的点加运算次数. 为了进一步提高标量乘算法效率, 根据标量的生成方式, 提出了一种结合伪随机数生成器改进算法. 利用斐波那契数列生成器的的循环迭代相加可以将标量乘运算由反复的点加和倍点运算转化为单一的点加运算. 实验结果表明, 改进算法相比传统的窗口 NAF 算法能够降低 60%以上的运算量.

关键词: 标量乘; 椭圆曲线密码体制; 点加; 倍点; 伪随机数生成器; 斐波那契数列生成器

Improved Algorithm of Scalar Multiplication Based on Pseudo-Random Number Generator

LI Hui, LIU Zhong-Hua, YI Jun-Kai

(College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract: Scalar multiplication is the dominant and time consuming algorithm in elliptic curve cryptography, including point addition and point doubling. Traditional improved methods reduced the calculation of point addition by transforming the representation of scalar to reduce the number of non-zero bits. To further improve the efficiency of the scalar multiplication, this paper proposes an improved algorithm combining pseudo-random number generator according to the generation of the scalar. With cyclical iterative addition of lagged Fibonacci generator, scalar multiplication can be transformed into simplex operation of point addition from repetitive point addition and point doubling. Experimental results show that the improved algorithm compared to the traditional window NAF algorithm can reduce more than 60% of the amount of computation.

Key words: scalar multiplication; elliptic curve cryptography; point addition; point doubling; pseudo-random number generator; lagged Fibonacci generator

椭圆曲线密码体制(Elliptic Curve Cryptography, ECC)是一种基于椭圆曲线上离散对数难题的公钥密码, 由 Koblitz[1]和 Miller[2]分别提出. 由于椭圆曲线上离散对数问题相比大整数分解和传统离散对数问题复杂性更高, 故此 ECC 具有最强的单比特安全性. 近年来人们对大整数分解问题的研究使得 1024 位密钥的 RSA 密码不再安全, 而增加密钥长度会使得系统实现速度减慢, 对设备的存储运算能力也有更高的要求. 密钥长度短小的特点使得 ECC 更适合应用于存储空间与计算能力受限的设备中, 如小型无线网络设备, 智能卡等等[3].

ECC 的核心运算标量乘包含两种基本的运算: 点加和倍点运算, 其最基础的算法是二进制算法. 之后又发展出了带有加减法链的方法, NAF 表示法[4]是一种典型加减法链的利用, 可以减少标量乘的点加次数. 而结合了 NAF 和窗口法的 w-NAF 算法通过存储预计算的点来一次完成数据块内多位的计算, 进一步减少了点加的运算次数. 在以上这些算法中, 标量乘都是通过反复的点加与倍点运算来计算结果的. 传统的标量乘算法通过改造标量的表达形式, 利用减少非零元位数的方式来降低点加的运算次数, 但是倍点的运算次数并没有明显的降低[5].

① 收稿时间:2014-04-26;收到修改稿时间:2014-05-16

目前国内外针对标量乘快速算法方面, 还包括固定基 Comb 法、多基数链法、Montgomery 等优化算法, 国内一些学者对 Montgomery 算法进行改进节省了一定的运算时间, 在多标量乘的快速实现方面, 有具体的联合带符号二进制表示的编码算法[6], 在一定程度上提高了标量乘的计算效率.

本文对标量乘算法研究后发现, 在很多情况下标量是小于椭圆曲线基点阶的一个随机整数, 利用这个随机数扰乱加密后的密文, 并不参与后面的传输和解密过程. 随机的标量完全可以由伪随机数生成器得到, 而标量乘的目的只是为了得到一个结果点, 在完成一次标量乘计算后这个随机数便不再起任何作用.

在目前流行的伪随机数生成器中, 斐波那契数列生成器(LFG)具有算法简单、速度快、随机周期大的特点. 其以加法作为二元运算的生成器(ALFG)更是可以把标量乘运算转化为单一的点加运算. 标量乘运算的本质就是椭圆曲线点自身连续相加多次, 只不过由于次数很多, 即标量太大造成算法实现的困难, 进而人们把标量乘转化为反复的点加和倍点运算, 具有了可实现性. 利用 ALFG 生成器, 将预计算的随机椭圆曲线点作为初始种子, 经过有限次的循环迭代相加, 得到的点仍然是随机的点, 并且无需关注随机标量的具体数值而直接获取到标量乘的结果.

1　椭圆曲线上的标量乘算法

定义在有限域 K 上的 Weierstrass 方程:

$$E: y^2 \equiv x^3 + ax + b \pmod p$$

所确定的平面曲线称为域 K 上的椭圆曲线, 其中域 K 的特征 p 大于 3, 参数 a 与 b 属于 K 且 $4a^3 + 27b^2 \neq 0$. 椭圆曲线 E 上的点连同其上所定义的一个特殊的无穷远点∞对椭圆曲线上点的加法构成一个交换群. 将椭圆曲线上的一个点 P 连续相加 k 次得到曲线上另外的一个点 Q, 即 Q=kP 的计算过程称为标量乘. 标量乘可以通过累次相加得到结果, 但是对于标准 256 位实数域下的椭圆曲线, 此种方法计算量太大无法实现. 为此, 人们将椭圆曲线上的标量乘算法转化为点加和倍点两种最基本运算, 以二进制表示标量 k, 并进行各种形式的改造, 来降低运算的复杂度.

标量乘的特点是曲线上的点连续多次相加得到最终结果, 相加的过程可以转化为点加和倍点两种运算, 但是仅仅给出初始点和结果点无法求出标量的值, 也就是连续相加的次数, 这也是椭圆曲线加密安全性的保障, 即椭圆曲线下离散对数(ECDLP)复杂难题的原理.

1.1 二进制算法

二进制方法是计算椭圆曲线上的标量乘最经典方法, 其原理是通过将标量 k 表示为二进制的形式, 即

$$k = \sum_{j=0}^{n-1} K_j 2^j, K_j \in \{0,1\}$$

从而反复利用点加与倍点运算得到标量乘结果.

算法 1. 二进制算法

输入: $P \in E(p)$, $k=(k_{n-1},k_{n-2},\ldots,k_1,k_0)_2$

输出: $Q = kP$

(1)$Q = \infty$

(2)For $i = n - 1$ to 0 do

① $Q = 2Q$

② If $k_i = 1$ then $Q = Q + P$

(3)Return Q

二进制算法平均计算量为: nD+(n/2)A. 其中 D 表示倍点运算, A 表示点加运算.

1.2 NAF 算法

使用二进制算法计算标量乘的运算量主要取决于标量转化为二进制表示的序列长度及序列中非零元的个数. 在使用二进制的前提下很难改变标量转化后的序列长度, 但是可以通过引入带符号的二进制序列来减少非零元的个数, 从而减少点加的运算次数. NAF 是一种非邻接形式的二进制序列, 即任何连续的两个位数最多只有一位非零元, 整个序列平均只有三分之一的非零元位数[7].

算法 2. 标量 k 转化 NAF 序列

输入: 标量 k

输出: k 的 NAF 序列

(1)Let $i = 0$

(2) While $k > 0$ do

① If k is odd $k_i = 2 - (k \bmod 4)$

$k = k - k_i$

② Else $k_i = 0$

③$k = k/2$, $i = i+1$

(3)Return $(k_{i-1}, k_{i-2},\ldots\ldots k_1, k_0)$

使用 NAF 计算标量乘的原理与二进制法类似, 只是当 k_i =-1 时, 计算 Q=Q–P. NAF 算法计算标量乘的运算量为 nD+(n/3)A.

1.3 窗口 NAF 算法(w-NAF)

窗口法[8]通过预先存储的点一次计算数据块内多比特位的方式加快运算速度, 是一种以空间换时间的处理方法. 结合 NAF 形式的窗口法[9]可以进一步减少点加的运算次数.

算法 3. 计算 w-NAF 序列

输入: window w, scalar k

输出: $\mathrm{NAF_w}(k)$

(1)Let $i = 0$

(2)While $k >= 1$

①If k is odd, then $k_i = k \bmod 2^w$

$k = k - k_i$

② Else $k_i = 0$

③ $k = k / 2$, $i = i+1$

(3) Return $(k_{i-1}, k_{i-2}, \ldots\ldots, k_1, k_0)$

算法 4. w-NAF 算法

输入: window w, $\mathrm{NAF_w}(k) = (k_{n-1}, k_{n-2}, \ldots\ldots, k_1, k_0)$, points $P_{ki} = \{1, 3, 5, \ldots\ldots, 2^{w-1}-1\}P$

输出: $Q = kP$

(1) $Q = \infty$

(2) For $i = n$-1 to 0 do

① $Q = 2Q$

② If $k_i \neq 0$, then:

If $k_i > 0$, then $Q = Q + P_{ki}$

Else $Q = Q - P_{ki}$

(3)Return Q

对于一个窗口大小为 w 的 w-NAF 序列, 其非零元位数的平均密度是 $1/(w+1)$, 所需的存储空间为$(2^{w-2} - 1)$, 平均运算量为 nD+$(n/w+1)$A. 以上所介绍的几种常见标量乘方法都是通过改造标量的表示形式来减少点加运算的次数.

2 基于伪随机数生成器标量乘改进算法

在基于 ECC 的密码系统中, 标量经常被加密一方随机生成并计算标量乘, 如 EC-ElGamal 加密、ECDH 密钥交换、ECIES 加密等. 而实际应用中, 随机数的生成是由伪随机数生成器来实现的, 其特点是算法简单、速度快并且具有较高的随机性[10]. 标量乘运算是有限个椭圆曲线点迭代相加而得, 而 addition lagged Fibonacci generator(ALFG)是一种基于 Fibonacci 序列循环迭代加法运算而产生伪随机数的伪随机数生成器. 本文将利用 ALFG 将标量乘由原来的反复点加与倍点运算转化为点的循环迭代相加, 从而消除倍点运算并减少点加的运算量, 在整体上使标量乘计算效率有大幅的提高.

2.1 斐波那契数列

斐波那契数列满足线性递归: $F_n = F_{n-1} + F_{n-2}$, 通常的表达形式为 $X_n = X_{n-s} \circ X_{n-r} \bmod 2^m$ $(0 < s < r)$. 其中二元运算○可以为加法、减法、乘法和异或. ALFG 就是建立在斐波那契数列上以加法为二元运算的一种伪随机数生成器, 其最大随机周期[10]是$(2^r - 1)*2^{m-1}$.

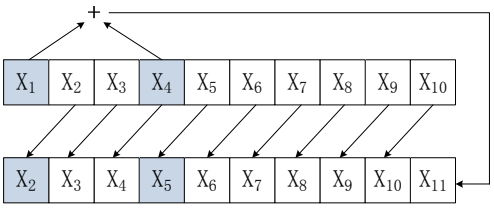
2.2 ALFG 生成随机数

在斐波那契数列中含有两个滞后量参数 r 和 s, ALFG 需要一组初始种子与滞后量 r、s 并循环迭代计算 $X_n = X_{n-s} + X_{n-r} \pmod{2^m}$, 在迭代一定的次数后, 产生的数列趋于随机. 滞后量 r、s 的选取需要满足一定的条件才能获得更好的随机数[10], 很多文献提供了经实验证明比较合理的组合, 如(10, 7), (55, 24), (127, 97), (607, 334)等等. 由于 ALFG 算法简单、运算速度快、随机周期较大, 本文用 ALFG 作为随机标量的生成器. Marsaglia 的测试表明当滞后量大于 55 时得到的随机数具有较高的随机性, Brent 建议滞后量应大于 100 能够得到更加满意的随机数, 随着滞后量的增大, 随机周期也越来越大, 随机效果越好[11]. 基于 ECC 的加密系统更适用于无线网络、智能卡等计算能力、存储空间受限的设备中, ALFG 参数的选取也要考虑平台的环境因素. 为了简便说明, 本文选取滞后量(10, 7)作为实验参数. 初始的随机种子需要用其他随机性较好的生成器生成, 本文使用了 Mersenne Twister 算法生成所需的随机数.

2.3 ALFG 改进算法流程

使用 Mersenne Twister generator 生成的 10 个随机种子: $X_1, X_2, \ldots, X_{10}$, 进而可以通过标量乘 $P_i = X_iG$(点 G 为基点)预先计算出对应的 10 个初始点. 在 ALFG 中使用能存储 10 个随机数的空间, 循环迭代相加 10 个种子一定次数就可以得到新的随机数序列.

类似地, 对 10 个初始椭圆曲线点做同样次数的迭代相加即可得到新的点序列, 如图 1 所示.



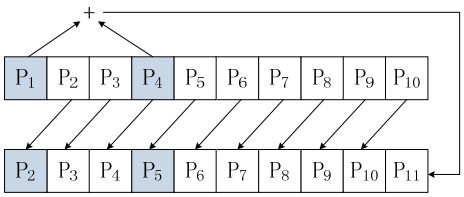


图 1　ALFG 迭代生成随机数 X_i 和曲线点 P_i

新得到的椭圆曲线点与随机数依然满足 $P_i = X_iP$ 关系式. 这改进算法就把标量乘运算转换成了 ALFG 的迭代点加运算, 如图 2 所示.

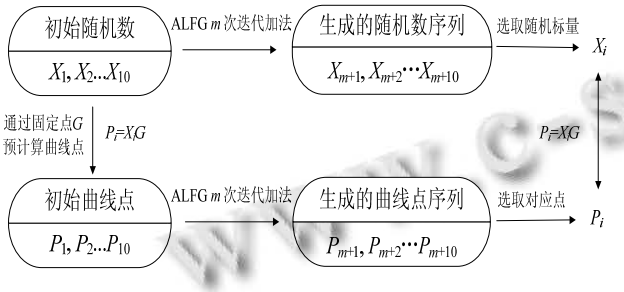


图 2　ALFG 计算标量乘原理

为了获得更好的随机性, Patrick Burns 建议 ALFG 的循环迭代次数不少于 100 次[12], 所以 ALFG 改进算法的计算量 mA, m 是一个常量, 并不随着标量大小的增加而变大, 各算法运算量比较如表 1 所示.

表 1　各标量乘算法比较

算法	运算量	预计算点
二进制算法	nD + (n/2) A	
NAF 算法	nD + (n/3) A	
w-NAF 算法	nD + (n/w + 1) A	$2^{w-2} - 1$
ALFG 算法	mA	r

其中改进算法所需预计算点和存储空间 r 为 ALFG 的滞后量参数. 从上表中可以看出, ALFG 改进算法的总体计算量是一个常数, 并且只含有单一的点加运算. 由于当前的标量乘优化算法只能在一定程度上减少点加运算次数, 对于倍点运算没有实际的优化效果, 而倍点运算又占据了标量乘的大部分计算量, 因此改进算法对标量乘的总体计算量有着明显的优化效果. 并且由于迭代次数是一个常数, 并不随着标量位数增多而增大, 在位数越高的标量乘计算中优化效果也越明显, 如图 3.

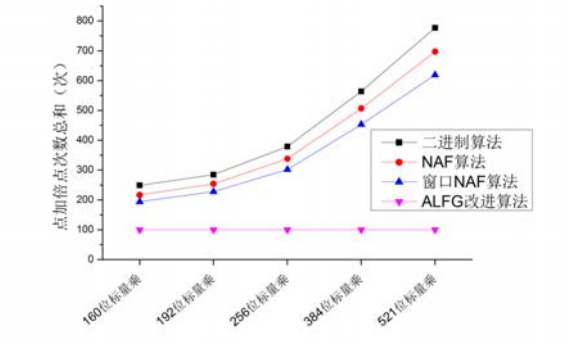


图 3　不同位数下各标量乘算法计算量

3　ALFG改进算法的应用

本节对椭圆曲线上的 ElGamal 加密系统应用了改进算法来完成部分标量乘的计算. 在此选用的椭圆曲线为 ANSI X9.62–2005 标准下的 P-256 位曲线. 具体参数如下:

实数域 $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. 椭圆曲线上的基点 $G(x_G, y_G)$, 公钥点 $Q(x_Q, y_Q)$:

x_G=6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296.

y_G=4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5.

x_Q=8f2501bd1e23ef5899571a5e170c45394f7f126851e75a814b3ea741791fad1f.

y_Q=64e0ed0def2eba3f035963a10f2b9a6fcf6ad8ea9ac3d14e830ccf5182fb31b5.

加密方将信息字段 m 嵌入椭圆曲线 $E(F_p)$中, 表示为点 $M(x_M, y_M)$:

x_M=13030ccb2c844134.

y_M=768fc93a373d02bf9f1c621b499eb3f468fafa3d1389f2d7de62ce95fedededea.

加密方在区间[1, n-1]之间选取一个随机数 k 并计算$(x_1, y_1) = kG$. 这里本文使用改进算法生成随机数 k 并计算标量乘 kG. 首先利用 Mersenne Twister generator 生成 10 个随机数 $X_1...X_{10}$ 作为 ALFG 初始随机种子.

首先根据新生成的随机数 $X_1…X_{10}$ 计算出对应的椭圆曲线点 $P_i = X_iG$, 得到 $P_1…P_{10}$. 使用 ALFG 将 $X_1…X_{10}$ 循环迭代相加 100 次, 得到新的随机序列 $X_{101}…X_{110}$. 同样使用 ALFG 将 $P_1…P_{10}$ 循环迭代相加 100 次, 得到新的椭圆曲线点 $P_{101}…P_{110}$. 关系式 $P_i = X_iG$ 依然成立.

当需要随机数时即可从 $X_{101}…X_{110}$ 中进行选择, 假设选取 X_{108} 作为随机数 k, 改进算法以从 $P_{101}…P_{110}$ 中

选择对应的 P_{108} 就是标量乘的结果了. 这样一来甚至不需要知道随机数 k 是多少, 就能直接得到标量乘运算的结果. 改进算法能应用在这里也是因为在ElGamal 加密原理中, $(x_1, y_1) = kG$ 完成后 k 便不再有任何作用, 它只是中间的一个变量, 解密时可以完全消除, 并且基点 G 是固定的点.

k=bf37b49aefc7ca2575444eece9f57196b0cf6aa47af2906e7af6ba28ba987141.

x_1=e5dc913984a8c769b43de9ce4d4f5df471bbeb4f3e636a4214f684e2681f0f73.

y_1=38cd2f10178a3b344b2937e3e2e42d6f361a7cb9d5de51cfd54004e5588d1b1.

令 $C_1 = (x_1, y_1)$, 并计算$(x_2, y_2) = M + kQ$.

x_2=99588e02e4694d202a66d08682db687658899d3db17e3a5135bd7a34428b695a.

y_2=b725347b2cd0b7420510cc1332716fc68c1f041b13dc9201483561c2a521ed66.

$C_2 = (x_2, y_2)$, 加密完成, 加密方将 C_1 与 C_2 传送给解密方.

解密方得到 C_1、C_2 并计算 $M(x_M, y_M) = C_2 - dC_1$, 将得到的明文点 M 解码成明文信息字段 m, 解密完成.

图 4 反映了在上例中完成 10 组不同标量乘运算, 各种算法所需要的点加与倍点总次数.

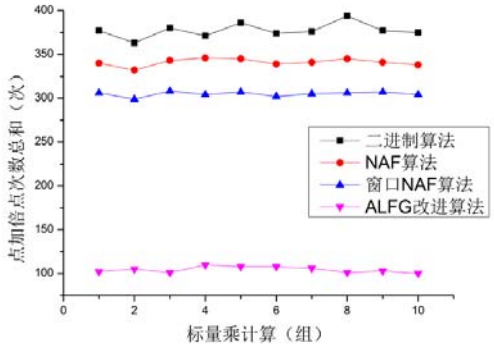


图 4 不同算法计算 10 组标量所需运算次数的比较

以 256 位曲线下的 EC-ElGamal 加密为例, 完成一次加密解密流程不同算法所需的点加倍点计算量如下表 2 所示:

表 2 各标量乘算法比较

算法	倍点	点加	总次数	减少量(%)
二进制算法	764	403	1167	--
NAF 算法	760	275	1035	11.3
w-NAF 算法	757	207	964	6.9
ALFG 算法	0	300	300	68.9

由于改进算法中不含倍点次数, 使其运算量大大减少, 而点加的次数是常量, 并不会因为标量的增大而升高, 这也是比传统方法更优越的一个特点.

4 结语

本文首先分析了标量乘在椭圆曲线加密中的原理以及一些传统的标量乘优化算法. 这些算法关注于改造标量的表现形式, 无论是二进制、NAF 或窗口法, 多基数法都在一定程度上减少了点加的运算次数, 而倍点运算并没有得到明显的改进.

在研究了标量的产生过程后, 本文发现在很多情况下, 随机标量的生成与标量乘运算是由同一角色连续完成的, 而且这个随机标量只是一个临时中间变量, 不需要参与后续的解密运算. 由此, 本文利用算法简单、速度较快的 ALFG 生成随机变量并通过对预先计算出的椭圆曲线点进行循环迭代点加运算得到标量乘的结果.

在标量随机且基点固定的情况下, ALFG 改进算法将传统标量乘的反复点加与倍点运算转化为单一的点加运算, 相比窗口 NAF 算法降低了 60%以上的总体计算量. 并且点加运算的次数, 即 ALFG 的循环迭代次数是一个常量, 并不会因为标量的增长而增大, 从而在位数更高的椭圆曲线标量乘中节省更多的计算量.

参考文献

1 Koblitz N. Elliptic curve cryptosystems. Mathematics of Computation, 1987, 48(177): 203–209.
2 Miller VS. Use of elliptic curves in cryptography. Advances in Cryptology-CRYPTO'85 Proc. Springer Berlin Heidelberg. 1986. 417–426.
3 Menezes AJ. Elliptic curve public key cryptosystems. New York: Springer, 1993: 356–382.
4 Hankerson D, Vanstone S, Menezes AJ. Guide to elliptic curve cryptography. New York: Springer, 2004: 416–489.
5 潘晓君.一种新的基于椭圆曲线的数字签名方案.计算机系统应用,2008,17(1):35–37.
6 刘铎,戴一奇.计算椭圆曲线上多标量乘的快速算法.计算机学报,2008,31(7):1131–1137.
7 殷新春,赵荣,侯红祥,等.基于折半运算的快速双基数标量乘算法.计算机应用,2009,29(5):1285–1288.
8 陈厚友,马传贵.椭圆曲线密码中一种多标量乘算法.软件学报,2011,22(4):782–788.
9 Longa P. Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields. University of Ottawa, 2007.
10 Marsaglia G. A current view of random number generators, Computer Science and Statistics. Sixteenth Symposium on the Interface. Elsevier Science Publishers. North–Holland, Amsterdam. 1985. 3–10.
11 Brent RP. Uniform random number generators for supercomputers. Proc. Fifth Australian Supercomputer Conference. SASC Organizing Committee. 1992. 95–104.
12 Burns PJ. Lagged Fibonacci Random Number Generators. http://lamar.colostate.edu/~grad511/lfg.pdf.