双基数链算法计算 Tate 对的一种改进

翁 江,豆允旗,马传贵
(信息工程大学信息工程学院,河南郑州 450002)

摘 要: 双线性对在基于身份的密码体制中有着广泛的应用.Miller 算法是计算双线性对的核心算法.本文在双基数链计算 Tate 对的基础上给出了一种高效的 Miller 算法.通过范函数和共轭技巧的应用,减少了 Miller 算法中有理函数直线和垂线的数量并用共轭代替了求逆运算.结果表明新算法与已有算法相比效率提高了 10%以上.

关键词: 双基数链; Miller 算法; Tate 对; 椭圆曲线
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112 (2012)09-1775-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2012.09.012

Refinements of Double-Base Chains Algorithm for Computing Tate Pairing

WENG Jiang, DOU Yun-qi, MA Chuan-gui
(*Institute of Information Engineering, Information Engineering University, Zhengzhou, Henan* 450002, *China*)

Abstract: Pairings have been widely used in the study of identity-based cryptosystems (IBC). Miller algorithm is the key of computing pairings. We propose an efficient Miller algorithm computing Tate pairing based on the double-base chain. Through the application of norm function and conjugate technique, our refinements reduce the total number of lines and vertical lines in the rational function, and replace the inverse by its conjugate in Miller algorithm. Results show that the efficiency of our algorithm can be improved by more than 10% compared with the previous method.

Key words: double-base chains; Miller algorithm; Tate pairing; elliptic curve

1 引言

近些年来,基于椭圆曲线构造的双线性对在公钥密码学中得到了广泛的应用.椭圆曲线上的双线性对早期被用来攻击椭圆曲线有理点群上的离散对数问题.1991 年 Menezes, Okamato 和 Vanstone[1] 基于双线性 Weil 对将椭圆曲线上的离散对数问题规约到有限域上的离散对数问题,即 MOV 约化.1999 年 Frey 和 Ruck[2] 利用双线性 Tate 对给出了 FR 约化.这种规约使得某些椭圆曲线有理点群中的离散对数问题变得相对容易.

1984 年,Shamir[3] 首次提出了基于身份加密体制的思想,该方法在密码学中有着广泛的应用背景,但是直到 2000 年 Sakai, Ohgish, Kasahara[4] 和 Joux[5] 才借助双线性对分别给出该类体制的具体实现方案.此后,双线性对被用来构造各种各样的密码协议,比如基于身份的加密体制[6],短签名[7,8] 等.由此双线性对引起了密码学者们的极大兴趣,也成为近年来公钥密码体制研究的热点之一.

由于双线性对作为一个基本工具在协议设计中取得了广泛应用,为了使这些协议具有实用性,就必然要求提高双线性对的计算效率.计算双线性对的关键在于计算有理函数在一些特定除子或者有理点的赋值.最普通的想法是首先给出有理函数的确定形式,然后将有理点的坐标带入有理函数.当有理函数的次数较小时,这样做是可行的.但是如果有理函数的次数非常大(比如在目前的安全强度下,相应的次数的数量级为 2^{160} 大小),显然此时要给出有理函数的确定形式难度比较大,故要寻求别的多项式时间算法.

目前,基于椭圆曲线构造的双线性对有两种算法:一种是 Miller[9] 算法,另一种是椭圆网[10] 算法.相对椭圆网算法,Miller 算法以其效率高,应用范围广的特点,引起了许多学者的关注.因此目前关于双线性对快速计算的研究主要集中在 Miller 算法的改进上.Miller 算法的本质就是通过有理函数倍乘和加法的反复迭代,得到有理函数 f_i 在某一除子的赋值.迭代的次数和每次迭代运算的复杂度是整个 Miller 算法耗时的主要部分.

收稿日期:2011-09-20;修回日期:2012-02-15
基金项目:国家高技术研究发展计划(863 计划)课题(No.2009AA01Z417)

传统的 Miller 算法采用二进制表示来计算双线性对;若采用双基数链表示,利用其表示稀疏的优势可以减少椭圆曲线群中的点加运算和相应有理函数的点加运算,从而可以提高双线性对的计算效率.2008 年赵昌安[11]等人首次将双基数链用于双线性对的计算中,与一些经典的算法相比使得计算效率提高了大约 9% ~ 38%.2011 年陈厚友[12]等人进一步研究了 Tate 对快速计算,又将运算效率提高了约 10.6% ~ 20.3%.

另外许多学者从缩短 Miller 循环的角度构造新的双线性对来加速计算.如:Barreto[13]提出的适用于超奇异曲线计算的 Eta 对,Hess[14]等人推广 Eta 对到一般的椭圆曲线上,提出了著名的 Ate 对,以及 Vercauteren[15]提出的 Optimal 对.本质上都是 Tate 对的变形,故 Miller 算法同样可用于这些双线性对计算.

本文在双基数链算法计算 Tate 对的基础上,提出了合并连续相邻项的方法,利用范函数减少 Miller 迭代中有理函数直线和垂线的数量,结合共轭技巧将求逆转化为扩域中共轭元的乘法,优化了 Miller 算法,提高了计算效率.新算法不仅能应用于超奇异椭圆曲线上而且能应用于一般的椭圆曲线上,结果表明本文提出的新算法比现有的算法[11,12]效率至少高 10%以上.

2 背景知识

2.1 双基数链简介

首先我们引进双基数链表示,接着将其应用于双线性对的计算中.双基数链[16]是整数的一种表示方法,它能够大大减少整数表示中所需要的加号和减号.Dimitrov[17]将其用于加速椭圆曲线上点的标量乘快速计算,赵昌安等[11]将这样的表示推广到加速双线性对的计算.本文在文献[11,12]的基础上进一步改进了基于双基数链计算双线性对的算法.

任意一个整数 n 都可表示成基于{2,3}幂次(即形如 2^a3^b 的整数)的和,如下式:

$$l=\sum_{i=1}^{m} k_i 2^{a_i}3^{b_i},$$

表 1 文献[17]不同正整数双基数链表示时的参数

$a_{\max}$	$b_{\max}$	m	$a_{\max}$	$b_{\max}$	m
57	65	45	95	41	37
76	53	38	103	36	39

其中 $k_i\in\{-1,1\}$, $a_1\geqslant a_2\geqslant\cdots\geqslant a_m\geqslant 0$, $b_1\geqslant b_2\geqslant\cdots\geqslant b_m\geqslant 0$.

由于任意整数基于双基数链的表示并不唯一,Dimitrov 等人给出了一个有效的贪心算法,使得任意的正整数 l 通过此算法都可以用双基数链较为稀疏的表示.假设 $a_{\max}$ 和 $b_{\max}$ 分别表示 2 和 3 在双基数链中的最高幂次,m 表示在随机选取的 10000 个 l 中,l 展开式中 2,3 最高幂次相同的双基数的个数.对于任意 160 比特的素数,文献[17]给出了一些统计规律,为了方便后面的比较,我们引用这些数据.

2.2 双线性 Tate 对简介

设 $q=p^m$,其中 p 为素数,m 为一正整数.E 是有限域 F_q 上的椭圆曲线.E 的方程定义为

$$E: y^2=x^3+ax+b,$$

其中点 ∞ 记作曲线 E 的无穷远点,$a,b\in F_q$ 且 $4a^3+27b^2\neq 0$.

设 l 为一大素数且满足 $(l,q)=1$.取椭圆曲线 $E(F_q)$中的点 P,它的阶为 l,且 $l\nmid(q-1)$,k 是 E 的嵌入次数,即满足条件 $l\mid(q^k-1)$的最小正整数.椭圆曲线 E 的 l – 挠群定义为 $E[l]$.

传统的双线性 Tate 对 e_l 定义为如下的非退化双线性映射:

$$e_l: E(F_{q^k})[l]\times E(F_{q^k})/lE(F_{q^k})\to F_{q^k}^*/(F_{q^k}^*)^l,$$
$$e_l(P,Q)=f_l(D_Q).$$

由上面的定义可知,计算传统的双线性 Tate 对结果为陪集值,即对相同的 P 和 Q,计算 $e(P,Q)$可能得到不同的数值,但这些数值属于同一个陪集.而在密码学应用中要求在双线性映射之后得到唯一值,因此有如下约化的双线性 Tate 对定义:

$$\hat{e}_l: E(F_{q^k})\times E(F_{q^k})/lE(F_{q^k})\to\mu_l,$$
$$\hat{e}_l(P,Q)=f_l(D_Q)^{q^k-1/l},$$

其中 μ_l 为 $F_{q^k}^*$中的 l 次单位根群,即 $\mu_l=\{\mu\in F_{q^k}^*\mid\mu^l=1\}$.

如果限制 $P\in E(F_q)$,Barreto[19]定义如下变形的约化双线性 Tate 对:

$$\tilde{e}_l: E(F_q)[l]\times E(F_{q^k})/lE(F_{q^k})\to\mu_l,$$
$$\tilde{e}_l(P,Q)=f_l(Q)^{q^k-1/l}.$$

Barreto[19]等人的工作表明在限定 $P\in E(F_q)$条件下,有理函数 f_l 在除子 D_Q 的赋值可以用有理函数 $f_{l,P}$ 在点 Q 的赋值来替代,这样可以使传统的 Miller 算法得到优化.

由于 Weil 对相当于两次 Tate 对的计算,而最近几年提出的 Eta 对,Ate 对和 Optimal 对本质上都是 Tate 对的变形,故本文中只讨论最典型的双线性 Tate 对.

应用在密码学中的双线性 Tate 对需要满足下面的性质:

(1)双线性:对任意的 $P\in E(F_q)[l]$, $Q\in E(F_{q^k})$和 $n\in\mathbb{Z}$,有 $e_l(nP,Q)=e_l(P,nQ)=e_l(P,Q)^n$.

(2)非退化性:一定存在某个 $P\in E(F_q)[l]$和 $Q\in E(F_{q^k})$满足 $e_l(P,Q)\neq 1$.

目前存在很多基于双线性对构造的密码协议,为了保证这些方案的安全性,一般要求 G_1, G_2 和 G_T 这三个群中的离散对数问题是难解的.具体到双线性 Tate 对定义,若取 $G_1 = E(F_q)[l]$, $G_2 = E(F_{q^k})[l]$和 $G_T = \mu_l$,要求 l 至少是 160 比特大小,对应嵌入扩域中的 q^k 至少是 1024 比特大小.

2.3 除子

定义 1 设 E 是定义在有限域 F_q 上的椭圆曲线.对任意的点 $P\in E(F_q)$,定义形式化符号$\langle P\rangle$.则 E 上的除子 div(E)定义为如下的形式和:

$$\mathrm{div}(E) = \sum_{P_j\in E} n_j\langle P_j\rangle, n_j\in\mathbb{Z},$$

其中仅有有限多个 $n_j\neq 0$. div(E)被称为 E 的除子群,D 为 E 的除子,D 的次数定义为 $\deg(D) = \sum_{P_j\in E} n_j$. div($E$)是由 E 上的点所生成的自由交换群.

除子与函数

设 f 是 E 上的非零有理函数,其除子定义为

$$\mathrm{div}(f) = \sum_{P\in E} \mathrm{ord}_P(f)\langle P\rangle,$$

其中 $\mathrm{ord}_P(f)$是函数 f 在零点或者极点的阶.

对于任意的定义在 E 上的非零有理函数f 和g 有

$$\mathrm{div}(fg) = \mathrm{div}(f) + \mathrm{div}(g),$$

$$\mathrm{div}\left(\frac{f}{g}\right) = \mathrm{div}(f) - \mathrm{div}(g).$$

2.4 Miller 算法

Miller算法的核心是构造 F_{q^k}上的有理函数f_l,其除子满足 $\mathrm{div}(f_l) = l\langle P\rangle - \langle lP\rangle - (l-1)\langle\infty\rangle$,其中 $P\in E(F_q)$.

设过点 iP 和 jP 的直线为 $l_{iP,jP}$,当 $i = j$, $l_{iP,iP}$表示过点 iP 的切线;用 $v_{(i+j)P}$表示过点$(i+j)P$ 的垂线,当 $i = j$ 时,v_{2iP}表示过 $2iP$ 的垂线.

对于任意的 $i,j\in\mathbb{Z}$,有

$$\begin{aligned}\mathrm{div}(l_{iP,jP}) &= \langle iP\rangle + \langle jP\rangle + \langle -(i+j)P\rangle - 3\langle\infty\rangle,\\ \mathrm{div}(v_{(i+j)P}) &= \langle (i+j)P\rangle + \langle -(i+j)P\rangle - 2\langle\infty\rangle.\end{aligned} \tag{1}$$

由式(1)可得

$$\mathrm{div}(f_{i+j}) = \mathrm{div}(f_i) + \mathrm{div}(f_j) + \mathrm{div}(l_{iP,jP}) - \mathrm{div}(v_{(i+j)P}), \tag{2}$$

因此 $f_{i+j} = f_i\cdot f_j\dfrac{l_{iP,jP}}{v_{(i+j)P}}$.

算法 1 (**Miller 算法**)

输入:素数 $l = \sum_{i=0}^{t} k_i 2^i$,其中 $k_i\in\{0,1\}$, $k_t = 1$,点 $P\in E(F_q)[l]$和 $Q\in E(F_{q^k})$.

输出:$f = f_l(Q)$.

[1] $f\leftarrow 1, T\leftarrow P$;

[2] for $i\leftarrow t-1$ to 0 do

[2.1] $f\leftarrow f^2\cdot\dfrac{l_{T,T}(Q)}{v_{2T}(Q)}$;

$T\leftarrow 2T$;

[2.2] If $k_i = 1$ do

$f\leftarrow f\cdot\dfrac{l_{T,P}(Q)}{v_{T+P}(Q)}$;

$T\leftarrow T+P$;

End if

End for

[3] Return $f^{(q^k-1)/l}$.

3 范函数及其应用

本章主要介绍范函数的性质和共轭技巧,它们将在快速计算双线性对和复杂度评估中发挥重要的作用.

3.1 范函数性质

引理 1[20] 设椭圆曲线 E 上过点 $P = (a,b)$, $Q = (c,d)$和 $-(P+Q)$的直线为 $l(x,y) = 0$,其中 $P+Q = (\alpha,\beta)$,则 $N_{K(x,y)/K(x)}(l) = l(x,y)\bar{l}(x,y) = -(x-a)(x-c)(x-\alpha)$,其中 $\bar{l}(x,y)$是 $l(x,y)$的共轭.对于任意的 $R\in E$, $l(R) = \overline{l(-R)}$, $N_{K(x,y)/K(x)}(l)$称作范函数.

进一步我们给出两个重要的引理,它们将在算法 2 的计算中发挥重要的作用.

引理 2 如果 $\mathrm{div}(f_i) = i\langle P\rangle - \langle iP\rangle - (i-1)\langle\infty\rangle$,则 $f_{i-1} = f_i\cdot\dfrac{1}{v_P}\cdot\dfrac{l_{iP,-P}}{v_{(i-1)P}}$.

证明:因为

$$\begin{aligned}&\mathrm{div}\left(f_i\cdot\frac{1}{v_P}\cdot\frac{l_{iP,-P}}{v_{(i-1)P}}\right)\\ &= i\langle P\rangle - \langle iP\rangle - (i-1)\langle\infty\rangle - [\langle P\rangle + \langle -P\rangle - 2\langle\infty\rangle]\\ &\quad + [\langle iP\rangle + \langle -P\rangle + \langle -(i-1)P\rangle - 3\langle\infty\rangle] - [\langle (i-1)P\rangle + \langle -(i-1)P\rangle - 2\langle\infty\rangle]\\ &= (i-1)\langle P\rangle - \langle -(i-1)P\rangle - (i-2)\langle\infty\rangle,\end{aligned}$$

所以 $f_{i-1} = f_i\cdot\dfrac{1}{v_P}\cdot\dfrac{l_{iP,-P}}{v_{(i-1)P}}$.

引理 3 设 $T\in E[n]$且 $Q\neq T, 2T, \cdots, nT$ 则

$$\frac{l_{T,T}(Q)l_{2T,2T}(Q)}{v_{2T}(Q)v_{3T}(Q)} = -\frac{l_{T,T}(Q)v_T(Q)}{l_{T,2T}(-Q)} \tag{3}$$

$$\left(\frac{l_{T,T}(Q)}{v_{2T}(Q)}\right)^2\frac{l_{2T,2T}(Q)}{v_{4T}(Q)} = -\frac{l_{T,T}^2(Q)}{l_{2T,2T}(-Q)} \tag{4}$$

$$\frac{l_{T,T}(Q)l_{2T,P}(Q)}{v_{2T}(Q)v_{2T+P}(Q)} = -\frac{l_{T,T}(Q)v_P(Q)}{l_{2T,P}(-Q)} \tag{5}$$

$$\frac{l_{T,T}(Q)l_{2T,-P}(Q)}{v_{2T}(Q)v_{2T-P}(Q)} = -\frac{l_{T,T}(Q)v_{-P}(Q)}{v_P(Q)l_{2T,-P}(-Q)} \tag{6}$$

证明:由引理 1 知:

$$\frac{l_{T,T}(Q)l_{T,2T}(Q)}{v_{2T}(Q)v_{3T}(Q)}=\frac{l_{T,T}(Q)l_{T,2T}(Q)l_{T,2T}(-Q)}{v_{2T}(Q)v_{3T}(Q)l_{T,2T}(-Q)}$$
$$=\frac{l_{T,T}(Q)N_{K(x,y)/K(x)}(l_{T,2T})(Q)}{(x_Q-x_{2T})(x_Q-x_{3T})l_{T,2T}(-Q)}$$
$$=-\frac{l_{T,T}(Q)v_T(Q)}{l_{T,2T}(-Q)}$$
$$(\frac{l_{T,T}(Q)}{v_{2T}(Q)})^2\frac{l_{2T,2T}(Q)}{v_{4T}(Q)}=\frac{l_{T,T}^2(Q)l_{2T,2T}(Q)l_{2T,2T}(-Q)}{v_{2T}^2(Q)v_{4T}(Q)l_{2T,2T}(-Q)}$$
$$=\frac{l_{T,T}^2(Q)l_{2T,2T}(Q)N_{K(x,y)/K(x)}(l_{2T,2T})(Q)}{(x_Q-x_{2T})^2(x_Q-x_{4T})l_{2T,2T}(-Q)}$$
$$=-\frac{l_{T,T}^2(Q)}{l_{2T,2T}(-Q)}$$
$$\frac{l_{T,T}(Q)l_{2T,P}(Q)}{v_{2T}(Q)v_{2T+P}(Q)}=\frac{l_{T,T}(Q)l_{2T,P}(Q)l_{2T,P}(-Q)}{v_{2T}(Q)v_{2T+P}(Q)l_{2T,P}(-Q)}$$
$$=\frac{l_{T,T}(Q)N_{K(x,y)/K(x)}(l_{2T,P})(Q)}{(x_Q-x_{2T})(x_Q-x_{2T+P})l_{2T,P}(-Q)}$$
$$=-\frac{l_{T,T}(Q)v_P(Q)}{l_{2T,P}(-Q)}$$

如式(5)证明,同理可证式(6).

为了减少有理函数分子分母中直线和垂线的数量,在算法 2 我们中提出了合并连续相邻两项的方法,其主要基于引理 3 中这 4 个公式.

3.2 共轭与求逆

如果椭圆曲线 E 的嵌入次数 k 是偶数,设 $k=2d$,即 F_{q^k}是 F_{q^d}的二次扩域.那么 F_{q^k}中的元素 ω 可以表示为 $\omega=a+bi$,其中 $a,b\in F_{q^d}$,i 是二次非剩余且 $\delta=i^2$.那么 ω 在 F_{q^d}中的共轭元为 $\bar{\omega}=\overline{(a+bi)}=a-bi$.如果 $\omega\neq 0$,则 $\frac{1}{\omega}=\frac{\bar{\omega}}{a^2-\delta b^2}$,其中 $a^2-\delta b^2\in F_{q^d}$.由于 F_{q^d}中的元素在最后幂运算下等于 1,故可以用 $\bar{\omega}$ 代替 $\frac{1}{\omega}$.在实际应用中通常选取的椭圆曲线的嵌入次数为偶数,故本文只考虑嵌入次数为偶数的情况.

在本文给出的提升算法中,由于 $x_P\in F_q$,$\bar{x}_P=x_P\in F_q$,故分母中的垂线 x_Q-x_P 可用其共轭元 $\overline{x_Q-x_P}=\overline{x_Q}-x_P$ 来代替;同理分母中的直线 $l(-Q)$用其共轭元 $\overline{l(-Q)}=l(Q)$来代替.从而省去了 F_{q^k}[18,23]中复杂度较高的求逆运算.

对于任意的非零常数 c,$\mathrm{div}(f)=\mathrm{div}(cf)$,所以引理 3 中各式的负号不影响双线性对的计算,故我们在计算中忽略不计.

4 提升算法

2008 年赵昌安[11]等将双基数链表示用于双线性对计算并给出了完整的算法,使得双线性对的计算效率提高了大约 9% ~38%;2011 年陈厚友[12]等人进一步研究了 Tate 对快速计算,又将运算效率提高了约 10.6% ~20.3%.

本文在其基础上提出了合并连续相邻项的方法,结合范函数的性质以及引理 3 进一步提升了双线性对的计算效率,算法如下.

算法 2 (提升的 Miller 算法)

输入:素数 $l=\sum_{i=1}^{m}s_i2^{a_i}3^{b_i}$,其中 $s_i\in\{-1,1\}$,$a_1\geqslant a_2\geqslant\cdots\geqslant a_m\geqslant 0$,$b_1\geqslant b_2\geqslant\cdots\geqslant b_m\geqslant 0$,
$P=(x_P,y_P)\in E(F_q)[l]$,$Q=(x_Q,y_Q)\in E(F_{q^k})$
输出:$f=f_l(Q)$

[1] $f\leftarrow 1$,$f_{-1}\leftarrow\frac{1}{v_P(Q)}$,$T\leftarrow P$;
[2] {for $i=1,\cdots,m-1$ do
  [2.1] $a\leftarrow a_i-a_{i+1}$,$b\leftarrow b_i-b_{i+1}$;
  [2.2] if $b\geqslant 1$ then
    [2.2.1] for $j=1,\cdots,b$ do
    [2.2.2] $f\leftarrow f^3\frac{l_{T,T}(Q)v_T(Q)}{l_{T,2T}(-Q)}$,$T\leftarrow 3T$;
  [2.3] { if $a\geqslant 1$ then
    [2.3.1] for $j=1,\cdots,\left\lfloor\frac{a}{2}\right\rfloor$ do
    [2.3.2] $f\leftarrow f^4\frac{l_{T,T}^2(Q)}{l_{2T,2T}(-Q)}$,$T\leftarrow 4T$;
    [2.3.3] if $1\equiv a \bmod 2$;
      [2.3.3.1] if $s_i=1$;
      [2.3.3.2] $f\leftarrow f^2\frac{l_{T,T}(Q)v_P(Q)}{l_{2T,P}(-Q)}$,$T\leftarrow 2T+P$;
      [2.3.3.3] if $s_i=-1$;
      [2.3.3.4] $f\leftarrow f^2\frac{1}{v_P(Q)}\cdot\frac{l_{T,T}(Q)v_{-P}(Q)}{l_{2T,-P}(-Q)}$,$T\leftarrow 2T-P$;
    }
  [2.4] if $0\equiv a \bmod 2$;
    [2.4.1] if $s_i=1$;
    [2.4.2] $f\leftarrow f\frac{l_{T,P}(Q)}{v_{T+P}(Q)}$,$T\leftarrow T+P$;
    [2.4.3] if $s_i=-1$;
    [2.4.4] $f\leftarrow f\frac{1}{v_P(Q)}\cdot\frac{l_{T,-P}(Q)}{v_{T-P}(Q)}$,$T\leftarrow T-P$;
}
[3] Return $f^{(q^k-1)/l}$.

5 复杂度分析

在算法 2 的复杂度分析中,我们忽略有限域中的加法和减法运算,因为这些运算与有限域中的乘法和求逆相比可以忽略不计.用 M,S,I 分别表示有限域 F_q 上

的乘法,平方以及求逆;用 M_k, S_k, I_k 分别表示有限域 F_{q^k} 中的乘法,平方以及求逆;M_b 表示基域 F_q 中的元素和扩域 F_{q^k} 中的元素相乘,故 $M_b = kM$;通常 $M_k = k^2M$,利用 Karatsuba[21] 算法可化简为 $M_k = k^{\log_2 3}M$,因此在复杂度分析中将假设 $M_k = k^{1.6}M$.

在算法复杂度分析中涉及到有理函数赋值的加法和二倍乘,我们分别用 ECADD 和 ECDBL 表示,下表中给出这些运算所需的运算量.

表 2　文献[22] $E(F_q)$ 群中的基本运算

基本运算	运算量
ECADD	$I+2M+S$
ECDBL	$I+2M+2S$

算法 2 将整个迭代过程中分为 6 个部分:"TADD","TSUB","迭代 TDBL","TDBL + P","TDBL – P"和"TTRL",下面具体分析每一部分的复杂度.由于预计算可以在算法执行前完成,故本文在复杂度分析中不考虑预计算的开销.

5.1　TADD 的运算量

设 $T=(x_1,y_1)$ 和 $P=(x_P,y_P)\in E(F_q)$. TADD 即算法 2 中第[2.4.2]步,其计算如下:

输入:$f\in F_{q^k}^*$, $T=(x_1,y_1)\in E(F_q)$, $P=(x_P,y_P)$ 和 $Q=(x_Q,y_Q)\in E(F_{q^k})$

输出:更新的 f

(1) $T_4=(x_4,y_4)\leftarrow$ECADD(T,P)

(2)
$$\frac{l_{T,P}(Q)}{v_{T+P}(Q)}=\frac{(y_Q+y_4)-\lambda_4(x_Q-x_4)}{x_Q-x_4}$$
$$=\frac{y_Q+y_4}{x_Q-x_4}-\lambda_4$$
$$=(y_Q+y_4)(\bar{x}_Q-x_4)-\lambda_4$$
$$=y_Q\bar{x}_Q+y_4\bar{x}_Q-(y_Q+y_4)x_4-\lambda_4$$

(3)更新的 $f=f\cdot[y_Q\bar{x}_Q+y_4\bar{x}_Q-(y_Q+y_4)x_4-\lambda_4]$

直线 $l_{T,P}$ 经过点 T 和 P,其斜率为 λ_4. 由于 $P+T=T_4=(x_4,y_4)$,故 $P, T, -T_4$ 三点共线. 即:$l_{T,P}=(y_Q-y_P)-\lambda_4(x_Q-x_P)=(y_Q+y_4)-\lambda_4(x_Q-x_4)$,为了减少 TADD 的运算量,在上面的计算中我们选择 $l_{T,P}=(y_Q+y_4)-\lambda_4(x_Q-x_4)$.

由于 $P+T=(x_4,y_4)$,即计算 T_4 需要一次 ECADD;因为 $y_Q\bar{x}_Q$ 可预计算;计算 $y_4\bar{x}_Q$ 和 $(y_Q+y_4)x_4$ 需要 $2M_b$;另外计算 $f\cdot\frac{l_{T,P}(Q)}{v_{T+P}(Q)}$ 需要 $1M_k$.

因此,TADD 算法的总运算量为 M_k+2M_b+ ECADD. 若再运用伪求逆技巧[24],运算量可约减到 $M_k+1.5M_b+$ ECADD.

5.2　TSUB 的运算量

设 $T=(x_1,y_1)$ 和 $P=(x_P,y_P)\in E(F_q)$. TSUB 即算法 2 中第[2.4.4]步,其计算如下:

输入:$f\in F_{q^k}^*$, $T=(x_1,y_1)\in E(F_q)$, $P=(x_P,y_P)$, $Q=(x_Q,y_Q)\in E(F_{q^k})$

输出:更新的 f

(1) $T_5=(x_5,y_5)\leftarrow$ECADD($T,-P$)

(2)
$$\frac{1}{v_P}\cdot\frac{l_{T,-P}}{v_{T-P}}=\frac{y_Q+y_P-\lambda_5(x_Q-x_P)}{(x_Q-x_5)(x_Q-x_P)}$$
$$=(\frac{y_Q+y_P}{x_Q-x_P}-\lambda_5)(\bar{x_Q}-x_5)$$
$$=\frac{y_Q+y_P}{x_Q-x_P}\bar{x_Q}-\lambda_5\bar{x_Q}-(\frac{y_Q+y_P}{x_Q-x_P}-\lambda_5)x_5$$

(3)更新的 $f=f\cdot[\frac{y_Q+y_P}{x_Q-x_P}\bar{x_Q}-\lambda_5\bar{x_Q}-(\frac{y_Q+y_P}{x_Q-x_P}-\lambda_5)x_5]$

直线 $l_{T,-P}$ 经过点 T 和 $-P$ 的直线 $l_{T,-P}$ 的斜率为 λ_5,直线方程为 $l_{T,-P}=y_Q+y_P-\lambda_5(x_Q-x_P)$. 由引理 2 知 $f_{i-1}=f_i\cdot\frac{1}{v_P}\cdot\frac{l_{iP,-P}}{v_{(i-1)P}}$.

由于 $T-P=(x_5,y_5)$,即计算 T_5 需要一次 ECADD,又因为 $\frac{y_Q+y_P}{x_Q-x_P}\bar{x}_Q$ 可以预计算;计算 $\lambda_5\bar{x}_Q$ 和 $(\frac{y_Q+y_P}{x_Q-x_P}-\lambda_5)x_5$ 需要 $2M_b$;另外计算 $f\cdot\frac{1}{v_P}\frac{l_{T,-P}}{v_{T-P}}$ 需要 $1M_k$.

因此,TSUB 算法的总量为 M_k+2M_b+ ECADD.

5.3　迭代 TDBL 的运算量

设 $T=(x_1,y_1)$ 和 $2T=(x_2,y_2)\in E(F_q)$. 迭代 TDBL 即算法 2 中第[2.3.3.2]步,其计算如下:

输入:$f\in F_{q^k}^*$, $T=(x_1,y_1)\in E(F_q)$, $Q=(x_Q,y_Q)\in E(F_{q^k})$

输出:更新的 f

(1) $T_2=(x_2,y_2)\leftarrow$ECDBL(T)

(2)
$$\frac{l_{T,T}^2(Q)}{l_{2T,2T}(-Q)}=l_{T,T}^2(Q)\overline{l_{2T,2T}(-Q)}$$
$$=l_{T,T}^2(Q)l_{2T,2T}(Q)$$
$$=l_{T,T}^2(Q)[y_Q-y_2-\lambda_2(x_Q-x_2)]$$

(3)更新的 $f=f^2l_{T,T}^2(Q)[y_Q-y_2-\lambda_2(x_Q-x_2)]$

直线 $l_{T,T}$ 经过点 T 和 $-2T$,其斜率为 λ_1;λ_2 是直线 $l_{2T,2T}$ 的斜率,由于 $l_{2T,2T}$ 过点 $2T$ 和 $-4T$,计算 λ_2 需先计算 $2T$,即一次 ECDBL 运算;计算 $\lambda_2\cdot(x_Q-x_2)$ 需要 $1M_b$;另外计算 $(f^2\cdot l_{T,T}(Q))^2\cdot l_{2T,2T}(Q)$ 需要 $2M_k+2S_k$.

因此,迭代 TDBL 算法的总量为 $2M_k+2S_k+M_b+$ ECDBL.

5.4　TDBL + P 的运算量

设 $T=(x_1,y_1)$, $2T=(x_2,y_2)$ 和 $P=(x_P,y_P)\in E(F_q)$, TDBL + P 即算法 2 中第[2.3.3.2]步,其计算如下:

输入:$f\in F_{q^k}^*$, $T=(x_1,y_1)\in E(F_q)$, $P=(x_P,y_P)$, $Q=(x_Q,y_Q)\in E(F_{q^k})$

输出:更新的 f

(1) $T_2=(x_2,y_2)\leftarrow$ECDBL(T)

(2) $\dfrac{l_{T,T}(Q)v_P(Q)}{l_{2T,P}(-Q)}=\dfrac{l_{T,T}(Q)(x_Q-x_P)}{(-y_Q-y_P)-\lambda_6(x_Q-x_P)}$

$=\dfrac{l_{T,T}(Q)}{(-y_Q-y_P)\overline{(x_Q-x_P)}-\lambda_6}$

$=l_{T,T}(Q)[(\overline{y_Q}+y_P)(x_P-x_Q)-\lambda_6]$

(3)更新的 $f=f^2l_{T,T}(Q)[(\overline{y_Q}+y_P)(x_P-x_Q)-\lambda_6)]$

直线 $l_{T,T}$同上,λ_6 是直线 $l_{2T,P}$的斜率,由于 $l_{2T,P}$过点 $2T$ 和 P,计算 λ_6 需先计算 $2T$,即一次 ECDBL 运算;又因为$(\overline{y_Q}+y_P)(x_P-x_Q)$可以预计算;另外计算 $f=f^2l_{T,T}(Q)\cdot[(\overline{y_Q}+y_P)(x_P-x_Q)-\lambda_6]$需要 $2M_k+S_k$.

因此,TDBL+P 算法的总量为 $2M_k+S_k$+ECDBL.

5.5 TDBL-P 的运算量

设 $T=(x_1,y_1)$,$2T=(x_2,y_2)$和 $P=(x_P,y_P)\in E(F_q)$. TDBL-P 即算法 2 中第[2.3.3.4]步,其计算如下

输入:$f\in F_{q^k}^*$, $T=(x_1,y_1)\in E(F_q)$, $P=(x_P,y_P)$, $Q=(x_Q,y_Q)\in E(F_{q^k})$

输出:更新的 f

(1) $T_2=(x_2,y_2)\leftarrow$ECDBL(T)

(2) $\dfrac{l_{T,T}(Q)v_{-P}(Q)}{v_P(Q)l_{2T,-P}(-Q)}=\dfrac{l_{T,T}(Q)(x_Q+x_P)}{(x_Q-x_P)l_{2T,-P}(-Q)}$

$=l_{T,T}(Q)(x_Q+x_P)\overline{(x_Q-x_P)l_{2T,-P}(-Q)}$

$=l_{T,T}(Q)(x_Q+x_P)(\overline{x_Q}-x_P)l_{2T,-P}(Q)$

$=l_{T,T}(Q)(x_Q+x_P)(\overline{x_Q}-x_P)[(y_Q+y_P)-\lambda_7(x_Q-x_P)]$

$=l_{T,T}(Q)[(x_Q+x_P)(y_Q+y_P)(\overline{x_Q}-x_P)-\lambda_7(x_Q^2-x_P^2)(\overline{x_Q}-x_P)]$

(3)更新的 $f=f^2l_{T,T}(Q)[(x_Q+x_P)(y_Q+y_P)(\overline{x_Q}-x_P)-\lambda_7(x_Q^2-x_P^2)(\overline{x_Q}-x_P)]$

直线 $l_{T,T}$同上,λ_7 是直线 $l_{2T,-P}$的斜率,由于 $l_{2T,-P}$过点 $2T$ 和 $-P$,计算 λ_7 需先计算 $2T$,即一次 ECDBL 运算;又因为 $(x_Q+x_P)(y_Q+y_P)(\overline{x_Q}-x_P)$和 $(x_Q^2-x_P^2)(\overline{x_Q}-x_P)$都可以预计算;计算 $\lambda_7\cdot(x_Q^2-x_P^2)(\overline{x_Q}-x_P)$需要 $1M_b$;另外计算 $f^2\cdot l_{T,T}(Q)\cdot[(x_Q+x_P)(y_Q+y_P)-\lambda_7(x_Q^2-x_P^2)]$需要 $2M_k+S_k$.

因此,TDBL-P 算法的总量为 $2M_k+S_k+M_b$+ECDBL.

5.6 TTRL 的运算量

设 $T=(x_1,y_1)$,$2T=(x_2,y_2)\in E(F_q)$. TTRL 即算法 2 中第[2.2.2]步,其计算如下:

输入:$f\in F_{q^k}^*$, $T=(x_1,y_1)\in E(F_q)$, $Q=(x_Q,y_Q)\in E(F_{q^k})$

输出:更新的 f

(1) $T_2=(x_2,y_2)\leftarrow$ECDBL(T)

(2) $\dfrac{l_{T,T}(Q)v_T(Q)}{l_{T,2T}(-Q)}=\dfrac{l_{T,T}(Q)(x_Q-x_1)}{(-y_Q-y_1)-\lambda_3(x_Q-x_1)}$

$=\dfrac{l_{T,T}(Q)}{(-y_Q-y_1)(\overline{x_Q}-x_1)-\lambda_3}$

$=\dfrac{l_{T,T}(Q)}{-y_Q\overline{x_Q}-y_1\overline{x_Q}+x_1(y_Q+y_1)-\lambda_3}$

$=l_{T,T}(Q)[-x_Q\overline{y_Q}-y_1x_Q+x_1(\overline{y_Q}+y_1)-\lambda_3]$

(3) $f=f^3l_{T,T}(Q)[(-x_Q\overline{y_Q}-y_1x_Q+x_1(\overline{y_Q}+y_1)-\lambda_3)]$

直线 $l_{T,T}$同上,其斜率为 λ_1;λ_3 是直线 $l_{T,2T}$的斜率,由于 $l_{T,2T}$过点 T 和 $2T$,计算 λ_3 需先计算 $2T$,即一次 ECDBL 运算;又因为 $x_Q\overline{y_Q}$是可以预计算的;计算 y_1x_Q 和 $x_1(\overline{y_Q}+y_1)$需要 $2M_b$;另外计算 $f^2\cdot f\cdot l_{T,T}(Q)[-x_Q\overline{y_Q}-y_1x_Q+x_1(\overline{y_Q}+y_1)-\lambda_3]$需要 $3M_k+S_k$.

因此,TTRL 算法的总量为 $3M_k+S_k+2M_b$+ECDBL.

6 算法效率比较

本章我们给出算法 2 的运算量并与已有的算法[11,12]进行比较.根据目前椭圆曲线密码安全性要求,椭圆曲线群阶 l 至少 160 比特,而 q^k 至少是 1024 比特.为了和现有算法相比较我们取 $\log_2 l=160$.

表 3 新旧算法中 Miller 迭代中各个部分的运算量

	基本运算	复杂度	$k=4$	$k=6$	$k=8$
文献[11]算法	TADD	$M_k+2.5M_b+I+3M+S$	32.8M	46.8M	60.8M
	TSUB	$M_k+I+(2k+3)M+S$	30.8M	43.8M	56.8M
	迭代 TDBL	$3M_k+2S_k+4M_b+I+4M+2S$	73M	122.4M	171.8M
	TDBL+P	$2M_k+S_k+6M_b+2I+7M+3S$	78.6M	115.8M	153M
	TDBL−P	$2M_k+S_k+3.5M_b+2I+(2k+7)M+3S$	76.6M	112.8M	149M
	TTRL	$3M_k+S_k+2M_b+I+9M+4S$	64.4M	102.6M	140.8M
文献[12]算法	TADD	$M_k+2M_b+I+2M+S$	29.8M	42.8M	55.8M
	TSUB	$M_k+1.5M_b+I+2M+S$	27.8M	39.8M	51.8M
	迭代 TDBL	$3M_k+2S_k+4M_b+I+4M+2S$	73M	122.4M	171.8M
	TDBL+P	$2M_k+S_k+4M_b+2I+4M+3S$	67.6M	100.8M	134M
	TDBL−P	$2M_k+S_k+3.5M_b+2I+4M+3S$	65.6M	97.8M	130M
	TTRL	$3M_k+S_k+2M_b+I+7M+4S$	55.8M	94M	132.2M
算法 2	TADD	$M_k+1.5M_b+I+2M+S$	27.8M	39.8M	51.8M
	TSUB	$M_k+2M_b+I+2M+S$	29.8M	42.8M	55.8M
	迭代 TDBL	$2M_k+2S_k+M_b+I+2M+2S$	50M	84.4M	118.8M
	TDBL+P	$2M_k+S_k+I+2M+2S$	38.8M	64M	89.2M
	TDBL−P	$2M_k+S_k+M_b+I+2M+2S$	42.8M	70M	97.2M
	TTRL	$3M_k+S_k+2M_b+I+2M+2S$	55.8M	94M	132.2M

为了便于后面总体运算量的评估,表 3 首先给出了 Miller 迭代中各个部分的运算量,其中取 $M_4 = 9M$, $M_6 = 18M$, $M_8 = 27M$, $S = 0.8M$, $S_k = 0.8M_k$, $I = 10M$, $M_b = kM$ 和 $I_k = I + k^2M$.

从表 3 中可以看出算法 2 中各个部分迭代的运算量都不超过文献[11,12]相对应部分的运算量.

假设 l 被双基数链表示时所需要的加号和减号均相等,且 $a_i - a_{i+1}, i = 1, \cdots, m-1$ 中的奇数偶数的个数相等.则算法 2 的总开销大约为:

$$b_{\max}\cdot \mathrm{TTRL} + \left\lfloor \frac{a_{\max}}{2} \right\rfloor \cdot \text{迭代 } \mathrm{TDBL} + \frac{m}{2}\cdot\left[\frac{1}{2}\cdot((\mathrm{TDBL}+P)+\mathrm{TADD}) + \frac{1}{2}\cdot((\mathrm{TDBL}-P)+\mathrm{TSUB})\right]$$

$$= (3b_{\max} + 2\cdot\left\lfloor \frac{a_{\max}}{2} \right\rfloor + \frac{3}{2}m)M_k + (b_{\max} + 2\cdot\left\lfloor \frac{a_{\max}}{2} \right\rfloor + \frac{m}{2})S_k + (2b_{\max} + \left\lfloor \frac{a_{\max}}{2} \right\rfloor + \frac{9}{8}m)M_b + (b_{\max} + \left\lfloor \frac{a_{\max}}{2} \right\rfloor + m)I + (2b_{\max} + 2\cdot\left\lfloor \frac{a_{\max}}{2} \right\rfloor + 2m)M + (2b_{\max} + 2\cdot\left\lfloor \frac{a_{\max}}{2} \right\rfloor + \frac{3}{2}m)S.$$

表 4 统计了在不同嵌入次数下算法 2 的总运算量,并与目前最快的算法[11,12]进行了比较.

表 4　不同嵌入次数下算法的复杂度

	$a_{\max}$	$b_{\max}$	m	$k=4$	$k=6$	$k=8$
文献[11]算法	57	65	45	8160M	12507M	16846M
	76	53	38	8223M	12317M	16709M
	95	41	37	8268M	12315M	16676M
	103	36	39	8366M	12439M	16810M
文献[12]算法	57	65	45	7386M	11506M	15606M
	76	53	38	7321M	11151M	15269M
	95	41	37	7180M	10859M	14921M
	103	36	39	7307M	10902M	14917M
算法 2	57	65	45	6593M	10910M	15227M
	76	53	38	6180M	10247M	14314M
	95	41	37	5925M	9824M	13723M
	103	36	39	5916M	9800M	13685M

表 4 表明算法 2 与文献[11]的算法相比较效率提高大约 10% ~ 29%,与文献[12]的算法相比较效率提高大约 10%,并且新算法同样适用于嵌入次数 $k = 2$ 的情况.

7　结论

本文在基于双基数链计算 Tate 对的基础上,通过范函数和共轭技巧的应用进一步提高了 Miller 算法的计算效率.新算法减少了 Miller 迭代中有理函数分子分母中直线和垂线的数量并且用共轭代替了耗时较多的求逆运算,提高了双线性对的计算效率.本文给出的新算法不仅可以应用于 Tate 对的快速计算,同时可以应用于近年来学者们构造的新双线性对,如 Eta 对,Ate 对和 Optimal 对的快速计算.结果表明本文给出的新算法比已有的算法快 10%以上.

参考文献

[1] A J Menezes, T Okamoto, S A Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field [J]. IEEE Transaction on Information Theory, 1993, 39(5): 1639 – 1646.

[2] G Frey, H G Ruck. A remark concerning m-divisibility and the discrete logarithm in divisor class group of curves [J]. Mathematics of Computation, 1994, 62 (206): 865 – 874.

[3] A Shamir. Identity-based cryptosystems and signature schemes [A]. Advances in Cryptology-Crypto'84 [C]. Berlin Heidelberg: Springer-Verlag, 1984. 47 – 53.

[4] R Sakai, K Ohgishi, M Kasahara. Cryptosystems based on pairing [A]. Symposium on Cryptography and Information Security [C]. Japan: Okinawa, 2000. 26 – 28.

[5] A Joux. A one round protocol for tripartite Diffie-Hellman [A]. ANTS-4: Proceedings of the 4th International Symposium on Algorithmic Number Theory [C]. Berlin Heidelberg: Springer-Verlag, 2000. 385 – 394.

[6] D Boneh, M Franklin. Identity-based encryption from the Weil pairing [A]. Advances in Cryptology-Crypto'2001 [C]. Berlin Heidelberg: Springer-Verlag, 2001. 213 – 229.

[7] F Zhang, R Safavi-Naini, W Susilo. An efficient signature scheme from bilinear pairings and its applications [A]. Public Key Cryptography-PKC' 2004 [C]. Berlin Heidelberg: Springer-Verlag, 2004. 277 – 290.

[8] 祁明, L Harn. 基于离散对数的若干新型代理签名方案 [J]. 电子学报, 2000, 28(11): 114 – 118.
Qi Ming, L Harn. Some new proxy signature schemes based on discrete logarithms [J]. Acta Electrionica Sinica, 2000, 28 (11): 114 – 118. (in Chinese)

[9] V S Miller. The Weil pairing and its efficient calculation [J]. Journal of Cryptology, 2004, 17(04): 235 – 261.

[10] K E Stange. The Tate pairing via elliptic nets [A]. Pairing-Based Cryptography Pairing 2007 [C]. Berlin Heidelberg: Springer-Verlag, 2007. 329 – 348.

[11] C A Zhao, F G Zhang, J W Huang. Efficient Tate paring computation using double-base chains [J]. Science in China series F-Information sciences, 2008, 51(08): 1096 – 1105.

[12] 陈厚友, 马传贵. 基于双基数链的 Tate 对快速算法 [J]. 电子学报, 2011, 39(2): 408 – 413.
Chen Hou-you, Ma Chuan-gui. Fast Tate pairing algorithm using double-base chains [J]. Acta Electrionica Sinica, 2011, 39 (2): 408 – 413. (in Chinese)

[13] P S L M Barreto, S D Galbraith, C Eigeartaig, M Scott. Effi-

cient pairing computation on supersingular abelian varieties [J]. Designs, Codes Cryptography, 2007, 42(3): 239 – 271.
[14] F Hess, N P Smart, F Vercauteren. The Eta pairing revisited [J]. IEEE Transactions on Information Theory, 2006, 52(10): 4595 – 4602.
[15] F Vercauteren. Optimal pairings [J]. IEEE Transactions on Information Theory, 2010, 56(1): 455 – 461.
[16] V Dimitrov, G Jullien. Loading the bases: A new number representation with applications [J]. IEEE Circuits and Systems Magazine, 2003, 3(2): 6 – 23.
[17] V S Dimitrov, L Imbert, P K Mishra. Efficient and secure elliptic curve point multiplication using double-base chains [A]. ASIACRYPT 2005 [C]. Berlin Heidelberg: Springer-Verlag, 2005. 59 – 78.
[18] M Scott. Implementing cryptographic pairings [A]. Pairing 2007 [C]. Tokyo: LCNS, 2007. 177 – 196.
[19] P S L M Barreto, H Y Kim, B Lynn, M Scott. Efficient algorithms for pairing based cryptosystems [A]. Cryptology-Cryto'2002 [C]. Berlin Heidelberg: Springer-Verlag, 2002. 354 – 368.
[20] I F Blake, V K Murty, G W Xu. Refinements of Miller's algorithm for computing the Weil/Tate pairing [J]. Journal of Algorithms, 2006, 58(2): 34 – 149.
[21] D E Knuth. The Art of Computer Programming [M]. Massachusetts: Addison-Wesley, 1988.
[22] IEEE P1363, Standard Specifications For Public Key Cryptography [S].
[23] M Ciet, M Joye, K Lauter, P L Montgomery. Trading inversions for multiplications in elliptic curve cryptography [J]. Designs, Codes and Cryptography, 2006, 39(2): 189 – 206.
[24] T Kobayashi, K Aoki, H Imai. Efficient algorithms for tate pairing [J]. IEICE Trans on Fundamentals, 2006, E89-A(1): 134 – 143.

作者简介

翁 江 男, 1986 年 3 月出生于陕西省西安市. 现为信息工程大学信息工程学院博士研究生. 主要研究方向为椭圆曲线双线性对快速计算及其侧信道攻击.
E-mail: wengjiang858@163.com

豆允旗 男, 1987 年 9 月出生于河南省商丘市. 现为信息工程大学信息工程学院硕士研究生. 主要研究方向为椭圆曲线侧信道攻击.
E-mail: douyunqi@126.com

马传贵 男, 1962 年 4 月出生于山东省菏泽市. 现为信息工程大学信息工程学院教授、博士生导师. 主要研究方向为密码学和无线网络安全.
E-mail: chuanguima@sina.com