一类安全椭圆曲线的选取及其标量乘法的快速计算

白国强[1,2],周 涛[1],陈弘毅[1]
(1. 清华大学微电子学研究所,北京 100084;2. 西安交通大学数学系,陕西西安 710049)

摘 要: 安全椭圆曲线的选取和标量乘法的快速计算是有效实现椭圆曲线密码体制的两个主要问题. 本文将二者结合起来考虑给出了一类适合普通 PC 机实现的安全椭圆曲线,并详细给出了选取这类曲线的具体步骤和基于"大步-小步法"思想构造了一种新的计算这类曲线上标量乘法的快速算法. 这类曲线不仅选取容易而且利用本文所提出方法计算其标量乘法时能使所需椭圆曲线运算次数大大减少. 此外,选用这类曲线后基域中元素不再需要专门的表示方法,各种运算能非常快地得到实现,从而能极大地提高体制的整体实现速度.

关键词: 椭圆曲线密码; 安全椭圆曲线; 标量乘法; Frobenius 展式
中图分类号: TN918.1 **文献标识码**: A **文章编号**: 0372-2112 (2002) 11-1654-04

A Selection of the Secure Elliptic Curve and Fast Calculation of Scalar Multiplication

BAI Guo-qiang[1,2], ZHOU Tao[1], CHEN Hong-yi[1]
(1. *Institute of Microelectronics, Tsinghua Univ., Beijing* 100084, *China*;
2. *Department of Mathematics, Xi'an Jiaotong Univ. Xi'an, Shaanxi* 710049, *China*)

Abstract: The selection of secure elliptic curves and the scalar multiplications of elliptic curves are two important problems in the practice of efficiently implementing an elliptic curve cryptosystems. In this paper, we study those two problems jointly, give a class of secure elliptic curves mainly based on the computer words, describe a detailed process of how to selecting those curves, and present a new method, which is based on the idea of "baby step-giant step", of computing the scalar multiplication concerning those curves. With the new method, the amount of scalar multiplications based on those curves can be reduced greatly. Besides, when those curves are used, special representation method for the elements in the base field is no longer needed, and all the arithmetic in the field can be quickly accomplished.

Key words: elliptic curve cryptosystem; secure elliptic curves; scalar multiplication; Frobenius expansions

1 引言

椭圆曲线密码[1]是一种基于椭圆曲线离散对数问题的公钥密码,近年来它在解决信息安全问题实际需求的推动下成了密码学中的一个研究热点. 其中尤其是关于有效实现椭圆曲线密码体制的研究,是人们围绕这一课题研究的主要内容. 在有效实现椭圆曲线密码体制的研究中,安全椭圆曲线的选取和标量乘法的快速计算又是其中的两个主要问题. "子域曲线"是最近的研究中受到关注的一类曲线. 对"子域曲线"上的标量乘法,一种基于 Frobenius 展式的计算方法[2~4]受到了人们注意. 然而已有研究基本上都将安全"子域曲线"的选取和 Frobenius 展式下标量乘法的快速计算分开来考虑. 本文我们将二者结合起来考虑,首先给出了一类基于计算机字长的安全椭圆曲线,并详细给出了选取这类曲线的具体步骤. 进一步对这类曲线,基于 Frobenius 方法文中又提出了快速计算其标量乘法的一种新方法. 这类曲线不仅选取容易而且利用文中所提出方法计算其标量乘法时能使所需椭圆曲线运算次数大大减少. 对普通 CPU 而言,选用这类曲线后基域中元素不再需要专门的表示方法,各种运算能非常快地得到实现,从而能极大地提高体制的整体实现速度.

本文余下部分是这样安排的:第 2 节介绍椭圆曲线密码体制及基于 Frobenius 展式下计算标量乘法的快速算法. 第 3 节首先给出了基于计算机字长的一类安全椭圆曲线的选取方法,然后构造了针对这类曲线利用 Frobenius 方法计算其标量乘法时完成最后阶段计算的新方法. 第 4 节是本文的结论部分.

2 椭圆曲线密码体制及其实现

设 p 是一素数, n 是一正整数,记 $q = p^n$,记 F_q 是特征为 p 的有限域. 所谓定义在有限域 F_q 上的椭圆曲线 $E(F_q)$ 是指

收稿日期:2001-12-05;修回日期:2002-03-04
基金项目:中国博士后科学基金(No. 023202001);国家 863 项目(No. 2002AA141040)

下面的方程(1)在 F_q 上的解连同一个特殊元素 O 所组成的集合.

$$y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6, \quad a_i \in F_q \qquad (1)$$

在 $E(F_q)$ 的元素之间有一个自然的群运算法则能使 $E(F_q)$ 构成一个加法群,其中 O 是单位元. 因此,简单说椭圆曲线 $E(F_q)$ 是一个有限加法群. 关于椭圆曲线及椭圆曲线密码的进一步内容请参阅有关文献[1,5,6],这里不再赘述.

现设 $P \in E_q$ 是椭圆曲线 $E(F_q)$ 上的一点,ω 为一整数. 令 $Q = \omega P$. 则椭圆曲线 $E(F_q)$ 上的标量乘法是指由 ω 和 P 对 Q 的计算. 在椭圆曲线密码体制的实现中,标量乘法是关键. 相对于标量乘法所用时间,其它计算所用时间都可以忽略不计. 因此,本文中关于椭圆曲线密码体制的实现将主要考虑对标量乘法的计算.

设 $p > 3$ 是一素数,n 是一正整数,现考虑方程

$$y^2 = x^3 + Ax + B, (\text{其中}, 4A^3 + 27B^2 \neq 0, A, B \in F_p) \qquad (2)$$

在 F_p 的 n 次扩域 F_{p^n} 上的解连同单位元"O"所构成的椭圆曲线. 这种曲线习惯上称为"子域曲线". 对这类曲线,近年来发展了一种新的计算标量乘法的快速算法[2,3,4,7],即基于 Frobenius 展式下的快速算法. 下面我们对这一算法先作一简要介绍.

用 φ 表示方程(2)在 F_p 的代数闭域 $\bar{F}_p$ 上的 Frobenius 自同态,即 φ 表示下列映射:

$$\varphi: \quad (x, y) \to (x^p, y^p)$$
$$O \to O$$

其中,$x, y \in \bar{F}_p$ 且满足方程(2). 则 φ 满足如下的同态方程

$$\varphi^2 - c\varphi + p = 0$$

其中 $|c| \leq 2\sqrt{p}$ 是 Frobenius 同态 φ 的迹,并且有 $\#E(F_p) = p + 1 - c$. 这里 $\#E(F_p)$ 表示曲线 $E(F_p)$ 上点的个数.

设 $P = (x, y) \in E(F_{p^n})$,则有

$$\varphi(P) = (x^p, y^p)$$
$$\varphi^2(P) = (x^{p^2}, y^{p^2})$$
$$\cdots\cdots$$
$$\varphi^n(P) = (x^{p^n}, y^{p^n}) = (x, y)$$

由此可见在 F_p 中,当采用正规基表示域中元素时,对上述各式子右边各个分量的计算是非常容易的.

为计算椭圆曲线标量乘法 ωP,Frobenius 展式下的这一方法首先将 ω 在式(2)的自同态环中把 ω 展开为

$$\omega = a_0 + a_1\varphi + a_2\varphi^2 + \cdots + a_t\varphi^t, (\text{其中}, |a_i| \leq (p-1)/2) \qquad (3)$$

然后按照下式计算 ωP:

$$\omega P = a_0 P + a_1\varphi(P) + a_2\varphi^2(P) + \cdots + a_t\varphi^t(P) \qquad (4)$$

因为 $|a_i| \leq (p-1)/2$, $t \leq 2n + 1$[2] 以及在正规基表示下对 $\varphi(P)$, $\varphi^2(P)$, ……, $\varphi^t(P)$ 的计算所用时间可以忽略不计,所以用这一方法能极大地提高对 ωP 的计算速度.

但是,采用上述方法计算 ωP 时,还需要解决以下一些问题:

(1)如何求出 ω 的展开式(3)或 ωP 的展开式(4);

(2) t 的最小值是多少;

(3)如何由式(4)求出 ωP.

本文我们主要考虑第三个问题,即如何由式(4)求出 ωP.

文[3]讨论了如何求出 ω 的展开式(4)以及 t 的最小值问题. 对适当大的 p 值,文[3]证明了对定义在 F_p 上的椭圆曲线(2),点 $P \in E(F_{p^n})$ 以及正整数 ω,有

$$\omega P = a_0 P + a_1\varphi(P) + a_2\varphi^2(P) + \cdots + a_t\varphi^t(P) \qquad (5)$$

其中 $|a_i| \leq (p-1)/2$, $i = 0, 1, \cdots, t$, $t \leq n + 2$. 进一步,文[4]还给出了计算展式(5)的一种算法. 但是如何有效计算式(5)的右边从而最终计算出 ωP 目前仍是这一方法是否有效的关键. 本文在假定已有式(5)的情况下,构造了在两种不同情况下计算式(5)的有效方法.

3　一类安全椭圆曲线的选取及其标量乘法的计算

3.1　基本假定

为了能够较具体地给出这类曲线,我们作以下两个基本假定

(1)根据安全性要求,假定对即将要给出的椭圆曲线,其阶的素因子分解式中至少应包含一个 160 比特位以上的因子. 就目前的计算能力和对椭圆曲线离散对数问题的求解方法而言,160 比特位已足够安全了. 此外,160 比特也是目前公认的一个标准[7].

(2)假定普通 PC 机 CPU 的字长是 32 位的.

3.2　曲线的选取

根据以上假定,本文给出的曲线将是定义在 F_p 上的"子域曲线" $E(F_{p^n})$. 因为 $|\#E(F_{p^n}) - p^n| \leq 2\sqrt{p^n}$,我们有 $\#E(F_{p^n}) = O(p^n)$,并且下面我们将看到 $p \approx 2^{32}$,因此根据上述第一个假定这里可取 $n = 7$. 于是,对这类曲线可按如下办法选取. 其中主要的算法来自文献[1,5]等.

(1)选取一个形如 $p = 2^{32} - c$ 的素数. 其中 c 是一个尽可能小的非负整数. 因为 $p < 2^{32}$,这样的素数 p 采用搜索的办法是非常容易找到的.

(2)随机选取一条定义在 F_p 上的椭圆曲线 E 如下:

$$y^2 = x^3 + Ax + B, 4A^3 + 27B^2 \neq 0, A, B \in F_p \qquad (6)$$

(3)求出 $\#E(F_p)$. 因为 p 比较小,所以对 $\#E(F_p)$ 的计算可采用如下公式[5]直接求出:

$$\#E(F_p) = p + 1 + \sum_{x=0}^{p-1}\left(\frac{x^3 + Ax + B}{p}\right).$$

这里 (x/p) 表示 Legendre 符号. 以下为了方便,记 $N_p = \#E(F_p)$.

(4)取 $n = 7$,并计算 $\#E(F_{p^n})$. 因为 N_p 已经求出,所以利用文[1]等指出的方法可以很容易地计算出 $\#E(F_{p^n})$. 具体地,令 $c_0 = 2$, $c_1 = p + 1 - N_p$, $c_k = c_1 c_{n-1} - pc_{n-2}$,则

$$\#E(F_{p^n}) = p^n + 1 - c_n$$

其中,$n = 7$.

(5)分解 $\#E(F_{p^n})$. 如果 $\#E(F_{p^n})$ 的素因子分解式中包含一个 160 比特位以上的素因子,则选定 $\#E(F_{p^n})$. 否则,更换方程(6)中系数 A, B,并继续上面的(3)和(4),直到找到合适的 $\#E(F_{p^n})$ 为止.

$\#E(F_p^n)$ 是方程(6)中系数 A, B 的函数. 如何选取 A, B 才能使 $\#E(F_p^n)$ 中包含一个大素因子的可能性更大目前尚无任何结果. 注意到 $\#E(F_p^n) = O(2^{224})$, 利用现有的大数分解算法和计算技术对如此规模的 $\#E(F_p^n)$, 分解它不会有困难, 但具体的分解仍会比较复杂. 上面最后一步中, 因为 $E(F_p) \subset E(F_p^n)$ 是 $E(F_p^n)$ 的子群, 所以 $\#E(F_p^n)$ 中自然地会包含一个大小约为 $O(p) = O(2^{224})$ 的因子. 这一因子的存在不会妨碍 $\#E(F_p^n)$ 中仍可能包含一个 160 比特位以上素因子存在的可能性. 这是因为, $\#E(F_p^n) = O(p^n) = O(2^{224})$, 在 $\#E(F_p^n)$ 中去掉一个 32 位的因子后, $\#E(F_p^n)$ 中的大素因子最大仍可达到 $O(2^{192})$. 所以这时, 在 $\#E(F_p^n)$ 中存在一个 160 比特位以上素因子的可能性仍不会太小.

以下假定已选定了这样的一条曲线 E, 并假定 P 是利用 E 构造椭圆曲线密码时的基点.

3.3 标量乘法的快速计算

对任一整数 $\omega \in [1, \#E(F_p^n)]$, 标量乘法要求计算 ωP. 如前所述, 对 ωP, 本文假定已有式(5)存在. 下面分两种情况给出计算 ωP 的具体办法.

(1) 固定基点的情况. 这时 P 是系统参数, 为此可先作一些预计算. 记 $N = \sqrt{p/2}$, 对 $k = 1, 2, \cdots, N$, 分别计算 kP 并将其列表保存. 记这一表格为 T.

(a) 将 $a_i (i = 0, 1, \cdots, t)$ 表示为

$$a_i = b_i N + c_i$$

(b) 查表格 T 分别求出 $b_0 P, b_1 P, \cdots, b_t P$ 和 $c_0 P, c_1 P, \cdots, c_t P$, 并计算

$$Q = b_0 P + \phi(b_1 P) + \phi^2(b_2 P) + \cdots + \phi^t(b_t P)$$

$$R = c_0 P + \phi(c_1 P) + \phi^2(c_2 P) + \cdots + \phi^t(c_t P)$$

(c) 注意到

$$\begin{aligned}\omega P &= a_0 P + a_1\phi(P) + a_2\phi^2(P) + \cdots + a_t\phi^t(P)\\ &= a_0 P + \phi(a_1 P) + \phi^2(a_2 P) + \cdots + \phi^t(a_t P)\\ &= (b_0 N + c_0) P + \phi((b_1 N + c_1) P) + \cdots + \phi^t((b_t N + c_t) P)\\ &= (b_0 N + c_0) P + (N\phi(b_1 P) + \phi(c_1 P)) + \cdots\\ &\quad + (N\phi^t(b_t P) + \phi^t(c_t P))\\ &= NQ + R\end{aligned}$$

最后计算 $NQ + R$ 即可求出 ωP.

虽然 N 是一个固定值, 但因 Q 是一个随机点, 对上面最后一步中 NQ 的计算仍需当作普通标量乘法对待. $n = 7$ 当时, 因为 $t \le n + 2 = 9$, 所以在上述计算中, 对 Q 和 R 的计算各需要 8 次点加运算. 注意到 $N = \sqrt{p/2} < 2^{16}$, 对 NQ 的计算最多需要 16 次倍点运算和 15 次点加运算. 于是按照上述方法借助于预计算表格 T 对 ωP 的计算最多需要 48 次椭圆曲线中的点加运算或倍点运算. 如果选用其它类曲线并采用常规方法计算标量乘法时, 对 160 比特位的密钥, 一般至少需 $\frac{4}{3} \times 160 = 214$ 次椭圆曲线中的点加运算或倍点运算[1]. 由此可见这种方法能极大地减少计算 ωP 时所需椭圆曲线中的运算次数. 这种方法我们主要根据"大步小步法"(baby step-giant step)的思想构造.

在上述方法中, 注意 $N = \sqrt{p/2} < 2^{16} = 65536$, 所以表格 T 非常小. 生成和存储表格 T 都不会有实际困难, 查表时间可以忽略不计.

(2) 随机基点的情况. 这时基点 P 是在体制实现过程中随机产生的, 不再是系统参数, 上述方法不能适用于这种情况. 现记 $P_0 = P$, $P_1 = \phi(P)$, $P_2 = \phi^2(P)$, $\cdots$, $P_t = \phi^t(P)$, 则式(5)可写为

$$\omega P = a_0 P_0 + a_1 P_1 + a_2 P_2 + \cdots a_t P_t \tag{7}$$

令 $a_0, a_1, a_2, \cdots, a_t$ 的不相连扩展二进制表示[11]分别为

$$\begin{aligned}a_0 &= d_{00}2^r + d_{01}2^{r-1} + \cdots + d_{0r}\\ a_1 &= d_{10}2^r + d_{11}2^{r-1} + \cdots + d_{1r}\\ &\cdots\cdots\\ a_t &= d_{t0}2^r + d_{t1}2^{r-1} + \cdots + d_{tr}\end{aligned}$$

其中, $d_{ij} \in \{-1, 0, 1\}$. 则,

$$\begin{aligned}\omega P &= a_0 P_0 + a_1 P_1 + a_2 P_2 + \cdots + a_t P_t\\ &= \sum_{i=0}^{t} a_i P_i = \sum_{i=0}^{t}\sum_{j=0}^{r}(d_{ij}2^{r-j}) P_i = \sum_{j=0}^{r} 2^{r-j}\sum_{i=0}^{t} d_{ij} P_i\\ &= 2\left(\cdots 2\left(2\left(2\sum_{i=0}^{t} d_{i0} P_i + \sum_{i=0}^{t} d_{i1} P_i\right)\right.\right.\\ &\quad \left.\left. + \sum_{i=0}^{t} d_{i2} P_i\right) + \sum_{i=0}^{t} d_{i3} P_i\right)\cdots\Bigg) + \sum_{i=0}^{t} d_{ir} P_i\Bigg)\end{aligned} \tag{8}$$

由此可得这种情况下计算 ωP 的算法如下.

Frobenius 展式下计算 ωP 的算法(P 为随机点的情况)

输入: P, ω; $a_0, a_1, a_2, \cdots, a_t$.

输出: ωP.

(a) 计算 $a_0, a_1, a_2, \cdots, a_t$ 的不相连扩展二进制表示.

设 a_i 的不相连扩展二进制表示为 $(d_{i0}, d_{i1}, \cdots, d_{ir})$.

(b) $j \leftarrow 0$, $Q \leftarrow 2\sum_{i=0}^{t} d_{ij} P_i$.

(c) If $j \le r - 1$ do;

$j \leftarrow j + 1$;

$Q \leftarrow Q + \sum_{i=0}^{t} d_{ij} P_i$;

$Q \leftarrow 2Q$;

(d) End if, output $Q = \omega P$.

在这一算法中, 当 $d_{ij} = -1$ 时需要计算 $(-P_i)$. 由椭圆曲线的性质, 由 P_i 计算 $(-P_i)$ 的时间可以忽略不计. 因此, $d_{ij} = -1$ 不会增加算法的复杂度.

下面对这一算法需要的点加运算和倍点运算作一简单分析. 记

$$D = \begin{bmatrix} d_{00} & d_{01} & \cdots & d_{0r}\\ d_{10} & d_{11} & \cdots & d_{1r}\\ d_{t0} & d_{t1} & \cdots & d_{tr}\end{bmatrix}$$

由扩展二进制表示的性质, 对每个 a_i 的扩展二进制表示 $a_i = d_{i0}2^r + d_{i1}2^{r-1} + \cdots + d_{ir}$, 向量 $(d_{i0}, d_{i1}, \cdots, d_{ir})$ 中非零分量的个数[8]平均是 $\frac{1}{3}(r+1)$. 于是矩阵 D 中非零元素的个数平均是 $\frac{1}{3}(r+1)t$. 因此由(8)可见, 上述算法共需进行 r 次倍点

运算和 $\frac{1}{3}(r+1)t - t = \frac{1}{3}rt - \frac{2}{3}t$ 次点加运算.

由式(5),$|a_i| \leq \frac{p-1}{2} < \frac{1}{2}p < 2^{31}$, $i = 0, 1, \cdots, t$. 所以对上述 r 显然有 $r \leq 32$. 于是再由 $n = 7$, $t \leq n + 2 = 9$, 与固定基点的情况类似不难得出,这时计算 ωP 最多约需

$$r + \frac{1}{3}r(n+2) - \frac{2}{3}(n+2)$$

$$= 32 + \frac{1}{3} \times 32 \times (7+2) - \frac{2}{3} \times (7+2) = 122$$

次椭圆曲线中的点加运算或倍点运算. 这一次数仅是上面 214 次的 57 %.

对本节所给出的曲线,虽然有限域 F_{p^n} 上的元素仍需要在有限域 F_p 的基础上采用多项式基或正规基表示,但是对有限域 F_p 中的元素,如果使用 32 位字长的普通 PC 机实现这类曲线上的椭圆曲线密码时, F_p 中的元素不再需要专门的表示方法. 这在一定程度上能够加快 F_{p^n} 中的运算. 此外,采用软件方式实现时,生成和存储前述表格 T 都不会有实际困难. 因此,本文所给出的曲线非常适合用普通 PC 机采用软件方式实现.

4　结论

本文给出了一类适合普通 PC 机采用软件方式实现的安全椭圆曲线,并详细给出了选取这类曲线的具体步骤和这类曲线上标量乘法的具体计算方法. 选用这类曲线不仅能使计算标量乘法所需椭圆曲线中的运算次数大大减少,而且还有这样两个特点:(1) 基域中元素不再需要专门的表示方法,元素之间的运算可通过计算机直接进行, F_{p^n} 中的运算能非常快地得到实现,从而能极大地提高体制的整体实现速度;(2) 曲线的可选空间比较大. F_p 上总共大约有 p^2 条曲线,在这些曲线中符合安全性要求的曲线会很丰富. 这一特点克服了 Koblitz 曲线数量稀少的弱点,能很好地满足实际中一定场合的需求. 因此,与其它类曲线相比这类曲线不仅选取容易,而且具有很高的实现速度.

参考文献:

[1] I Blake, G Seroussi, N Smart. Elliptic Curves in Cryptography[M]. Cambridge, United Kingdom: Cambridge University Press, 1999. 2 - 10.

[2] W Meier, O Staffelbach. Efficient Multiplication on Certain Nonsupersingular Elliptic Curves[A]. Advances in Cryptology-crypto '92, LNCS 740, Springer-Verlag[C]. Berlin. 1992, 333 - 344.

[3] N Smart. Elliptic curve cryptosystems over small fields of odd characteristic[J]. Journal of Cryptology, 1999, 12: 141 - 151.

[4] D Bailey, C Paar. Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms[A]. Advances in Cryptology-Crypto '98, LNCS 1462, Springer-Verlag[C]. Berlin: 1998. 472 - 485.

[5] J Silverman. The Arithmetic of Elliptic Curves[M]. New York: GTM 106, Springer-Verlag, 1986. 45 - 63.

[6] Guoqiang Bai, Yupu HU, Guozhen Xiao. Method of improving an elliptic curve cryptosystem over the ring zn[J]. Chinese Journal of Electronics. 2000, 19(1): 89 - 91.

[7] ANSI-X9.63-1998, Public Key Cryptography for Financial Service Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography[S].

[8] J Solinas. Improved algorithm for arithmetic on anomalous binary curves[R]. Canada: CACR of University of Waterloo, 2000.

作者简介:

白国强　男,1963 年 9 月生于陕西清涧,2000 年 12 月获西安电子科技大学密码学博士学位,现为清华大学微电子学研究所博士后,主要研究领域包括信息论、编码与密码学,发表论文二十余篇,近期感兴趣的研究领域为椭圆曲线密码的实现.

周　涛　男,1976 年 10 月生于江苏东台,1998 年 9 月获清华大学学士学位,并且开始在清华大学微电子学研究所攻读博士学位,研究方向为超大规模集成电路设计和密码算法的 VLSI 实现.

陈弘毅　男,1942 年 9 月出生于重庆市,教授,博士生导师,中国电子学会高级会员,清华大学微电子学研究所所长,近年来感兴趣的研究领域为密码算法的 VLSI 实现与系统的芯片集成,已完成多种密码芯片的设计,发表论文九十余篇.