特征 3 有限域上椭圆曲线的 co-Z Montgomery 算法

于 伟[1),2)] 李 宝[1)] 王鲲鹏[1)] 李维晅[1)] 田 松[1)]

[1)](中国科学院信息工程研究所 北京 100093)

[2)](中国科学院 DCS 中心 北京 100093)

摘 要 椭圆曲线公钥密码是公钥密码体制的主流方向之一.由于密钥短、计算速度快,该体制在智能卡和手机存储卡等受限的环境中得到了广泛的应用.椭圆曲线密码体系中最耗时的运算是标量乘.标量乘需要安全、有效、快速的实现算法.Montgomery 算法是计算椭圆曲线标量乘的算法之一,它能够有效地抵抗简单能量分析.在 Montgomery 算法结构的基础上,文中首次利用统一 Z 坐标技巧和循环中间阶段不计算 Y 坐标的技巧,改进了有限域 $GF(3^m)$ 上椭圆曲线的点加和倍点公式,构造了抵抗简单能量攻击的 co-Z Montgomery 算法.设 I,M,C 分别表示有限域上的求逆、乘法、立方.当域上的平方和乘法使用相同的算法时,理论分析表明每轮循环中,co-Z Montgomery 算法比仿射 Montgomery 算法快 $I+C-5M$,比射影 Montgomery 算法快 $C+2M$,比使用"Selected Areas in Cryptography" 2012 上快速点加、倍点公式的 Montgomery 算法快 $2C+M$.在文章"特征 3 有限域上椭圆曲线的 Montgomery 算法"的模拟实验环境下,结果表明该算法比上述算法分别快 26.3%、19.0%、20.6%;Sage 云平台的实验结果表明该算法比上述算法分别快 24.1%、20.1%、23.1%.

关键词 椭圆曲线;Montgomery 算法;标量乘;简单能量攻击;co-Z

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2017.01121

co-Z Montgomery Algorithm on Elliptic Curves Over Finite Fields of Characteristic Three

YU Wei[1),2)] LI Bao[1)] WANG Kun-Peng[1)] LI Wei-Xuan[1)] TIAN Song[1)]

[1)](*Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093)

[2)](*Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing* 100093)

Abstract Elliptic curve cryptosystem is one of the main directions of public key cryptography. Because of the short key and efficient arithmetic, it has attracted increasing attention, particularly in resource-limited hardware environments such as smart cards and phone cards. Scalar multiplication is the most time-consuming operation in elliptic curve cryptosystems, which should be implemented safely, efficiently, and fast. Montgomery algorithm is a scalar multiplication algorithm on elliptic curves which is resistant to simple power analysis. Based on the structure of Montgomery algorithm, new formulas of point operations including point addition and point doubling of elliptic curves defined on finite fields $GF(3^m)$ are first introduced by using same Z-coordinate and not calculating Y-coordinate. Hence co-Z Montgomery algorithm which is resistant to simple power analysis is proposed. When squaring algorithm is implemented through multiplication algorithm over a finite field, co-Z Montgomery algorithm saves $I+C-5M$ more than affine Montgomery algorithm, saves $C+2M$ more than projective Montgomery algorithm, and saves $2C+M$ more than Montgomery

收稿日期:2015-04-08;最终修改稿收到日期:2015-12-08.本课题得到国家自然科学基金(61502487、61272040)、国家"九七三"重点基础研究发展规划项目基金(2013CB338001)资助.于 伟,男,1987 年生,博士,助理研究员,主要研究方向为椭圆曲线密码学.E-mail:yuwei_1_yw@163.com.李 宝,男,1962 年生,博士,研究员,博士生导师,主要研究领域为椭圆曲线密码学、可证明安全方法的理论与应用、安全协议的设计与分析.王鲲鹏,男,1971 年生,博士,研究员,博士生导师,主要研究领域为椭圆曲线密码学.李维晅,女,1994 年生,博士研究生,主要研究方向为椭圆曲线密码学.田 松,男,1987 年生,博士研究生,主要研究方向为椭圆曲线密码学.

algorithm using the formulas of "Selected Areas in Cryptography 2012" where I, M, C stand for field inversion, multiplication and cube respectively. Experimental results on the platform of "Montgomery algorithm on elliptic curves over finite fields of character three" show that co-Z algorithm are 26.3%, 19.0%, 20.6% faster than the previous algorithms respectively. Experimental results on Sage cloud platform indicate that co-Z algorithm are 24.1%, 20.1%, 23.1% faster than the previous algorithms respectively.

Keywords elliptic curve; Montgomery algorithm; scalar multiplication; simple power analysis; co-Z

1 引 言

1986 年, Koblitz[1] 和 Miller[2] 同时独立地提出椭圆曲线密码体制. 在实际应用中,160 位的椭圆曲线密码体制和 1024 位的 RSA 密码体制的安全强度是相同的;256 位的椭圆曲线密码体制和 3072 位的 RSA 密码体制具有相同的安全强度;600 位的椭圆曲线密码体制则和 21000 位的 RSA 密码体制具有相同的安全强度. 由此可见随着密钥长度的增加,椭圆曲线密码体制和 RSA 密码体制的密钥长度的比例越来越小,其优势也越来越大. 在密钥长度越大的情况下,密码体制所能提供的安全性也就越高. 因而,这样更有利于在手机智能卡、安全存储芯片等资源受限的环境中,使用椭圆曲线密码算法. 由于椭圆曲线密码体制的短密钥、高效率,以及目前在一般椭圆曲线上没有亚指数时间攻击椭圆曲线离散对数的算法,椭圆曲线密码体制的应用越来越广泛.

美国国家标准协会给出了 FIPS 186-2 椭圆曲线密码标准;美国国家标准组织给出了 ANSI X9.62 和 X9.63 椭圆曲线标准;电子电器工程师协会给出了 FIPS IEEE P1363 椭圆曲线密码标准. 为了进一步推动椭圆曲线密码体制在我国的广泛应用,国家密码管理局于 2010 年颁布了中国椭圆曲线密码标准 SM2①.

椭圆曲线密码体制主要包括椭圆曲线密钥生成算法、密钥交换算法、加密算法、解密算法、签名算法和验签名算法. 在这些椭圆曲线密码算法中,最耗时的部分是标量乘运算. 椭圆曲线密码体制的实现效率很大程度上取决于椭圆曲线标量乘的计算速度. 椭圆曲线密码能否进一步广泛应用,也主要取决于其密码算法效率能否提高. 因而,如何改进椭圆曲线标量乘的计算效率一直是公钥密码学的研究热点.

点的标量乘 kP,指计算 k 个 P 的和,其中 k 是整数,P 是椭圆曲线上的点. 提高标量乘效率的算法主要有以下 3 类:(1)降低标量 k 的表示的 Hamming 重量[3-6]. Hamming 重量是影响标量乘效率的主要因素之一,通过使用非邻接形表示、双基表示等降低标量的 Hamming 重量能够有效地减少标量乘的计算量;(2)利用可计算的有效自同态,比如素域上的 GLV 算法[7-12] 和特征为 2 的有限域上椭圆曲线的 Frobenius 自同态都能够有效地降低标量乘的计算量;(3)利用点的特定坐标表示. 不同的坐标表示所带来的点加和倍点的计算速度是不同的,进而会影响整个标量乘的效率. 点的坐标表示一般包括仿射坐标表示、射影坐标表示、雅可比坐标表示和混合坐标表示[13]. 计算点乘时,利用仿射坐标,一般需要比较多的求逆运算,当求逆运算的速度比较快时,可以选择仿射坐标计算标量乘. 当求逆运算花费较大时,射影坐标和雅可比坐标计算速度较快. 选择坐标系时,很多时候还要考虑密码算法的实际应用环境.

1999 年,Coron[14] 研究了利用差分能量分析来攻击椭圆曲线密码体制,并且给出了抵抗差分能量分析的具体方法. 2006 年,Lars② 引入了简单能量分析的方式用于椭圆曲线公钥密码算法的攻击. 简单能量分析通过检测密码设备消耗的能量进而获取密码设备内部的相关秘密信息,从而成功攻击椭圆曲线密码算法或者部分的攻击椭圆曲线密码算法. 这种密码分析方法基于密码算法的实现特性,需要在算法实现的过程中进行防护. 目前简单能量分析严重威胁了广泛使用的椭圆曲线公钥密码算法的

① State Cryptography Administration Office of Security Commercial Code Administration. Public key cryptographic algorithm SM2 based on elliptic curves. http://www.oscca.gov.cn, 2010

② Lars Elmegaard-Fessel. Efficient scalar multiplication and security against power analysis in cryptosystems based on the NIST elliptic curves over prime fields. eprint, 2006/313. http://eprint.iacr.org/2006/313

安全.

文献[3-12]中的标量乘算法均无法有效地防止简单能量分析. 当前的标量乘算法主要有 3 类可以有效地防止简单能量分析：(1) 使用统一的点加和倍点公式，如 Edwards 曲线[15]，让攻击者无法恢复出 k 的二进制信息；(2) 随机化方法，比如 Joye 和 Tymen[16]提出的点、曲线、标量随机化方法；(3) 每一轮循环均使用点加和倍点，使得每个循环消耗的能量相同，例如 Montgomery 算法[17].

Montgomery 算法用于计算有限域上的椭圆曲线点的标量乘. 文献[18]提出了省略计算 Y 坐标的点加和倍点公式，并且去除了求逆运算，进而改进了特征 3 有限域上的 Montgomery 算法. 文献[19]改进了特征 3 域上的点加、倍点操作.

co-Z 技巧是指点的射影坐标表示下统一 Z 坐标的方法. 该方法首先由 Meloni[20]提出，提高了椭圆曲线点加、倍点和多倍点(三倍点、五倍点)的计算效率. Longa 等人[6]利用该技巧进一步提高多倍点的计算速度. Lin 和 Zhang[21]利用该技巧提高了多标量乘的计算效率.

Smart 和 Westwood[22]指出特征 3 上的椭圆曲线比相同安全强度的特征 2 上的椭圆曲线的速度至多慢 50%，并且指出了特征 3 上的椭圆曲线也适合密码学应用.

考虑到简单能量分析对椭圆曲线密码算法的威胁以及定义在有限域 $GF(3^m)$ 上的椭圆曲线的标量乘的计算效率问题，我们提出了 co-Z Montgomery 算法，提高了标量乘的计算速度. 本文的创新点是使用 co-Z 技巧和循环中间阶段不计算 Y 坐标的技巧优化特征 3 有限域上的射影 Montgomery 算法. 用 I,M,S,C 分别表示特征 3 有限域上的求逆，乘法，平方和立方运算. 特征 3 有限域上的 Montgomery 算法的每轮循环需要的运算为 $I+6M+2C+2S$，射影 Montgomery 算法需要 $13M+2C+2S$，Montgomery 算法使用文献[19]最近提出的公式需要 $14M+3C$. 本文提出的 co-Z Montgomery 算法需要 $10M+C+3S$. 若平方和乘法按相同的花费计算，co-Z Montgomery 算法的每个循环比射影 Montgomery 算法少 2 个乘法和 1 个立方运算，比 Montgomery 算法利用文献[19]的公式快 1 个乘法和 2 个立方运算. 当求逆的速度大于 4 个乘法时，co-Z Montgomery 算法比仿射 Montgomery 算法快. 在文献[18]中的环境下，co-Z Montgomery 算法比仿射 Montgomery 算法快 26.3%；比射影 Montgomery 算法快 19.0%；比 Montgomery 算法利用文献[19]的公式快 20.6%. Sage 云平台的实验结果表明该算法比仿射 Montgomery 算法快 24.1%，比射影 Montgomery 算法[18]快 20.1%，比使用“Selected Areas in Cryptography” 2012 上快速点加、倍点公式的 Montgomery 算法快 23.1%.

本文第 2 节介绍特征为 3 的有限域上的椭圆曲线的基础知识；第 3 节介绍 Montgomery 算法的由来、特征 3 有限域上的仿射 Montgomery 算法和射影 Montgomery 算法；第 4 节讲述利用 co-Z 技巧和循环中间阶段不计算 Y 坐标的技巧优化特征 3 有限域上的射影 Montgomery 算法的详细过程及输出时恢复 Y 坐标的技术细节；第 5 节讲述 co-Z Montgomery 算法和仿射 Montgomery 算法、射影 Montgomery 算法及利用目前最新的点加、倍点公式的 Montgomery 算法的效率比较；第 6 节总结本文的工作.

2　椭圆曲线

椭圆曲线是指亏格为 1 并且具有一个基点的代数曲线. 域 K 上的椭圆曲线一般由 Weierstrass 方程表示为 $y^2+a_2xy+a_4y=x^3+a_1x^2+a_3x+a_5$.

当域 K 的特征为 2 时，约化后的非超奇异椭圆曲线方程为 $y^2+xy=x^3+ax^2+b$，其中判别式 $\Delta=b\neq 0$.

当域 K 的特征大于 3 时，约化后的非超奇异椭圆曲线方程为 $y^2=x^3+ax+b$，其中判别式 $\Delta=-16(4a^3+27b^2)\neq 0$.

特征 3 上的椭圆曲线约化后的曲线方程为 $y^2=x^3+a_1x^2+a_5$ 或者 $y^2=x^3+a_3x+a_5$. 其中，$y^2=x^3+a_3x+a_5$ 是超奇异椭圆曲线，其 j 不变量为 0. 特征 3 上的非超奇异椭圆曲线的定义如下.

定义 1.　当 K 的特征等于 3 时，定义在 K 上的非超奇异椭圆曲线 E 的仿射 Weierstrass 方程为

$$y^2=x^3+ax^2+b \tag{1}$$

其中 $a,b\in K$，判别式 $\Delta=-a^3b\neq 0$，$j=\dfrac{-a^3}{b}$.

在代数闭域上，j 不变量相同的椭圆曲线是同构的. j 不变量不同的椭圆曲线不同构. 判别式则标记了方程是否有重根，如果有重根，则亏格为 0，不能称之为椭圆曲线. 根据 $\Delta=-a^3b\neq 0$，$j=\dfrac{-a^3}{b}$，可知超奇异椭圆曲线的 j 不变量非 0. 该类曲线与超奇异椭圆曲线不可能同构.

由于特征 2 和特征大于 3 的域上的椭圆曲线的软硬件实现的效率较高,已经被广泛研究.特征 3 上的椭圆曲线由于其效率因素的影响,研究的一直比较少.如何快速安全有效地实现特征 3 有限域上的椭圆曲线的标量乘运算,成为当前的研究热点之一.

上文中我们介绍了仿射 Weierstrass 方程表示的椭圆曲线.很多时候,我们还需要考虑标准射影坐标表示下的椭圆曲线.标准射影坐标$(X:Y:Z)$表示的点对应于仿射坐标 $P(x,y)$表示的点,其中 $x=\frac{X}{Z}$,$y=\frac{Y}{Z}$.例如,$P(x,y)$的标准射影坐标可以表示为$(x:y:1)$.

由于超奇异椭圆曲线受到 MOV 攻击[23]的威胁,我们主要研究非超奇异椭圆曲线.特征 3 上的非超奇异椭圆曲线仿射方程的等价标准射影形式的方程为

$$Y^2Z=X^3+aX^2Z+bZ^3.$$

椭圆曲线 E 的全部 K-有理点的集合添加一个无穷远点,记作 $E(K)=\{(x,y)\in K\times K \mid y^2=x^3+ax^2+b\}\cup\{\mathcal{O}\}$组成一个交换群.$\mathcal{O}$在仿射坐标表示下,没有具体的表现形式;$\mathcal{O}$在标准射影坐标下,可以用$(0:1:0)$表示.

有限域 $GF(3^m)$上的非超奇异椭圆曲线的群运算法则由弦切律定义如下:

若 $P=(x,y)\in E$,则$-P=(x,-y)\in E$.

单位元为无穷远点$\mathcal{O}$.对任意的点 P,有下式成立 $P+\mathcal{O}=P$.

令 $P_i=(x_i,y_i)\in E$,$i=1,2,3$,$P_3=P_1+P_2$,则

$$x_3=\begin{cases}\left(\frac{y_2-y_1}{x_2-x_1}\right)^2-x_1-x_2-a, & P_1\neq P_2\\ \left(\frac{ax_1}{y_1}\right)^2+x_1-a, & P_1=P_2\end{cases},$$

$$y_3=\begin{cases}\left(\frac{y_2-y_1}{x_2-x_1}\right)(x_1-x_3)-y_1, & P_1\neq P_2\\ \left(\frac{ax_1}{y_1}\right)(x_1-x_3)-y_1, & P_1=P_2\end{cases}.$$

从上述公式可以看出,特征 3 有限域上的椭圆曲线的点加的开销为有限域上的 1 个求逆、2 个乘法和 1 个平方;倍点的开销为有限域上的 1 个求逆、3 个乘法和 1 个平方.由于特征 3 有限域上的加法、减法相对于乘法、平方和求逆等运算花费的时间可以忽略不计.因而,这两种运算的能量消耗是有显著区别的.利用二进制展开计算椭圆曲线标量乘 kP 时,算法中的每个循环都执行一次倍点运算,加法运算则由该二进制比特的值来决定:当二进制比特为 l 时,执行一次加法运算操作;当二进制比特为 0 时,不执行加法运算操作.由于执行的操作不同,简单能量分析能够根据二进制计算标量乘的算法判断出该次循环是只执行了一个倍点运算,还是执行了一个倍点和一个点加运算,进而确定该二进制位是 0,还是 1.因而,简单能量分析利用算法执行过程中消耗的能量进而推断出标量 k 的比特信息.

由于不带防护的标量乘算法无法有效地抵抗简单能量分析,Montgomery 在 1987 年就提出了 Montgomery 算法用于抵抗简单能量分析,同时兼顾算法的效率.Montgomery 算法通过在每个循环中均存在一个点加和倍点运算的方法,让攻击者无法甄别出该二进制比特位的信息,从而有效防止简单能量分析.下文中我们首先回顾一些现有的 Montgomery 算法.

3 仿射和射影的 Montgomery 算法

算法 1 给出了经典的 Montgomery 算法[17].此 Montgomery 算法中的标量 k 采用二进制表示 $k=(k_{l-1},\cdots,k_0)_2$,其中 $k_{l-1}=1$,$k_i\in\{0,1\}$,$i=0,1,\cdots,l-2$,点 $P\in E(GF(q))$,其中 q 为一个素数的方幂.也就是说该算法适用于特征为 2 的有限域、特征为 3 的有限域和特征大于等于 5 的有限域上的椭圆曲线.

Montgomery 算法(算法 1)中的主要中间参数为 P_1,P_2.P_1,P_2的初始值为 P,$2P$,这时 P_1,P_2的关系为 $P_2-P_1=P$.算法 1 中在每轮循环的开始和结束均保持一个固有性质 $P_2-P_1=P$.这一性质是省略 y 坐标的计算的基础.由于 $k_{l-1}=1$,相当于 P_1 在开始时赋的初始值为$(k_{l-1})_2P$,P_1在循环过程中始终保存着当前计算的结果$(k_{l-1}k_{l-2}\cdots k_i)_2P$.

以利用算法 1 计算 $23P$ 为例.23 的二进制表示为 $k=(10111)_2$.循环开始之前,$l=5$,$k_4=1$,$k_3=0$,$k_2=1$,$k_1=1$,$k_0=1$,P_1被赋的初始值为$(k_4)_2P=P$,P_1被赋的初始值为$2P$.循环初始时,$i=l-2=3$.

(1) $i=3$ 时,$k_3=0$,此时执行 $P_2\leftarrow P_1+P_2=3P$,$P_1\leftarrow 2P_1=2P$;

(2) $i=2$ 时,$k_2=1$,此时执行 $P_1\leftarrow P_1+P_2=5P$,$P_2\leftarrow 2P_2=6P$;

(3) $i=1$ 时,$k_1=1$,此时执行 $P_1\leftarrow P_1+P_2=11P$,$P_2\leftarrow 2P_2=12P$;

(4) $i=0$ 时,$k_0=1$,此时执行 $P_1\leftarrow P_1+P_2=$

$23P$，$P_2 \leftarrow 2P_2 = 24P$.

循环结束.

可以看出该例子符合上面对算法的分析. 在每轮循环后 P_1 的值为 $(k_{l-1}k_{l-2}\cdots k_i)_2 P$，$P_2$ 的值为 P_1+P. 循环中，如果当前比特位为 0，则执行 $P_2 \leftarrow P_1+P_2$，$P_1 \leftarrow 2P_1$；如果当前比特位为 1，则执行 $P_1 \leftarrow P_1+P_2$，$P_2 \leftarrow 2P_2$. 由此可见. 该算法能够有效地计算出标量乘 kP 的结果. 并且无论当前比特位的值是 0，还是 1，算法 1 都能够只执行 1 次点加和 1 次倍点操作，并且 P_1 储存当前结果，P_2 的储存着 P_1+P 的结果.

由于算法 1 每轮循环中都恰好只执行了一次点加和倍点运算，这样就有效地抵抗了简单能量攻击. 在算法中，每行符号'//'后面的部分表示注释.

算法 1.　Montgomery 算法[17].

输入：$P \in E(GF(q))$ and $k=(k_{l-1},\cdots,k_0)_2$，$k_{l-1}=1$

输出：$kP \in E(GF(q))$

$P_1 \leftarrow P$，$P_2 \leftarrow 2P$；

FOR i from $l-2$ down to 0 do

IF $k_i=0$ THEN

$P_2 \leftarrow P_1+P_2$，$P_1 \leftarrow 2P_1$；//$P_2-P_1=P$

ELSE

$P_1 \leftarrow P_1+P_2$，$P_2 \leftarrow 2P_2$；//$P_2-P_1=P$

RETURN $P_1=(x_1,y_1)$

考虑到进一步提高标量乘的效率，Lopez 和 Dahab[24]优化了 $GF(2^m)$ 上椭圆曲线的 Montgomery 算法. 在 Montgomery 算法的每轮循环中，他们省略了 y 坐标的计算. 循环结束后恢复出 y 坐标，极大地降低了标量乘运算的开销. Okeya 和 Sakurai[25]把 Lopez 和 Dahab 的思想应用到特征大于等于 5 的有限域上 Montgomery 曲线上. 该技巧在 Montgomery 曲线上取得了良好的结果.

汪宏等人[18]也使用该技巧提出了特征 3 有限域上椭圆曲线的两类 Montgomery 算法，分别是仿射坐标下的 Montgomery 算法如算法 2 所示和射影坐标下的 Montgomery 算法如算法 3 所示. 本文中，仿射坐标下的 Montgomery 算法简记做仿射 Montgomery 算法，射影坐标下的 Montgomery 算法简记做射影 Montgomery 算法. 当特征 3 有限域上的求逆算法消耗的时间大于 7 个乘法时，射影 Montgomery 算法比仿射 Montgomery 算法快.

算法 2 和算法 1 的实质相同，在算法 1 的基础上省略了 y 坐标的计算，提高了运算效率. 算法 2 把算法 1 中的点加、倍点公式的具体形式给了出来. 在特征为 3 的有限域上，当 $k_i=0$ 时，只计算 P_1 的 x_1 坐标 $x_1 \leftarrow \dfrac{a^2x_1^2(x_2-x_1)^2}{(x_2-x_1)^2(x_1^3+ax_1^2+b)}+x_1-a$ 和 P_2 的 x_2 坐标 $x_2 \leftarrow \dfrac{(-(x_1+x_2)^3+ax_1x_2+b)(x_1^3+ax_1^2+b)}{(x_2-x_1)^2(x_1^3+ax_1^2+b)}+x_1+x_2-x$. 当 $k_i=1$ 时，也只计算 P_1 的 x_1 坐标 $x_1 \leftarrow \dfrac{(-(x_1+x_2)^3+ax_1x_2+b)(x_2^3+ax_2^2+b)}{(x_2-x_1)^2(x_2^3+ax_2^2+b)}+x_1+x_2-x$ 和 P_2 的 x_2 坐标 $x_2 \leftarrow \dfrac{a^2x_2^2(x_2-x_1)^2}{(x_2-x_1)^2(x_2^3+ax_2^2+b)}+x_2-a$. 从 P_1 的 x_1 坐标和 P_2 的 x_2 坐标的计算可以看出计算过程中用不到 y 坐标. 但是，在循环结束之后，就需要恢复出 $P_1=kP$ 的 y 坐标. 因为算法 2 中一直暗含关系式 $P_2-P_1=P$，并且知道 P 的 x 坐标、P_1 的 x_1 坐标和 P_2 的 x_2 坐标，这样就可以通过椭圆曲线的方程，恢复出 $P_1=kP$ 的 y 坐标 $y_1 \leftarrow \dfrac{(x_1-x)^2x_2+(a-x-x_1)xx_1+b}{y}$. 这样算法 2 的实质就和算法 1 相同.

算法 2.　仿射坐标下 $GF(3^m)$ 上椭圆曲线 Montgomery 算法[18].

输入：$P(x,y) \in E(GF(3^m))$ and $k=(k_{l-1},\cdots,k_0)_2$

输出：$kP \in E(GF(3^m))$

$x_1 \leftarrow x$，$x_2 \leftarrow \dfrac{a^2x^2}{x^3+ax^2+b}+x-a$，

FOR i from $l-2$ down to 0 do

IF $k_i=0$ THEN

$x_1 \leftarrow \dfrac{a^2x_1^2(x_2-x_1)^2}{(x_2-x_1)^2(x_1^3+ax_1^2+b)}+x_1-a$，

$x_2 \leftarrow \dfrac{(-(x_1+x_2)^3+ax_1x_2+b)(x_1^3+ax_1^2+b)}{(x_2-x_1)^2(x_1^3+ax_1^2+b)}+x_1+x_2-x$，

//$P_2 \leftarrow P_1+P_2$，$P_1 \leftarrow 2P_1$

ELSE

$x_1 \leftarrow \dfrac{(-(x_1+x_2)^3+ax_1x_2+b)(x_2^3+ax_2^2+b)}{(x_2-x_1)^2(x_2^3+ax_2^2+b)}+x_1+x_2-x$，

$x_2 \leftarrow \dfrac{a^2x_2^2(x_2-x_1)^2}{(x_2-x_1)^2(x_2^3+ax_2^2+b)}+x_2-a$，

//$P_1 \leftarrow P_1+P_2$，$P_2 \leftarrow 2P_2$

$y_1 \leftarrow \dfrac{(x_1-x)^2x_2+(a-x-x_1)xx_1+b}{y}$，

//恢复 y_1 坐标

RETURN $P_1=(x_1,y_1)$.

算法 3 也和算法 1 的实质相同，在算法 2 的基础上，利用射影坐标省略了求逆运算，进一步提高了特征 3 有限域上 Montgomery 算法的运行效率. 具体情况如下. 把算法 2 中的 x_1 坐标表示为 X_1/Z_1，

x_2 坐标表示为 X_2/Z_2，即得到算法 3 中的 P_1 点的 X_1, Z_1 坐标和 P_2 点的 X_2, Z_2 坐标. 算法 3 的最后需要计算 P_1 点的 x_1 坐标 $x_1 \leftarrow X_1(yZ_2)(yZ_1Z_2)^{-1}$ 和 P_1 点的 y_1 坐标 $y_1 \leftarrow [Z_1(x_1-x)^2X_2+((a-x-x_1)xx_1+b)Z_1Z_2](yZ_1Z_2)^{-1}$，从而正确的返回 $P_1=kP$ 的仿射坐标值.

算法 3. 射影坐标下 $GF(3^m)$ 上椭圆曲线 Montgomery 点乘[18].

输入：$P(x,y)\in E(GF(3^m))$ and $k=(k_{l-1},\cdots,k_0)_2$

输出：$kP\in E(GF(3^m))$

$X_1\leftarrow x$, $Z_1\leftarrow 1$,

$X_2\leftarrow x^4+bx-ab$, $Z_2\leftarrow x^3+ax^2+b$.

For i from $l-2$ down to 0 do

IF $k_i=0$ THEN

$T\leftarrow Z_2$,

$X_2\leftarrow -(X_1T+X_2Z_1)X_1X_2+$
$(aX_1X_2+bZ_1T)Z_1T-xZ_2^2$,

$T\leftarrow Z_1$, $Z_1\leftarrow (X_1^3+aX_1^2Z_1+bZ_1^3)Z_1$,

$X_1\leftarrow X_1X_1^3+bT^3(X_1-aT)$,

//$P_2\leftarrow P_1+P_2$, $P_1\leftarrow 2P_1$

ELSE

$T\leftarrow Z_1$, $Z_1\leftarrow (X_2Z_1-X_1Z_2)^2$,

$X_1\leftarrow -(X_2T+X_1Z_2)X_1X_2+$
$(aX_1X_2+bZ_2T)Z_2T-xZ_1^2$,

$T\leftarrow Z_2$, $Z_2\leftarrow (X_2^3+aX_2^2Z_2+bZ_2^3)Z_2$,

$X_2\leftarrow X_2X_2^3+bT^3(X_2-aT)$,

//$P_1\leftarrow P_1+P_2$, $P_2\leftarrow 2P_2$

$x_1\leftarrow X_1(yZ_2)(yZ_1Z_2)^{-1}$,

$y_1\leftarrow [Z_1(x_1-x)^2X_2+((a-x-x_1)xx_1+b)Z_1Z_2]\cdot$
$(yZ_1Z_2)^{-1}$,

RETURN $P_1=(x_1,y_1)$.

在特征 3 有限域上的仿射 Montgomery 算法和射影 Montgomery 算法的基础上，我们提出了更高效的抗简单能量分析的 Montgomery 算法.

4 特征 3 有限域上椭圆曲线的 co-Z Montgomery 算法

本节利用 co-Z 技巧、射影坐标表示和循环中间阶段不计算 Y 坐标的技巧改进有限域 $GF(3^m)$ 上椭圆曲线的点加和倍点公式，进而构造 co-Z Montgomery 算法.

Montgomery 算法的输入和输出都是仿射坐标表示的有理点. 利用射影坐标表示，就需要先把仿射坐标表示的点转换为射影坐标表示的点. 一般情况下，$P(x,y)$ 的标准射影坐标可以表示为 $(x:y:1)$. 本文中也采取这种转换方式. 在计算过程中，借助标准射影坐标能有效地减少有限域中的求逆运算次数. 这在特征 3 有限域上的求逆运算比乘法、平方等运算的花销更大的情况下是意义深远的，并且能够有效地提高标量乘的计算速度. 在标量乘计算结束时，我们就需要把 kP 的射影坐标表示 $(\alpha:\beta:\gamma)$ 转换为仿射坐标表示 $\left(\frac{\alpha}{\gamma},\frac{\beta}{\gamma}\right)$. 这样算法就成功地完成了从仿射坐标点的输入，到返回的结果也是仿射坐标点的过程.

在 co-Z Montgomery 算法的每轮循环开始时都依然和算法 1 的结构相同，有关系式 $P_2=P_1+P$. 由于这个关系的存在，我们就可以利用椭圆曲线方程和 $P_2=P_1+P$，得到只需要 X, Z 坐标的射影坐标下的点加、倍点公式，这样就可以在循环中不计算中间的点的 Y 坐标. 循环结束时，也是因为知道 P_1 的 X_1, Z_1 坐标、P_2 的 X_2, Z_2 坐标、P 点的坐标 $(x:y:1)$ 以及 $P_2=P_1+P$，再由特征 3 上的椭圆曲线方程，就可以求出 P_1 的 Y_1 坐标. 在循环结束时，P_1 点储存着标量乘 kP 的值，这时已经求出了 P_1 的坐标 $(X_1:Y_1:Z_1)$，我们只需要把标准射影坐标表示转换为仿射坐标表示，就可以得到标量乘 kP 的计算结果 $\left(\frac{X_1}{Z_1},\frac{Y_1}{Z_1}\right)$. 所以，我们可以只在循环结束的时候恢复出 P_1 的 Y_1 坐标，就成功的实现了中间阶段不计算 Y 坐标的过程.

应用标准射影坐标表示时，可以保持 Montgomery 算法每轮循环结束时点加运算结果的 Z 坐标和倍点运算结果的 Z 坐标相同. 具体情况是循环开始时，P_1 点和 P_2 点的 Z 坐标相同，循环结束时，P_1 点和 P_2 点的 Z 坐标也相同. 能做到这点的原因如下. 如果 P_1 点的射影坐标表示为 $(X_1:Y_1:Z_1)$，P_2 点的射影坐标表示为 $(X_2:Y_2:Z_2)$，则 P_1 点的仿射坐标表示为 $\left(\frac{X_1}{Z_1},\frac{Y_1}{Z_1}\right)$，$P_2$ 点的仿射坐标表示为 $\left(\frac{X_2}{Z_2},\frac{Y_2}{Z_2}\right)$. 很明显 P_1 点的仿射坐标表示也可以写成 $\left(\frac{X_1Z_2}{Z_1Z_2},\frac{Y_1Z_2}{Z_1Z_2}\right)$，$P_2$ 点的仿射坐标表示也可以写成 $\left(\frac{X_2Z_1}{Z_1Z_2},\frac{Z_1Y_2}{Z_1Z_2}\right)$. 再转换为射影坐标表示，就得到 P_1 点的射影坐标表示为 $(X_1Z_2:Y_1Z_2:Z_1Z_2)$，P_2 点的射影坐标表示为 $(X_2Z_1:Y_2Z_1:Z_1Z_2)$. 这样就成功地把两个 Z 坐标互不相同的点转换为 Z 坐标相同的点. 由于每轮循环的结果 P_1 点和 P_2 点的 Z 坐标相

同，下轮循环 P_1 点和 P_2 点作为输入，很自然地有每轮循环开始之前的 P_1 点和 P_2 点的 Z 坐标相同. 在 P_1 点和 P_2 点的 Z 坐标相同时，计算点加和倍点的效率能够明显提高. 这也是我们提出效率更高的抗简单能量分析的标量乘算法的基础.

基于上述利用标准射影坐标表示、循环中间阶段不计算 Y 坐标和每轮循环的输入输出点的 Z 坐标相同 3 个技巧，我们给出了射影坐标下特征 3 有限域上椭圆曲线的 co-Z Montgomery 标量乘算法，如算法 4 所示. 后文中，我们把该算法简记作 co-Z Montgomery 算法.

算法 4.　射影坐标下 $GF(3^m)$ 椭圆曲线上 co-Z Montgomery 算法.

输入：$P(x,y)\in E(GF(3^m))$ and $k=(k_{l-1},\cdots,k_0)_2$

输出：$kP\in E(GF(3^m))$

$X_1\leftarrow x(x^3+ax^2+b)$, $X_2\leftarrow x^4+b(x-a)$,

$Z\leftarrow x^3+ax^2+b$.

FOR i from $l-2$ down to 0 do

IF $k_i=0$ THEN

$Z'\leftarrow[(X_2-X_1)^2Z][(X_1+aZ)X_1^2+bZ^3]$,

$X_{2'}\leftarrow[(aZ-X_1-X_2)X_1X_2+bZ^3-x[(X_2-X_1)^2Z]][(X_1+aZ)X_1^2+bZ^3]$,

$X_{1'}\leftarrow(X_2-X_1)^2[bZ^3(X_1-aZ)+X_1^4]$,

//$P_2\leftarrow P_1+P_2$, $P_1\leftarrow 2P_1$

ELSE

$Z'\leftarrow[(X_2-X_1)^2Z][(X_2+aZ)X_2^2+bZ^3]$,

$X_{1'}\leftarrow[(aZ-X_1-X_2)X_1X_2+bZ^3-x(X_2-X_1)^2Z][(X_2+aZ)X_2^2+bZ^3]$,

$X_{2'}\leftarrow(X_2-X_1)^2[bZ^3(X_2-aZ)+X_2^4]$,

//$P_1\leftarrow P_1+P_2$, $P_2\leftarrow 2P_2$

$Z\leftarrow Z'$, $X_1\leftarrow X_{1'}$, $X_2\leftarrow X_{2'}$,

$$x_1\leftarrow\frac{X_1}{Z},$$

$$y_1\leftarrow\frac{(X_1-xZ)^2X_2+((a-x)Z-X_1)xX_1Z+bZ^3}{yZ^3},$$

RETURN $P_1=(x_1,y_1)$.

算法 4 以仿射坐标下特征 3 有限域上椭圆曲线上的点 P 和标量 k 的二进制表示 $k=(k_{l-1},\cdots,k_0)_2$ 作为输入，算法的输出为标量乘 kP 的值. 我们仍然利用 Montgomery 算法的框架，加入效率更高的计算公式，达到提高标量乘计算速度的目的. 类似于算法 1、2 和算法 3，首先计算 $P_1=P$, $P_2=2P$ 并且保证 P_1 一直储存着当前的计算结果，P_2 满足关系式 $P_2=P_1+P$.

计算 $P_1=P$, $P_2=2P$ 并且保证 P_1 和 P_2 的 Z 坐标相同，保证了该算法在 while 循环之前的初始 Z 坐标相同. 算法要求每轮循环的开始时，满足 $P_2=P_1+P$ 并且 P_1 和 P_2 的 Z 坐标相同. 该初始化的过程的 P_1, P_2 的射影坐标中的 X, Z 坐标的计算公式如引理 1 所示.

引理 1.　在算法 4 的初始化阶段，$P_1=P$, $P_2=2P$, $P_i=(x_i,y_i)=(X_i:Y_i:Z)$, $i=1,2$，其中 $x_i=\frac{X_i}{Z}$, $y_i=\frac{Y_i}{Z}$，且满足 $P=P_2-P_1$，记 P 的坐标表示为 $P(x,y)=(x:y:1)$. 则 P_1, P_2 的 X, Z 坐标可以由如下公式表示：

$$\begin{cases}X_1=x(x^3+ax^2+b)\\ X_2=x^4+b(x-a)\\ Z=x^3+ax^2+b\end{cases}.$$

证明.　$P_1=P$ 可得 P_1 的仿射坐标表示为 (x,y).

$P_2=2P$ 可得 P_2 的仿射坐标 (x_2,y_2) 中的 x_2 的具体表达式为 $x_2=\frac{a^2x^2}{y^2}+x-a$.

由仿射坐标和射影坐标的转换关系，我们知道

$$\begin{cases}x=\frac{X_1}{Z}\\ x_2=\frac{X_2}{Z}\end{cases}.$$

为了求出 P_1, P_2 点的 X, Z 坐标，我们需要对 P_1, P_2 点的仿射坐标形式进行统分. 也就是 P_1 的仿射坐标 (x,y) 表示为 $\left(\frac{xy^2}{y^2},y\right)$，把 P_2 的仿射坐标表示为 $\left(\frac{a^2x^2}{y^2}+x-a,y_2\right)$，等价于 $\left(\frac{a^2x^2+y^2(x-a)}{y^2},y_2\right)$. 由于只需要考虑 X, Z 坐标，我们可以不用考虑 y 坐标.

这样就得到

$$\begin{cases}X_1=xy^2\\ X_2=a^2x^2+y^2(x-a)\\ Z=y^2\end{cases}.$$

又由于特征 3 有限域上的椭圆曲线的方程为

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

故 $\begin{cases}X_1=xy^2=x(x^3+ax^2+b)\\ X_2=a^2x^2+y^2(x-a)=x^4+b(x-a)\\ Z=x^3+ax^2+b\end{cases}$，

可得引理正确.　证毕.

由引理 1 知初始化后的 P_1, P_2 点的 Z 坐标相同，并且有 $P_1=P$, $P_2=2P$，也就是 P_1, P_2 点满足关系式 $P_2=P_1+P$. 这种情况下，进入循环后，在算法 4 的每轮循环中，如果 $k_i=0$，执行操作 $P_1'=2P_1$, $P_2'=P_1+P_2$，否则执行操作 $P_1'=P_1+P_2$, $P_2'=2P_2$. 每轮

循环都执行一次点加 P_1+P_2，都执行一次倍点 $2P_1$ 或者 $2P_2$. 由于在每轮循环的开始时，我们有 P_1，P_2 点的 Z 坐标相同，这样我们就可以利用 P_1，P_2 点的 Z 坐标相同的特点来构造 P_1+P_2 的公式，再根据 k_i 的值，确定其倍点执行的操作是 $2P_1$ 还是 $2P_2$. 根据 k_i 的不同分类，相应 P_1'，P_2' 的计算由引理 2 给出.

每轮循环结束得到结果 P_1'，P_2' 满足 $P_2'=P_1'+P$ 并且 Z 坐标相同. P_1'，P_2' 分别赋值给 P_1，P_2，进入下一轮循环. 这样保证了 co-Z Montgomery 算法得以正确运行. 循环结束后，恢复出 P_1 点的 Y_1 坐标.

算法循环结束后恢复出 P_1 的 $y_1=\dfrac{Y_1}{Z_1}$ 坐标的公式如引理 3 所示. 在 co-Z Montgomery 算法返回时，将射影坐标转换为仿射坐标.

在每轮循环的开始时 $P_2=P_1+P$ 并且 P_1，P_2 有共同的 Z 坐标. 经过引理 2 的公式计算之后，无论是 $k_i=0$ 时，$P_1'=2P_1$，$P_2'=P_1+P_2$，还是 $k_i=1$ 时，$P_1'=P_1+P_2$，$P_2'=2P_2$，每轮循环结束时，输出 P_1'，P_2' 也需要有相同的 Z 坐标并且满足 $P_2'=P_1'+P$. 在 k_i 不同的情况下 P_1'，P_2' 的 X，Z 坐标的详细计算公式如引理 2 所示.

引理 2. 记在算法 4 的每个循环节中的输入为 $P_i=(x_i,y_i)=(X_i:Y_i:Z)$，$i=1,2$，其中 $x_i=\dfrac{X_i}{Z}$，$y_i=\dfrac{Y_i}{Z}$，且满足 $P=P_2-P_1$，记 P 的坐标表示为 $P=(x,y)=(x:y:1)$. 设算法每个循环节的输出为 $P_{i'}=(x_i',y_i')=(X_i':Y_i':Z')$，$i=1,2$，则 P_1'，P_2' 的 X，Z 坐标可以由如下公式表示：

当 $P_1'=2P_1$，$P_2'=P_1+P_2$ 时，

$$\begin{cases}X_2'=[(aZ-X_1-X_2)X_1X_2+bZ^3-x(X_2-X_1)^2Z]\\ \qquad (X_1^3+aX_1^2Z+bZ^3)\\ X_1'=(X_2-X_1)^2[bZ^3(X_1-aZ)+X_1^4]\\ Z'=(X_2-X_1)^2Z(X_1^3+aX_1^2Z+bZ^3)\end{cases}.$$

当 $P_1'=P_1+P_2$，$P_2'=2P_2$ 时，

$$\begin{cases}X_1'=[(aZ-X_1-X_2)X_1X_2+bZ^3-x(X_2-X_1)^2Z]\\ \qquad (X_2^3+aX_2^2Z+bZ^3),\\ X_2'=(X_2-X_1)^2[bZ^3(X_2-aZ)+X_2^4],\\ Z'=(X_2-X_1)^2Z(X_2^3+aX_2^2Z+bZ^3)\end{cases}.$$

证明. 当 $P_1'=2P_1$，$P_2'=P_1+P_2$ 时，

$$x_2'=\left(\frac{y_2-y_1}{x_2-x_1}\right)^2-x_1-x_2-a,$$

$$x=\left(\frac{y_2+y_1}{x_2-x_1}\right)^2-x_1-x_2-a.$$

将两式相加得

$$x+x_2'=\frac{2(y_1^2+y_2^2)}{(x_2-x_1)^2}-2(x_1+x_2+a).$$

由于域的特征为 3，有

$$x+x_2'=-\frac{y_1^2+y_2^2}{(x_2-x_1)^2}+(x_1+x_2+a) \tag{2}$$

又 $y_1^2=x_1^3+ax_1^2+b$，$y_2^2=x_2^3+ax_2^2+b$，代入式(2)，得

$$x_2'=\frac{-x_1x_2(x_1+x_2)+ax_1x_2+b-x(x_2-x_1)^2}{(x_2-x_1)^2} \tag{3}$$

由于 $x_1=\dfrac{X_1}{Z}$，$x_2=\dfrac{X_2}{Z}$，$x_2'=\dfrac{X_{2'}}{Z'}$，代入 x_2' 的表达式(3)，得

$$\frac{X_{2'}}{Z'}=\frac{(aZ-X_1-X_2)X_1X_2+bZ^3-x(X_2-X_1)^2Z}{(X_2-X_1)^2Z} \tag{4}$$

因为 $P_1'=2P_1$，所以

$$x_1'=\left(\frac{ax_1}{y_1}\right)^2+x_1-a,$$

由于 $y_1^2=x_1^3+ax_1^2+b$，此时得 x_1' 的表达式为

$$x_1'=\frac{a^2x_1^2}{x_1^3+ax_1^2+b}+x_1-a=\frac{x_1^4+b(x_1-a)}{(x_1^3+ax_1^2+b)} \tag{5}$$

又 $x_1=\dfrac{X_1}{Z}$，$x_1'=\dfrac{X_1'}{Z'}$，代入 x_1' 的表达式(5)，得

$$\frac{X_1'}{Z'}=\frac{bZ^3(X_1-aZ)+X_1^4}{Z(X_1^3+aX_1^2Z+bZ^3)} \tag{6}$$

由式(4)和(6)可以得到 P_1'，P_2' 的 X，Z 坐标由如下公式表示

$$\begin{cases}X_2'=[(aZ-X_1-X_2)X_1X_2+bZ^3-x(X_2-X_1)^2Z]\\ \qquad (X_1^3+aX_1^2Z+bZ^3)\\ X_1'=(X_2-X_1)^2[bZ^3(X_1-aZ)+X_1^4]\\ Z'=(X_2-X_1)^2Z(X_1^3+aX_1^2Z+bZ^3)\end{cases}.$$

当 $P_1'=P_1+P_2$，$P_2'=2P_2$ 时，

$$x_1'=\left(\frac{y_2-y_1}{x_2-x_1}\right)^2-x_1-x_2-a,$$

$$x=\left(\frac{y_2+y_1}{x_2-x_1}\right)^2-x_1-x_2-a.$$

将上述两式相加，并考虑到域的特征为 3，有

$$x+x_1'=-\frac{y_1^2+y_2^2}{(x_2-x_1)^2}+(x_1+x_2+a).$$

又 $y_1^2=x_1^3+ax_1^2+b$，$y_2^2=x_2^3+ax_2^2+b$，因而

$$x_1'=\frac{-x_1x_2(x_1+x_2)+ax_1x_2+b-x(x_2-x_1)^2}{(x_2-x_1)^2}.$$

因为 $x_1=\dfrac{X_1}{Z}$，$x_2=\dfrac{X_2}{Z}$，$x_2'=\dfrac{X_2'}{Z'}$，所以得

$$\frac{X_1'}{Z'}=\frac{(aZ-X_1-X_2)X_1X_2+bZ^3-x(X_2-X_1)^2Z}{(X_2-X_1)^2Z}.$$

又 $P_2'=2P_2$，所以

$$x_2'=\left(\frac{ax_2}{y_2}\right)^2+x_2-a,$$

因为 $y_2^2=x_2^3+ax_2^2+b$，此时得 x_2'的表达式为

$$x_{2'}=\frac{a^2x_2^2}{x_2^3+ax_2^2+b}+x_2-a=\frac{x_2^4+b(x_2-a)}{(x_2^3+ax_2^2+b)}.$$

又 $x_2=\frac{X_2}{Z}$，$x_2'=\frac{X_2'}{Z'}$，故有

$$\frac{X_2'}{Z'}=\frac{bZ^3(X_2-aZ)+X_2^4}{Z(X_2^3+aX_2^2Z+bZ^3)}.$$

由于$\frac{X_1'}{Z'}$和$\frac{X_2'}{Z'}$的表达式已经求出，只需要把两表达式的分母相通，即可得到 X_1'，X_2'，Z'坐标的表达式

$$\begin{cases}X_1'=[(aZ-X_1-X_2)X_1X_2+bZ^3-x(X_2-X_1)^2Z]\\ \qquad(X_2^3+aX_2^2Z+bZ^3)\\ X_2'=(X_2-X_1)^2[bZ^3(X_2-aZ)+X_2^4]\\ Z'=(X_2-X_1)^2Z(X_2^3+aX_2^2Z+bZ^3)\end{cases}.$$

证毕.

在算法 4 中，当 $k_i=0$ 时，执行公式

$$\begin{cases}X_{2'}=[(aZ-X_1-X_2)X_1X_2+bZ^3-x(X_2-X_1)^2Z]\\ \qquad(X_1^3+aX_1^2Z+bZ^3)\\ X_{1'}=(X_2-X_1)^2[bZ^3(X_1-aZ)+X_1^4]\\ Z'=(X_2-X_1)^2Z(X_1^3+aX_1^2Z+bZ^3)\end{cases}.$$

当 $k_i=1$ 时，执行公式

$$\begin{cases}X_1'=[(aZ-X_1-X_2)X_1X_2+bZ^3-x(X_2-X_1)^2Z]\\ \qquad(X_2^3+aX_2^2Z+bZ^3)\\ X_2'=(X_2-X_1)^2[bZ^3(X_2-aZ)+X_2^4]\\ Z'=(X_2-X_1)^2Z(X_2^3+aX_2^2Z+bZ^3)\end{cases}.$$

由此可见，无论是 $k_i=0$，还是 $k_i=1$，算法 4 的每轮循环中执行的运算的花销都是相同的. 并且该过程与算法 3 的每一轮循环相比，少计算了一个 Z 坐标，其点加的公式也比算法 3 中的点加的花销少.

引理 2 保证了在 co-Z Montgomery 算法的每轮循环结束时，P_1'，P_2'分别赋给下轮循环的 P_1，P_2，此时依然满足 $P_2=P_1+P$. 由于 P_1'，P_2'的 Z 坐标相同，故下轮循环开始时 P_1，P_2的 Z 坐标相同. 引理 1 给出了初始的 P_1，P_2点的 X，Z 坐标的表达式，并且其 Z 坐标相同，这样整个循环就能够正确的运行.

循环结束后，需要返回的是 kP 的计算结果，由于 P_1点中储存着 kP 的值，因而需要返回 $P_1(x_1,y_1)$ 的值. 循环过程中使用的是标准射影坐标表示 $P_1(X_1:Y_1:Z)$. 并且 Y_1还未求出，X_1，Z 坐标的表达式，已经通过 For 循环给出. 由于 $x_1=\frac{X_1}{Z}$可以直接算出，接下来需要通过一些已知量计算出 $y_1=\frac{Y_1}{Z}$的表达式.

引理 3 给出了已知 $P_2=P_1+P$，P_1点的 X_1，Z 坐标，P_2 点的 X_2，Z 坐标并且其 Z 坐标相同时 $y_1\left(y_1=\frac{Y_1}{Z}\right)$的计算公式.

引理 3.　令 $P_i=(x_i,y_i)=(X_i:Y_i:Z)$，$i=1,2$，$P=(x,y)$，且 $P_2-P_1=P$. 则 P_1的 y_1坐标为

$$y_1=\frac{(X_1-xZ)^2X_2+((a-x)Z-X_1)xX_1Z+bZ^3}{yZ^3}.$$

证明.　根据椭圆曲线上的点加公式，由于 $P_2=P_1+P$，有

$$x_2=\left(\frac{y_1-y}{x_1-x}\right)^2-x_1-x-a,$$

所以

$$(y_1-y)^2=(x_1-x)^2(x_1+x_2+x+a).$$

又 $y_1^2=x_1^3+ax_1^2+b$，$y^2=x^3+ax^2+b$，得

$$\begin{aligned}(y_1-y)^2&=y_1^2+y^2-2y_1y_2\\&=y_1^2+y^2+y_1y_2\\&=x_1^3+x_2^3+a(x_1^2+x_2^2)-b+y_1y_2.\end{aligned}$$

因此，

$$\begin{aligned}&(x_1^3+x^3)+a(x_1^2+x^2)-b+y_1y=\\&\quad(x_1^2+x^2+xx_1)x_2+x_1^3+(x^2+xx_1)x_1+x^3+\\&\quad(x_1^2+xx_1)x+a(x_1^2+x^2)+axx_1.\end{aligned}$$

故知 $y_1y=(x_1-x)^2x_2+(a-x-x_1)xx_1+b$，因而有

$$y_1=\frac{(x_1-x)^2x_2+(a-x-x_1)xx_1+b}{y},$$

代入 $x_1=\frac{X_1}{Z}$，$x_2=\frac{X_2}{Z}$，得到

$$y_1=\frac{(X_1-xZ)^2X_2+((a-x)Z-X_1)xX_1Z+bZ^3}{yZ^3}.$$

证毕.

co-Z Montgomery 算法执行完 for 循环之后，已经得到 $P_1=kP$ 的 X_1，Z 坐标. 根据引理 3，恢复出 P_1的 y_1坐标. 算法最后返回 P_1的仿射坐标表示 $\left(\frac{X_1}{Z},y_1\right)$.

接下来，我们将对利用统一 Z 坐标表示、使用标准射影坐标表示和循环中间阶段不计算 Y 坐标 3 个技巧改进的 co-Z Montgomery 算法进行效率分析，并对本文列出的 4 个算法进行实验分析及相互之间的效率比较.

5 算法效率分析

考虑到比较的公平性与合理性，与文献[18]一样，采用文献[22]中 Smart 和 Westwood 给出的定义在 $GF(3^{97})=GF_3[\theta]/(\theta^{97}+\theta^{12}+2)$ 上的椭圆曲线，其方程为 $E: y^2=x^3+x^2+b$，其中 $b=$ 0x5C6A21D1BF0967068295B8EAA7253DD2BD7A72. 系数 b 按三进制展开对应多项式基表示. 该椭圆曲线的阶为 $\sharp E(GF(3^{97}))=3\times 636268544113594235847488166718193849291632297 9$. 在椭圆曲线方程 $y^2=x^3+ax^2+b$ 中，一般情况下 a 较小(本实例中 $a=1$)，因此不同算法中的计算公式中的 a 乘以域中元素的运算消耗的时间可以忽略. 而当 b 很大时，以 b 为乘数的乘法看作一个普通乘法 M. 因为特征 3 有限域上的加法运算与乘法、立方、求逆等运算相比，其计算时间很少，为了方便比较，将其忽略不计.

为了便于和文献[18]的结果进行公平合理的比较，我们主要选择两种实验平台. 其中的一种测试平台为文献[18]的实验平台. 文献[18]中的测试环境为 1096.728 MHz 英特尔奔腾 III；多精度运算函数库是 GMP① 4.2.2. 另一种测试平台为 Sage 云平台②，采用浏览器作为 GUI 界面，融入了云计算的思想，可以在线使用数学软件 Sage 来实现特征 3 有限域上椭圆曲线的标量乘算法.

表 1 中列出了文献[18]中 $GF(3^m)$ 上基础运算的时间和 Sage 云平台的基础运算时间. 特征 3 有限域上的平方采用与域上乘法相同的算法. 也就是说平方和乘法花费的时间相同.

表 1 不同平台上特征 3 域上的基础运算运行时间

(单位：μs)

	文献[18]的结果	Sage 云平台运行的结果
I	10121	149.0
M	1157	18.7
C	1584	32.1

从表 1 中可知：文献[18]中，$S/M=1$，$C/M=1.37$，$I/M=8.75$；Sage 云平台上，$S/M=1$，$C/M=1.72$，$I/M=7.97$. 这两种平台上，乘法、平方、立方和求逆的比例符合特征 3 有限域上的一般估计.

在算法 2、3、4(分别对应仿射 Montgomery 算法、射影 Montgomery 算法、co-Z Montgomery 算法)中，恢复 y 坐标的时间和从射影坐标恢复成仿射坐标的时间，其花费分别为 $I+4M+S$，$I+11M+S$，$I+10M+S+C$. 当标量 k 的位数较大时，忽略循环之外的计算时间，包括循环开始前计算 P_1，P_2 的 X，Z 坐标和循环结束后恢复 y 坐标的时间及其从射影坐标恢复成仿射坐标的时间. 这 3 个算法的各自的每轮循环都相同，包含一个点加运算和一个倍点运算，因此标量 k 的二进制权重对算法间的比较没有影响. 每轮循环的运算时间直接反映了整个标量乘 kP 的花销. 因此，比较这几个算法之间标量乘的相对运行时间，只需比较每轮循环的运行时间即可.

在算法 1 中，利用原始的特征 3 有限域上的点加和倍点公式. 其点加的花销为 $I+2M+S$，倍点的花销为 $I+3M+S$，每轮循环的花销为 $2I+5M+2S$. 该过程虽然不需要恢复 y 坐标，省略了 y 坐标的计算花销，但是其每轮循环的花费太多. 我们考虑算法 1 时，其点加和倍点使用较新的文献[19]中的计算公式. 文献[19]中给出了雅可比坐标下的点加、倍点公式，其点加的花销为 $11M+C$，倍点的花销为 $3M+2C$. 每轮循环的花费为 $14M+2C$. 这种没有利用 Montgomery 算法的特性，只在普通意义下优化的点加、倍点公式应用在 Montgomery 算法中是效率低下的. 域上的平方和乘法使用相同的操作时，我们提出的算法 4 每轮循环的花费比算法 1 少 $I+C-5M$，比算法 2 少 $C+2M$，比算法 3 少 $2C+M$. 算法 1 利用文献[19]中的计算公式，循环结束之后，还需要把雅可比坐标转换为仿射坐标. 雅可比坐标表示转换为仿射坐标表示的过程需要花费 $(I+M+S)+(I+M+C)=2I+2M+S+C$. 这样利用文献[19]的公式的算法 1 的优势就比较小.

通过上面的分析，比较算法 1～4 的效率时，我们只需要比较循环阶段的运行时间. 由于标量相同，也就是说循环的轮数相同，又由于各自算法的每轮循环都执行一次点加和倍点，因此只需要比较各自算法每轮循环的花费即可.

不同的 Montgomery 算法的循环阶段的每轮循环的花销如表 2 所示. 当 $S=M$ 时，理论结果表明 co-Z Montgomery 算法比利用文献[19]的公式的原 Montgomery 算法节省的计算量为 $I+C-5M$，比射影 Montgomery 算法节省 $C+2M$，比利用文献[19]的公式的 Montgomery 算法快 $2C+M$.

① The GNU Multiple Precision Arithmetic Library. https://gmplib.org/

② SageMathCloud? collaborative computational mathematics. https://cloud.sagemath.com/

表 2　不同的 Montgomery 算法运行时间

算法	每轮循环的花费	文献[20]环境的每轮循环的运行时间/μs	Sage 云平台每轮循环的运行时间/μs
算法 2	$I+6M+2C+2S$	22545	362.8
算法 3	$13M+2C+2S$	20523	344.7
利用文献[19]的公式的算法 1	$14M+3C$	20950	358.1
算法 4	$10M+C+3S$	16625	275.2

从表 2 中的结果可以看出：在文献[18]中的环境下，co-Z Montgomery 算法比仿射 Montgomery 算法快 26.3%；比射影 Montgomery 算法快 19.0%；比利用文献[19]的公式的 Montgomery 算法快 20.6%. 算法 1、2、3 之间的比较可以看出，射影 Montgomery 算法和利用文献[19]的公式的算法 1 都比仿射 Montgomery 算法快. 射影 Montgomery 算法则比利用文献[19]的公式的算法 1 效率更高.

在 Sage 云平台环境下，co-Z Montgomery 算法比仿射 Montgomery 算法快 24.1%；比射影 Montgomery 算法快 20.1%；比 Montgomery 算法利用文献[19]的公式快 23.1%. 算法 3 比利用文献[19]的加法、倍点公式的算法 1 和仿射 Montgomery 算法的计算速度都要快.

综上分析知，co-Z Montgomery 算法是这 4 个抗能量攻击的标量乘算法中计算速度最快的.

本文的实验都是软件实现的. 由于我们提出的 co-Z Montgomery 算法比利用文献[19]的公式的原 Montgomery 算法节省的计算量为 $I+C-5M$，比射影 Montgomery 算法节省 $C+2M$，比 Montgomery 算法利用文献[19]的公式快 $2C+M$. 因此，在硬件实现中，算法 4 的计算速度也是最有优势的.

接下来，我们将对本文进行总结，并且对未来的工作进行展望.

6　结　论

本文改进了有限域 $GF(3^m)$ 上椭圆曲线的 Montgomery 算法. 在射影坐标下，使用 co-Z 技巧和每轮循环中不计算 Y 坐标的技巧，提出了统一 Z 坐标的椭圆曲线点操作公式. 在该公式的基础上，设计了 co-Z Montgomery 标量乘算法. 当 $S=M$ 时，理论结果表明 co-Z Montgomery 算法比原 Montgomery 算法利用文献[19]的公式节省的计算量为 $I+C-5M$，比射影 Montgomery 算法节省 $C+2M$，比 Montgomery 算法利用文献[19]的公式快 $2C+M$. 利用文献[18]的基础运算时间，实验结果表明该算法比原算法的效率提高 26.3%，比射影 Montgomery 算法快 19.0%，比 Montgomery 算法利用文献[19]的公式快 20.6%. 利用 Sage 云平台的基础运算时间，结果表明 co-Z Montgomery 算法比上述算法分别快 24.1%、20.1%、23.1%. co-Z 技巧，射影坐标技巧等都可广泛应用于椭圆曲线标量乘算法效率的研究中.

本文所提出的 co-Z Montgomery 算法能够有效地抵抗简单能量攻击. 如果还需要抵抗差分能量攻击，直接对标量进行随机化[16]即可.

参　考　文　献

[1] Koblitz N. Elliptic curve cryptosystems. Mathematics of Computation, 1987, 48(177): 203-209

[2] Miller V S. Uses of elliptic curves in cryptography//Proceedings of the CRYPTO 85. Santa Barbara, USA, 1985: 417-428

[3] Doche C. On the enumeration of double-base chains with applications to elliptic curve cryptography//Proceedings of the 20th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2014). Kaoshiung, China, 2014: 297-316

[4] Dimitrov V, Imbert L, Mishra P K. The double-base number system and its application to elliptic curve cryptography. Mathematics of Computation, 2008, 77(262): 1075-1104

[5] Meloni N, Hasan M A. Efficient double bases for scalar multiplication. IEEE Transactions on Computers, 2015, 64(8): 2204-2212

[6] Doche C, Sutantyo D. New and improved methods to analyze and compute double-scalar multiplications. IEEE Transactions on Computers, 2014, 63(1): 230-242

[7] Gallant R, Lambert R, Vanstone S. Faster point multiplication on elliptic curves with efficient endomorphisms//Proceedings of the CRYPTO 2001. Santa Barbara, USA, 2001: 190-200

[8] Galbraith S, Lin X, Scott M. Endomorphisms for faster elliptic curve cryptography on a large class of curves//Proceedings of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2009). Cologne, Germany, 2009: 518-535

[9] Hankerson D, Karabina K, Menezes A. Analyzing the Galbraith-Lin-Scott point multiplication method for elliptic curves over binary fields. IEEE Transactions on Computers, 2009, 58(10): 1411-1420

[10] Longa P, Sica F. Four-dimensional Gallant-Lambert-Vanstone scalar multiplication//Proceedings of the 18th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2012). Beijing, China, 2012: 718-739

[11] Hu Z, Longa P, Xu M. Implementing the 4-dimensional GLV method on GLS elliptic curves with j-invariant 0. Designs, Codes and Cryptography, 2012, 63(3): 331-343

[12] Guillevic A, Ionica S. Four-dimensional GLV via the weil restriction//Proceedings of the 19th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2013). Bengaluru, India, 2013: 79-96

[13] Cohen H, Miyaji A, Ono T. Efficient elliptic curve exponentiation using mixed coordinates//Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 98). Beijing, China, 1998: 51-65

[14] Coron J S. Resistance against differential power analysis for elliptic curve cryptosystems//Proceedings of the Cryptographic Hardware and Embedded Systems 1999. Worcester, USA, 1999: 292-302

[15] Bernstein D J, Lange T. Faster addition and doubling on elliptic curves//Proceedings of the 13th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2007). Kuching, Malaysia, 2007: 29-50

[16] Joye M, Tymen C. Protections against differential analysis for elliptic curve cryptography an algebraic approach//Proceedings of the Cryptographic Hardware and Embedded Systems 2001. Paris, France, 2001: 377-390

[17] Montgomery P. Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation, 1987, 48(177): 243-264

[18] Wang Hong, Li Bao, Yu Wei. Montgomery algorithm on elliptic curves over finite fields of character three. Journal on Communications, 2008, 29(10): 25-29(in Chinese)
(汪宏, 李宝, 于伟. 特征 3 有限域上椭圆曲线的 Montgomery 算法. 通信学报, 2008, 29(10): 25-29)

[19] Farashahi R R, Wu H F, Zhao C A. Efficient arithmetic on elliptic curves over fields of characteristic three//Proceedings of the Selected Areas in Cryptography 2012. Windsor, Canada, 2013: 135-148

[20] Meloni N. New point addition formulae for ECC applications//Proceedings of the International Workshop on the Arithmetic of Finite Fields 2007. Madrid, Spain, 2007: 189-201

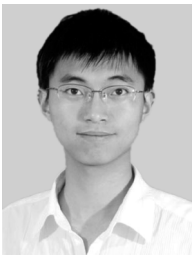
[21] Lin Q, Zhang F. Efficient precomputation schemes of $kP+lQ$. Information Processing Letters, 2012, 112(11): 462-466

[22] Smart N P, Westwood E J. Point multiplication on ordinary elliptic curves over fields of characteristic three. Applicable Algebra in Engineering Communication and Computing, 2003, 13(6): 485-497

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

[24] Lopez J, Dahab R. Fast multiplication on elliptic curves over $GF(2^n)$ without precomputation//Proceedings of the Cryptographic Hardware and Embedded Systems 1999. Worcester, USA, 1999: 316-327

[25] Okeya K, Sakurai K. Efficient elliptic curve cryptosystems from a scalar multiplication algorithm with recovery of the y-coordinate on a Montgomery-form elliptic curve//Proceedings of the Cryptographic Hardware and Embedded Systems 2001. Paris, France, 2001: 126-141



YU Wei, born in 1987, Ph. D., assistant professor. His main research interests focus on elliptic curve cryptography.

LI Bao, born in 1962, Ph. D., professor, Ph. D. supervisor. His main research interests include elliptic curve cryptography, the theory and application of provable security, and design and analysis of cryptographic protocols.

WANG Kun-Peng, born in 1971, Ph. D., professor, Ph. D. supervisor. His main research interests focus on elliptic curve cryptography.

LI Wei-Xuan, born in 1994, Ph. D. candidate. Her main research interests focus on elliptic curve cryptography.

TIAN Song, born in 1987, Ph. D. candidate. His main research interests focus on elliptic curve cryptography.

Background

Elliptic Curve Cryptography (ECC) was introduced independently by Koblitz [1] and Miller[2] in 1985. The 160-bit key in ECC provides 1024-bit key security for RSA systems, and the 256-bit key on ECC provides 3072-bit key security for RSA systems. Because of the shorter key, ECC has attracted increasing attention, particularly in resource-limited hardware environments (such as smart cards and phone cards).

The efficiency of ECC mainly depends on its scalar multiplication for calculating kP, where P is a point on the elliptic curve and scalar k is an integer. There are a variety of methods to accelerate scalar multiplication. In the first method, the Hamming weight of k is reduced to decrease the number of point additions[4-7]. The second method changes the representation of points on the elliptic curve using affine

coordinates, standard projective coordinates, Jacobi coordinates, and so on. The third method uses a nontrivial efficiently computable endomorphism, which was first introduced by Gallant, Lambert, and Vanstone and is called the GLV method[8]. Later, other efficiently computable endomorphisms were proposed[9-13] to speed up scalar multiplication.

However, none of these algorithms can resist simple energy attacks[15] on elliptic curves.

Montgomery[18] proposed an algorithm called the Montgomery algorithm, whose basic idea is to compute a point doubling and a point addition in each step of the cycle. Since the energy consumption in each cycle is basically the same, the algorithm can resist simple energy attacks. However, the calculation speed of the Montgomery algorithm is slower than that of other algorithms for computing scalar multiplication.

Later, Lopez and Dahab[23] optimized the Montgomery algorithm. They eliminated the calculation of the y-coordinate in the point addition and doubling formulae. This greatly improved the computing efficiency of the scalar multiplication. Wang, Li, and Yu[19] provided the Montgomery algorithm on elliptic curves over finite fields of character 3 and improved the Montgomery algorithm using projective coordinates.

In this work, we present new point addition and doubling formulae on elliptic curves over finite fields of character 3 by the tricks of using same Z-coordinate and not calculating Y-coordinate. Using these new formulas, co-Z Montgomery algorithm is proposed. Experimental results on Sage cloud platform indicate that co-Z algorithm is faster than affine Montgomery algorithm using Lopez and Dahab's idea[23] by 24.1%, faster than projective Montgomery algorithm using Lopez and Dahab's idea by 20.1%, and faster than Montgomery algorithm using the formulas of "Selected Areas in Cryptography 2012" by 23.1%.