代　号　10701　　学　号　0809420499

分类号　TN918　　密　级　公开

西安电子科技大学

硕士学位论文

题（中、英文）目　椭圆曲线上标量乘快速算法研究

The Study of Fast Algorithm of Scalar Multiplication on Elliptic Curve

作者姓名　王立川　指导教师姓名、职称　李学俊 副教授

学科门类　军事学　学科、专业　密码学

提交论文日期　二〇一一年一月

西安电子科技大学

学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切的法律责任。

本人签名：王亦川　　日期 2011.1.15

西安电子科技大学

关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律署名单位为西安电子科技大学。

（保密的论文在解密后遵守此规定）。

本人签名：王亦川　　日期 2011.1.15

导师签名：李国俊　　日期 2011.1.15

摘要

椭圆曲线密码体制(ECC)是迄今为止单比特具有最高安全强度的密码系统。与其他公钥密码系统相比，椭圆曲线密码体制具有安全性高、计算负载小、密钥尺寸短、占用带宽少等众多优点，深入研究基于椭圆曲线离散对数问题的公钥密码体制具有很大的现实意义。标量乘法是椭圆曲线密码体制实现过程中最基本、最耗时的运算，也是椭圆曲线密码体制快速实现最关键的运算，提高标量乘法的运算速度对椭圆曲线密码体制的推广意义重大。研究标量乘法有两个切入点：一是研究标量k的有效表示：二是寻求底层域快速运算算法。本文将研究重点放在标量k的有效表示上。

本文主要针对双标量乘快速算法和k的多基编码标量乘算法两个方面进行了研究：首先介绍了椭圆曲线上原有的单标量乘算法如二元法、NAF 法、窗口法等，对 Shamir 快速双标量乘算法和交错 NAF 方法的优缺点进行了详细分析，结合二者优点给出了 Shamir 快速算法的一种改进方案。改进方案与 Shamir 快速算法相比，在没有明显增加运算量的前提下，至少减少点存储量 50%以上，是一种适合于内存受限的手持设备的方案。其次，介绍了双基和多基编码标量乘算法的发展和研究现状，并对与之相关的底层域快速算法$kP+Q$的研究进展做了说明。之后分析了 Ciet 等人的双基编码标量乘算法和一种多基编码标量乘算法的运算效率，给出了一种改进的多基编码标量乘算法。对此三种算法进行定量效率分析，分析结果表明，改进方案较原有多基标量乘，每次点乘能减少两个逆运算，从而达到降低运算量的目的。当$I/M=60$时，改进方案比原有的多基标量乘算法提高效率 7%左右。

关键词：椭圆曲线密码体制 标量乘 双标量乘 双基编码 多基编码

Abstract

Elliptic Curve Cryptography(ECC) provides the highest security strength per-key-bit of any cryptography known.Compared with other public-key cryptographies,ECC not only has the higher security but also has less computation overheads,shorter key size and narrower bandwidth.Therefore,making an intensive study about public key cryptography has great practical significance.The basic and most time-consuming computation in elliptic curve cryptosystems is the scalar multiplication,and scalar multiplication algorithm is the most important operation in elliptic curve cryptosystem.Improving the speed of scalar multiplication operation on elliptic curve is of great significance to the promotion of elliptic curve cryptosystem.Gerenrally speaking,there are two strategies to speed up the scalar multiplication:one is to investigate efficient the scalar k's representation,the other is to seek fast algorithms over bottom filed.We mainly study efficient representation of k.

This paper studys scalar multiplication mainly focus on double scalar multiplication algorithms and representation of k with multi-Base.First,we introduce the elliptic curve scalar multiplication algorithms,including binary method,NAF method and window method.And we make an analysis about the classic double scalar multiplication algorithms-Shamir fast algorithm and interlacing NAF method.Based on Shamir fast algorithm,we give a improved algorithm.Compared with Shamir fast algorithm,the improved algorithm can reduce the storage of points by 50% without significant increase of quantity of computation.It is suitable for handheld devices which has few memory.Secondly,we introduce the development and research situation about double-base and multi-base system.And analyzes the existing Ceit’s scalar multiplication algorithm with double-base coding and a kind of scalar multiplication algorithm with multi-base coding,and then gives a kind of improved scalar multiplication algorithm with multi-base coding.By quantitative analysis about the three algorithms,we find the improved scheme can minimize the number of inverse calculate.When I/M is 60, the improved algorithm can improve the efficiency by 7%.

Keywords:Elliptic Curve Cryptography Scalar Multiplication Double Scalar Multiplication Double-Base Coding Multi-Base Coding

目录

第一章　绪论

本章简单介绍了公钥密码学的基本概念及研究现状，对公钥密码学的发展历史做了简单介绍，回顾了椭圆曲线密码学的发展历程和一些重要的算法，最后对论文的章节安排做了说明。

1.1　公钥密码学及现状

随着计算机技术的发展和应用，互联网的安全性越来越受到人们的重视。由于互联网环境的开放性，使得非法授权访问、冒充合法用户、破坏数据完整性、干扰系统正常运行等计算机安全问题应运而生。因此，人们对信息加密技术提出了更高的要求。

密码学是信息安全技术的核心技术。密码学包括密码编码学和密码分析学这两个相互独立又相互依存的分支。前者致力于建立难以被敌方或对手攻破的安全密码体制；后者则力图破译敌方或对手已有的密码体制。

密码科学具有悠久的历史。1949 年香农发表的“保密系统的通信理论”为密码学奠定了理论基础，从此密码学成为一门科学。继 Shannon 之后，许多研究者设计了各种基于密钥的对称密码体制，它的保密性取决于密钥的保密性。对称密码体制具有运算量较小的优点，但是从密钥管理角度来看，由于对称密码体制需要事先分配密钥，使其在很多场合应用并不方便。1976 年，Diffie 和 Hellman 发表的“New Direction in Cryptography”[1]，首次提出了公钥密码的思想，是现代密码学的里程碑，使得在开放的环境中确保信息的保密性、认证性、完整性和不可否认性成为可能。公钥密码体制不仅可以方便地实现密钥分配，而且还可以实现数字签名，因而特别受到人们的重视。但是公钥密码体制中常包含代数结构，例如群、环、或者有限域等，在这些代数结构中存在着复杂的大整数或多项式的运算，运算效率较低。于是，人们开始着眼于设计快速高效、安全的密码算法，同时提高公钥密码体制的效率成为当今的一大课题。

自从公钥密码问世以来，学者们提出了许多基于公钥的加密方法，它们的安全性是基于复杂的数学难题的。根据所基于的数学难题来分类，有以下三类系统目前被认为是安全和有效的。

(1)　大整数因式分解问题(IFP)：例如 RSA[2]与 Rabin-Williams[3,4]体制。

(2)　有限域离散对数问题(DLP)：例如美国政府数字签名算法 DSA[5]，ElGamal 加密体制和签名方案[6]，Diffie-Hellman 密钥交换方案[7]。

(3)　有限域上椭圆曲线的离散对数问题(ECDLP)：该问题的困难性是椭圆曲线

公钥密码[8,9]安全基础，如 ECC。

1978 年，Rivet、Shamir、Adelman 提出了 RSA 系统，它的安全性决定于大整数素因子分解的难解性，而大整数因子分解问题是著名的数学难题，至今仍然没有有效的方法能够解决，这一事实确保了 RSA 算法的安全性。RSA 系统是公钥密码系统具有典型意义的方法，大多数公钥密码都使用 RSA 算法进行签名和加密。RSA 方法的特点主要在于原理简单，易于使用，因而得到大规模的普及。

自从 RSA 体制被提出以后，人们对于大整数分解问题产生了极大的兴趣，由于计算机计算能力的大幅度提升以及各种新的破解算法的出现，人们求解大整数分解问题的能力相比过去有了很大的进步。1984 年 Carl Pomeranee 设计出了二次筛选法[10]，同年，Aden Lenstra 利用二次筛选法成功地分解了 129 位十进制数，该工作由全世界范围的 1600 台计算机耗费大约 8 个月的时间完成。1996 年该研究小组使用数域筛选法[11]分解了 130 位的十进制整数（对应二进制 432bit），将分解的效率提高了 15%。1999 年，一个国际研究小组经过研究发现使用通用的数域筛选法分解 155 位十进制数需要总的运算时间约为 8000MIPS 年，因此 512bit 的模长在基于因式分解问题的 RSA 体制中，仅仅能够提供勉强的安全性。2002 年 Chris MoIlico[12]等人领导的研究小组成功地解决了著名的 Certieom ECC-p109 挑战难题，因此为长期安全性，在离散对数密码体制中，也应该使用 1024bit 或更大的模 p。

目前数域筛选法[13-15]和相应的整数分解算法一样，具有相同的渐进运行时间。而另外一种著名的 Pollard rho 算法[16]，在计算机中可以容易的并行实现。因此，随着分解大整数方法的进步及完善、计算机速度的提高以及计算机网络的发展，对 RSA 加解密安全起决定作用的大整数要求越来越大。为了确保 RSA 使用的安全性，其密钥的位数也一直在增加，例如，目前一般认为 RSA 需要 1024 位以上的字长才能保证安全性。通常建议使用 1024bit 模长，预计要保证 20 年的安全性需要选择 2048bit 的模长，由于模长的增加使得加解密效率大幅降低，同时存储量也变大，因此该算法的硬件实现也变得困难，这成为了 RSA 广泛应用需要解决的一个大问题。

而 DSA（Data Signature Algorithm）作为一个数字签名标准是基于有限域上离散对数问题的，它仅仅可以提供数字签名，但并不提供数据加密功能。

1985 年由 Koblitz（美国华盛顿大学）和 Miller（IBM 公司）[17]两人分别独立提出了公钥系统椭圆曲线加密算法 ECC。椭圆曲线加密算法提出后，人们发现很多其他公钥加密系统所不具有的优点，因此迅速成为了公钥密码学的一个重要分支。椭圆曲线公钥密码的有效实现是近几年密码学研究的热点内容，其研究主要集中在以下几个方向：①研究提高标量乘的运算效率。②研究随机曲线的选取方法。③研究 ECC 系统的实现问题。④研究如何快速得到 ECC 参数。⑤研究设计好的椭圆曲线密码协议。

椭圆曲线公钥密码体制的实现问题可以分为三个层次，其中有限域中的基本运算是最底层的，主要有加、减、乘、求逆等运算；而 ECC 上的点加、倍点和标量乘运算为其第二层次的运算，其中可以由椭圆曲线的几何性质得到点加和倍点运算的方法，而在点加运算和点乘运算的基础上又可以得到标量乘运算方法，它是椭圆曲线密码系统实现中最为耗时的运算，我们研究椭圆曲线密码体制的快速算法主要研究椭圆曲线中点加、倍点乘和标量乘运算；第三层则是 ECC 应用的各种密码协议，主要包括加解密、数字签名、密钥交换等协议，这些协议也是在椭圆曲线上的基本运算的基础上实现的。本文将研究重点放在椭圆曲线的标量乘算法的快速实现上。

1.2　椭圆曲线密码的产生背景和研究价值

1.2.1　椭圆曲线密码的产生背景

1986 年，Koblitz 和 Miller[18]提出了椭圆曲线密码体制，它是以椭圆曲线离散对数问题的难解性为安全基础的，该密码体制使用有限域上椭圆曲线有限群替代离散对数问题密码体制中的有限循环群，从而获得安全性。椭圆曲线密码是在 RSA 算法提出多年后才被提出来，同时该理论日趋成熟，当时人们对大整数分解能力有限，因此 RSA 具有较高的安全性，同时 RSA 理论简单，操作方便，使得复杂的椭圆曲线密码没有了市场。但随着计算机计算能力的飞速提高，人们处理大整数因子分解问题的能力也有了很大提高，从而导致原来 512 比特就能保证足够安全的 RSA 算法到后来不得不将密钥长度增加到 1024 位或 2048 位来增强安全性，要达到相同的安全级别，160 比特长椭圆曲线密码体制的密钥相当于 1024 比特的 RSA 算法的密钥，210 比特长椭圆曲线密码体制密钥安全性相当于 2048 比特 RSA 算法的密钥。这样椭圆曲线密码体制密钥短、单位比特安全强度高的优势逐渐显出。ECC 的这些优点使得人们普遍认为，ECC 会取代 RSA 而成为 21 世纪最主要的公钥密码体制。因此 ECC 作为密码学的重要分支逐步引起广大学者的兴趣。经过学者们的深入研究，ECC 越来越走向成熟，显示出许多其他密码体制没有的优势。

椭圆曲线从被提出到现在得到了快速发展，大致可以分成两个阶段：

(1) 1985～1995 年，这一段时间人们对椭圆曲线密码技术以理论研究为主展开研究。经过约十年的深入研究 ECC 的两大弱点逐渐得到了克服：一是无法找到一种合适的计算椭圆曲线点 P 的阶的算法导致人们在选取曲线时遇到了不可克服的困难，从而导致 ECC 的实用性不强。计算椭圆曲线点 P 的阶的算法最早是在 1985 年被 Schoof[19]提出的，1989 年到 1992 年之间 Atkin 和 Elkies[20-22]对其作出了进一

步的改进，也就是被人们称之为 SEA(Schoof-Elkies-Atkin)的算法。后来 Satoh[23]、Harley、Gaudry[24]等人又进一步完善了该算法，到 1995 年人们已经能容易地计算出满足密码要求的任意椭圆曲线的阶了。二是椭圆曲线中的加法运算以及点乘运算比较复杂致使 ECC 实现效率较低。随着对椭圆曲线运算研究的进一步深入，ECC 的实现速度也不断得到提高。同时，这段时间内人们还探讨了 ECC 的安全性问题，提出了许多实现 ECC 操作的算法[25-27]，其中在域操作算法方面的研究成果特别显著。

(2) 1995 年至今，国外学者继续对椭圆曲线加密方法展开研究[28-34]，这段时间学者们不仅研究如何改善椭圆曲线密码算法的性能，还对椭圆曲线密码的应用进行了研究。同时出现了多种椭圆曲线加密、签名、密钥交换的算法。加拿大的 Certicom 安全公司和工业界联合研制并生产出以椭圆曲线密码算法为核心的密码产品。

椭圆曲线密码发展的两个阶段中，ECC 的快速实现是学者研究的着重点，特别是对标量乘算法的研究，所谓标量乘法 kP，就是给定的基点 P 和整数 k，求另一个点 kP 的运算。随着人们对标量乘算法研究的不断深入，对 kP 的计算方法已有了较大的改善。从最初的仿射坐标，到后来的各种投影坐标（标准投影坐标、雅可比坐标等），以及各种坐标体系的混合模式，都对加速计算 kP 起到了一定的作用。

1.2.2 椭圆曲线密码的优缺点

总体而言，相对于其他公钥体制，椭圆曲线密码具有需要更少的计算时间、更少的密钥存储空间、更小的密钥传输带宽要求等特点。

第一，和其他公钥密码相比，对于单比特的密钥长度，椭圆曲线密码具有最高的安全性，其单比特安全性强度与其它公钥相比如下表 1.1 所示[35]

表 1.1 RSA 与 ECC 和 DSA 的比较

攻破时间 (MIPS 年)	RSA 与 DSA 密钥长度(bit)	ECC 密钥长度(bit)	RSA 与 ECC 密钥长度比
10^4	512	106	5:1
10^8	768	132	6:1
10^{11}	1024	160	7:1
10^{20}	2048	210	10:1
10^{78}	21000	600	35:1

从表中可以看出，为了保证同样的安全强度，所需要 RSA、DSA 公钥密码的密钥长度必须是椭圆曲线密码密钥长度的 5 倍以上，并且密钥长度越长，ECC 相

对于 RSA 和 DSA 所表现出来的单比特安全性就越明显，由于椭圆曲线密码需要密钥长度短的优点使得它很适用存储空间受限的智能卡等小型手持设备。因此椭圆曲线公钥密码体制引起了学者的广泛兴趣。

第二，在同一基域下，椭圆曲线密码算法具有多种多样的选择。就是说，对给定的素数 p 或 q，我们可以唯一地确定一个 RSA 算法或 ElGamal 类算法。但是对于在同一基域上的椭圆曲线密码体制，只要改变椭圆曲线方程的系数，我们就能够得到更多具体的算法。

第三，可以自主选取基域 $GF(p)$ 中的素数 p，从而可以选取到更适合计算机处理的素数，例如 Mersenne 素数等。但 RSA 体制和 ElGamal 类体制则不能这样选取。因为对于 RSA 体制，素数的选取必须是随机的，对于 ElGamal 类体制又必须选取使 $p-1$ 个数中包含一个大素因子的素数。

第四，经过分析发现，RSA 和 DSA 密码系统等均存在亚指数算法[36,37]，而目前分析没有发现椭圆曲线密码系统存在亚指数算法。

第五，椭圆曲线密码体制的带宽要求低。对于长消息的加解密，这三类公钥密码体制具有相同的带宽要求，但对短消息进行加解密，椭圆曲线密码体制的带宽要求低得多，因为公钥加密体制多用于短消息，例如数字签名和密钥交换等，带宽要求低的特点使得椭圆曲线密码体制在无线网络领域具有良好的应用前景。椭圆曲线的这些优点使得人们对它的研究产生了浓厚的兴趣。

虽然如此，相比于 RSA 等其他公钥密码体制，椭圆曲线密码体制还有一些不足之处，主要是以下几点：

(1)　在保证同样的安全强度前提下，在运算速度上椭圆曲线密码体制并不比 RSA 算法占优势。例如，在签名方案中，虽然椭曲线密码体制签名过程比 RSA 算法略快，但是 RSA 算法的验证过程却比椭圆曲线密码算法快。这是因为 RSA 算法可以尽可能地选取较小一点的公钥，从而使得验证速度加快。在实际应用中，往往会对验证过程有较高的效率要求。总的来说，就签名和验证过程的总时间而言，椭圆曲线密码要快于 RSA 算法。

(2)　椭圆曲线的群运算法比 RSA 算法要复杂许多。RSA 算法中没有椭圆曲线有限域中的乘法和求逆运算，只有整数的求模运算。这一事实影响到椭圆曲线密码体制的效率。

(3)　RSA 算法允许事先选定公钥，然后根据公钥确定私钥。而椭圆曲线密码体制不可以事先选定公钥，只能够首先确定私钥，然后再确定公钥，这样就使得椭圆曲线密码公钥的产生有了一定的随机性。同时虽然椭圆曲线相对于其他公钥密码具有单比特安全性强的优势，但是和对称密码体制相比较，公钥密码的计算速度远远慢于对称密码体制。公钥加密系统的安全是是基于数学问题的难解性的，计算过程较为复杂，他们的实现效率比对称加密系统要落后许多，例如 RSA 算法

模数是 512 比特，当硬件实现时，DES 实现速度约为 RSA 的 1000 倍[38]；当软件实现时，DES 实现速度约为 RSA 的 100 倍[39]。其中制约椭圆曲线快速发展的因素之一就是标量乘运算，它在很大程度上决定了椭圆曲线密码的执行速度，当今已有不少学者提出了多种计算椭圆曲线标量乘的快速算法。

1.2.3 椭圆曲线公钥密码实现算法介绍

制约椭圆曲线快速发展的因素之一就是标量乘运算。当今已有不少学者提出了多种计算椭圆曲线标量乘的快速算法。对标量乘法的研究是从二进制方法开始的，1990 年 Morain 和 Olivos[40]提出使用非邻接表示型（NAF）方法取代二进制方法，将二进制的 1/2 的汉明重量降低到 NAF 方法的 1/3，到此为止，NAF 成为计算 kP 的经典方法。1998 年，Knuth[41]提出使用从右到左的二进制方法计算标量乘，随后又把此方法扩展到 m 进制，Cohen[42]研究了基数为 $2k$ 的从左到右的方法。同年，Gordont[43]发明了一种高效的求幂方法。而事实上 1996 年 Menezes[44]等人已提出比 Gordon 的求幂方法更具有普遍性的求幂方法。2000 年，Solinas[45]提出了 NAF 窗口方法，把 k 的汉明重量从 NAF 的 $1/3$ 减小到 $1/(w+1)$，但该方法需要使用一定的存储容量。窗口 NAF 方法也可进一步改进为滑动窗口方法，它需要 $(2(2^w-(-1)^w)/3-2)/2$ 个预计算，但是二者之间有细微的区别。Miller[46]提出了一种比窗口 NAF 方法更有效的窗口碎片技术，对滑动窗口和窗口 NAF 方法进行了扩展，此方法在预计算时拥有较大的弹性，在存储受限的情况下更能体现其优势，对于窗口宽度 w 和给定的奇数 $v \le 2^{w-1}-3$，碎片窗口技术的平均密度为 $1/(w+1+(v+1)/2^{w-1})$。与此同时，Brickell 等人又提出了固定基窗口法，此方法可以认为是 Lim-Lee 的一种特例，但这两种方法的共同缺点是需要存储太多。1996 年 Dimitrov 等人开始提出 DBNS[47-49] (Double-Base Number System)。DBNS 是基于 Tijdeman[50]提出的小素数组成整数的最佳间隔思想，它把 k 表示成 2^x3^y 的形式，结合了 3^yP，这种算法比传统的 NAF 算法有了较大的进步。

另外就是通过提高底层域上的运算速度来提高椭圆曲线标量乘的运算速度。1997 年 Guajardo 和 Paar[51]提出了直接计算 $4P$、$8P$、$16P$ 的有效算法，之后大量学者对底层域上 2^kP 以及 $kP+Q$ 的直接快速算法进行了比较深入的研究[52-54]。文献[54]中使用了牺牲乘法操作来降低求逆次数的思想，将计算 2^kP 的运算量从 $kI+2kS+2kM$ 降低到 $1I+(4k+1)S+(4k+1)M$，其中 I、S、M 分别表示椭圆曲线基域中的求逆、平方、乘法运算，I/M 表示曲线上求逆和乘法运算的复杂度比。2003 年 Eisentrger 给出了一种利用仿射坐标计算 $2P+Q$ 的快速算法[55]，能够节约基域运算中的 1 次乘法和 1 次平方，将运算量从 $2I+3S+4M$ 降低到 $2I+2S+3M$，而 Ciet[56]基于与 Yasuyuki 同样的思想，将这一运算的运算量调整为 $1I+2S+9M$。

在给定P和Q情况下，Ciet 等根据同样的思想，给出了仿射坐标下直接计算$3P+Q$的算法，把其运算量下降为$2I+4S+9M$，比计算$(3P)+Q$节省一次求逆运算。

从国外和国内学者的研究结果可知，对标量乘法的研究方法可归纳为①寻求标量k的有效表示，即利用重新编码求标量乘（如二进制、三进制、NAF、w-NAF 等）；②利用坐标变换计算标量乘法（投影坐标、仿射坐标、Jacobian 坐标等）；③使用加减法链方法及固定基 comb 等算法；④ 研究曲线上底层域上的快速算法如$(2P、2P+Q、3P、3P+Q$等)。虽然目前 ECC 的计算速度已经有了很多的提高，但是由于现代信息量的飞速膨胀，导致对 ECC 的运行效率提出了更高的要求。在 ECC 快速实现中仍有一些关键为题尚待进一步研究。因此在相当长的一段时间内，快速实现 ECC 还是密码学者研究的重点，而标量乘法kP是研究的核心。

1.2.4　椭圆曲线密码体制的建立

一、体制建立

建立椭圆曲线密码体制：首先需要选择一个基域F_q，它可以是一个素域F_p，也可以是一个特征为 2 的域F_{2^m}，且m是素数。然后从F_q上选取一条椭圆曲线E，使该曲线的阶为一大素数n，或者是一个小整数与大素数n乘积的形式。之后，从曲线E上选取一个阶为该大素数n的点P。把有限域F_q、曲线E、点P以及阶n作为公开信息。

二、密钥生成

每一个参与者（或者说用户）A 完成下述过程：

1. 随机选择某一整数$d_A \in [1, n-1]$；
2. 计算点$Q_A = d_A P$；
3. 将点Q_A作为参与者的公钥；
4. 将整数d_A作为参与者的公钥；

三、椭圆曲线加密方案(ECES）

现在设用户 B 要发送信息M给用户 A，则 B 的加密过程如下：

1. 选定 A 的公钥Q_A；
2. 将发送信息M表示成域元素$m \in F_q$的形式；
3. 选取一个随即的素数$k \in [1, n-1]$；
4. 计算点$(x_1, y_1) = kP$；
5. 计算点$(x_2, y_2) = kQ_A$,若$x_2 = 0$，则返回第三步；
6. 计算$c = mx_2$；
7. 将已加密数据(x_1, y_1, c)发送给 A。

而 A 收到密文(x_1, y_1, c)后的解密过程如下：

1. 计算点 $(x_2, y_2) = d_A(x_1, y_1)$，得到 $x_2 \in F_q$；
2. 计算 $m = cx_2$,得出信息 M 。

1.3 论文所作的工作和章节安排

为方便后面章节讨论，本文前两章对椭圆曲线密码的基本概念和研究现状做了简要介绍，其中第一章介绍了椭圆曲线密码学的背景知识、研究现状和研究方法，第二章介绍了椭圆曲线的定义、椭圆曲线上点的表示和基本运算。本文所做的工作主要安排在第三章和第四章，具体安排如下：

第三章，介绍了标量乘算法的研究历程和几种经典的标量乘算法。如“二元法”，“NAF 方法”，“NAF 窗口方法”。认真分析了双标量乘 Shamir 快速算法和交错 NAF 方法的优缺点，将二者的优点相结合，在 Shamir 快速算法的基础上提出一种改进方案。在没有明显增加点乘运算量的前提下，改进方案比 Shamir 快速算法节省点存储量 50%以上，可以作为一种应用于手持设备等内存受限设备的点乘方案。

第四章，介绍了双基和多基编码标量乘算法的研究现状，指出多基编码标量乘与底层域快速算法相结合的研究趋势。对 Ciet 等人的双基编码标量乘算法和一种多基编码标量乘算法进行效率分析，在 Ciet 等人双基编码标量乘算法的基础上，提出了一种新的基于多基标量乘算法，将算法与底层域快速算法相结合最大程度地减少了点乘运算中逆运算的个数。通过定量分析，当 $I / M = 60$ 时，改进方案比原有方案速度提高 7%左右。同时因为多基表示的高度冗余性，k 可以有多种不同的多基链表示形式，这使得多次计算 k，基本运算的运算顺序可以完全不同，因此天然能够抵抗某些边信道攻击。因此多基编码的标量乘同时具有了更好的安全性。

第五章，对全文的研究工作做了细致总结，并对目前椭圆曲线标量乘算法的研究方法和研究趋势做了总结和展望。

第二章　椭圆曲线的定义和基本计算

本章介绍了椭圆曲线密码数学基础，椭圆曲线的定义和椭圆曲线上加法群的运算法则，并详细介绍了各种坐标形式下点的运算法则。本章所介绍的均为椭圆曲线密码理论的基础理论，是作为后续章节的讨论方便之用，其内容参考了文献[57]和[58]。

2.1　椭圆曲线密码数学基础

为后面章节讨论方便，本节先给出椭圆曲线密码理论的数学基础--有限群和有限域的相关概念。

2.1.1　有限群

定义 2.1 假定G是具有有限个或无限个元素的集合，在G的元素间给出了一个代数运算*（加、乘或其他运算），使得

(1) 满足封闭性，即满足对任意的$a,b\in G$，$a*b\in G$；

(2) 满足结合律，即满足对任何$a,b\in G$，有$a*b*c=(a*b)*c=a*(b*c)$；

(3) 满足含有单位元，即该代数结构中存在一个元素$e\in G$，对任意元素$a\in G$，有$a*e=e*a=a$，此元素 e 称为单位元；

(4) 元素有逆元，即任意$a\in G$，存在一个元素$a^{-1}\in G$，使得$a*a^{-1}=a^{-1}*a=e$，这样的元素a称为逆元。

如果某代数结构满足上述条件则称为群，记作$<G,*>$。

定义 2.2 如果该群结构$<G,*>$满足交换律，即对任何$a,b\in G$，有$a*b=b*a$，则把该群 G 称为交换群（或 Abel 群、加法群等）。群具有以下性质：

(1) 群中的单位元是惟一的；

(2) 消去律成立，即对任意的$a,b,c\in G$，如果$ab=ac$，则$b=c$。如果$ba=ca$，则$b=c$；

(3) 群中的每一元素的逆元是惟一的。

定义2.3用$|G|$来表示群的阶，即代表该群的元素个数。包含元素数为无穷的群称为无限群，包含元素个数为有限的称为有限群。

定义2.4 对于任意一个元素$g\in G$，满足$g^i=e$的最小的正整数，i称为元素g的阶，用$|g|$表示。

定义 2.5 元素$g\in G$，当且仅当群 G 中的任何一个元素都可以表示成g^i的形式且 i 为整数时，元素 g 被称为群 G 的生成元，并且$|g|=|G|$。

例 2.1 对于群$G=Z_5^*=\{1,2,3,4\}$，构成了一个乘法群，该群阶$|G|=4$。由于

$2^0 \bmod 5 = 1$，$2^1 \bmod 5 = 2$ $2^2 \bmod 5 = 4$，$2^3 \bmod 5 = 3$，$2^4 \bmod 5 = 1$，因此元素 2 的阶是 4。又由于 $4^0 \bmod 5 = 1$，$4^1 \bmod 5 = 4$，$4^2 \bmod 5 = 1$，$4^3 \bmod 5 = 4$，$4^4 \bmod 5 = 1$，因此元素 4 的阶是 2。因为群中每一个元素均可以由 2 的某次方而得到，因此 2 是该群的生成元，又因为通过元素 4 不能得到这个群中的全部元素，因此元素 4 不是生成元。

2.1.2 有限域

如果一个非空集合 K，对于加法和乘法代数运算满足：$<K,+>$ 对于加法运算构成加法交换群，单位元用 0 表示；$<K\setminus 0,\cdot>$ 对于乘法运算构成乘法交换群，单位元用 1 表示；对于所有的 $a,b,c \in K$，都有 $(a+b)c = ac+bc$，即分配律成立，则称 K 对于加法和乘法构成域。若集合 K 是有限集合，则称域 K 为有限域，这里用 $GF(q)$ 表示，有限域的元素个数称为有限域的阶。常用的有限域是素数域 $GF(p)(q=p)$ 和二进制域 $GF(2^m)$，即 $q = 2^m$。

(1) 素数域

设 p 是一个素数，以 p 为模，则模 p 的全体余数的集合 $\{0,1,2,\cdots,p-1\}$ 关于模 p 的加法和乘法构成一个 p 阶有限域，并用符号 $GF(p)$ 表示。我们称 p 为 $GF(p)$ 的模。对于任意的整数 a，$a \bmod p$ 表示用 p 除 a 所得到的余数，$0 \le r \le p-1$，并称这种运算为求模 p 的剩余。

(2) 二进制域

阶为 2^m 的域称为二进制域或特征值为 2 的有限域，用 $GF(2^m)$ 表示。构成 $GF(2^m)$ 的一种方法是采用多项式基表示法。在这种表示方法中，域 $GF(2^m)$ 的元素是次数最多为 $m-1$ 次的二进制多项式(多项式的系数取自 $GF(2)=\{0,1\}$)。

$$GF(2^m) = \{a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \cdots + a_2 2^2 + a_1 2 + a_0 : a_i \in \{0,1\}\}$$

选择一个二进制域上的 m 次既约多项式 $f(z)$，$f(z)$ 的既约性意味着 $f(z)$ 不能被分解成次数低于 m 的二进制域上的多项式的乘积。域元素的加法是两个多项式系数的模 2 相加。域元素的乘法是两个多项式相乘后取模 $f(z)$。对于任意一个二进制域上的多项式 $a(z)$，$a(z) \bmod f(z)$ 表示一个次数低于 m 的余式多项式 $r(z)$。余式多项式 $r(z)$ 可用 $f(z)$ 辗转相除 $a(z)$ 得到，这一运算称为求模 $f(z)$ 的余式。

2.2 椭圆曲线定义

定义 2.6 给出域 K 上的椭圆曲线 E 的定义如下[57]：

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2-1}$$

其中 $a_1, a_2, a_3, a_4, a_6 \in K$ 且 $\Delta \neq 0$，Δ 是 E 的判别式，具体定义如下：

$$\left.\begin{aligned}\Delta &= -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6 \\ d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4 + a_1 a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2\end{aligned}\right\} \tag{2-2}$$

如果L是K的扩域，那么E上的L有理点的集合是

$$E(L) = \{(x, y) \in L \times L : y^2 + a_1 xy + a_3 y - x^3 - a_2 x^2 - a_4 x - a_6 = 0\} \cup \{\infty\}$$

其中，∞是无穷远点。

式(2-1)称为*Weierstrass*方程，E是域K上的椭圆曲线，这是因为系数a_1, a_2, a_3, a_4, a_6均为域K上的元素。我们将椭圆曲线简单记为E/K以强调椭圆曲线E定义在域K上，并称K为E的基础域。因此，若椭圆曲线E定义在域K上，则E也定义在K的扩域上。

条件$\Delta \neq 0$能够保证椭圆曲线是“光滑”的，即曲线的所有点都没有两个或者两个以上不同的切线。

点∞是曲线唯一的一个无穷远点，它满足投影式的*Weierstrass*方程。曲线E的L有理点数是满足曲线方程且坐标x和y属于L的点(x, y)，并认为无穷远点是K的所有扩域L上的L有理数点。

定义 2.7 由*Weierstrass*方程给出的定义在域K上的两个椭圆曲线E_1和E_2[57]

$$E_1 : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6$$

$$E_2 : y^2 + \bar{a}_1 xy + \bar{a}_3 y = x^3 + \bar{a}_2 x^2 + \bar{a}_4 x + \bar{a}_6$$

若存在$u, r, s, t \in K$，且$u = 0$,使得变量变换

$$(x, y) \to (u^2 x + r, u^3 y + u^2 st + t) \tag{2-3}$$

把方程E_1变成E_2，则称E_1和E_2在域K上同构，式(2-3)的变换成为变量的相容性变换。

定义在域K上的一个*Weierstrass*方程

$$E : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6$$

能够用变量的相容性变换来简化，下面将分别考虑域K的特征等于 2 或 3 和域的特征不等于 2 或 3 两种情况。

1. 若域K的特征不等于 2 或者 3，则变量的相容性变换

$$(x, y) \to \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1 x}{216} - \frac{a_1^3 + 4a_1 a_2 - 12a_3}{24}\right)$$

把E变换为曲线

$$y^2 = x^3 + ax + b \tag{2-4}$$

其中 $a,b \in K$ 。曲线的判别式是 $\Delta = -16(4a^3 + 27b^2)$

2. 若域的特征是 2，则有两种情况要考虑。

若 $a_1 \neq 0$，则变量的相容性变换

$$(x,y) \to (a_1^2 x + \frac{a_3}{a_1}, a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3})$$

把 E 变为曲线

$$y^2 + xy = x^3 + ax^2 + b \tag{2-5}$$

其中 $a,b \in K$ 。这样的曲线成为超奇异的，且判别式为 $\Delta = c^4$ 。

若 $a_1 = 0$，则变量的相容性变换

$$(x,y) \to (x + a_2, y)$$

把 E 变换为曲线

$$y^2 + cy = x^3 + ax + b \tag{2-6}$$

其中 $a,b,c \in K$ 。这样的曲线称为超奇异的，且判别式为 $\Delta = c^4$ 。

3. 若域 K 的特征是 3，则也有两种情况要考虑。

若 $a_1^2 \neq -a_2$,则变量的相容性变换

$$(x,y) \to (x + \frac{d_4}{d_2}, y + a_1 x + a_1 \frac{d_4}{d_2} + a_3)$$

其中 $d_2 = a_1^2 + a_2$， $d_4 = a_4 - a_1 a_3$，把 E 变换为曲线

$$y^2 = x^3 + ax^2 + b \tag{2-7}$$

其中 $a,b \in K$ 。这样的曲线成为非超奇异的，且判别式为 $\Delta = -a^3 b$ 。

若 $a_1^2 = -a_2$，则变量的相容性变换

$$(x,y) \to (x, y + a_1 x + a_3)$$

把 E 变换为曲线

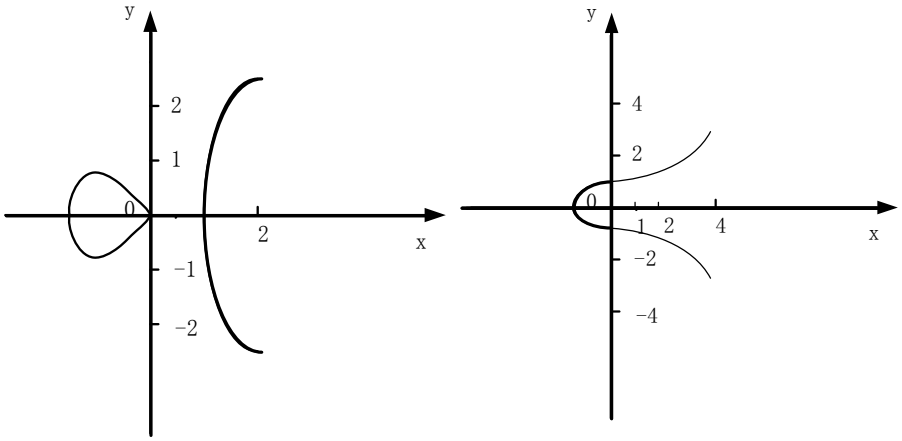
$$y^2 = x^3 + ax + b \tag{2-8}$$

其中 $a,b \in K$ 。这样的曲线成为超奇异的，且判别式为 $\Delta = -a^3$ 。

下面给出常见椭圆曲线的几何图形，这里考虑定义在实数域 R 上的椭圆曲线。

$$E_1 : y^2 = x^3 - x \qquad\qquad E_2 : y^2 = x^3 + x/4 + x/5$$

曲线 $E_1(R) \setminus \{\infty\}$ 和 $E_2(R) \setminus \{\infty\}$ 如下图 2.1(a)和(b)所示。



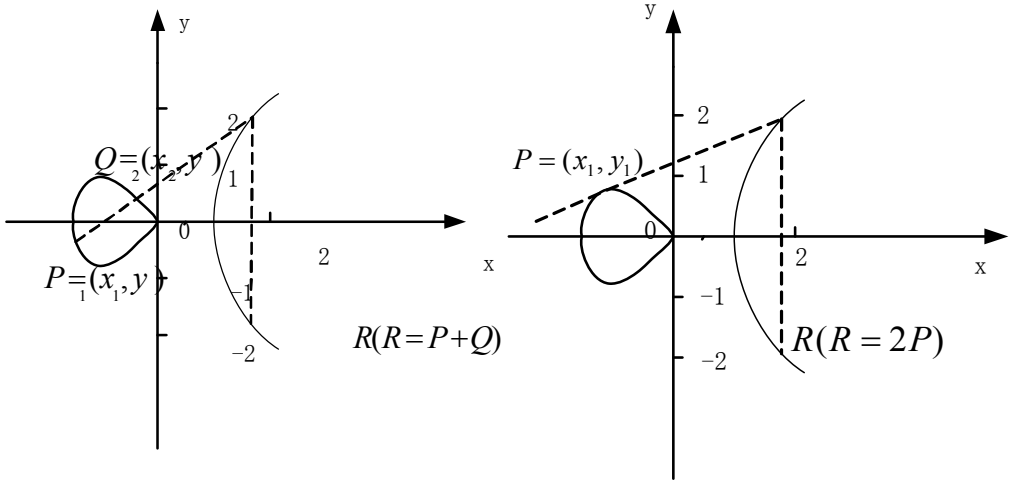
(a)$E_1: y^2 = x^3 - x$　　$(b)E_2: y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

图 2.1　R 上的椭圆曲线

2.3　椭圆曲线加法群的运算法则

设 E 是一个定义在域 K 上的椭圆曲线。依据“弦和切线”规则，$E(K)$ 上的两个点相加得到 $E(K)$ 上的第三个点。点集合 $E(K)$ 及这种加法运算法则构成了一个加法交换群，并且∞为其无穷远点。就是这个群被用来构建椭圆曲线密码体制。

群的加法法则可以用几何方法来说明。假设 $P = (x_1, y_1)$ 和 $Q = (x_2, y_2)$ 是椭圆曲线上的两个不同的点，则点 P 与 Q 的和 $R(R = P + Q)$ 按照如下来定义：首先画一条连接 P 和 Q 的直线，这条直线与椭圆曲线相交于第三点，则这个交点关于 x 轴的对称点就是它们的和 R 点。这一几何表示示于下图 2.2(a)中。



(*a*)相加：P+Q=R　　(b)倍点：P + P = R

图 2.2 椭圆曲线的点加和倍点运算的几何表示

而按照上图 2.2(b)所示方法则可以求出 P 的倍点 $R(R = 2P)$：首先在 P 点画椭

圆曲线的切线，这条切线与椭圆曲线相交于第二点，这个交点关于 x 轴的对称点即是其二倍点 R 点。这一几何表示示于下图 2.2(b)中。

从上面的几何描述可以得出群运算的代数公式。下面分别对于以下三种简化的*Weierstrass* 方程给出其群运算代数公式：

(i) $y^2 = x^3 + ax + b$ 的加法规则（基础域 K 的特征不等于 2 或 3，如 $K = F_p, P$ 是大于 3 的素数）

1. 零元。对于所有的 $P \in E(K), P + \infty = \infty + P = P$ 。
2. 负元素。若 $P = (x, y) \in E(K)$ ，则 $(x, y) + (x, -y) = \infty$ 。记点 $(x, -y)$ 为 $-P$ ，并称其为 P 的负。$-P$ 也是 $E(K)$ 上的一个点，此外，$\infty = -\infty$ 。
3. 点加。令 $P = (x_1, y_1) \in E(K)$ ， $Q = (x_2, y_2) \in E(K)$ ， $P \neq \pm Q$ ，则 $P + Q = (x_3, y_3)$ 。其中

$$x_3 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2 \quad 和 \quad y_3 = (\frac{y_2 - y_1}{x_2 - x_1})(x_1 - x_2) - y_1$$

4. 倍点。令 $P = (x_1, y_1) \in E(K)$，$P \neq -P$ ，则 $2P = (x_3, y_3)$ 。其中

$$x_3 = (\frac{3x_1^2 + a}{2y_1})^2 - 2x_1 \quad 和 \quad y_3 = (\frac{3x_1^2 + a}{2y_1})(x_1 - x_3) - y_1$$

(ii) $y^2 + xy = x^3 + ax^2 + b$ 的加法规则（非超奇异椭圆曲线 E ， $K = F_{2^m}$ ）

1. 零元。对于所有的 $P \in E(F_{2^m})$ ， $P + \infty = \infty + P = P$
2. 负元素。若 $P = (x, y) \in E(F_{2^m})$ ，则 $(x, y) + (x, x + y) = \infty$ 。记点 $(x, x + y)$ 为 $-P$ ，并称其为 P 的负。注意， $-P$ 也是 $E(F_{2^m})$ 上的一个点，此外 $-\infty = \infty$ 。
3. 点加。令 $P = (x_1, y_1) \in E(F_{2^m})$ ， $Q = (x_2, y_2) \in E(F_{2^m})$ ， $P \neq \pm Q$ ，则 $P + Q = (x_3, y_3)$ 。其中

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad 和 \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1 ,$$

而 $\lambda = \dfrac{y_1 + y_2}{x_1 + x_2}$

4. 倍点。令 $P = (x_1, y_1) \in E(F_{2^m})$ ， $P \neq -P$ ，则 $2P = (x_3, y_3)$ 。其中，

$$x_3 = \lambda^2 + \lambda + a = x_1^2 + \frac{b}{x_1^2} \quad 和 \; y_3 = x_1^2 + \lambda x_3 + x_3$$

而 $\lambda = \dfrac{x_1 + y_1}{x_1}$

(iii) $y^2 + cy = x^3 + ax^2 + b$ 的加法规则（超奇异椭圆曲线 E ， $K = F_{2^m}$ ）

1. 零元。对于所有的 $P=(x_1,y_1)\in E(F_{2^m})$，$P+\infty=\infty+P=P$。
2. 负元素。若 $P=(x_2,y_2)\in E(F_{2^m})$，则 $(x,y)+(x,y+c)=\infty$。记点 $(x,y+c)$ 为 $-P$，并称其为 P 的负。注意，$-P$ 也是 $E(F_{2^m})$ 上的一个点，此外 $-\infty=\infty$。
3. 点加。令 $P=(x_1,y_1)\in E(F_{2^m})$，$Q=(x_2,y_2)\in E(F_{2^m})$，$P\neq\pm Q$，则 $P+Q=(x_3,y_3)$。其中

$$x_3=(\frac{y_2+y_1}{x_2+x_1})+x_1+x_2 \quad 和 \quad y_3=(\frac{y_2+y_1}{x_2+x_1})(x_1+x_3)+y_1+c。$$

4. 倍点。令 $P=(x_1,y_1)\in E(F_{2^m})$，$P\neq -P$，则 $2P=(x_3,y_3)$。其中

$$x_3=(\frac{x_1^2+a}{c})^2-2x_1 \quad 和 \quad y_3=(\frac{x_1^2+a}{c})(x_1+x_3)+y_1+c$$

2.4　点的坐标表示和群的运算符法则

假定域 K 的特征为 2 或 3，我们在上一节给出了定义在域 K 上的椭圆曲线 $y^2=x^3+ax+b$ 的两个点的加法公式，还给出了二进制域上椭圆曲线 $y^2+xy=x^3+ax^2+b$ 的两个点的加法公式。对于这两类曲线，点加（两个互不相同且互不为负的点相加）和倍点运算需要用到域上的求逆和乘法。若域 K 上的求逆比乘法费时，则采用投影坐标表示点将是非常好的一种解决办法。

2.4.1　投影坐标

令 K 是一个域，令 c 和 d 是正整数，集合 $K^3\setminus\{(0,0,0)\}$ 是域 K 上非零三维向量的集合。我们可以定义一个 $K^3\setminus\{(0,0,0)\}$ 上的等价关系~如下：

$(X_1,Y_1,Z_1)\sim(X_2,Y_2,Z_2)$，若对某个 $\lambda\in K^*$ 有 $X_1=\lambda^c X_2$，$Y_1=\lambda^d Y_2, Z_1=\lambda Z_2$

包含 $(X,Y,Z)\in K^3\setminus\{(0,0,0)\}$ 的等价类是

$$(X:Y:Z)=\{(\lambda^c X,\lambda^d Y,\lambda Z):\lambda\in K^*\}$$

$(X:Y:Z)$ 称为投影点，(X,Y,Z) 称为 $(X:Y:Z)$ 的代表元。所有投影点的集合 $P(K)$ 表示。注意，若 $(X',Y',Z')\in(X:Y:Z)$，则 $(X':Y':Z')=(X:Y:Z)$，即等价类的任一元素都可以作为它的代表元。实际上，若 $Z\neq 0$,则 $(X/Z^c,Y/Z^d,1)$ 是投影点 $(X:Y:Z)$ 的代表元，并且是 Z 坐标为 1 的唯一的代表元。于是，我们有投影点集合

$$P(K)^* = \{(X:Y:Z): X,Y,Z \in K, Z \neq 0\}$$

和仿射点集合

$$A(K) = \{(x,y): x,y \in K\}$$

之间的一一映射。

投影点集合

$$P(K)^0 = \{(X:Y:Z): X,Y,Z \in K, Z = 0\}$$

称为无穷远的线，因为它的点并不对应于任何仿射点。通过用 X/Z^c 代替 x 和用 Y/Z^d 代替 y，并去掉分母，可以得到定义在 K 上的椭圆曲线 E 的*Weierstrass* 方程(3.1)的投影坐标形式。现在，若 $(X,Y,Z) \in K^3 \setminus \{(0,0,0)\}$ 满足投影方程，则任何 $(X',Y',Z') \in (X:Y:Z)$。所以，可以说投影点 $(X:Y:Z)$ 在曲线 E 上。因此 $A(K)$ 中的仿射点和 $P(K)^*$ 中的投影点之间有一一映射关系。$P(K)^0$ 中的投影点是 E 的无穷远点。

例 2.1 （标准投影坐标） 令 c=1,d=1.则定义在 K 上的*Weierstrass* 方程

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2-9}$$

的投影形式为

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

曲线 E 上唯一的一个无穷远点是(1:1:0)。这个投影点对应于定义 2.1 中的点 ∞。

投影坐标的点加和倍点运算不涉及域上的求逆。首先把点转换为仿射坐标，然后利用 2.1 节的公式进行仿射点加，最后去掉分母，便得到投影坐标点的计算公式。

例 2.2 （利用雅克比坐标的加法公式）令 c=2,d=3，投影点 $(X:Y:Z)$，$Z \neq 0$ 与仿射点 $(X/Z^2, Y/Z^3)$ 对应。则定义在 K 上的*Weierstrass* 方程

$$E: y^2 = x^3 + ax + b \tag{2-10}$$

的投影形式是

$$Y^2 = X^3 + aXZ^4 + bZ^6。$$

无穷远点与点(1:1:0)对应，而点 $(X:Y:Z)$ 的负点是 $(X:-Y:Z)$。

倍点。令 $P = (X_1:Y_1:Z_1) \in E$，并假设 $P \neq -P$。因为 $P = (X_1/Z_1^2 : Y_1/Z_1^3 : 1)$,所以我们可以利用仿射坐标形式的倍点公式计算 $2P = (X_3' : Y_3' : 1)$，得到

$$X_3^{'}=(\frac{3\frac{X_1^2}{Z_1^4}+a}{2\frac{Y_1}{Z_1^3}})-2\frac{X_1}{Z_1^2}=\frac{(3X_1^2+aZ_1^4)^2-8X_1Y_1^2}{4Y_1^2Z_1^2}$$

和 $Y_3^{'}=(\frac{3\frac{X_1^2}{Z_1^4}+a}{2\frac{Y_1}{Z_1^3}})(\frac{X_1}{Z_1^2}-X_3^{'})-\frac{Y_1}{Z_1^3}=\frac{3X_1^2+aZ_1^4}{2Y_1Z_1}(\frac{X_1}{Z_1^2}-X_3^{'})-\frac{Y_1}{Z_1^3}$

为了在 $X_3^{'}$ 和 $Y_3^{'}$ 的表达式中消去分母，我们令 $X_3=X_3^{'}\cdot Z_3^2$，$Y_3=Y_3^{'}\cdot Z_3^3$，其中 $Z_3=2Y_1Z_1$，于是得到雅克比坐标形式的计算 $2P=(X_3:Y_3:Z_3)$ 的公式：

$$\left.\begin{aligned}X_3&=(3X_1^2+aZ_1^4)^2-8X_1Y_1^2\\Y_3&=(3X_1^2+aZ_1^4)(4X_1Y_1^2-X_3)-8Y_1^4\\Z_3&=2Y_1Z_1\end{aligned}\right\}\tag{2-11}$$

通过存储某些中间结果，可通过如下 6 次域的平方和 4 次域的乘法计算出 X_3，Y_3，Z_3：

$$A\leftarrow Y_1^2,B\leftarrow 4X_1\cdot A,C\leftarrow 8A^2,D\leftarrow 3X_1^2+aZ_1^4,$$
$$X_3\leftarrow D^2-2B,Y_3\leftarrow D\cdot(B-X_3)-C,Z_3\leftarrow 2Y_1\cdot Z_1$$

2.4.2　椭圆曲线 $y^2=x^3+ax+b$

本节讨论定义在域 K（特征不等于 2 且不等于 3）上的椭圆曲线 E：$y^2=x^3+ax+b$ 的坐标系统和加法公式。在这里引入了集中投影坐标。

1. 标准投影坐标。这里 c=1 且 d=1。投影点 $(X:Y:Z)$，$Z\neq 0$ 与仿射点 $(X/Z,Y/Z)$ 对应。则椭圆曲线的投影方程是

$$Y^2Z=X^3+aXZ^2+bZ^3\tag{2-12}$$

无穷远点 ∞ 与点(0:1:0)对应，而点 $(X:Y:Z)$ 的负点是 $(X:-Y:Z)$。

2. 雅克比投影坐标。这里 c=2，且 d=3。投影点 $(X:Y:Z)$，$Z\neq 0$ 与仿射点 $(X/Z^2,Y/Z^3)$ 对应。则椭圆曲线的投影方程是

$$Y^2=X^3+aXZ^4+bZ^6\tag{2-13}$$

无穷远点 ∞ 与点(1:1:0)，而点 $(X:Y:Z)$ 的负点是 $(X:-Y:Z)$。倍点和点加的公式在例 2.2 中已给出。若 $a=-3$，则通过 1 次域乘法和 1 次平方运算就可以计算出式(2-9)中的表达式 $3X_1^2+aZ_1^4$，因为

$$3X_1^2 - 3Z_1^4 = 3(X_1 - Z_1^2)\cdot(X_1 + Z_1^2)。$$

自此以后，我们将假设椭圆曲线 $y^2 = x^3 + ax + b$ 中的 $a = -3$。利用 $2Y_1$ 在式(2-9)中出现了几次的事实，并用乘 4 和乘 8 的计算替换除 2 的计算，则倍点的计算将会提高速度。修改后的倍点公式是

$$A \leftarrow 3(X_1 - Z_1^2)\cdot(X_1 + Z_1^2), B \leftarrow 2Y_1, Z_3 \leftarrow B\cdot Z_1, C \leftarrow B^2,$$

$$D \leftarrow C\cdot X_1, X_3 \leftarrow A^2 - 2D, Y_3 \leftarrow (D - X_3)\cdot A - C^2/2$$

$a = -3$ 时的点加和倍点计算过程在算法 3.2.1 和算法 3.2.2 中给出，其中努力使临时变量 T_i 的数最小。算法用域的基本运算来书写，然而用基本运算集成的专用程序将是更优的。

3. *Chudnovsky* 坐标。这里雅克比点 $(X:Y:Z)$ 被表示为 $(X:Y:Z:Z^2:Z^3)$。这种表示的冗余性在某些采用投影坐标的点乘方法中是有益的。

算法 2.1 倍点（$y^2 = x^3 - 3x + b$，雅克比坐标）

输入：椭圆曲线 E/K：$y^2 = x^3 - 3x + b$，雅克比坐标中的点 $P = (X_1 : Y_1 : Z_1)$

输出：雅克比坐标中的点 $2P = (X_3 : Y_3 : Z_3)$

计算过程如下所示：

1.若 $P = \infty$, 则返回（∞）	
2. $T_1 \leftarrow Z_1^2$	$\{T_1 \leftarrow Z_1^2\}$
3. $T_2 \leftarrow X_1 - T_1$	$\{T_2 \leftarrow X_1 - Z_1^2\}$
4. $T_1 \leftarrow X_1 + T_1$	$\{T_1 \leftarrow X_1 + Z_1^2\}$
5. $T_2 \leftarrow T_2 \cdot T_1$	$\{T_2 \leftarrow X_1^2 - Z_1^4\}$
6. $T_2 \leftarrow 3T_2$	$\{T_2 \leftarrow A = 3(X_1 - Z_1^2)(X_1 + Z_1^2)\}$
7. $Y_3 \leftarrow 2Y_1$	$\{Y_3 \leftarrow B = 2Y_1\}$
8. $Z_3 \leftarrow Y_3 \cdot Z_1$	$\{Z_3 \leftarrow BZ_1\}$
9. $Y_3 \leftarrow Y_3^2$	$\{Y_3 \leftarrow C = B^2\}$
10. $T_3 \leftarrow Y_3 \cdot X_1$	$\{T_3 \leftarrow D = CX_1\}$
11. $Y_3 \leftarrow Y_3^2$	$\{Y_3 \leftarrow C^2\}$
12. $Y_3 \leftarrow Y_3 / 2$	$\{Y_3 \leftarrow C^2/2\}$
13. $X_3 \leftarrow T_2^2$	$\{X_3 \leftarrow A^2\}$
14. $T_1 \leftarrow 2T_3$	$\{T_1 \leftarrow 2D\}$
15. $X_3 \leftarrow X_3 - T_1$	$\{X_3 \leftarrow A^2 - 2D\}$

16.$T_1 \leftarrow T_3 - X_3$　　$\{T_1 \leftarrow D - X_3\}$
17.$T_1 \leftarrow T_1 \cdot T_2$　　$\{T_1 \leftarrow (D - X_3)A\}$
18.$Y_3 \leftarrow T_1 - Y_3$　　$\{Y_3 \leftarrow (D - X_3)A - C^2/2\}$
19.返回$(X_3 : Y_3 : Z_3)$

算法 2.2 点加（$y^2 = x^3 - 3x + b$，仿射-雅克比坐标）
输入：椭圆曲线 E/K： $y^2 = x^3 - 3x + b$,雅克比坐标中的点 $P = (X_1 : Y_1 : Z_1)$，仿射坐标中的点 $Q = (x_2, y_2)$
输出：雅克比坐标中的 $P + Q = (X_3 : Y_3 : Z_3)$
1.若$Q = \infty$,则返回$(X_1 : Y_1 : Z_1)$
2.若$P = \infty$，则返回（$x_2 : y_2 : 1$）
3.$T_1 \leftarrow Z_1^2$　　$\{T_1 \leftarrow A = Z_1^2\}$
4.$T_2 \leftarrow T_1 \cdot Z_1$　　$\{T_2 \leftarrow B = Z_1 \cdot A\}$
5.$T_1 \leftarrow T_1 \cdot x_2$　　$\{T_1 \leftarrow C = X_2 \cdot A\}$
6.$T_2 \leftarrow T_2 \cdot y_2$　　$\{T_2 \leftarrow D = Y_2 B\}$
7.$T_1 \leftarrow T_1 - X_1$　　$\{T_1 \leftarrow E = C - X_1\}$
8.$T_2 \leftarrow T_2 - Y_1$　　$\{T_2 \leftarrow F = D - Y_1\}$
9.若$T_1 = 0$，则

9.1若$T_2 = 0$，则用算法2.1计算$(X_3 : Y_3 : Z_3) = 2(x_2 : y_2 : 1)$,并且返回$(X_3 : Y_3 : Z_3)$
9.2否则，返回（∞）

10.$Z_3 \leftarrow Z_1 \cdot T_1$　　$\{Z_3 \leftarrow Z_1 E\}$
11.$T_3 \leftarrow T_1^2$　　$\{T_3 \leftarrow G = E^2\}$
12.$T_4 \leftarrow T_3 \cdot T_1$　　$\{T_4 \leftarrow H = E^3\}$
13.$T_3 \leftarrow T_3 \cdot X_1$　　$\{T_3 \leftarrow I = X_1 G\}$
14.$T_1 \leftarrow 2T_3$　　$\{T_1 \leftarrow 2I\}$
15.$X_3 \leftarrow T_2^2$　　$\{X_3 \leftarrow F^2\}$
16.$X_3 \leftarrow X_3 - T_1$　　$\{X_3 \leftarrow F^2 - 2I\}$
17.$X_3 \leftarrow X_3 - T_4$　　$\{X_3 \leftarrow F^2 - (H + 2I)\}$
18.$T_3 \leftarrow T_3 - X_3$　　$\{T_3 \leftarrow I - X_3\}$
19.$T_3 \leftarrow T_3 \cdot T_2$　　$\{T_3 \leftarrow F(I - X_3)\}$
20.$T_4 \leftarrow T_4 \cdot Y_1$　　$\{T_4 \leftarrow Y_1 H\}$
21.$Y_3 \leftarrow T_3 - T_4$　　$\{Y_3 \leftarrow F(I - X_3) - Y_1 H\}$
22.返回$(X_3 : Y_3 : Z_3)$

在不同坐标系中点加和倍点所需的域运算的量列于表 2.1 中。其中符号 $C_1 + C_2 \to C_3$ 说明要加的点在 C_1 坐标和 C_2 坐标中，而和在 C_3 坐标中。例如

$J+A\to J$ 表示雅克比坐标与仿射坐标的点加，而结果是雅克比坐标。我们看到雅克比坐标的倍点计算速度最快，而雅克比与仿射坐标的点加计算最快。

表 2.1 椭圆曲线 $y^2=x^3-3x+b$ 的点加和倍点的运算量。其中 A 表示仿射坐标，P 表示标准投影坐标，J 表示雅克比坐标，C 表示 *Chudnovsky* 坐标，I 表示求逆，M 表示点乘，S 表示平方。

倍　点		普通点加		混合点加	
$2A\to A$	$1I,2M,2S$	$A+A\to A$	$1I,2M,1S$	$J+A\to J$	$8M,3S$
$2P\to P$	$7M,3S$	$P+P\to P$	$12M,2S$	$J+C\to J$	$11M,3S$
$2J\to J$	$4M,4S$	$J+J\to J$	$12M,4S$	$C+A\to C$	$8M,3S$
$2C\to C$	$5M,4S$	$C+C\to C$	$11M,3S$		

2.5　本章小结

本章首先介绍了椭圆曲线的基本概念，椭圆曲线上点加和点乘运算在代数和几何上的表示，对于几种经典域上的椭圆曲线的运算进行了介绍。最后介绍了椭圆曲线上点的坐标表示和群的运算法则。

第三章　双标量乘算法研究及改进

本章前面两节介绍了标量乘算法中已经比较成熟的算法，包括单标量乘算法和双标量乘算法，第三节在充分分析 Shamir 快速算法和交错 NAF 方法的优缺点的基础上，基于 Shamir 快速算法提出了一种改进的双标量乘方法，该方法在没有明显增加点乘运算量的前提下，至少节约内存 50%以上，降低了对内存设备的要求。

3.1　单标量乘经典算法

设 k 是一个大整数，P 是定义在域 F_q 上的椭圆曲线 E 上的一个点。求点 P 的倍点 kP 这一计算称为点乘或标量乘，它决定着椭圆曲线密码体制的运算速度。$kP+lQ$ 的计算被称为多点乘，它决定着某些椭圆曲线密码体制的效率，例如 ECDSA 的签名验证。本节先讨论单标量乘算法。本节所讨论的点乘均针对未知点。

3.1.1　二元法

类似于乘法群中求模幂运算的平方--乘算法，整数 k 可以表示成长度为 1 的二进制序列 $k=\sum_{i=0}^{i=t-1}a_i 2^i, a\in\{0,1\}$，则 $kP=\sum_{i=0}^{i=t-1}a_i 2^i P$

算法 3.1.1 和算法 3.1.2[41]是计算幂的反复平方乘算法的扩充。算法 3.1.1 从右到左处理 k 的各位，而算法 3.1.2 却是从左向右处理。假定 m 为 k 的二进制位数。

算法 3.1.1 计算点乘的从右向左的二进制方法

输入：$k=(k_{t-1},...,k_1,k_0)_2$，$P\in E(F_q)$

输出：kP

1. $Q\leftarrow\infty$
2. 对于 i 从 0 到 $t-1$，重复执行

   2.1 若 $k_i=1$，则 $Q\leftarrow Q+P$

   2.2 $P\leftarrow 2P$
3. 返回 (Q)。

算法 3.1.2 计算点乘的从左向右的二进制方法

输入：$k=(k_{t-1},...,k_1,k_0)_2$，$P\in E(F_q)$

输出：kP

1. $Q \leftarrow \infty$
2. 对于i从$t-1$到0，重复执行
   2.1 $Q \leftarrow 2Q$
   2.2 若$k_i = 1$，则$Q \leftarrow Q + P$
3. 返回(Q)

k的二进制表示中1的个数的期望值为$t/2 \approx m/2$，因而算法3.1.2的期望运行时间近似为$m/2$次点加和m次二倍点，将加法运算用A表示，倍点运算用D表示，则可表示为：

$$\frac{m}{2}A + mD$$

3.1.2 非相邻表示型(NAF)

设$P = (x, y) \in E(F_q)$，若F_q是二进制域，则$-P = (x, x + y)$；若F_q的特征大于3，则$-P = (x, x - y)$。因此椭圆曲线点的减法与加法一样有效。这促使算法可以使用k的带符号的数字表示$k = \sum_{i=0}^{l-1} k_i 2^i$，其中$k \in \{0, \pm 1\}$。一种特别有用的带符号数字表示是非相邻表示型$(NAF)$[9]。

定义 3.1 一个正整数k的非相邻表示型为表达式$k = \sum_{i=0}^{l-1} k_i 2^i$，其中$k \in \{0, \pm 1\}$，$k_{l-1} \neq 0$，并且没有两个连续的数字$k_i$是非零的，$NAF$的长度是$l$。

定理 3.1 (NAF)的性质，令k是一个正整数

① k具有唯一的NAF表示形式，并记为$NAF(k)$。

② $NAF(k)$在k的所有带符号表示中具有最小的非零位。

③ $NAF(k)$的长度最多比k的二进制表示的长度大1。

④ 若$NAF(k)$的长度是l，则$2^l/3 < k < 2^{l+1}/3$。

⑤ 所有长度为l的NAF中非零数字的平均密度约为1/3。

该定理的详细证明见文献[9]，此处不再赘述。

利用下面给出的算法3.1.3可以有效地计算$NAF(k)$。$NAF(k)$的各位数字可以通过连续用2去除且允许余数取0或± 1来得到。若k是奇数，则余数$r \in \{-1, 1\}$，以使商$(k-r)/2$是偶数，这将确保下一个NAF数字是0。

算法 3.1.3 计算一个正整数的NAF

输入：一个正整数k

输出：$NAF(k)$

1. $i \leftarrow 0$

2. 当$k \geq 1$时，重复执行

2.1 若k是奇数，则$k_i \leftarrow 2-(k \bmod 4)$，$k_i \leftarrow k-k_i$

2.2 否则，$k_i \leftarrow 0$

2.3 $k_i \leftarrow k/2$，$i \leftarrow i+1$

3. 返回$(k_{l-1}, k_{l-2}, ..., k_1, k_0)$

接下来的算法修改了从左到右的二进制点乘方法（算法 3.1.2），用$NAF(k)$代替了k的二进制表示。

算法 3.1.4 用二进制NAF方法计算点乘

输入：一个正整数k，$P \in E(F_q)$

输出：kP

1. 用算法 3.1.3 计算

2. $Q \leftarrow \infty$

3. 对于i从$l-1$到 0，重复执行

3.1 $Q \leftarrow 2Q$

3.2 若$k_i = 1$，则$Q \leftarrow Q+P$

3.3 若$k_i = -1$，则$Q \leftarrow Q-P$

4. 返回(Q)

算法 3.1.4 的期望运行时间近似为

$$\frac{m}{3}A + mD$$

3.1.3　NAF 窗口方法

若可以利用额外的存储器，采用一次处理k的w位的窗口方法，则算法 3.1.4 的运行时间还可以减少。

定义 3.2 令$w \geq 2$是正整数，正整数k的宽度为w的NAF是表达式 $k = \sum_{i=0}^{l-1} k_i 2^i$，其中每一个非零系数$k_i$都是奇数，$|k_i| < 2^{w-1}$，$k_{l-1} \neq 0$，并且任何连续$w$个数字中最少 1 位为非零。宽度为$w$的$NAF$的长度是$l$。

定理 3.2 （宽度w的 NAF 的性质）令k是一个正整数

① k 有唯一的宽度为 w 的 NAF，并记为 $NAF_w(k)$。

② $NAF_2(k) = NAF\ (k)$。

③ $NAF_w(k)$ 的长度最多比 k 的二进制表示的长度大 1。

④ 在所有长度为 l 的宽度 w 的 NAF 中，平均非零数字的密度近似为 $l/(w+1)$。

定理 3.2 的证明类似于定理 3.1 的证明，详见[56], 不再赘述。

算法 3.1.5 计算一个正整数的窗口宽度 w 的 NAF

输入：窗口宽度 w，一个正整数 k

输出：$NAF_w(k)$

1. $i \leftarrow 0$

2. 当 $k \geq 1$ 时，

2.1 若 k 是奇数，则 $k_i \leftarrow k \bmod 2^w$， $k \leftarrow k - k_i$

2.2 否则，$k_i \leftarrow 0$

2.3 $k \leftarrow k/2$， $i \leftarrow i+1$

3. 返回 $(k_{l-1}, k_{l-2}, ..., k_1, k_0)$

利用算法 3.1.5 可以有效地计算 $NAF_w(k)$，这里 $k \bmod 2^w$ 表示一个正整数 u，u 满足 $u \equiv k (\bmod\ 2^w)$ 且 $-2^{w-1} \leq u \leq 2^{w-1}$。$NAF_w(k)$ 的各位数字通过 k 连续除 2^w 并使余数 r 在区间 $-2^{w-1} \leq u \leq 2^{w-1}$ 内来获得。若 k 是奇数并且余数 $r \equiv k (\bmod 2^w)$，则 $(k-r)/2$ 将能被 2^{w-1} 整除，从而使后面的 $w-1$ 个数字为零。

算法 3.1.6 计算标量乘的窗口 NAF 方法。

输入：窗口宽度 w，一个正整数 k， $P \in E(F_q)$

输出：kP

1. 用算法 3.1.5 计算 $NAF_w(k) = \sum_{i=0}^{l-1} k_i 2^i$

2. 对于 $i \in \{1,3,5,...,2^{w-1}-1\}$，计算 $P_i = iP$

3. $Q \leftarrow \infty$

4. 对于 i 从 $l-1$ 到 0，重复执行

4.1 $Q \leftarrow 2Q$

4.2 若 $k_i \neq 0$，则

若 $k_i > 0$，则 $Q \leftarrow Q + P_{k_i}$

否则，$Q \leftarrow Q - P_{k_i}$

5. 返回 (Q)

算法 3.1.6 通过用 $NAF_w(k)$ 代替 $NAF(k)$，扩展了二进制 NAF 方法。若依据定理 3.2 的③和④，则算法 3.1.6 的期望运行时间近似为

$$\frac{l}{w+1}A+lD$$

3.2 多标量乘经典算法

3.2.1 直接算法

简单地说，这个算法“差不多”就是分别计算 m 个标量乘；区别在于采用了同时多标量乘的技巧，即同时进行倍点运算。这是最简单，却也是最慢的算法。

算法 3.2.1 计算多标量乘的直接方法

输入：$GF(q),E,P,(k_{i,L-1},k_{i,L-2},\cdots,k_{i,1},k_{i,0})_2,1\le i\le m$

输出：$k_1P_1+k_2P_2+\cdots k_mP_m$

1. $Q\leftarrow O$
2. For $j=L-1$ downto 0
3. $Q\leftarrow 2Q$
4. For $i=1$ to m
5. If $k_{i,j}=1$ then $Q=Q+P_i$
6. Output Q

一般性地假设 $k_{i,j}\in\{0,1\}$ 取值为 0 和取值为 1 是等概率的，于是算法 3.2.1 平均要进行的点加次数是 $mL/2$，倍点次数是 L。

3.2.2 Shamir 快速算法

Shamir 算法的最主要思想是对于 $(P_1,P_2,\cdots,P_m)\cdot(k_{1,j},k_{2,j},\cdots,k_{m,j})^T$ 进行预计算和存储。

算法 3.2.2 计算多标量乘的 Shamir 方法

输入：$GP(q),E,P,(k_{i,L-1},k_{i,L-2},...,k_{i,1},k_{i,0})_2,1\le i\le m$

输出：$k_1P_1+k_2P_2+\cdots k_mP_m$

1. 预计算过程：
2. For $i=1$ to 2^m-1
3. Let $i=(i_1,i_2,\cdots,i_m)_2$

4. Compute and store $Q_i = (P_1, P_2, \cdots, P_m) \cdot (i_1, i_2, \cdots, i_m)^T$

5. 主计算过程：

6. $Q \leftarrow O$

7. For $j = L-1$ downto 0

8. $Q \leftarrow 2Q$

9. If $I_j \neq 0$ then $Q \leftarrow Q + Q_{I_j}$

10. *Output* (Q)

同样地，算法 3.2.2 需要L次倍点运算，而其点加运算的平均次数则取决于$I_j \neq (0,\cdots,0)^T$。

（下文中，在不引起混下的前提下，也简记为$I_j \neq 0$）的概率，显然有$P(I_j = 0) = 1/2^m$，于是点加运算的平均次数为$(1-1/2^m)L$。

3.2.3 Shamir-NAF 算法

椭圆曲线上点的减法与加法具有同等时间复杂度，因此可以使用整数的带符号二进制（*signed binary*）表示方法。即将一个正整数k表示成为$(K_{t-1}, K_{t-2}, \cdots, K_1, K_0)_s$，使得$k = \sum_{l=0}^{t-1} K_l 2^l$， 其中$K_l \in \{0,1,-1\}$ (为简洁期间，用$\bar{1}$表示-1)，$0 \le l \le t-1$。

与正整数的二进制表示方法不同，带符号的二进制表示具有以下特性：

①一个正整数的带符号二进制表示方法不是唯一的，例如$5 = (1,0,1)_s = (1,1,\bar{1})_s$

②如果令$k = (k_{L-1}, k_{L-2}, \cdots, k_1, k_0)_2$是$k$的二进制表示，且$k_{L-1} \neq 0$，而$(K_{t-1}, K_{t-2}, \cdots, K_1, K_0)_s$是$k$的一个带符号的二进制表示方法，则明显地有$t \ge L$；

③ 一个正整数的所有带符号二进制表示方法的长度（即t）没有上界--因为对于任意的$T > 1$，都可以将 1 表示为$1 = (K_{T-1}, K_{T-2}, \cdots, K_0)$，其中$K_{T-1} = 1$，$K_j = \bar{1}$，$0 \le j \le T-2$。

因此，提出了正整数的非相邻表示形式(Non Adiacent Form,NAF)表示方法：$k = (K_{t-1}, K_{t-2}, \cdots, K_1, K_0)_{NAF}$。

在 3.1.3 节给出正整数的 NAF 表示方法后，下面给出 Shamir-NAF 算法：

假设$k_i = (K_{i,L}, K_{i,L-1}, \cdots, K_{i,1}, K_{i,0})_{NAF}$，$1 \le i \le m$。在不引起混淆的前提下，仍然记为$I_j = (k_{1,j}, k_{2,j}, \cdots, k_{m,j})^T$，即$(k_1, k_2, \cdots, k_m)^T$的*NAF*表示的第$j$列，$0 \le j \le L+1$。

算法 3.2.3 计算多标量乘的 Shamir-NAF 算法

输入：$GF(q), E, P, (K_{i,L}, K_{i,L-1}, \cdots, K_{i,1}, K_{i,0})_{NAF}, 1 \le i \le m$

输出：$k_1P_1+k_2P_2+\cdots k_mP_m$

1. 预计算

2. For all $T=(T_1,T_2,\cdots,T_m)$，$T_l\in\{0,1,\bar{1}\}$，$1\le l\le m$

3. Compute and store $Q_T=(P_1,P_2,\cdots,P_m)\#(T_1,T_2,\cdots,T_m)^T$

4. 主循环

5. $Q\leftarrow O$

6. For $j=L$ downto 0

7. $Q\leftarrow 2Q$

8. If $I_j\neq(0,0,\cdots,0)$ then $Q\leftarrow Q+Q_{I_j}$

9. Output (Q)

算法 3.2.3 需要$L+1$次倍点运算，而$I_j=0$的概率为$P(I_j=0)=(2/3)^m$，于是点加运算的平均次数为$(1-(2/3)^m)L$。与算法 3.2.3 相比，倍点次数至多增加一次，而点加运算次数则大幅度减少。

3.2.4　Solinas 算法

对于$m=2$的情形，Solinas 定义了(k_1,k_2)“联合稀疏形式”[45],并且给出了生成算法。这也是一种带符号的二进制表示方法，$k_i=(K_{i,t-1},K_{i,t-2},\cdots,K_{i,1},K_{i,0})_{Sol}$，$i=0,1$。Solinas 已经证明：在所有的$(k_1,k_2)$的带符号的二进制表示方法中，“联合稀疏形式”的全“0”列的数目是最多的，列$(K_{i,0},K_{j,0})^T=(0,0)^T$的概率为 1/2。使用 Solinas 表示法计算$k_1P_1+k_2P_2$的算法如下（Solinas 的生成算法过长，且仅适合$m=2$的情形，因此在此文中不予尽述）

算法 3.2.4 计算多标量乘的 Solinas 算法

输入：$GP(q),E,P,(K_{i,t-1},K_{i,t-2},\cdots,K_{i,1},K_{i,0})_{Sol}$，$i=0,1$

输出：$k_1P_1+k_2P_2$

1. 预计算：

2. Compute and store $Q_{(1,1)}=P_1+P_2$，$Q_{(1,\bar{1})}=P_1-P_2$，$Q_{(\bar{1},\bar{1})}=-Q_{(1,1)}$，$Q_{(\bar{1},1)}=-Q_{(1,\bar{1})}$

3. 主循环。

4. $Q\leftarrow O$

5. For $j=t-1$ downto 0

6. $Q\leftarrow 2Q$

7. If $I_j \neq (0,0)$, then $Q \leftarrow Q + Q_{I_j}$

8. Output (Q)

算法 3.2.4 需要 L 或者 $L+1$ 次倍点运算，平均 $L/2$ 次点加运算。

3.3 改进的双标量乘算法

3.3.1 一种改进的双标量乘算法

下面先给出已有的经典算法 Shamir 双标量乘快速算法[9]和交错 NAF 方法，然后给出作者提出的改进算法，之后对三种算法进行对比，分析其优缺点。

给出 Shamir 快速双标量乘算法如下：

算法 3.3.1 Shamir 快速双标量乘算法

输入：窗口宽度 w, $k=(k_{t-1},...,k_1,k_0)_2$, $l=(l_{t-1},...,l_0)_2$, $P,Q \in E(F_q)$

输出：$kP+lQ$

1. 对于所有 $i,j \in [0,2^w-1]$, 计算 $iP+jQ$

2. 记 $k=(K^{d-1},...,K^1,K^0)$, $l=(L^{d-1},...,L^1,L^0)$, 其中每个 K^i， L^i 是长度为 w 的位串， $d=\lceil t/w \rceil$

3. $R \leftarrow \infty$

4. 对于 i 从 $d-1$ 到 0,重复执行

   4.1 $R \leftarrow 2^w R$

   4.2 $R \leftarrow R+(K^iP+L^iQ)$

5. 返回(R)

下图为 Shamir 快速双标量乘算法的示意图

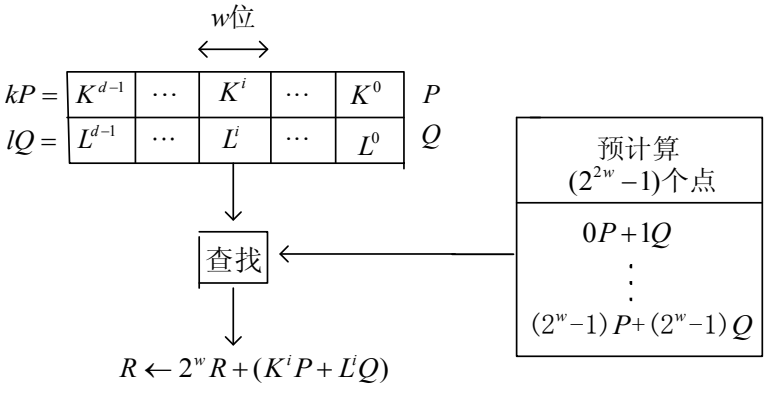


图 3.1 同时点乘累加步

Shamir 快速算法需要存储的点个数为[57]

$$2^{2w}-1 \tag{3-1}$$

Shamir 快速算法的期望运行时间近似为[57]

$$[(3\cdot 2^{2(w-1)}-2^{w-1}-1)A+(2^{2(w-1)}-2^{w-1})D]+[(\frac{2^{2w}-1}{2^{2w}}d-1)A+(d-1)wD] \tag{3-2}$$

Shamir 快速算法是一种巧妙提高计算 $(kP+lQ)$ 速度的多点乘方法。也被称 Shamir 窍门方法。若 k 和 l 为 t 位数，则它的二进制表示可写成一个 $2\times t$ 代表矩阵。给定宽度 w，对于 $0\le i$， $j<2^w$，计算值 $iP+jQ$。在 $\lceil t/w\rceil$ 步中的每一步，累加器从值 $iP+jQ$ 的表接收 w 次倍点，一次点加，而这些值由移过代表矩阵的 $2w$ 窗口来确定。

对于 Shamir 快速算法，当窗口宽度 w 较小时，点的存储量较小，同时存储点的利用效率也比较高，此时该方法具有较高的效率和较强的实用性。但是随着窗口宽度 w 变大，该方法就表现出预计算量大，点存储量大的特点。之所以预计算量和点存储量较大是因为预计算中交叉项 $iP+jQ$ 的个数太多。

另一方面，Shamir 快速算法划分的窗口个数为 $\lceil t/w\rceil$，当窗口宽度 w 变大时，划分窗口个数减少，从而可使主计算循环次数减少。而循环中包含点加和倍点运算，因而 w 增大可使点加和倍点次数减少。因此，可以看到，w 的增大导致预计算量和点存储量增加的同时，也使得点加和倍点次数减少。但是预计算量和点存储的大量增加制约了 w 的进一步增加。

算法 3.3.2 交错 NAF 方法计算双标量乘

输入：v,整数 k^j,宽度 w^j, 点 P^j, $1\le j\le v$

输出：$\sum_{j=1}^{v}k^jP_j$

1. 对于 $i\in\{1,3,...,2^{w_j-1}-1\}$, 计算 iP_j, $1\le j\le v$
2. 利用算法 3.1.5 计算, $NAF_{w_j}(k^j)=\sum_{i=0}^{l_j-1}k_i^j2^i$， $1\le j\le v$
3. 令 $l=\max\{l_j:1\le j\le v\}$
4. 定义 $k_i^j=0, l_j\le i<l, 1\le j\le v$
5. $Q\leftarrow\infty$
6. 对于 i 从 $l-1$ 到 0，重复执行

   6.1 $Q\leftarrow 2Q$

   6.2 对于 j 从 1 到 v，重复执行

若 $k_i^j \neq 0$,则

若 $k_i^j > 0$，则 $Q \leftarrow Q + k_i^j P_j$

否则，$Q \leftarrow Q - k_i^j P_j$

7. 返回(Q)。

下图为交错 NAF 方法的示意图

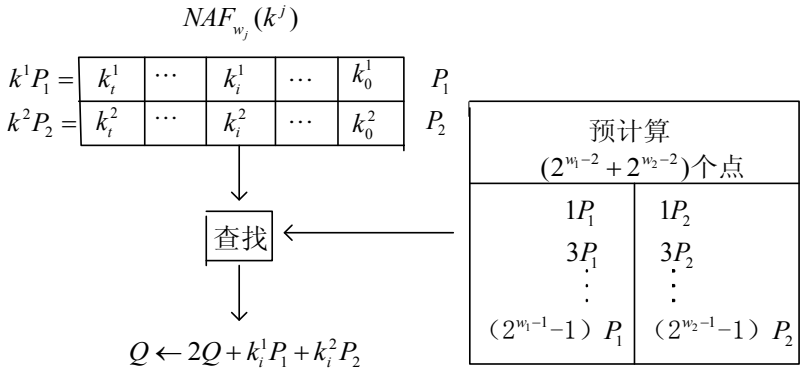


图 3.2 用交错 NAF 方法计算$k^1P_1 + k^2P_2$。对于$v = 2$个点示出了点乘的累加步骤。标量k^j被写为宽度为w_j的 NAF 形式

从图 3.2 可知，当多点标量乘为双标量乘形式时，交错 NAF 算法需要保存的点的个数为[57]

$$2^{w_1-2} + 2^{w_2-2} \tag{3-3}$$

该算法的期望运行时间近似为[57]

$$[|\{j : w_j > 2\}| D + \sum_{j=1}^{v}(2^{w_j-2} - 1)A] + [\max_{1 \le j \le v} l_j D + \sum_{j=1}^{v} \frac{l_j}{w_j + 1} A] \tag{3-4}$$

Shamir 快速算法需要保存点的个数为 $2^{2w} - 1$，假设此时的 $w = w_1 = w_2$，则 Shamir 快速算法需要保存的点个数大约是交错 NAF 方法的 $(2^{2w} - 1)/(2^{w_1-2} + 2^{w_2-2})$ 倍，大约为 2^{w+1} 倍 。因此可以看到交错 NAF 方法能够大量减少点存储量，同时预计算量也大为减少。这是因为交错 NAF 方法预计算时不计算和存储交叉项 $iP + jQ$ 。

另一方面，假设整数 k 和 l 的位数大约为 t 位，则交错 NAF 方法所需二倍点乘次数为 t 次，而 Shamir 方法则需要 $d = \lceil t/w \rceil$ 次 2^w 倍点乘，Shamir 方法所需点乘次数更少，大约是交错 NAF 方法的 $1/w$。

因此，可以设想将交错 NAF 方法和 Shamir 快速算法相结合，以达到既减少点

存储量同时又减少点乘次数的目的。如果能减少点存储量，就可以使窗口宽度w尽量增加，从而得到减少主计算循环次数的好处。

因此提出一种改进的双标量乘算法 3.3.3 如下：

算法 3.3.3 改进的双标量乘算法

输入：窗口宽度w，$k=(k_{t-1},...,k_1,k_0)_2$，$l=(l_{t-1},...,l_0)_2$，$P,Q\in E(F_q)$

输出：$kP+lQ$

1. 对于所有$i,j\in[0,2^w-1]$, 计算iP以及jQ
2. 记$k=(K^{d-1},...,K^1,K^0)$, $l=(L^{d-1},...,L^1,L^0)$, 其中每个K^i,L^i是长度为w的位串，$d=\lceil t/w\rceil$
3. $R\leftarrow\infty$
4. 对于i从$d-1$到 0, 重复执行

   4.1 $R\leftarrow 2^wR$

   4.2 $R\leftarrow R+(K^iP+L^iQ)$
5. 返回(R)

该方法综合了 Shamir 快速算法和交错 NAF 的优点。相对于 Shamir 快速算法，改进算法在预计算阶段只计算和存储P和Q的倍点，不计算交叉项$iP+jQ$，减少了预计算量和点存储量。相对于交错 NAF 方法，在主计算时，改进算法进行了$d=\lceil t/w\rceil$次2^w倍点乘，交错 NAF 方法进行了t次二倍点乘。对于改进双标量乘算法中的2^w倍点乘，当逆运算开销较大时，可通过底层域快速算法2^kP将逆运算转换为乘运算，从而降低运算量。

但是，改进算法将交叉项$iP+jQ$的计算放到了主计算中进行也可能导致主计算中计算量的增加。因此下面将具体分析当w分别等于2,3,4时，改进算法与Shamir快速算法在点的存储量和点乘计算量两个方面的优劣。其中，用A表示点加运算，用D表示二倍点乘。分别用M,S,I表示素域上的乘法、平方和求逆运算。在域$GF(p)$上，点加运算A的开销为$1I+1S+2M$[56]，二倍点乘的开销为$1I+2S+2M$。为不失一般性，我们假设$S=0.8M$, $I/M=6$。

(1) 当$w=2$时

为讨论方面，假设k和l均为160bit，则Shamir快速算法划分的窗口个数为160/2=80个。

(i) 对于 Shamir 快速算法，根据式(3-1)，当$w=2$时，需要保存的点个数为$2^{2w}-1$=15 个。

根据式(3-2)，预计算运算量约为

$$[(3\cdot 2^{2(w-1)}-2^{w-1}-1)A+(2^{2(w-1)}-2^{w-1})D] \tag{3-5}$$

主计算运算量为

$$[(\frac{2^{2w}-1}{2^{2w}}d-1)A+(d-1)wD] \tag{3-6}$$

代入$w=2$,得到预计算量为

$$9A+2D=9(1I+1S+2M)+2(1I+2S+2M)=98.4M$$

代入$w=2$，得到主计算量为

$$74A+158D=74(1I+1S+2M)+158(1I+2S+2M)=2168M$$

因此当$w=2$时，总的计算开销为$98.4M+2168M=2266.4M$

(ii) 对于改进双标量乘算法，需要存储P、Q、$2P$、$2Q$、$3P$、$3Q$共六个点。并且划分窗口个数为160/2=80个窗口。

预计算需要计算的点为$2P$、$3P$、$2Q$、$3Q$。其中计算点$2P$、$2Q$,共需两次二倍点乘，计算$3P$、$3Q$，共需两次点加。

因此预计算的计算量为

$$2A+2D=2(1I+1S+2M)+2(1I+2S+2M)=36.8M$$

主计算分为点乘步骤$R\leftarrow 2^wR$和点加步骤$R\leftarrow R+(K^iP+L^iQ)$两部分。

点乘步骤$R\leftarrow 2^wR$需要$(d-1)$次2^w倍点乘。为了方便，计算$(d-1)$次2^w倍点可通过计算$(d-1)*w$次二倍点来实现。当k和l大约为160bit时，改进双标量乘算法需要的二倍点乘次数与 Shamir 快速算法相同，均为$(d-1)*w$=(160/2-1)*2=158次。

点加步骤$R\leftarrow R+(K^iP+L^iQ)$计算过程分析如下：

计算点加步骤$R\leftarrow R+(K^iP+L^iQ)$时，交叉项$K^iP+L^iQ$的计算放到主计算中进行。交叉项$K^iP+L^iQ$的系数$K^i\in\{0,1,2,3\}$，$L^i\in\{0,1,2,3\}$。因此交叉项$K^iP+L^iQ$共有4*4=16种可能的组合，列表3.1如下：

表 3.1

(K^i,L^i)	(K^i,L^i)	(K^i,L^i)	(K^i,L^i)
(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
(2,0)	(2,1)	(2,2)	(2,3)
(3,0)	(3,1)	(3,2)	(3,3)

因为划分窗口数为80个，故每种组合平均出现约80*1/16=5次。

对于点加运算$R\leftarrow R+(K^iP+L^iQ)$，所耗费的计算量如下表3.2所示（其中每种组合平均出现5次）：

表 3.2

$R \leftarrow R+(K^iP+L^iQ)$		$R \leftarrow R+(K^iP+L^iQ)$	
$(0,0)\rightarrow(0P,0Q)$	0A	$(0,1)\rightarrow(0P,1Q)$	5A
$(0,2)\rightarrow(0P,2Q)$	5A	$(0,3)\rightarrow(0P,3Q)$	5A
$(1,0)\rightarrow(1P,0Q)$	5A	$(1,1)\rightarrow(1P,1Q)$	5A+5A
$(1,2)\rightarrow(1P,2Q)$	5A+5A	$(1,3)\rightarrow(1P,3Q)$	5A+5A
$(2,0)\rightarrow(2P,0Q)$	5A	$(2,1)\rightarrow(2P,1Q)$	5A+5A
$(2,2)\rightarrow(2P,2Q)$	5A+5A	$(2,3)\rightarrow(2P,3Q)$	5A+5A
$(3,0)\rightarrow(3P,0Q)$	5A	$(3,1)\rightarrow(3P,1Q)$	5A+5A
$(3,2)\rightarrow(3P,2Q)$	5A+5A	$(3,3)\rightarrow(3P,3Q)$	5A+5A

共计 120 次点加运算。

因此主计算中的计算量为

$$120A+158D=120(1I+1S+2M)+158(1I+2S+2M)=2572.8M$$

总计算量为 $36.8M+2572.8M=2609.6M$

当 $w=2$ 时，改进双标量乘算法比 Shamir 快速算法增加的计算量为

$$(2609.6M-2266.4M)/2266.4M\approx 14\%$$

同时改进双标量乘算法比 Shamir 快速算法节省点存储量为(15-6)/15=60%。

(2) 当 $w=3$ 时

为了讨论方面，假设 k 和 l 均为 192bit，此时 Shamir 算法划分窗口个数为 192/3=64。

(i) 对于 Shamir 算法，根据式(3-1)，当 $w=3$ 时，需要保存的点个数为 63 个。

根据式(3-2)，预计算运算量约为

$$[(3\cdot 2^{2(w-1)}-2^{w-1}-1)A+(2^{2(w-1)}-2^{w-1})D] \tag{3-7}$$

主计算运算量约为

$$[(\frac{2^{2w}-1}{2^{2w}}d-1)A+(d-1)wD] \tag{3-8}$$

代入 $w=3$ 得到预计算量为

$$43A+12D=43(1I+1S+2M)+12(1I+2S+2M)=493.6M$$

代入 $w=3$ 得到主计算量为

$$62A+189D=62(1I+1S+2M)+189(1I+2S+2M)=2360M$$

总计算量为 $493.6M+2360M=2853.6M$

(ii) 对于改进标量乘算法，需要存储 $\{P,2P,3P,4P,5P,6P,7P\}$ $\{Q,2Q,3Q,4Q,5Q,6Q,7Q\}$ 共 14 个点。并且划分窗口个数为 192/3=64 个。其中计算点 $2P,2Q$ 共需两次点加，其余各点需要一次点加即可得到。

因此预计算的计算量为

$10A+2D=10(1I+1S+2M)+2(1I+2S+2M)=107.2M$

主计算分为点乘步骤 $R\leftarrow 2^wR$ 和点加步骤 $R\leftarrow R+(K^iP+L^iQ)$ 两部分。

点乘步骤 $R\leftarrow 2^wR$ 需要 $(d-1)$ 次 2^w 倍点乘。为了方便，计算 $(d-1)$ 次 2^w 倍点可通过计算 $(d-1)*w$ 次二倍点来实现。设 k 和 l 大约为 192bit，则改进双标量乘算法需要 $(d-1)*w$=(192/3-1)*3=189 次二倍点乘法。

点加运算 $R\leftarrow R+(K^iP+L^iQ)$ 分析如下：

点加运算 $R\leftarrow R+(K^iP+L^iQ)$ 中 $K^iP\in\{0,P,2P,3P,4P,5P,6P,7P\}$，$L^iQ\in\{0,Q,2Q,3Q,4Q,5Q,6Q,7Q\}$，因此 K^iP+L^iQ 一共可能有 8*8 共 64 种可能的组合。且每种组合出现的平均次数为窗口个数乘以出现概率，即(192/3)*(1/64)=1。计算交叉项 K^iP+L^iQ 所需计算量列表 3.3 如下：

表 3.3

	$0Q$	$1Q$	$2Q$	$3Q$	$4Q$	$5Q$	$6Q$	$7Q$
$0P$	0A	0A	0A	0A	0A	0A	0A	0A
$1P$	0A	1A	1A	1A	1A	1A	1A	1A
$2P$	0A	1A	1A	1A	1A	1A	1A	1A
$3P$	0A	1A	1A	1A	1A	1A	1A	1A
$4P$	0A	1A	1A	1A	1A	1A	1A	1A
$5P$	0A	1A	1A	1A	1A	1A	1A	1A
$6P$	0A	1A	1A	1A	1A	1A	1A	1A
$7P$	0A	1A	1A	1A	1A	1A	1A	1A

故计算 $R\leftarrow R+(K^iP+L^iQ)$ 所需计算量列表 3.4 如下：

表 3.4

	$0Q$	$1Q$	$2Q$	$3Q$	$4Q$	$5Q$	$6Q$	$7Q$
$0P$	0A	1A	1A	1A	1A	1A	1A	1A
$1P$	1A	2A	2A	2A	2A	2A	2A	2A
$2P$	1A	2A	2A	2A	2A	2A	2A	2A
$3P$	1A	2A	2A	2A	2A	2A	2A	2A
$4P$	1A	2A	2A	2A	2A	2A	2A	2A
$5P$	1A	2A	2A	2A	2A	2A	2A	2A
$6P$	1A	2A	2A	2A	2A	2A	2A	2A
$7P$	1A	2A	2A	2A	2A	2A	2A	2A

一共需要 112 次点加，因此总的计算量为

$$112A+189D=112(1I+1S+2M)+189(1I+2S+2M)=2800M$$

改进双标量乘算法总的计算量为$107.2M+2800M=2907.2M$

因此当$w=3$时，相对于 Shamir 快速方法，改进双标量乘算法增加的计算量为$(2907.2M-2853.6M)/2853.6M\approx 1.8\%$。节省的点存储量为$(63-14)/63\approx 77\%$。

(3) 当$w=4$时

为了讨论方面，我们假设k和l均为 512 位，此时划分的窗口个数为 512/4=128 个。

(i) 对于 Shamir 快速算法，当$w=4$时，需要保存的点的个数是$2^{2w}-1$，为 255 个。根据式(3-2)，预计算运算量约为

$$[(3\cdot 2^{2(w-1)}-2^{w-1}-1)A+(2^{2(w-1)}-2^{w-1})D] \tag{3-9}$$

主计算运算量为

$$[(\frac{2^{2w}-1}{2^{2w}}d-1)A+(d-1)wD] \tag{3-10}$$

代入$w=4$得到预计算量为

$$183A+56D=183(1I+1S+2M)+56(1I+2S+2M)=2148M$$

代入$w=4$得到主计算量为

$$127A+508D=127(1I+1S+2M)+508(1I+2S+2M)=5994.4M$$

需要总的计算量为$2148M+5994.4M=8142.4M$。

(ii) 对于改进双标量乘算法，通过归纳得到，改进双标量乘算法需要存储的点的个数为

$$2\cdot 2^{w}-2 \tag{3-11}$$

在预计算$2\cdot 2^{w}-2$个点时，P，Q不需预计算，$2P$和$2Q$共需两次二倍点乘，其余各点均只需一次点加即可得到，因此需要的预计算量为

$$2D+(2\cdot 2^{w}-6)A \tag{3-12}$$

当窗口宽度为w时，交叉项$iP+jQ$共有$2^{w}\cdot 2^{w}$种组合。其中形如$iP+0Q$和$0P+jQ$的组合共有$2\cdot(2^{w}-1)$种。这两种组合计算$R\leftarrow R+(K^{i}P+L^{i}Q)$时，需要一次点加运算。形如$iP+jQ$（且$i,j$均不为0）的组合有$(2^{w}-1)(2^{w}-1)$种，这种组合计算$R\leftarrow R+(K^{i}P+L^{i}Q)$时，需要两次点加运算。假设每种组合都是等概率出现，则当窗口个数为d时，每种组合平均出现$d\cdot 1/(2^{w}\cdot 2^{w})$次。通过以上分析，可知主计算中计算$R\leftarrow R+(K^{i}P+L^{i}Q)$需要的计算量为

$$[1\cdot 2\cdot(2^{w}-1)+2\cdot(2^{w}-1)(2^{w}-1)]\cdot\frac{d}{2^{w}\cdot 2^{w}}A \tag{3-13}$$

而主计算总的计算开销为

$$[1\cdot 2\cdot(2^w-1)+2\cdot(2^w-1)(2^w-1)]\cdot\frac{d}{2^w\cdot 2^w}A+(d-1)wD \tag{3-14}$$

因此根据式(3-11)，当$w=4$，改进双标量乘算法需要存储点为 30 个。

根据式(3-12)，可知需要预计算量为

$2D+26A=2(1I+2S+2M)+26(1I+S+2M)=248M$ 。

根据式(3-14)，可知需要主计算量为

$240A+508D=240(1I+1S+2M)+508(1I+2S+2M)=6988.8M$ 。

总的计算量为 $248M+6988.8M=7236.8M$ 。而 Shamir 快速算法总的计算量为 $8142.4M$ 。

因此当$w=4$时，相对于 Shamir 快速方法，改进双标量乘算法节省的计算量为 $(8142.4M-7236.8M)/8142.4M\approx 11\%$。节省的点存储量为 $(63-14)/63\approx 77\%$。

可以看到 Shamir 快速算法的运算量超过改进双标量乘算法的运算量。这是因为在 Shamir 快速算法中，随着窗口宽度增大，预计算量迅速增大。并且 k 和 l 位数不大，从而预计算表中存储点利用率不高所致。

下面将 w 分别等于 2、3、4 时，Shamir 快速算法和改进算法所需计算量和点存储量分别列表 3.5、表 3.6 如下：

表 3.5

计算量	Shamir 快速算法	改进标量乘	增加的计算量
$w=2$ (k、l约160bit)	2266.4M	2609.6M	14%
$w=3$ (k、l约192bit)	2853.6M	2907.2M	1.8%
$w=4$ (k、l约512bit)	8142.4M	7236.8	-11%

表 3.6

存储点数	Shamir 快速算法	改进标量乘	减少存储量
$w=2$ (k、l约160bit)	15	6	60%
$w=3$ (k、l约192bit)	63	14	77%
$w=4$ (k、l约512bit)	30	255	88%

由以上对比可得，如果窗口宽度 w 比较大，且 k 和 l 的位数较少，此时用 Shamir 快速算法，点存储量和预计算量都较大，且存储点利用率不高。此时如果改用改进标量乘算法，不仅点乘运算量没有明显增加，而且可以大幅减少点的存储量。其所需点存储量为 $2\cdot 2^w-2$，大约占 Shamir 快速算法的比例为 $(2\cdot 2^w-2)/(2^{2w}-1)$，约为 $1/2^{w-1}$，可见 $w\geq 2$，至少减少点存储 50%以上，因此改进算法可以作为一种

用于内存受限的手持设备的解决方案。

3.3.2　新算法的进一步优化

近些年出现了一些底层域上的快速算法，这些快速算法通过多做乘运算来减少逆运算的个数，其运算形式和运算量如下表 3.7 所示：

表 3.7

算法	运算量
$P \pm Q$ [56]	$1I + 1S + 2M$
$2P \pm Q$ [56]	$1I + 2S + 9M$
$3P$ [56]	$1I + 4S + 7M$
$2^k P$ [52]	$1I + (4k+1)S + (4k+1)M$
$3^k P$ [63]	$1I + (5k+1)S + 12kM$

由上表可知，当 I/M 较大时，在改进算法的点乘步 $R \leftarrow 2^w R$ 可结合上述的快速算法 $2^k P$，即计算倍点 $R \leftarrow 2^w R$ 时，不采用 $(d-1)\cdot w$ 次二倍点运算，而直接采用 $(d-1)$ 次 2^w 倍点运算。

例如，假设窗口宽度 $w=2$。由上表可知，当 $I/M=9$ 时，一次四倍点运算开销等于两次二倍点的运算开销；当 $I/M>9$ 时，一次四倍点运算开销则小于两次二倍点运算开销。因此当 $I/M>9$ 时，倍点运算 $R \leftarrow 2^w R$ 处采用四倍点计算方式，可以进一步降低运算开销。同样当窗口宽度 $w=3$ 时，当 $I/M>6$ 时, 一次八倍点的运算开销低于三次二倍点运算开销。因此当 $I/M>6$ 时，在倍点运算 $R \leftarrow 2^w R$ 处采用直接八倍点计算方式，可以进一步降低运算开销。并且 I/M 越大，则该方法节省运算开销越多。

3.4　本章小结

本章介绍了椭圆曲线单标量乘和双标量乘的经典方法，对经典的 Shamir 快速双标量乘算法以及交错 NAF 方法的优缺点进行了分析，在 Shamir 快速算法的基础上提出一种改进算法，通过定量分析，发现该算法在没有明显增加运算量的前提下，可大幅降低点存储量，可以作为一种内存受限的手持设备的解决方案。在本章末尾，将近些年出现的一些底层域快速算法和改进算法相结合，进一步对改进算法进行了优化。

第四章　多基编码标量乘算法研究及改进

4.1　多基链标量乘基本概念及研究现状

4.1.1　多基链的基本概念

设 $\{b_i\}$、$\{t_i\}$、$\{q_i\}$ 为 3 个单调递减序列，$s_i = \pm 1$，则整数 $n = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i} 5^{q_i}$ 称为整数 n 的多基链，m 称为多基链的长度，基集 $B = \{2,3,5\}$。当 $B = \{2,3\}$ 时，相应的表示称为双基链。双基链是高冗余的，而且表示长度非常短。如仅考虑 $s_i = 1$ 情况下，100 共有 402 个双基链表示，而它的多基链表示就有 8425 个。当 $B = \{2,3,5,7\}$ 时，100 有 43777 种对应表示。b_i、t_i、q_i 的大小影响标量乘中二倍点、三倍点和五倍点运算的运算次数，而 $m-1$ 为标量乘中点加的次数。对于一个 160bit 的大整数，如果使用双基链表示需要大约 23 项，而如果使用多基链则大约需要 15 项，因此使用多基链能够大大提高椭圆曲线标量乘法的运算效率。

4.1.2　多基链标量乘的研究现状

标量乘法是椭圆曲线密码体制中最基本最耗时的运算，其运算过程分为两个层次：一是上层运算，主要是椭圆曲线上点之间的运算：另一个层次称为底层域运算，主要是在有限域中为实现椭圆曲线上点之间的运算而做的一些求逆、乘法、平方等的操作，本章中分别记作 I、M、S。

研究标量乘法有两条思路：一是研究标量 k 的有效表示，尽量减少上层运算，如 NAF、$DBNS$；二是底层域快速算法研究，其根本目标是减少底层域运算量。目前一种方法是适当增加底层域乘法与平方运算来减少求逆运算，如在域中直接计 $3P$、$4P$、$4P \pm Q$、$2^k P$、$2P \pm Q$、$3P \pm Q$。标量 k 有效表示与底层域快速算法相融合，实现“多元优化”是目前研究标量乘法快速实现的一个趋势。2005 年 Dimitrov 等人将底层域快速算法 $3P$、$4P$、$4P \pm Q$、$2^k P$、$2P \pm Q$、$3P \pm Q$ 与双基数相融合，有效地减少了底层域运算量，从而加快了标量乘法的运行速度。

在 k 的表示方面，最典型的是非邻接格 $NAF(Non-Adjacent-Form)$ 以及窗口 NAF 方法，而作为一种选择，将滑动窗口技术用于 k 的 NAF，产生了计算标量乘的滑动窗口方法。NAF 表示已经相对成熟，目前 NAF 的研究主要集中在与底层快速算法的结合和用来计算多标量乘上。另外一种是双基数数字系 DBNS ($Double-base\ \ Number\ \ System$)，Dimitrov 等人一向致力于 k 的双基数表示，此

系统最大的优点是能在$O(\log k / \log\log k)$的时间复杂度内表示一个大整数k，应用到标量乘法上就体现为大大降低了点加运算的次数。

针对底层运算主要是寻求各种底层域快速算法，底层快速算法研究是近些年来的研究热点，如$3P, mP+Q$等。1997 年 Guajardo 和 Para 利用引入多个中间变量，多做乘法运算以减少求逆运算的方法，提出了F_{2^m}域上直接计算$4P$、$8P$、$16P$的算法，之后，这一“有偿代换”的思想再现在 1999 年 Han 和 Tan[34]给出的F_{2^m}域上$3P$、$5P$、$6P$和$7P$的直接计算和 Lopez、Dahab[35]的改进上。2005 年在双基数表示基础上提出了双基数链(Double-base chain)的概念，将$mP \pm Q(m=2,3,4)$的快速算法与双基数系统结合起来，体现了“多元优化″的思想。而 Ciet 等人分析了双基数链在超椭圆曲线上的应用。给出了一种基于双基数链的大整数k的分解方式。

无论是上层运算还是底层运算，一个不可忽略的因素是坐标的选取。常用的表示椭圆曲线点的坐标有：仿射坐标、射影坐标、Jacobin 坐标。不同的曲线，不同的运算在不同的坐标系中运行的时间也不相同，所以采用混合坐标、寻求各种坐标的最优组合是加快标量乘法运算的一种手段。

考虑到椭圆曲线倍点运算比点加运算耗时的特点，Ciet 等人给出了计算$2P+Q$的有效算法，即先计算$P+Q$，再计算$(P+Q)+P$，而不是先计算$2P$，再计算$2P+Q$，其中P、Q都是椭圆曲线上的有理点，该方法能够减少一个域乘。2005 年 Ciet、Joye 和 Lauter[56]将牺牲乘法运算以换取求逆运算这一思想明朗化，巧妙地设计出了$2P+Q$、$3P$、$3P+Q$、$4P$、$4P+Q$的快速计算算法。同时给出了一种基于双基链的分解k的方法。

Ciet 等人给出的形如$mP \pm Q$的底层域运算的运算开销如下表所示[56]：

表 4.1 底层域运算的运算开销

Operation	GF(p) cost	Binary field cost
$P+Q$	$1I+1S+2M$	$1I+1S+2M$
$2P$	$1I+2S+2M$	$1I+1S+2M$
$2P+Q$	$1I+2S+9M$	$1I+2S+9M$
$3P$	$1I+4S+7M$	$1I+4S+7M$
$3P+Q$	$2I+4S+9M$	$2I+3S+9M$
$4P$	$1I+9S+9M$	$1I+5S+8M$
$4P+Q$	$2I+4S+11M$	
$5P$	$2I+4S+11M$	

其中假设$I=6M, S=0.8M$，运算$2P$用D表示，$2P \pm Q$用DA表示，$3P$用T表示，$3P \pm Q$用TA表示，$4P$用Q表示，$4P \pm Q$用QA表示，$5P$用F表示。

使用二进制和三进制的混合表示可以与$3P$、$4P$、$4P \pm Q$、$2^k P$、$2P \pm Q$、$3P \pm Q$的快速计算相结合，减少求和项的个数，因此比仅使用二进制的表示方法节省开销。同时由于双基链的高度冗余性，k可以有多种不同的双基链表示形式，这使得多次计算k，基本运算的运算顺序可以完全不同，因此天然能够抵抗某些边信道攻击。

本章将 Ciet 等人给出的基于双基标量乘算法进行扩展，得到了基于多基(2,3,5)的标量乘算法，与目前已有多基标量乘算法相比，每次点乘，新算法所需要的逆运算至少减少两个。当I/M较大时，该算法能够明显地减少点乘运算开销。

4.2　典型双基多基标量乘算法

在下面的讨论中，我们假定$I/M=6$。

给出n的非相邻表示算法 4.1 如下：

算法 4.1　n的非相邻表示型(NAF)

设n是一个大于零的整数，n的非相邻表示形式(NAF)为：

$n = 2^{e_k} \pm 2^{e_{k-1}} \pm \cdots \pm 2^{e_2} \pm 2^{e_1}$

其中$0 \le e_1 < e_2 < \cdots < e_k$，且没有两个$e_i$是相邻的。

n的表示算法如下：

①　n除以 2；

②　如果 2 不能整除n，再让$n-1$或者$n+1$除以 2，并表示成 2 的多少次方乘以一个数的形式，选择 2 的次数较高的n的表示。

例 1. 设n=314159, 则 314159 的非相邻形式(NAF)为

$$314159 = 2^{18} + 2^{16} - 2^{14} + 2^{12} - 2^{10} - 2^8 + 2^6 - 2^4 - 2^0$$

可以使用二进制的方法(NAF)来计算$314159P$，计算过程如表 4.1 所示

表 4.1

公式运算	所有运算
314159=2^4*19635-1	3D，DA
19635=2^2*4909-1	D，DA
4909=2^2*1227+1	D,DA
1227=2^2*307-1	D,DA
307=2^2*37-1	D,DA
77=2^2*19+1	D,DA
19=2^2*5-1	D,DA
5=2^2+1	D,DA

计算$314159P$总的运算量为

$$10D+8DA=10(1I+2S+2M)+8(1I+2S+9M)=228.8M$$

例 2．设$n=1069493$,则 1069493 的非相邻形式(NAF)为

$$1069493=2^{20}+2^{14}+2^{12}+2^{9}-2^{6}-2^{4}+2^{2}+2^{0}$$

可以使用二进制的方法(NAF)来计算$1069493P$，计算过程如表 4.2 所示

表 4.2

公式运算	所有运算
1069493=2^2*267373+1	D，DA
267373=2^2*66843+1	D，DA
66843=2^2*16711-1	D,DA
16711=2^3*2089-1	2D,DA
2089=2^3*261+1	2D,DA
261=2^2*65+1	D,DA
65=2^6+1	5D，DA

计算$1069493P$总的运算为

$$13D+7DA=13(1I+2S+2M)+7(1I+2S+9M)=241M$$

算法 4.2 n的二进制和三进制的混合表示[56]

设n是一个大于零的整数，n的二进制和三进制的混合表示为：

$n=\sum_{i=1}^{m}S_i\cdot 2^{b_i}\cdot 3^{t_i}$，其中$S_i\in\{1,-1\}$，且$b_1\geq b_2\geq\cdots b_m\geq 0$，$t_1\geq t_2\geq\cdots t_m\geq 0$。

n的表示算法如下

①先让n除以 2；

②如果 2 不能整除n, 再让n除以 3；

③如果 3 不能整除n，再让$n-1$或$n+1$除以 3，并表示成 3 的多少次方乘以一个数的形式，选择 3 的次数较高的n的表示。

例 1．我们可以使用二进制和三进制的方法计算$314159P$，计算过程如下表

表 4.3

公式运算	所有运算
314159=6*52360-1	T，DA
52360=8*6545	3D
6545=6*1091-1	T,DA
91=18*5+1	T,T,DA
5=6-1	T,DA

计算$314159P$总的计算量为

$$
\begin{aligned}
&6T+4D+5DA\\
&=6(1I+4S+7M)+4(1I+2S+2M)+5(1I+2S+9M)\\
&=218.6M
\end{aligned}
$$

例 2. 我们可以使用二进制和三进制的方法计算1069493P，计算过程如下表4.4 所示

表 4.4

公式运算	所有运算
1069493=6*178249-1	DA，T
178249=6*29708+1	DA,T
29708=2^2*7427	2D
7427=6*1238-1	DA,T
1238=2*619	D
619=6*103+1	DA,T
103=6*17+1	DA,T
17=6*3-1	DA,T
3=3	T

计算1069493P总的计算量为

$$
\begin{aligned}
&7T+3D+6DA\\
&=7(1I+4S+7M)+3(1I+2S+2M)+6(1I+2S+9M)\\
&=241.8M
\end{aligned}
$$

算法 4.3 二进制、三进制和五进制的混合表示[33]

设n是一个大于零的整数，n的改进的二进制和三进制和五进制的混合表示为：

$$n=\sum_{i=1}^{m} S_i 2^{a_i} 3^{b_i} 5^{c_i}$$

其中$S_i \in \{1,-1\}$，且$a_1 \geq a_2 \geq \cdots a_m \geq 0$，$b_1 \geq b_2 \geq \cdots b_m \geq 0$，$c_1 \geq c_2 \geq \cdots c_m \geq 0$。

n的表示算法

①先让n除以2；

②如果2不能整除n，再让n除以3；

③如果3不能整除n，再让n除以5；

④如果5不能整除n，再让$n-1$或$n+1$除以2，并表示成2的多少次方乘以一个数的形式，选择2的次数较高的n的表示。

例 1. 我们可以使用二进制、三进制和五进制的方法计算314159P，计算过程

如下表 4.5 所示

表 4.5

公式运算	所有运算
314159=2^4*19635-1	3D，DA
19635=3*6545	T
6545=5*1309	F
1309=2^2*327+1	D,DA
327=3*109	T
109=2^2*27+1	D,DA
27=3^3	3T

计算314159P总的计算量为

$$5D+5T+3DA+F$$
$$=5(1I+2S+2M)+5(1I+4S+7M)+3(1I+2S+9M)+(2I+4S+11M)$$
$$=205M$$

例2. 我们可以使用二进制、三进制和五进制的方法计算106493P，计算过程如下表4.6所示

表 4.6

公式运算	所有运算
1069493=2^2*267373+1	D，DA
267373=2^2*66843+1	D,DA
66843=3^2*7427	2T
7427=2^2*1857-1	D,DA
1857=3*619	T
619=2^2*155-1	D,DA
155=5*31	F
31=2^5-1	4D,DA

计算106493P总的计算量为

$$8D+3T+5DA+F$$
$$=8(1I+2S+2M)+3(1I+4S+7M)+5(1I+2S+9M)+(2I+4S+11M)$$
$$=234.6M$$

4.3　一种改进的多基标量乘算法

算法 4.4 改进的二进制、三进制和五进制的混合表示

设 n 是一个大于零的整数， n 的改进的二进制和三进制和五进制的混合表示为： $n=\sum_{i=1}^{m} S_i 2^{a_i} 3^{b_i} 5^{c_i}$

其中 $S_i \in \{1,-1\}$ ， 且 $a_1 \geq a_2 \geq \cdots a_m \geq 0$, $b_1 \geq b_2 \geq \cdots b_m \geq 0$, $c_1 \geq c_2 \geq \cdots c_m \geq 0$ 。

n 的表示算法如下：

① If $n=1$ then return P；

② If(n%2!=0&&n%3!=0&&n%5==0){n=n/5} else goto

③ Switch(n mod6)

cases 0mod6,3mod6:　return 3((n/3)P)

cases 2mod6,4mod6:　return 2((n/2)P)

case 1mod6,n=6m+1：　return 2((3m)P)+P

case 5mod6,n=6m-1:　return 2((3m)P)-P

例 1. 我们可以使用改进的二进制、三进制和五进制的方法计算314159P，计算过程如下表 4.7 所示

表 4.7

公式运算	所有运算
314159=6*52360-1	DA，T
52360=2^3*6545	3D
6545=5*1309	F
1309=6*218+1	DA,T
218=2*109	D
109=6*18+1	DA,T
18=2*9	D
9=3^2	2T

计算314159P 总的其运算量为

$$5D+5T+3DA+F$$
$$=5(1I+2S+2M)+5(1I+4S+7M)+3(1I+2S+9M)+(2I+4S+11M)$$
$$=205M$$

例2. 我们可以使用改进的二进制、三进制和五进制的方法计算106493P，计算过程如下表4.8所

表 4.8

公式运算	所有运算
1069493=6*178249-1	DA，T
178249=6*29708+1	DA,T
29708=2^2*7427	2D
7427=6*1238-1	DA,T
1238=2*619	D
619=6*103+1	DA,T
103=6*17+1	DA,T
17=6*3-1	DA,T
3=3	T

计算106493P总的其运算量为

$$3D+7T+6DA=3(1I+2S+2M)+7(1I+4S+7M)+6(1I+2S+9M)=241.8M$$

通过对以上三种算法计算过程的比较，发现算法 4.2 和算法 4.4 在计算过程中所需逆运算的个数更少。如果逆运算开销较大，即 I/M 较大时，则改进算法 4.4 应该会比原有的算法 4.3 效率更高。

为了比较此三种算法在点乘中用到的逆运算个数，随即地选取了以下一些数字，并在程序下运行，可得所需运算开销如下表 4.9 所示：

表 4.9

	算法4.2	算法4.3	算法4.4
314159	(15I,42S,95M)	(15I,40S,83M)	(15I,40S,83M)
1069493	(16I,46S,109M)	(18I,42S,93M)	(16I,46S,109M)
42315991	(20I,58S,134M)	(23I,50S,119M)	(20I,56S,129M)
56314991	(21I,58S,131M)	(23I,54S,115M)	(21I,56S,126M)
147113572	(22I,62S,138M)	(24I,58S,122M)	(22I,60S,133M)
243113574	(22I,64S,136M）	(25I,58S,119M)	(22I,62S,145M)
1211472472	(25I,68S,151M	(27I,60S,139M)	(25I,64S,141M)
2111382462	(24I,72S,150M)	(28I,64S,139M)	(24I,70S,152M)
1725472372	(24I,70S,159M)	(27I,64S,135M)	(24I,70S,159M)

通过上表对比可以发现，在点乘中算法4.2和算法4.4所需求逆个数基本相同，而算法4.4比原有算法4.3节省至少两个求逆运算。通常情况下，在域计算中求逆运算的的开销比较大。因此，可以推知，当 I/M 较大时，算法4.4会比算法4.2有更

好的效率。

下面分别取 I/M 为12、30、60和80时，对比各算法运算开销。

$I/M=12$ 时各算法点乘开销如下表4.10所示：

表 4.10

	算法4.2	算法4.3	算法4.4
314159	308.6M	295M	295M
1069493	337.8M	342.6M	337.8M
42315991	420.4M	435 M	413.8M
56314991	429.4	434.2M	422.8M
147113572	451.6M	456.4M	445M
1211472472	505.4M	511M	492.2M
2111382462	495.6M	526.2M	496M

$I/M=60$ 时各算法点乘开销如下表4.11所示：

表 4.11

	算法4.2	算法4.3	算法4.4
314159	1028.6M	1015M	1015M
1069493	1105.8	1206.6M	1105.8M
42315991	1380.4	1539M	1373.8M
56314991	1437.4	1538.2M	1430.8M
147113572	1507.6M	1608.4M	1501M
1211472472	1705.4M	1807M	1692.2M
2111382462	1647.6M	1870.2M	1648M

I/M=80时各算法点乘开销如下表4.12所示：

表 4.12

	算法4.2	算法4.3	算法4.4
314159	1328.6M	1315M	1315M
1069493	1425.8M	1566.6M	1425.8M
42315991	1780.4M	1999M	1773.8M
56314991	1857.4M	1998.2M	1850.8M
147113572	1947.6M	2088.4M	1941M
1211472472	2205.4M	2347M	2182.2M
2111382462	2127.6M	2430.2M	2128M

对比以上表中数据并对三种算法7次点乘开销做统计平均，并列表4.13如下：

表4.13

	算法4.2	算法4.3	算法4.4
$I/M=12$	421.1M	428.6M	416.4M
$I/M=60$	1402M	1512M	1395M
$I/M=80$	1810.4M	1963.5M	1802.2M

当$I/M=12$时，算法 4.4 比算法 4.2 运算开销略低，但基本相当，而算法 4.4 比算法 4.3 节省运算开销约 3%左右。

当$I/M=60$时，算法 4.4 比算法 4.2 运算开销略低，但基本相当，而算法 4.4 比算法 4.3 节省运算开销约 7.6%左右。

当$I/M=80$时，算法 4.4 比算法 4.2 运算开销略低，但基本相当，而算法 4.4 比算法 4.3 节省运算开销约 8.2%左右。

而算法 4.4 因为采用了多基编码，安全性比算法 4.2 要好。因此算法 4.4 是一个安全性和计算效率都比较高的一个多基标量乘快速算法。

4.4 本章小结

本章第一节介绍了双基以及多基标量乘算法的产生、发展过程以及研究现状，对底层域快速算法如$3P$、$4P$、$4P\pm Q$、$2^k P$、$2P\pm Q$、$3P\pm Q$的研究现状做了介绍。在第二节分别给出了两种典型的基于双基和多基的大数分解方式。在第三节给出了一种新的基于多基的标量乘算法，并且与现有多基标量乘算法进行对比，发现新算法能够尽可能地减少点乘中逆运算个数，从而达到降低点乘开销的目的。

椭圆曲线标量乘是椭圆曲线密码体制中最基本同时也是最耗时的运算，多基链标量乘是双基链标量乘的一个推广，它的特点是标量表示长度短，非零比特数目更少，是一种特别适合椭圆曲线标量乘的快速算法。该文给出的新算法，以 2、3 和 5 作为基，由于多基数表示方法具有高度冗余性，该算法能够抵抗某些边信道攻击，与常用的标准倍点乘和非邻接形标量乘算法相比，该算法具有更小的运算量。

第五章　总结与展望

标量乘法包括单标量乘法和双标量乘法是椭圆曲线中最关键的运算，标量乘法的运算速度直接关系到椭圆曲线密码体制能否更进一步的推广。在一些签名方案中，需要计算 $kP+lQ$，在一些多重签名方案中，也可能会碰到求更多标量乘的情况，即求 $k_1P_1+k_2P_2+\cdots k_lP_l$ 的情况，于是研究多标量乘法算法显得极为重要。

本文首先介绍了椭圆曲线密码体制涉及到的理论基础知识，包括有限群、有限域和椭圆曲线的基本概念，对经典的单标量乘和双标量乘算法做了介绍，详细分析了 Shamir 快速算法和交错 NAF 方法的优缺点，针对 Shamir 快速算法需要预计算量大，存储点多的特点，在 Shamir 快速算法的基础上提出一种改进方案。经过分析发现该算法在没有明显增加点乘运算量的前提下，大幅减少点的存储，从而降低对内存的要求，可以作为存储容量受限的移动设备的解决方案。

随后，本文对近些年标量乘的一个新的研究热点--多基编码标量乘进行了研究。椭圆曲线标量乘是椭圆曲线密码体制中最耗时的运算，多基链作为双基链的一个推广，具有标量表示长度更短、非零比特数目更少的特点，非常适宜用于椭圆曲线标量乘的快速计算。研究表明，一个 160bit 的大整数如果使用双基链表示需要大约 23 项，而如果使用多基链则需要约 15 项就可以了，因此与使用双基链计算标量乘相比，使用多基链能够大大提高椭圆曲线标量乘法的计算效率。同时由于双基链和多基链的高度冗余性，k 可以有多种不同的双基或多基表示形式，这使得多次计算 k，基本运算的运算顺序可以完全不同，因此双基和多基编码标量乘天然能够抵抗某些边信道攻击。因此无论从提高运算速度或是提高安全性的角度来考虑，双基或多基编码的标量乘算法都不失为一个很有价值的研究方向。事实上，从上世纪 90 年代末开始，已经有大量学者投入到这方面的研究，并且取得了不少的成果。

多基编码标量乘经常与底层域上快速算法 $kP+Q$ 相结合。在研究 $kP+Q$ 快速算法的过程中，针对椭圆曲线上求逆运算的时间开销比较大的特点，一些学者尝试利用“有偿代换”的思想，对各种底层域快速算法展开研究，通过多做乘法运算，来减少求逆运算，从而降低计算开销。这不失为一种可取的做法。例如 2005 年 Ciet、Joye 和 Lauter[56]就将牺牲乘法运算以换取求逆运算这一思想明朗化，巧妙地设计出了 $2P+Q$、$3P$、$3P+Q$、$4P$、$4P+Q$ 的快速计算方法。同样地，我们也可以将减少求逆运算的思想运用到上层运算，例如通过对 k 更有效地编码，从而减少点乘运算中求逆的个数，从而达到减少计算开销的目的。基于这种思想，本文在分析了 Ciet 等人的双基编码标量乘基础上，对其拓展和改进，给出一种多基编码方

法，相对原有多基编码方式，每次点乘能减少两次求逆运算，提高了点乘效率。

经过本文的研究，作者感到以下方面仍可作进一步的研究：

(1)Shamir 快速双标量乘算法是一种巧妙的多点乘方法，能明显地提高点乘速度，但是随着 w 的增加，预计算量和点存储量迅速增加，因此如果要在存储受限的手持设备中应用该算法，应该努力设计更好的算法来降低对内存的要求。

(2)通过改进标量乘中 k 的编码方式来提高标量乘运算的速度一直是学者热衷的一个研究方向。1990 年 Morain 和 Olivos[40]提出了用 NAF 取代二进制，NAF 已成为计算 kP 最经典的方法，同时也把对 k 的有效表示的研究推向高潮。1996 年 Dimitrov 等开始提出 DBNS[47-49] (Double-Base Number System)后。DBNS 是基于 Tijdeman[50]提出的小素数组成整数的最佳间隔思想，它把 k 表示成 2^x3^y 的形式，结合了 $3^y P$，这种算法比传统的 NAF 算法有了较大的进步。之后多基编码与底层域上快速算法的有效结合成为了学者研究的热点，并且到目前为止取得了不少成果。并且由于多基数表示的高度冗余性，多基编码标量乘能够抵抗某些边信道攻击。因此多基和双基编码的标量乘与底层运算的有效结合是一个很有价值的研究方向。

(3) 相对于 RSA 等公钥加密算法的加解密速度，椭圆曲线加密算法虽然占一定优势，但是仍然比 DES,AES 等对称加密算法慢很多，所以在一定程度上限制了它的广泛应用。因此公钥加密算法与私钥加密算法相结合的混合加密方法是今后加密应用的一种趋势，也是 ECC 发展的一个方向。

椭圆曲线公钥密码系统由于其优良的单比特安全强度以及椭圆曲线理论的高深、精彩，使得它不论在理论研究还是在实际应用中都堪称一枝独秀。椭圆曲线上点的标量乘法是椭圆曲线密码系统的核心运算，其实现速度对该密码体制的推广有决定性的作用，因此我们期待有更多的学者加入到这方面的研究以推进椭圆曲线密码体制的推广！

本文由于作者水平有限，文中一定存在不少的不足之处，敬请各位老师和专家不吝指正！

致谢

从刚进实验室时的生疏，到现在硕士论文的完成，三年时光荏苒。我向所有曾给予我帮助和关怀的老师、同学和亲人致以深深的敬意和感谢！

深深感谢我的导师李学俊老师！从选题、论证到论文的撰写都是在李老师的精心指导下完成的，自始至终都倾注了李老师的辛劳、心血和希望。也正是她的远见卓识，热情鼓励和富有启发性的建议，对论文研究工作的顺利完成起了关键性的指导作用！

深深感谢李老师在我攻读硕士学位期间，对我学习上的悉心指导和对我生活上的亲切关怀和热情帮助。在学业上，李老师以她敏锐的洞察力帮助我确定了科学研究方向，以她精益求精的治学作风、不断攀登的治学态度、渊博的专业知识，在专业理论、科学思想和方法上给予我极大的指导、帮助和影响。在生活中，李老师以她高洁的人格魅力、宽厚待人的师者风范感染着我，潜移默化中教会了我许多做人做事的道理，为我树立了工作和生活中学习的楷模。对她的教诲我将铭记于心，谨此向她表示衷心的感谢和最深切的敬意！

由衷感谢李晖教授、张跃宇副教授和实验室的各位老师为我们的学习和科研工作提供了良好的条件，使得我在研究生期间学到了很多的专业知识，为我将来的工作、生活打下了良好的基础。

感谢我的父母。感谢你们对我的养育之恩以及多年来对我的谆谆教诲，感谢你们长期以来对我的信任与无限支持，你们是我事业和生活的源动力，正是由于有了你们，我才能有现在的成就和发展。

衷心感谢实验室的同学：魏鹏娟，金春花，莫灿，赵黎彬，来齐齐等，同时感谢朝夕相处的舍友刘达、张景瑞、张书亮，感谢你们对我学习和生活上的关心与帮助。和你们在一起的日子，使得我的研究生生活充满了快乐，受益良多。从你们那里我学到了很多书本中所没有的知识；这份深厚的友谊是我研究生生活中最最珍贵的东西。祝你们将来工作顺利，生活幸福美满。

再次向所有关心，帮助过我的老师、亲人、朋友和同学表示衷心的感谢！谢谢你们！有了你们，我才能顺利的走完研究生的求学之路，才会过的充实和快乐。

最后，衷心地感谢为评阅本论文而付出宝贵时间和辛勤劳动的专家和教授们！

参考文献

[1] W.Diffie,M.Hellman.New directions in cryptography[J].IEEE Transactions on Information Theory.Vol.22,No.6,1976:644-654

[2] R.L.Rivest,A.Shamir,L.M.Adleman.A method for obtaining digital signatures and public key cryptosystem[J]. Communications of the ACM,1987,21(2),120-126

[3] H.C.Williams.A modification of the RSA public-key encryption procedure[J]. IEEE. Transactions on Information Theory,vo1.26,1980:726-729

[4] K.Nybergk,R.Rueppelra.Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem.Advances in Cryptology-Crypto'92, Spinger-verla, 1994:182-193

[5] M.Rabin,Digitalized signatures and public-key functions as intractable as Factorization[J].MIT/LCS/TR-212,MIT Laboratory for Computer Science,1979

[6] T.EIGamal.A public key cryptosystem and a signature scheme based on discrete Logarithms[J].IEEE Transactions on Information Theory,vol.31,1985:469-472

[7] J.A.Muir,D.R.Stinson.New Minimal Weight Representations for Left-to-Right Window Methods.The Cryptographers'Track at the RSA Conference 2005,San Francisco,USA,2005.Berlin,Germany,Topics in Cryptology-CT-RSA 2005, Lecture Notes in ComputerScience 3376,Springer-Verlag,2005:366-383

[8] N.Koblitz.Elliptic Curve Cryptosystems[J].Mathematics of Computation,1987, 48:203-209

[9] V.Miller.Use of elliptic curves in cryptography[A].Advances in Cryptology-CRYPTO'85[C],UNCS,1986,218(483):417-426

[10] K.Nyberg and R.Rueppel.Message recovery for signature schemes based on the discrete logarithm problem[J].Designs,Codes and Cryptography,1996,7:61-81

[11] J.Crowie,B.Dodson,P.Montgomery etal. A world wide number field sieve factoring record:onto the 512 bits[A].Advances in Cryptology –ASIACRYPT '96 [C] ,1996,LNCS l163:382-394

[12] K.Okeya,K.S.Samoa,C.Spahn,T.Takagi.Signed Binary Representations Revisited.SantaBarbara.24th Annual International Cryptology Conference, California,USA,2004.Berlin,Germany,Advances in Cryptology-CRYPTO 2004, Spinger-Verlag,2004:123-139

[13] D.Gordon.Discrete logarithms in GF(p) using the number field sieve[J].SIAM Journal on Discrete Mathmatics,1993,6:124-138

[14] D.Coppersmith.Fast evaluation of logarithms in fields ofcharacteristic two[J]. IEEE. Transcations on Information Theory,1984,30:587-594

[15] E.Thome.Computation of discrete logarithms in F2607[A].Advances in Cryptology-ASCACRYPTO'2001[C],2001,LNCS 2248:107-124

[16] J.Pollard,M.Carlo.Methods for index computation(mod p)[J].Mathematics of Computatiohn,1978.32:918-924

[17] N.Koblitz. Ellitpic curve cryptosystems[J].Mathematics of Computation, 1987, 48:203-209

[18] Danniel Bailey,and Chrisfor Parr,"Inversion in optimal extension fields", Proceedings of the Conference on The Mathmetics of Public Key Cryptography,Toronto Canada June 12-17 1999

[19] R.Schoof. Elliptic Curves Over Finite Fields and the Computation of Square Roots mod[J].Mathematics of Computation,1985,44(170):483-494

[20] william JC,Edward PD, Scott AR.PKI Elliptic Curve Cryptography and Digital signatures.Compters and Security,1999,18:47-66

[21] Aol.Atkin.the number of points on an elliptic curve modulo a prime(ii)[J].Series of e-mail to the NMBRTHRY mailing list,1992

[22] N.D.Elkies.Elliptic and modular calves over finite fields and related computational issues[M].AMS/International Press,1998

[23] T.Satoh.The canonical lift of an ordinary elliptic curve over a finite field and its point counting[J].Journal of the Ramanujan Mathematicical Society,2000,15:483

[24] R.Harley,J.F.Mestre,P.Gaudry.Counting points with the arithmetic-geometric mean[J].Rump talk at EUROCRYPT,2001

[25] Higuchi,Akira,Takagi,Naofumi.A fast addition algorithm for elliptic curve arithmetic in GF(2n) using projective coordinates[J].Information Processing Letters,2000,76(3):101-103

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

[27] 王许书,王昭顺,曲英杰.基于复合域上的椭圆曲线密码体制的计算算法[J].小型微型计算机系统,2002,23(8):1007-1009

[28] 冯登国.国内外密码学研究现状及发展趋势[J].通信学报,2002,23(5):18-26

[29] A.Menezes,P.C.Van Oorschot,S.A.Vanstone.Handbook of applied cryptography [M].CRC Press,1997

[30] D.Hankerson,A.Menezes,S.Vanstone.Guide to Elliptic Curve Cryptography[M] Spdnger-Veflag,2004

[31] 白国强.椭圆曲线密码及其算法研究[J]:[博士学位论文].西安:西安电子科技大学,2000

[32] 陈晓峰,王育民.公钥密码体制研究与进展[J].通信学报,2004,25(8):109-118.

[33] WangYuan,Xin Xiao-long.Improvement of Scalar Multiplication Algorithm in Elliptic Curve.Coumputer Engineering.2008.9

[34] 王衍波.椭圆曲线上密码研究现状与展望[J].解放军理工大学学报（自然科学版）,2002,3(6):18-25

[35] Jean-Sebastien.Resistance Against Differential Power Analysis For Elliptic Curve Cryptosystems[A].Cryptographic Hardware and Embedded Systems-Ches'1 999,1,17 17.292-302

[36] Silverman.H.The Arithmetic of Elliptic Curves[C].New York: Springer-Verlag, 1986

[37] Balasub:amanian R,koblitz N.the improbability that an elliptic curve has subexponential diserate log problem under to Menezes-Okma otovanstane a logorithm[J].J of eryptograph y. 1998,(11):141-145

[38] E.D.Mastorvito.VLSI designs of multiplication over finite fields GF(2^m)[C].In LNCS-357,Proc.1988,(6):297-309

[39] Marc Joye,Christophe Tymen.Compact Encoding of Non-aajacent Forms with Applications to Elliptic Curve Cryptography[C].Public Key Cryptography. 2000, 1,1992.353-364

[40] F.Morain,J.Olivos.Speeding up the computations on an elliptic curve using addition-subtraction chains[J].Theoretical Informatics and Applications 1990, 24:531-544

[41] D.Knuth.The art of computer programming-seminumerical algorithms[M]. Addision-Wesley,3rd edition,1998

[42] H.Cohen.A Course in Computational Algebraic Number Theory[M]. Springer-Verlag,1993

[43] D.Gordon.A Survey of Fast Exponentiation Methods[J].Journal of Algorithms,1998,27 129-146

[44] A.Menezes,P.C.Van Oorschot,S.A.Vanstone.Handbook of applied cryptography [M].CRC Press,1997

[45] J.Solinas.Efficient arithmetic on Koblitz curves[J].Designs,Codes and Cryptography,2000,19:195-249

[46] V.Miller and N.Koblitz.Elliptic curve cryptosystems[J].Mathematics of Computation.1985.203-209

[47] V.S.Dimitrov,L.Imbert,PK.Mishra. Fast Elliptic Curve Point Multiplication using Double-Base Chains[R].Cryptology ePrint Archive,Report 2005/069,2005

[48] V.S.Dimitrov,G.A.Jullien,W.C.Miller.An algorithm for modular. Exponentitation [J].Information Processing Letters,1998,66(3):155-159

[49] V.S.Dimitrov,G.A.Jullien,W.C.Miller.A new number representation with applications[J].IEEE Circuits and Systems Magazine,2003,3(2):6-23

[50] R.Tijdeman.On the maximal distance between integers composed of small primes[J].Composition Mathematical and Computer Modelling,1974,28:159-162

[51] J.Guajardo,C.Paar.Efficent algorithms for elliptic curve cryptosystems[C]. CRYPTO'97,LNCS,Springer-Verlag,vol1294,1997:328-340

[52] Y.Sakai,K.Sakurai.Efficient scalar multiplications on elliptic curves with direct computerations of several doublings[C].IEICE Transactions Fundamentals E84-A,No.1(2001),2001:120-129

[53] K.Araki,T.Satoh,and S.Muria," Overview of elliptic curve cryptography",In Proceeding of Public Key Cryptography,LCNS 1431 Springer-Verlag 1999,pp 22-49

[54] Minimizing the use of random oracles in authenticated encryption schemes. Information and communications Security'97(LNCS 1334),1997,188:l-16

[55] K.Eisentrager, K.Lauter, P.L.Montgomery. Fast elliptic curve arithmetic and improved weil pairing evaluation, In M.Joye,editor,Topics in Cryptology- CT-RSA[A].Springer-Verlag,2003[C].2003:343-354

[56] M.Ciet,M.Joye,K.Lauter,P.L.Montgomery.Trading inversions for multiplications in elliptic curve cryptography[R]. Cryptology ePrint Archive, Report, 2003/ 257, 2003

[57] 张焕国,椭圆曲线密码学导论.北京:电子工业出版社.2005

[58] 裴定一,祝跃飞,算法数论,北京:科学出版社.2002

[59] 徐秋亮,李大兴.椭圆曲线密码体制[J].计算机研究与发展,1999,36(11) 1281-1288

[60] 王衍波.椭圆曲线上密码研究现状与展望[J].解放军理工大学学报,2002,1.3(6):18-25

[61] 郝林,罗平.椭圆曲线密码体制中点的数乘的一种快速算法[J].电子与信息学报,2003,25(2):275-278

[62] 丁勇.椭圆曲线密码体系中标量乘的快速算法研究[D]:[博士学位论文].西安:西安电子科技大学,2005

硕士期间论文发表情况及科研工作

一、参加科研情况

参与 115 基金项目《方便×××××的研究》.

二、发表论文情况

1.Wei Peng-juan,Li Xue-jun,Jin Chun-hua,Wang Li-chuan.An Improved Pairing-friendly Elliptic Curve Applying to Bilinear Pairings. Proc. of International conference on Information Security and Artifical-ISAI'2010, Chengdu, China, IEEE Press , Vol 1, pp. 506-510, 2010.

2.金春花,李学俊,魏鹏娟,王立川.新的无证书混合签密.计算机应用研究.2010.10.

3.Jin Chun-hua,Li Xue-jun,Wei Peng-juan,Wang Li-chuan.Identity-Based Hybrid Signcryption. Journal of Electronics,has been contributed,2010,9.

西安電子科技大學

地址：西安市太白南路2号
邮编：710071
网址：www.xidian.edu.cn