

电子科技大学

博士学位论文

有限域运算和椭圆曲线数乘运算研究

姓名：王庆先

申请学位级别：博士

专业：计算机应用技术

指导教师：孙世新

20060612

摘 要

本文主要研究有限域运算算法和椭圆曲线数乘运算算法。全文在强调作者所取得研究成果的同时给出了有限域运算理论和椭圆曲线密码体制理论的基本框架。全文共分八章，具体内容如下：

第一章介绍有限域运算，椭圆曲线密码体制及其数乘运算的研究背景、意义和现状，并指出论文的主要研究内容及作者得到的主要研究成果。

第二章给出有限域运算和椭圆曲线密码体制理论的基本知识和相关性质。

第三章研究素数域上取模运算。针对具有特殊形式的模数(如 Mersenne 数，伪 Mersenne 数，广义 Mersenne 数等)，深入分析了其取模运算规律，得到如下结果：对 Mersenne 数和伪 Mersenne 数，给出了取模运算转换为模加或模减运算的公式；对模数为任意首一三项式和五项式产生的广义 Mersenne 数，推导了相应取模运算的复杂度计算解析表达式。根据该表达式，对任意给定的不可约首一三项式和首一五项式，可以根据该多项式的系数分别得出以广义 Mersenne 数为模数的取模运算所需要的模加法(或模减法)的次数。

第四章研究有限扩域的乘法运算算法。利用广义 Mersenne 数代替伪 Mersenne 数，提出了广义最优扩域的概念，并研究其上的快速乘法运算和取模运算，为乘法运算给出了通用的复杂度公式，为取模运算给出了具体的运算公式，推广了 Bailey, Mihăilescu 和 Woodbury 等在最优扩域上的相应结果。

第五章研究有限扩域的求逆运算算法。深入分析和研究了有限域 $GF(q)$ 和二元扩域 $GF(2^m)$ 上的各种求逆运算算法。重点讨论了最优扩域，最优塔域上的求逆算法，并对各种求逆算法的性能进行了分析比较。

第六章研究串、并行正规基乘法器设计算法。基于正规基表示有限域 $GF(2^m)$ 上元素的方法，以增加异或门(XOR Gates, XG)个数达到减少与门(AND Gates, AG)个数的方式，提出了一个串行乘法器和一个并行乘法器。同时，得到一个 II 型最优正规基乘法器的算法设计，该乘法器要求 $(2m-2)$ 个 XG, m 个 AG。

第七章研究椭圆曲线数乘运算算法。相同椭圆曲线，采用不同的坐标表示，对应不同的点加公式，不同的点加公式具有不同的计算复杂度。本章分析了不同坐标下点加公式的计算复杂度，并以此为基础，首先分析和比较了数乘运算的点加及倍加方法(add and double methods)，加减方法和窗口方法等三种算法的计算复

杂性。接着比较了点加及倍加方法和 Montgomery 方法的计算开销,并指出,在仿射坐标下,当域元素求逆运算和乘法运算的比值大于等于 2 时,点加及倍加方法的计算开销少于 Montgomery 方法;但是,在投影坐标下,点加及倍加方法的计算开销却大于 Montgomery 方法。最后对同时计算多个数乘运算的几种算法进行了计算复杂性分析。

第八章全文总结和未来工作。对全文的工作进行了概要性总结,并指出未来的研究方向。

关键词: 椭圆曲线密码体制, 有限域运算, 正规基乘法器, 数乘运算, 算法复杂性

Abstract

The finite field arithmetic, elliptic curve scalar multiplication and the related algorithms are investigated in this dissertation. With the emphasis on the results obtained by the author, the basic scheme of the finite field theory and elliptic curve cryptosystems theory is sketched. It consists of eight chapters, which are enumerated as follows:

In chapter one, some introductive materials are presented, including the motivations and developments of the finite field arithmetic, Elliptic Curve Cryptosystems(ECC) and elliptic curve scalar multiplication, the intentions of the research work, the main contents of the paper and the list of the results obtained by the author.

In chapter two, some necessary and basic materials of the finite field arithmetic theory and elliptic curve cryptosystems theory are introduced.

In chapter three, the modular arithmetic algorithms over prime fields are studied. To the moduli with special forms (such as Mersenne numbers, pseudo-Mersenne numbers, generalized Mersenne numbers), we analysis and study modular arithmetic laws, and obtain the conclusions as follows: for Mersenne numbers and pseudo-Mersenne numbers, we get modular arithmetic formulas, and determine exact expressions of the number of modular addition to generalized Mersenne numbers generated by irreducible monic trinomials and pentanomials. Depending on the coefficient of the polynomial, we can easily compute the number of modular addition of $A \bmod p$ by the formulas for any given irreducible monic trinomial and monic pentanomial.

In chapter four, the multiplication algorithms over finite extension fields are investigated. By replacing pseudo-Mersenne numbers with generalized Mersenne numbers, we propose a new notion-Generalized Optimal Extension Fields(GOEFs), and study the fast arithmetic about multiplication and modular arithmetic in GOEFs. At last, we deduce common formulas for multiplication and some more general formulas for modular arithmetic in GOEFs. The results in this paper extend the corresponding work on arithmetic in Optimal Extension Fields(OEFs) made by Bailey, Mihăilescu and

Woodbury.

In chapter five, the inversion algorithms over finite extension fields are researched. We deep analysis many kinds of inverse arithmetic algorithms over $GF(p)$ and $GF(2^m)$, and compare their performance. Especially, we lay emphasis on the inverse arithmetic algorithms over OEFs and Optimal Tower Fields(OTFs).

In chapter six, the new type of serial and parallel multipliers are designed. Efficient hardware architecture for multiplication is extensively used to the smart cards. Based on the normal basis representation of the field elements, we put forward a new serial multiplier and a new parallel multiplier by increasing the numbers of XOR gates. In addition, we propose a new type II optimal normal serial multiplier algorithm, which needs the number of the XOR gates is $(2m-2)$ and AND gates m .

In chapter seven, the elliptic curve scalar multiplication algorithms are studied. Though the elliptic curve is same, different coordinate representations bring forth different addition formulas, and different addition formulas correspond to different complexity. In this chapter, we firstly analysis and compare the complexity of three kinds of scalar multiplication algorithms. Then, the cost of the “add and double methods” and Montgomery methods is compared in detail. We show that the Montgomery methods has more cost than the “add-and-double” methods if the ratio of the cost of field inversion and field multiplication is more than 2 under the affine coordinate representations; but the former has less cost than the latter under the projective coordinate representations. At last we discuss the complexities of multiple scalar multiplication algorithms.

In chapter eight, summarization and the future work. A general description covering the main points of the dissertation are presented and the future work are enumerated.

Keywords: Elliptic Curve Cryptosystems(ECC), finite field arithmetic, normal basis multiplier, scalar multiplication, algorithm complexity.

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，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作了明确的说明并表示谢意。

签名： 王旭 日期： 2006 年 6 月 7 日

关于论文使用授权的说明

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

签名： 王旭 导师签名： 孙世强
日期： 2006 年 6 月 8 日

第一章 引言

1.1 研究动机和意义

在公钥密码体制中,椭圆曲线密码系统(ECC)尤其适合计算能力和带宽受限的应用,其最突出的优点是高效率。与 RSA(Rivest-Shamir-Adleman public key cryptography)相比,ECC 不需要产生大素数和进行素性检测;能以更短的密钥长度提供同样的安全级别。事实上,ECC 是所有主流加密算法中能提供最高位加密强度(the highest cryptographic strength-per-bit)的算法。因此,ECC 被认为是最值得关注的加密算法,是目前信息安全领域研究的一个热点^[1-5]。

ECC 研究的主要内容是 ECC 的有效实现,其关键问题是数乘运算的快速计算。图 1-1 是现代密码学系统的一种自上而下的模型^[6]。

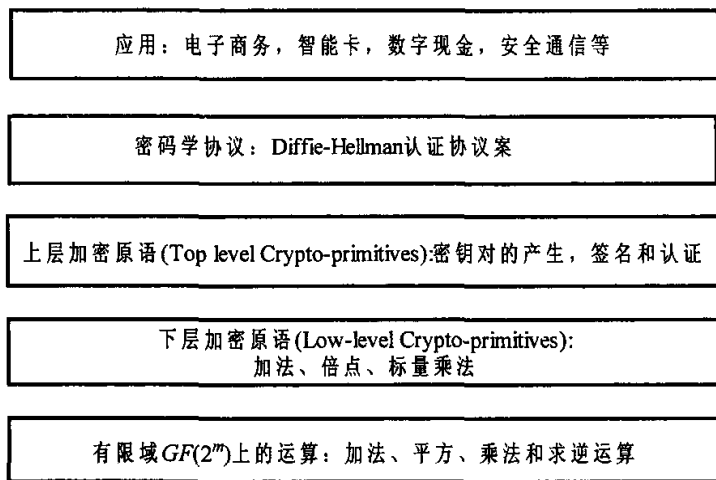


图 1-1 现代密码系统自上而下模型

从图 1-1 可以看出,基于 ECC 的具体应用依赖于数乘运算的快速实现,数乘运算又直接依赖于有限域加法、减法、乘法、求逆(除法)、取模等运算的高效算法。因此,要想使 ECC 具有能够接受的性能,设计高效的有限域运算算法是关键^[1-5]。

除 ECC 外,有限域上的运算(即加法、减法、乘法、取模和求逆运算)还被广泛用于其它公钥密码体制,如:著名的公钥密码体制 RSA 以及 ElGamal 算法。此外,编码理论,开关理论,超大规模集成电路 (Very Large Scale ntegration, VLSI)

测试以及各种数字签名算法,如 DSA(Digital Signature Algorithm)、GOST(Gosudarstvennyi Standard 或者 Government Standard)算法、Diffie-Hellman 算法等也大量使用有限域及其上的各种运算^[3-12]。因此有限域上各种运算的研究具有十分重要的理论意义和现实意义。

本文主要对 ECC 所依托的有限域上各种运算算法以及椭圆曲线数乘运算算法进行深入研究,内容主要涉及 ECC 的基域—素数域 $GF(p)$ 和二元扩域 $GF(2^m)$ 。论文首先介绍了有限域的基本理论和 ECC 的相关基础知识;随后对有限域上的取模运算算法、乘法运算算法、求逆运算算法进行深入研究,并得出相关结论;接着利用相应结果设计了新型的串、并行正规基乘法器;最后分析了不同坐标下点加运算的计算复杂度,并以此为基础,分析和比较了各种数乘算法的计算复杂性。

1.2 研究现状

由于论文所研究的是有限域运算算法以及 ECC 的数乘运算算法,因此论文分别从有限域运算、ECC 和 ECC 的数乘运算三个方面对相关工作的研究现状进行综述。

1.2.1 有限域运算算法的研究现状

在有限域的各种运算中,加法、取模和乘法运算是最基本的运算,指数运算,求逆运算均可转换为乘法运算完成。在这些运算中,开销最大的当属乘法运算、取模运算和求逆运算,因此,大量的文献都是针对这三种运算展开,具体可参见文献[3, 4, 12-27]等。

1.2.1.1 乘法运算算法

在有限扩域 $GF(p^m)$ 中,两个域元素相乘是两个多项式相乘(运算规则见论文第四章)。经典的两个 m 次多项式(或者两个 m 比特的整数)乘法是逐项相乘,然后合并次数相同的项。完成一次两个多项式相乘需要 m^2 次一位数乘法,即其时间复杂度是 $O(m^2)$ ^[6,28]。文献[29]给出了一个亚平方多项式乘法算法,被称为 Karatsuba-Ofman 算法。该算法的基本思想是通过增加加法次数达到减少乘法次数的目的,其时间复杂度为 $O(m^c)$,其中 $c = \log_2 3 \approx 1.585$ 。显然,该算法优于经典的多项式乘法。文献[23]给出了 Karatsuba 算法及其变体所需乘法次数的公式,我们的文章也对此做了详尽分析,并得到相应的计算复杂度公式。文献[28]以 Karatsuba

思想为基础,就两个五项式、六项式和七项式相乘得出两个多项式相乘所需乘法次数更少的结论,并采用递归的方式,就两个高次多项式的相乘得出了更优的乘法次数下限。

1.2.1.2 乘法器

由于有限域加法和乘法运算易于硬件实现,因此,设计时间和空间复杂性低的乘法器成为研究热点^[25],并已经包括在 IEEE 和 NIST 标准中,见文献[30, 31]。一般地,根据有限域元素不同的表示方法,乘法器可分为正规基乘法器、多项式基乘法器和对偶基乘法器;根据具体实现的方式,也可分为串行乘法器和并行乘法器^[32]。

正规基乘法器最大的优点是平方运算只须一个循环移位即可完成,从而得到更高效的乘法运算^[25]。在 $GF(2^m)$ 中,并行正规基乘法器的工作原理是:一旦乘法器接收到 $2m$ 个比特数据,经过各种逻辑门延迟和存取延迟后,同时输出乘积的 m 个比特数据;而串行正规基乘法器的工作原理则是首先将两个多项式的系数串行加载到移位存储器中,然后,在每一个时钟循环里,通过存储器的循环移位,再经过各种逻辑门延迟和存取延迟后,输出乘积的一个比特数据。

1986 年, Massey-Omura^[33]提出了第一个正规基乘法器,简称为 MO 乘法器。该乘法器需要 m^2 个 AG, $2m(m-1)$ 个 XG,其时间复杂度为 $T_A + (1 + \lceil \log_2(m-1) \rceil)T_X$, 其中, T_A , T_X 分别表示经过一个 AG 和 XG 的延迟。1993 年, Hasan 等人^[34]将不可约多项式限制为全一多项式 (All-One Polynomial, AOP, 系数全为 1 的多项式),提出一种新颖的乘法器结构,该乘法器大大降低了并行 MO 乘法器的复杂性。对由 AOP 产生的同类型有限域, Ç.K.Koç 和 B.Sunar^[35]将多项式基乘法器推广到正规基乘法器,提出一种新型并行正规基乘法器。同时, Mullin 等人^[36]给出了正规基乘法器的复杂度下限,并将能够达到该下限的正规基定义为最优正规基 (ONB)。他们还定义了两种类型的正规基,即 I 型最优正规基和 II 型最优正规基。

2001 年, B.Sunar 和 Ç.K.Koç^[37]提出一个 II 型最优正规基并行乘法器,该乘法器所需 XG 的个数比并行 MO 乘法器少 25%;2002 年, A.R.Masoleh 和 M.A. Hasan^[32]提出了一种简化的冗余 MO 并行乘法器,该乘法器可用于任意正规基和没有任何特殊限制的有限域,而且其复杂性低于并行 MO 乘法器。特别地,该乘法器的空间复杂度几乎是其它乘法器的一半。A.R.Masoleh 和 M.A.Hasan, B.Sunar 又对前面的工作做了进一步的发展,得到时间和空间复杂性都低于 MO 乘法器的几种正规基乘法器,具体可见文献[38-41]。

同时,国内许多作者对有限域乘法器的研究也做了大量工作。Fan 和 Dai^[42]在 $GF(2^m)$ 上重新定义了乘法器的输出函数(key function),并根据该定义,提出一种基于正规基表示的域元素乘法的快速软件实现算法。2004 年,鲁俊生等^[43]提出了一种有限复合域 $GF((2^{m_1})^{m_2})$ 上的快速乘法器,该乘法器采用串、并行计算相结合的原则,增加少量硬件规模将一次有限域乘法的计算速度由原来的 $m=m_2m_1$ 个时钟周期降低到 m_2 个时钟周期,极大提高了乘法器的计算速度。另外, Wu 等人也对正规基乘法器进行深入的研究,具体内容可参见文献[44]。

$GF(2^m)$ 上的并行多项式基乘法器最早由 Bartee 和 Schneider^[45]提出。根据不可约多项式,其实现需要 $GF(2)$ 上的 (m^3-m) 个二值输入的 XG^[46]。在文献[47, 48]中, Mastrovito 提出了一种多项式基乘法算法,并给出了其硬件体系结构。Sunar 和 Koç^[49]利用三项式,对 Mastrovito 算法给出了一个全新公式,并指出其对应的乘法器需要 (m^2-1) 个 XG 和 m^2 个 AG。在文献[50], Halbutogullari 和 Koç 推广了 Sunar 和 Koç 的思想,并找到对任意不可约多项式,构造 Mastrovito 乘法器的方法。迄今为止,对这些特定的多项式,就 XG 的数量和时间延迟来说, Halbutogullari 和 Koç 提出的算法具有最低的时间复杂度。在文献[51]中, Zhang 和 Parhi 提出一种设计 Mastrovito 乘法器的对称方法,而且,他们将这种方法应用于设计改进的 Mastrovito 乘法方案^[52],并对两类不可约五项式,提出了新的 Mastrovito 乘法器的复杂性结构。

不同于 Mastrovito 乘法器,通过首先直接相乘 $GF(2^m)$ 中的两个元素,随后再进行取模,许多文献如[53]等就采用的这种方式,作者 Wu 使用不可约三项式作为约简多项式,并指出 $GF(2^m)$ 上的一个取模运算需要 $(w-1)(m-1)$ 个加法,其中, w 是不可约多项式的汉明权重(多项式中非零系数的个数)。在硬件实现乘法运算时,需要 $((m^2-1)+(w-1)(m-1))$ 个 XG 和 m^2 个 AG。最近, Rodriguez, Henriquez 和 Koç^[54]对特殊的五项式提出了一个多项式基乘法器,并得出其时间延迟和所需要的门数。尽管作者都称其乘法器为 Mastrovito 乘法器,但是他们的体系结构与最初的 Mastrovito 乘法器是完全不同的,因为该乘法器通过单独使用两步乘法来完成乘法运算的。

由于本论文并不涉及对偶基乘法器,因此,此处略去对偶基乘法器的研究现状综述。

1.2.1.3 求逆运算算法

对大素数域和二元扩域,求逆技术都是基于欧氏算法,或者其变体如准求逆

算法(Almost Inverse Algorithm, *ALA*), 或者 Fermat 小定理^[2]。经典的二元扩域上的求逆算法有: 扩展欧氏求逆算法(Extended Euclidean Algorithm, *EEA*), 二进制欧氏求逆算法(Binary Euclidean Algorithm, *BEA*)和准求逆算法(*ALA*)。

Bailey 和 Paar^[46,47]对其提出的最优扩域, 利用 Frobenius 映射得到更有效的求逆算法, 具体可参见论文第五章。

1.2.2 椭圆曲线密码体制(ECC)的研究现状

与 RSA 等其它公钥密码体制相比, ECC 最吸引人之处在于: ECC 能够以较小的密钥尺寸提供和 RSA 相等的安全强度, 从而能有效的减少处理开销。同等安全强度下的密钥长度对照表见表 1-1。

表 1-1 同等安全强度下四种密码体制密钥长度的对比^[57]

ECC- p	ECC- 2^m	DH/DSA/RSA- n	Symmetric
192	163	1024	Skipjack 80
224	233	2048	3-DES 112
256	283	3072	AES-Small 118
384	409	0	AES-Medium 192
521	571	15360	AES-Large 256

从表 1-1 可以看出, 就安全性而言, 每一比特 ECC 的密钥安全强度至少相当于 5 比特的 DH/DSA/RSA- n 提供的安全强度, 并且这种比例关系随密钥长度的增加呈上升趋势。ECC 的这一特点, 使得它在未来计算能力逐渐提高的情况下具有更强的竞争力, 而且其短密钥优势特别能满足在带宽、计算能力或存储能力等受限的各种特殊应用场合。

从 1998 年起, 一些国际标准化组织开始了对椭圆曲线密码的标准化工作。其中, 1998 年底美国国家标准与技术研究所(American National Standards Institute, ANSI)公布了针对椭圆曲线密码的 ANSI-X9.62^[58]和 ANSI-X9.63 标准^[59]; 2000 年 2 月被 IEEE 确定为标准 IEEE1363-2000(包括了加密、签名、密钥交换协议方案)^[30], 同期, NIST 确定其为联邦数字签名标准(Federal Information Processing Standards)FIPS186-2^[31]。

此外, 电子商务协议 SET(Secure Electronic Transactions)的制定者已把它作为下一代 SET 协议中缺省的公钥密码算法, 异步传输模式 ATM(Asynchronous Transmission Mode)论坛技术委员会提出的 ATM 安全性规范中也支持 ECC^[60]; ISO/IEC 也确定 ECC 为数字签名标准 ISO14888-3^[60,61]。

在 ECC 的研究中, ECC 的构造、ECC 的分析和 ECC 的快速实现是 ECC 的三个主要研究方向^[62,63]。

1.2.3 数乘运算的研究现状

ECC 的有效实现是 ECC 研究的主要内容之一, 而 ECC 有效实现的关键问题之一则是数乘运算的快速计算, 其运行时间决定着 ECC 的实现时间。

针对一般群上的求幂运算, 已经存在许多快速算法, 这些方法均可以推广到 ECC 的数乘运算中^[2,3]。但是, 因为 ECC 的有理点群上的点加和点减运算开销相当, 所以还存在一些适宜于计算 ECC 数乘运算的独特方法, 例如各种带符号的二进制方法。

计算数乘运算的基本方法是“加—减”方法^[30]。对随机选取的乘数 k , 当使用仿射坐标时, 该算法平均需要执行 $8/3\log_2 k$ 个有限域乘法和 $4/3\log_2 k$ 个有限域求逆运算; 当使用投影坐标时, 需要 $25/3\log_2 k$ 个有限域乘法。

随后, 许多作者以二进制方法为基础, 又提出了许多新的方法, 例如: m -ary 方法, 标号窗口方法(signed window methods)等, 这些新方法都采用预计算^[64-69]和对乘数重新编码的方式完成^[112,113]。在文献[64-69]中, 提出了一些新的技术以加速 m -ary 方法, 标号窗口方法。这些技术有助于 $GF(2^m)$ 上随机产生椭圆曲线的软件实现。但是, 当存储空间有限时, 这些方法的优势就大大削弱了。

另外, 对于一些特殊的有限域和特殊曲线, 研究者们也设计了大量的有效方法。例如, 对反常椭圆曲线(Koblitz 曲线), 已知计算数乘运算最快速的算法见文献[66]; 对小特征域上的椭圆曲线, 有效的算法可参见文献[67-69]。

1987 年, Montgomery^[70]指出: 当待相加的两个点 P_1 和 P_2 总是满足 $P_2 - P_1 = P$ (已知的固定点), 计算 $P_3 = P_1 + P_2$ 的 x 坐标仅需 P_1 和 P_2 的 x 坐标, 从而得到计算 kP 的一种新方法。白国强等人^[71]以此为基础, 提出了计算多个数乘运算的新算法。

1.3 论文的主要工作

本文对 ECC 所依托的有限域理论、有限域上的各种运算以及椭圆曲线数乘运

算做了深入分析、研究,主要创新工作如下:

分析了素数域 $GF(p)$ 上的取模运算。针对具有特殊形式的模数(如 Mersenne 数, 伪 Mersenne 数, 广义 Mersenne 数等), 深入研究了其取模运算规律, 得到如下结果: 对 Mersenne 数和伪 Mersenne 数, 给出了取模运算转换为模加或模减运算的公式; 对模数为任意不可约首一三项式和五项式产生的广义 Mersenne 数, 推导了相应的取模运算复杂度计算公式, 并给出了正确性证明。根据该公式, 对任意给定的不可约首一三项式和五项式, 可以根据该多项式的系数分别得出以广义 Mersenne 数为模数的取模运算所需要的模加法(或模减法)的次数。

利用广义 Mersenne 数代替伪 Mersenne 数, 提出了广义最优扩域(GOEFs)的概念, 研究了其上的快速乘法运算和取模运算, 为乘法运算给出了通用的复杂度公式, 为取模运算给出了具体的运算公式, 推广了 Bailey, Mihăilescu 和 Woodbury 等在最优扩域(OEFs)上的相应结果。

基于正规基表示有限域 $GF(2^m)$ 上元素的方法, 以增加 XG 的个数达到减少 AG 个数的方式, 提出了一个串行正规基乘法器和一个并行正规基乘法器。同时, 得到一个 II 型最优正规基乘法器的算法设计, 该乘法器需要 $(2m-2)$ 个 XG, m 个 AG。

研究椭圆曲线数乘算法, 基于椭圆曲线点的不同坐标表示, 分析了相应点加公式所具有的计算复杂性, 并以此为基础, 分析和比较了计算数乘运算的各种算法的计算复杂性。

1.4 论文的章节安排

第二章 预备知识。对本文用到的有限域 $GF(p)$ 和 $GF(2^m)$ 的一些基本概念和基本性质进行了分析和介绍; 同时还介绍了 ECC 及数乘运算的相关基础知识。

第三章 素数域上的取模运算。主要研究了素数域 $GF(p)$ 上的取模运算, 针对 ECC 中经常使用的不可约首一三项式和五项式分别得出取模运算需要的模加法次数公式, 该公式仅与多项式的系数有关。

第四章 有限扩域上的乘法运算。基于广义 Mersenne 素数, 提出了 GOEFs 的概念, 并研究了 GOEFs 上的乘法运算。给出了乘法运算的复杂度计算公式, 并和相关结果进行了性能比较。

第五章 有限域上的求逆运算。深入分析和研究了有限域 $GF(p)$ 、 $GF(2^m)$ 和最优扩域(OEFs)上的各种求逆运算算法, 并对各种求逆算法的性能进行了分析比较。

第六章 正规基乘法器设计。提出并设计了新型的串行和并行正规基乘法器。

第七章 椭圆曲线数乘运算研究。研究各种椭圆曲线数乘算法的计算复杂性，重点比较了计算一个数乘运算的四种算法之间的性能。

第八章 全文总结。总结了论文的主要工作以及在本文基础上今后可进一步开展的研究工作。

第二章 有限域和 ECC

本章主要介绍有限域的基本概念、域元素的不同表示方法、有限域的相关性质、ECC 和数乘运算的基本概念与相关性质。限于篇幅考虑，对大部分性质的描述没有给出相应证明，具体可参见参考文献[3-14]。

我们首先介绍群和域的基本概念，然后介绍有限域和有限域的性质，其次介绍 ECC 和数乘运算的基本概念，最后介绍 ECC 和数乘运算的相关性质。本章的内容是后面章节的理论基础。

2.1 群和域

定义 2.1 任意给定一非空集合 G 和其上的二元运算 “ $*$ ”，如果满足：

- (1) 封闭性：对任意 $a, b \in G$ ，存在 $c \in G$ ，使得 $a * b = c$ ；
- (2) 结合律：对于任意 $a, b, c \in G$ ，都有 $(a * b) * c = a * (b * c)$ ；
- (3) 单位元 e 存在：即存在 $e \in G$ ，对于任意 $a \in G$ ，都有 $e * a = a * e = a$ ；
- (4) 逆元存在：对于任意 $a \in G$ ，存在 $b \in G$ ，使得 $a * b = b * a = e$ ，

则称集合 G 关于二元运算 “ $*$ ” 成群，记为： $\langle G, * \rangle$ 。

在群 $\langle G, * \rangle$ 中，如果对于任意 $a, b \in G$ ，都有 $a * b = b * a$ ，则称群 $\langle G, * \rangle$ 是交换群，也称为阿贝尔 (Abel) 群。

定义 2.2 设 “ $+$ ”，“ $*$ ” 是 G 上的二元运算，集合的基数 $|G| > 1$ ，如果满足：

- (1) $\langle G, + \rangle$ 是一个交换群，其单位元记为 0；
- (2) $\langle G - \{0\}, * \rangle$ 是交换群，其单位元记为 1；
- (3) 运算 “ $*$ ” 对 “ $+$ ” 可分配，即对任意 $a, b, c \in G$ ，都有

$$a * (b + c) = (a * b) + (a * c)$$

$$(a + b) * c = a * c + b * c$$

则称 $\langle G, +, * \rangle$ 是域。

例如： $\langle \mathbb{N}, + \rangle$ 是自然数集关于 “ $+$ ” 所作成的群， $\langle \mathbb{Q}^*, \times \rangle$ 是非零有理数集关于 “ \times ” 所作成的群，而 $\langle \mathbb{Q}, +, \times \rangle$ 则是有理数集上关于 “ $+$ ” 和 “ \times ” 所作成的域。

下面给出域的特征的定义。

定义 2.3 设 G 是任意域, 1 是 G 的单位元。

如果对于任何正整数 n , 有

$$\underbrace{1+1+\cdots+1}_{n\uparrow} \neq 0 \quad (2-1)$$

则称 G 的特征为 0;

如果存在正整数 n , 有

$$\underbrace{1+1+\cdots+1}_{n\uparrow} = 0 \quad (2-2)$$

则称满足(2-2)式的最小正整数 n 为 G 的特征, 记为 $ChG=n$ 。

定理 2.1^[7] 若 G 是域, 则 G 的特征为 0 或者是一个素数 p 。

2.2 有限域

如果域 G 中的元素个数有限, 则称 G 为有限域或伽罗华(Galois)域, 其中, G 中的元素个数称为有限域 G 的阶, 记为 $|G|$ 。

例如, 对任意给定的素数 p , 则 $\langle p, +_p, \times_p \rangle$ 是一个有限域, 它的阶为 p , 其中, $p = \{0, \dots, p-1\}$ 。对任意 $a, b \in p$, $a+_pb = (a+b) \bmod p$, $a \times_p b = (a \times b) \bmod p$ 。

2.2.1 有限域上的多项式

令 G 是任意有限域, G 上的多项式 $f(x)$ 定义如下:

$$f(x) = f_0 + f_1x + f_2x^2 + \cdots + f_{m-1}x^{m-1} + f_mx^m \quad (2-3)$$

其中, m 是非负整数, 系数 $f_i \in G$, $i = 0, 1, \dots, m$ 。我们称 m 是多项式 $f(x)$ 的次数, 记为 $m = \deg(f(x))$, f_0 是常数项, f_m 是首系数。如果 $f_m = 1$, 则称多项式 $f(x)$ 是首一多项式。记系数在 G 中的所有含 x 的多项式组成的集合为 $G[x]$ 。

假设 $f(x), g(x) \in G[x]$, 且 $f(x) = \sum_{i=0}^m f_i x^i, g(x) = \sum_{i=0}^m g_i x^i$, 则

$$f(x) + g(x) = \sum_{i=0}^m (f_i + g_i) x^i \quad (2-4)$$

$$f(x) \times g(x) = \sum_{i=0}^{2m} \left(\sum_{j=0}^i f_j g_{i-j} \right) x^i \quad (2-5)$$

下面给出后面章节将用到的概念和定理。

定义 2.4 令 $f \in G[x]$ 是非零多项式。

(1) 如果 $f(0) \neq 0$, 则使得 $f(x) | (x^e - 1)$ 的最小正整数 e 称为多项式 f 的阶。记为 $\text{ord}(f)$;

(2) 如果 $f(0) = 0$, 则 $f(x) = x^h g(x)$, $h \in N, g(x) \in G(x), g(0) \neq 0$ 是唯一确定的, 此时, $\text{ord}(f) = \text{ord}(g)$ 。

定义 2.5 $f(x)$ 是 G 上的不可约多项式, 如果 $f(x)$ 在 G 上不能再分解为次数更低的两个多项式的乘积。

定义 2.6 假设 $f(x), g(x) \in G[x]$, 如果

- $d(x) \in G[x]$ 是首一多项式;
- $d(x)$ 整除 $f(x)$, 且整除 $g(x)$;
- 如果对任意 $h(x) \in G[x]$, 如果 $h(x)$ 整除 $f(x)$, 且整除 $g(x)$, 则 $h(x)$ 整除 $d(x)$, 则称 $d(x)$ 是 $f(x)$ 和 $g(x)$ 的最大公因式 (Greatest Common Divisor, GCD), 记 $d(x) = \text{GCD}(f(x), g(x))$ 。

2.2.2 有限域的性质

下面给出的性质和定理可参见文献[7]。

定理 2.2 对任意 $f(x) \in G[x]$, 剩余类环 $G[x]/f(x)$ (任意 $h(x) \in G[x]/f(x)$, 当且仅当存在 $l(x) \in G[x]$ 使得 $h(x)$ 是 $l(x)$ 除以 $f(x)$ 后的余式) 是域当且仅当 $f(x)$ 是 G 上的不可约多项式, 其中, G 是有限域。

定理 2.3 (有限域的存在性和唯一性定理)

(1) 如果 G 是有限域, 则存在某个素数 p 和正整数 $m \geq 1$, 使得 G 包含 p^m 个元素;

(2) 对每一个素数幂阶 p^m , 存在唯一的(从同构意义上讲)阶为 p^m 的有限域, 记为 $GF(p^m)$ 。

定理 2.4 如果一个有限域 G 的特征是素数 p , 则有

$$(a+b)^{p^n} = a^{p^n} + b^{p^n}, a, b \in G, n \in N \quad (2-6)$$

可对 n 采用数学归纳法加以证明, 具体可参见文献[7]。

定理 2.5 (子域准则) 令 G 是有 $q = p^n$ 个元素的有限域, 则其每个子域的阶是 p^m , 且 $m|n$ 。反过来, 如果 $m|n$, 则一定存在含 p^m 个元素的子域。而且, 元素 $a \in G$ 在含 p^m 个元素的子域中, 当且仅当 $a^{p^m} = a$ 。

定理 2.6 如果有限域 G 是含 q 个元素的有限域, 则对每个 $a \in G$, 都满足 $a^q = a$ 。

定义 2.7 令 $\langle G, \circ \rangle$ 是群。如果存在 $a \in G$ ，对任意 $b \in G$ ，总存在非负整数 i ，使得

$$b = a^i \quad (2-7)$$

则称群 G 是循环群， a 称为群 G 的生成元，或者称为本原元。

定理 2.7 由有限域 G 的非零元构成的乘法群 G^* 是循环群。

定理 2.8 令 $f(x) \in G[x]$ 是有限域 G 上的不可约多项式， α 是 $f(x)$ 在 G 的扩域上的一个根。如果存在一个多项式 $h(x) \in G[x]$ ，且 $h(\alpha) = 0$ ，则有 $f(x) \mid h(x)$ 。

定理 2.9 令 $f(x) \in G[x]$ 是有限域 G 上的 m 次不可约多项式，则 $f(x) \mid (x^{q^m} - x) \Leftrightarrow m \mid n$ 。

2.2.3 有限域的类型

2.2.3.1 素数域 $GF(p)$

令 p 是一个素数，则 $\{0, 1, \dots, p-1\}$ 组成一个有限域，称为素域，记为 $GF(p)$ 。其上的加法、减法、乘法、除法等运算分别定义如下：

加法 如果 $a, b \in GF(p)$ ，则 $a + b = r$ ，其中， r 是 $a + b$ 除以 p 的余数，即， $0 \leq r \leq p-1$ ，该操作称为模 p 加法。

减法 域元素的减法运算可以转化为加法运算，即如果 $a, b \in GF(p)$ ，则 $a - b = a + (-b)$ ，其中， $-b \in GF(p)$ ，且 $-b + b = 0$ ($-b$ 称为 b 的逆元)。

乘法 如果 $a, b \in GF(p)$ ，则 $a \cdot b = s$ ，其中， s 是 $a \cdot b$ 除以 p 的余数，即， $0 \leq s \leq p-1$ ，该操作称为模 p 乘法。

除法 域元素的除法运算可以转化为乘法运算和求逆运算，即如果 $a, b \in GF(p), b \neq 0$ ，则 $a/b = a \cdot b^{-1}$ ，其中， $b^{-1} \in GF(p)$ ，且 $b^{-1} \cdot b = 1$ (b^{-1} 称为 b 的逆元)。因此，对除法运算的研究一般都转化为对求逆运算的研究，本文也是如此。

显然，在素数域 $GF(p)$ 中，主要的运算是加法、乘法，求逆和取模运算，其中，计算开销最大的是乘法、求逆和取模运算^{[4], [45]}。

2.2.3.2 二元扩域 $GF(2^m)$ (Binary Extension Fields)

二元扩域 $GF(2^m)$ ，称为特征为 2 的有限域，可看成 $GF(2) = \{0, 1\}$ 上的 m 维向量空间。在 $GF(2^m)$ 中存在一个含 m 个元素的集合 $\{\alpha_0, \alpha_1, \dots, \alpha_{m-1}\}$ ，使得任意 $A \in GF(2^m)$ 均可唯一写成如下形式：

$$A = \sum_{i=0}^{m-1} a_i \alpha_i, \text{ 其中 } a_i \in \{0,1\} \quad (2-8)$$

从而可将 A 表示为 m 维向量 $(a_0, a_1, \dots, a_{m-1})$ 。

2.2.3.3 扩域 $GF(p^m)$ (Extension fields)

推广二元扩域的思想, 可得到一般扩域的概念。

定义 2.8 令 p 是一个素数, $m \geq 2$, 令 $f(x)$ 是 $GF(p)[x]$ 上次数为 m 的不可约多项式, 则扩域 $GF(p^m)$ 中的元素由所有 $GF(p)$ 上次数小于 m 的多项式组成, 即

$$GF(p^m) = \{a_{m-1}x^{m-1} + \dots + a_2x^2 + a_1x + a_0 \mid a_i \in GF(p)\}$$

一般扩域与二元扩域上的运算不同之处在于: 二元扩域上的取模运算是模 2 运算, 而一般扩域上的取模运算是模 p 运算, 因此, 此处略去一般扩域运算的介绍。

2.2.4 有限域的基

有限域 $GF(p)$ 的有限扩域 $GF(p^m)$ 可以看成是有限域 $GF(p)$ 上的 m 维向量空间。如果 $\{\alpha_0, \alpha_1, \dots, \alpha_{m-1}\}$ 是 $GF(p^m)$ 在 $GF(p)$ 上的基, 则对任意元素 $A \in GF(p^m)$ 均可唯一表示为如下形式:

$$A = \sum_{i=0}^{m-1} a_i \alpha_i, \text{ 其中 } a_i \in GF(p) \quad (2-9)$$

一般地, $GF(p^m)$ 在 $GF(p)$ 上的基有多种, 但本文最有兴趣的是下面两种基: 正规基(Normal basis, NB)和多项式基(Polynomial Basis, PB)或者称为标准基(Standard or Canonical basis)。下面我们给出这两种基的基本概念和相关性质(此处仅讨论 $p=2$ 的情形)。

2.2.4.1 正规基 (Normal Basis)

假设 γ 是 $GF(2)$ 上次数为 m 的不可约多项式 $f(x)$ (称为正规多项式) 的根。如果集合 $N = \{\gamma, \gamma^2, \dots, \gamma^{2^{m-1}}\}$ 中的元素是线性独立的, 则集合 N 是扩域 $GF(2^m)$ 上的一个正规基。通过寻找正规多项式, 可以建立 $GF(2)$ 的扩域 $GF(2^m)$ 的正规基, 而检测一个不可约多项式是否正规的方法可参见文献[32]。

由于二元扩域 $GF(2^m)$ 被看作是定义在 $GF(2)$ 上的 m 维向量空间, 因此, 可以给出下面的定义:

定义 2.9 任意 $\gamma \in GF(2^m)$, 如果 $\gamma, \gamma^2, \dots, \gamma^{2^{m-1}}$ 是线性独立的, 则元素 γ 是正规

的。

定义 2.10 $N = \{\gamma, \gamma^2, \dots, \gamma^{2^m-1}\}$ 被称为由 γ 产生的正规基, 如果 γ 是正规的, 且 N 是基。

1、I 型正规基的构造

(1) I 型最优正规基的构造

定理 2.10^[32] 假设 $m+1$ 是素数, 2 是模 $m+1$ 的一个原根(即, 2 的幂模 $(m+1)$ 生成 $1 \sim m$ 中的元素), 则 $GF(2)$ 上 m 个非单位元的 $(m+1)$ 次单位根是线性无关的, 且组成 $GF(2^m)$ 到 $GF(2)$ 上的一组最优正规基, 记 $N = \{\alpha^{2^i} \mid i=0, \dots, m-1\} = \{\alpha^j \mid j=1, \dots, m\}$, 其中 α 是一个 $(m+1)$ 次本原单位根(i. e. $\alpha^{m+1}=1$, 但 $\alpha^i \neq 1 (1 \leq i < m+1)$), 称 N 是 $GF(2^m)$ 到 $GF(2)$ 上的一组 I 型最优正规基。

(2) II 型最优正规基的构造

定理 2.11^[32] 令 $GF(2^m)$ 是含 2^m 个元素的有限域。如果 $2m+1=p$ 是素数, 且下面两个条件之一成立:

- 2 是模 p 原根;
- $p \equiv 3 \pmod{4}$, 且 2 模 p 的次数为 m ,

则 $\gamma = \beta + \beta^{-1}$ 生成一个 $GF(2^m)$ 到 $GF(2)$ 上的一组最优正规基, 其中, β 是 $(2m+1)$ 次本原单位根。记 $N = \{\gamma, \gamma^2, \dots, \gamma^{2^m-1}\} = \{\beta + \beta^{-1}, \beta^2 + \beta^{-2}, \dots, \beta^m + \beta^{-m}\}$, 称 N 是 $GF(2^m)$ 到 $GF(2)$ 上的一组 II 型最优正规基。

下面我们介绍一种特殊正规基的构造。

(3) T 型高斯正规基的构造

更一般地, 可推广得到 T 型高斯最优正规基的定义。

定义 2.11 令 q 是一个素数或者素数的幂, m, T 是正整数, $Tm+1$ 是一个素数, 且不整除 q 。设 α 是 $GF(q)$ 的某个扩域上的 $Tm+1$ 次本原单位根, γ 是 $GF(Tm+1)$ 中 T 次本原单位根。则称

$$\beta = \sum_{i=0}^{m-1} \alpha^{\gamma^i}$$

为 $GF(q)$ 上的 (m, T) 型高斯周期(Gauss Period)。

由 (m, T) 型高斯周期(Gauss Period)诱导的正规基 $N = \{\beta, \beta^q, \beta^{q^2}, \dots, \beta^{q^{m-1}}\}$ 称为 T 型高斯正规基(Gaussian Normal Bases, GNB)。

高斯正规基(GNB)是一类特殊的正规基, 其上的有限域乘法运算尤其有效, 因此得到大量的研究, 具体可参见文献[25]。

众所周知, 对任意正整数 m , $GF(2^m)$ 的正规基总存在, 但是其 GNB 并非一定

存在。仅当 $0 \neq m \bmod 8$ 时, 才存在至少一个 GNB。对给定的 T 和 m , 则至多存在一个 T 型 GNB。对给定的 m , 如果存在多个 GNB, 则 T 值越小, 相应的有限域乘法越高效。

定理 2.12 T 型 GNB 的存在条件(见[25])

$GF(2^m)$ 的 T 型 GNB 存在, 当且仅当

- $p = Tm + 1$ 是素数;
- $\gcd(Tm/k, m) = 1$, 其中, k 是 2 模 p 的乘法阶。

2、 T 型高斯正规基 (GNB) 的运算

使用 T 型 GNB 表示域元素的运算定义如下:

假设 $a = (a_{m-1}, \dots, a_1, a_0)$ 和 $b = (b_{m-1}, \dots, b_1, b_0)$, 则

加法 $a + b = c = (c_{m-1}, \dots, c_1, c_0)$, $c_i = a_i + b_i \bmod 2$, 即加法按对应位异或。

平方 $a^2 = \left(\sum_{i=0}^{m-1} a_i \beta^{2^i} \right)^2 = \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} (a_{i-1} \bmod m) \beta^{2^i} = (a_0, a_{m-1}, \dots, a_1)$

即平方运算只是一个简单的循环移位。

乘法 令 $p = Tm + 1$, $u \in GF(p)$ 是阶为 T 的元素。定义序列 $F(1), F(2), \dots, F(p-1)$ 如下:

$$F(2^i u^j \bmod p) = i, 0 \leq i \leq m-1, 0 \leq j \leq T-1 \quad (2-10)$$

对每一个 $l, 0 \leq l \leq m-1$, 定义 A_l 和 B_l 如下:

$$A_l = \sum_{k=1}^{p-2} a_{F(k+1)+l} b_{F(p-k)+l} \quad (2-11)$$

和

$$B_l = \sum_{k=1}^{m/2} (a_{k+l-1} b_{m/2+k+p-1} + a_{m/2+k+p-1} b_{k+l-1}) + A_l \quad (2-12)$$

则 $a \cdot b = c = (c_{m-1}, \dots, c_1, c_0)$, 其中,

$$c_l = \begin{cases} A_l & (T \text{ 是偶数}) \\ B_l & (T \text{ 是奇数}) \end{cases}, \quad 0 \leq l \leq m-1, \text{ 下标是模 } m \text{ 的余数。}$$

使用正规基表示 $GF(2^m)$ 上元素的相应运算算法较好的综述文章, 请参见[1], 具体的软硬件执行参见[2, 72]。

3、计算复杂度

当使用正规基时, 计算一个元素的平方只需一次循环移位即可完成。但是,

计算两个不同元素的乘积却比较复杂,而且选用不同的正规基会导致不同的计算复杂度。

设 $N = \{\beta, \beta^2, \beta^{2^2}, \dots, \beta^{2^{m-1}}\}$ 是 $GF(2^m)$ 在 $GF(2)$ 上的一组正规基, $a = (a_0, a_1, \dots, a_{m-1}) \in GF(2^m)$, $b = (b_0, b_1, \dots, b_{m-1}) \in GF(2^m)$, $c = ab = (c_0, c_1, \dots, c_{m-1})$, 则存在 $\lambda_{i,j} \in GF(2)$, 使得:

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_{i,j} a_{i+k} b_{j+k} \quad (2-13)$$

令 $C_N = \{(i, j) | \lambda_{i,j} \neq 0, 0 \leq i, j \leq m-1\}$, 则 C_N 的基数(集合的元素个数), 记为 $|C_N|$, 反应了在基 N 下做乘法的复杂度。

显然, $|C_N| \leq m^2$ 。1988 年, R.Mullin 等人^[36]证明了 $|C_N| \geq 2m-1$, 如果一组正规基的 $|C_N|$ 达到了这个理论上的下界, 则称其为最优正规基。

T 型正规基的计算复杂度 $|C_N|$ 满足下面的界:

$$\begin{aligned} Tm - (T^2 - 3T + 3) &\leq |C_N| \leq Tm - 1 & T \text{ 是偶数} \\ (T+1)m - (T^2 - T + 1) &\leq |C_N| \leq (T+1)m - T & T \text{ 是奇数} \end{aligned}$$

2.2.4.2 多项式基

如果 $f(x) = x^m + \sum_{i=0}^{m-1} f_i x^i$ ($f_i \in \{0, 1\}, i = 0, 1, \dots, m-1$) 是不可约多项式, 则 $f(x)$ 被称为约化多项式。对每一个约化多项式, 总存在一个多项式基表示。即

设 $N = \{\alpha_0, \alpha_1, \dots, \alpha_{m-1}\}$ 是 $GF(2^m)$ 在 $GF(2)$ 上的一组基, 如果存在 $\beta \in GF(2^m)$, 使得 $\alpha_i = \beta^i$, $0 \leq i \leq m-1$, 则称 N 为一组多项式基。

如果使用多项式基表示, 则 $GF(2^m)$ 的每一个元素分别对应一个次数不超过 m , 系数为 0 或 1 的多项式, 即, 对任意 $a \in GF(2^m)$, 总存在 m 个数 $a_i \in \{0, 1\}$, $i = 0, 1, \dots, m-1$, 使得

$$a = a_{m-1}x^{m-1} + \dots + a_1x + a_0$$

一般地, 可将元素 a 表示为长度为 m 的比特串 $(a_{m-1}, \dots, a_1, a_0)$ 。

假设 $a = (a_{m-1}, \dots, a_1, a_0)$ 和 $b = (b_{m-1}, \dots, b_1, b_0)$, 则使用多项式基表示域元素运算的定义如下:

加法 $a + b = c = (c_{m-1}, \dots, c_1, c_0)$, 其中, $c_i = (a_i + b_i) \bmod 2$, $i = 0, 1, \dots, m-1$ 。即, 域元素加法是按对应位异或。

乘法 $a \cdot b = c = (c_{m-1}, \dots, c_1, c_0)$, 其中, $c(x) = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \dots + c_1x + c_0$ 是 $(\sum_{i=0}^{m-1} a_i x^i)(\sum_{j=0}^{m-1} b_j x^j) \bmod f(x)$ 的余式。

对多项式基下乘法运算的取模运算，我们将在第四章专门给出讨论。

2.2.4.3 正规基和多项式基间的关系

设 $f(x) = x^m + f_{m-1}x^{m-1} + \dots + f_1x + f_0$ 是 $GF(2)$ 上次数为 m 的不可约多项式，把 $f(x)$ 作为约化多项式可定义有限域 $GF(2^m)$ 为：

$$GF(2^m) = \{a_{m-1}x^{m-1} + \dots + a_1x + a_0 \mid a_i \in \{0,1\}\}$$

设 $N = \{\alpha_0, \alpha_1, \dots, \alpha_{m-1}\}$ 是 $GF(2^m)$ 在 $GF(2)$ 上的一组基，如果存在 $\beta \in GF(2^m)$ ，使得 $\alpha_i = \beta^i, 0 \leq i \leq m-1$ ，则称 N 为一组多项式基。如果存在 $\gamma \in GF(2^m)$ ，使得 $\alpha_i = \gamma^{2^i}, 0 \leq i \leq m-1$ ，则称 N 为一组正规基。当取定一组基后，可以把 $GF(2^m)$ 中的元素 $a_0\alpha_0 + a_1\alpha_1 + \dots + a_{m-1}\alpha_{m-1}$ 与 $GF(2)$ 上的 m 维向量 $(a_0, a_1, a_2, \dots, a_{m-1})$ 等同起来。

特别地，根据定理 2.11，II 型最优正规基与多项式基具有如下的转换形式。

因为 $\gamma = \beta + \beta^{-1}$ ，且 β 是 $(2m+1)$ 次本原单位根，于是有

$$\gamma^{2^t} = (\beta + \beta^{-1})^{2^t} = \beta^{2^t} + \beta^{-2^t} = \beta^t + \beta^{-t} \quad (0 < t < p = 2m+1, 2^s = t \pmod{p})$$

而且，当 $m+1 \leq t \leq 2m$ ，可以用 $(p-t)$ 代替 t ，从而，得到下面的定理。

定理 2.13 ^[25] 下面两个基集合

$$M = \{\beta + \beta^{-1}, \beta^2 + \beta^{-2}, \dots, \beta^{2^{m-1}} + \beta^{-2^{m-1}}\}$$

和

$$N = \{\beta + \beta^{-1}, \beta^2 + \beta^{-2}, \dots, \beta^m + \beta^{-m}\}$$

是等价的。

假设 $a = (a_{m-1}, \dots, a_1, a_0)$ 关于基 M 和 N 的表示分别为 $(a_{m-1}, \dots, a_1, a_0)$ 和 $(a_{m-1}', \dots, a_1', a_0')$ ，则由定理 2.13 和 $\beta^{2m+1} = 1$ ，可得到 a_j 和 a_i' 之间的关系为 $a_j = a_i'$ ，其中，

$$j = \begin{cases} k & (k \in [1, m]) \\ (2m+1) - k & (k \in [m+1, 2m]) \end{cases} \quad (2-14)$$

$$k = 2^{i-1} \pmod{2m+1} (i = 1, 2, \dots, m)。$$

于是，利用式(2-14)可将域元素的正规基表示和多项式基表示进行相互转换。

2.3 椭圆曲线密码体制(ECC)

1985 年, Miller^[73]和 Koblitz^[74]首先介绍了 ECC。作为一种对已有的公钥密码系统如 DSA 和 RSA 的替身提出的 ECC, 近年来已经在工业和学术上得到广泛研究^[1]。

目前, ECC 的安全性和优势得到了业界的认可, 且开始被采纳作为标准, 已经建立的采用 ECC 的标准有^[75]:

- 1998 年被 ISO/IEC 确定为数字签名标准 ISO14883-3^[60,61];
- 1999 年 2 月 ECDSA(Elliptic Curve Digital Signature Algorithm, 椭圆曲线数字签名算法)被 ANSI 确定为数字签名标准 ANSI X9.62-1998, ECDHP(Elliptic Curve Diffie-Hellman Protocol, 椭圆曲线 Diffie-Hellman 协议)被确定为密钥交换协议 ANSI X9.63^[58,59];
- 2000 年 2 月被 IEEE 确定为标准 IEEE1363-2000(包括了加密、签名、密钥交换协议方案), 同期, NIST 确定其为联邦数字签名标准 FIPS186-2^[30]。

ECC 之所以受到人们的欢迎, 其主要原因在于对恰当选择的椭圆曲线上的离散对数问题, 迄今为止, 还没有找到亚指数时间复杂性的算法。这意味着在同等安全情况下, ECC 中可以使用比其它有竞争力的密码系统如 RSA 和 DSA 小得多的参数。ECC 的这种高强度带来了许多优点:

- 可以使用较短的密钥;
- 数字签名和证书小;
- 运算速度快。

这些优点使得对有限制的环境如寻呼机, 掌上电脑, 蜂窝电话和智能卡, ECC 成为其理想的选择。另一方面, ECC 的具体执行要求选择基本有限域的类型、执行有限域运算的算法、椭圆曲线的类型、执行椭圆曲线有理点群操作的算法以及椭圆曲线协议等。不同的选择对最终性能的影响很大。本章将对椭圆曲线密码系统基础理论和数乘运算中用到的主要方法和技术给出较为详细的介绍。

2.3.1 椭圆曲线

设 $GF(q)$ 是含 q 个元素的有限域, $q=p^r$, p 为素数, r 为一个正整数。设 K 是一个域, 令 $K^* = K \setminus \{0\}$, \bar{K} 表示 K 的代数闭包。如果 $K=GF(q)$, 则 $\bar{K} = \bigcup_{r \geq 1} GF(q^r)$ 。

K 上的射影平面 $P^2(K)$ 是 $K^3 \setminus \{0,0,0\}$ 上的等价关系 R 的等价类集合, 其中, R 定义为 $(X_1, Y_1, Z_1)R(X_2, Y_2, Z_2)$ 当且仅当存在 $a \in K^*$ 使得 $(X_1, Y_1, Z_1) = (aX_2, aY_2, aZ_2)$ 。包含 (X, Y, Z) 的等价类记为 $(X:Y:Z)$ 。形如下式的 3 次齐次方程称为 K 上的 Weierstrass 方程:

$$Y^2Z + a_1XYZ + a_3YZ^3 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3 \quad (2-15)$$

其中 $a_i \in K, i=1,2,3,4,6$ 。

令 $F(X,Y,Z) = Y^2Z + a_1XYZ + a_3YZ^3 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3$ 。对所有满足 $F(X, Y, Z)=0$ 的射影点 $P=(X:Y:Z) \in P^2(\bar{K})$, 如果 $F(X,Y,Z)$ 在 P 点的 3 个偏导数 $\partial F(X,Y,Z)/\partial X$, $\partial F(X,Y,Z)/\partial Y$, $\partial F(X,Y,Z)/\partial Z$ 不全为 0, 则称方程(2-15)是平滑的或非奇异的, 否则称为奇异的。奇异和非奇异曲线的几何图形分别见图 2-1 和 2-2。

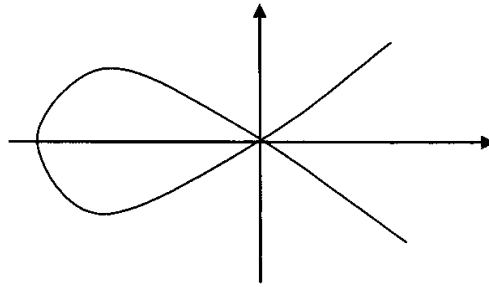


图 2-1 奇异曲线图

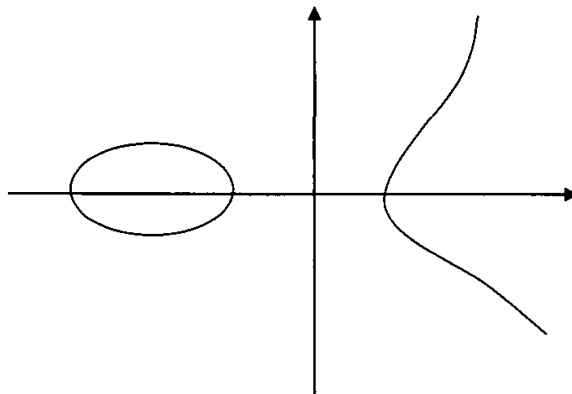


图 2-2 非奇异曲线图

此时方程(2-15)在 $P^2(\bar{K})$ 中的所有解组成下面的集合 E :

$$E = \{(X: Y: Z) \in P^2(\bar{K}) \mid F(X, Y, Z) = 0\} \quad (2-16)$$

称为域 K 上的一条椭圆曲线。在任一条椭圆曲线中, 存在唯一一点的 Z 坐标为 0, 该点表示为 $O = (0: 1: 0)$, 称为无穷远点。

设 $Z \neq 0$, $x = X/Z$, $y = Y/Z$, 则可将方程(2-15)以仿射坐标的形式改写为:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (2-17)$$

此时, 相应的椭圆曲线 E 即是方程(2-17)在仿射平面 $A^2(\bar{K})$ 中的所有解及无穷远点 O 组成的集合, 即:

$$E = \{(x, y) \in \bar{K} \times \bar{K} \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}$$

令 E/K 表示 K 上的椭圆曲线 E , E/K 上两个仿射坐标均属于 K 的点及无穷远点 O 称为 E/K 的 K -有理点。 E/K 的所有 K -有理点组成的集合记为 $E(K)$, 即

$$E(K) = \{(x, y) \in K \times K \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}$$

2.3.2 有限域上的椭圆曲线

ECC所使用的椭圆曲线都是定义在有限域上的, 有限域的类型及其大小决定了ECC的安全性和实现的有效性。ECC通常选用 $GF(p)$ (p 是一个大素数)和 $GF(2^m)$ (特征为2的有限域)作为其基域, 下面给出这两种域上的椭圆曲线方程。

2.3.2.1 $GF(p)$ 上的椭圆曲线

令 $p > 3$ 是一个奇素数, $a, b \in GF(p)$, 且满足 $4a^3 + 27b^2 \neq 0 \pmod{p}$, 则定义在 $GF(p)$ 上的椭圆曲线 $E(GF(p))$ 的方程为:

$$y^2 = x^3 + ax + b \quad (2-18)$$

对任意 $x, y \in GF(p)$, $P = (x, y) \in E(GF(p))$, 当且仅当 $P = (x, y)$ 的 x 和 y 坐标满足方程(2-18)。

例 2.1 令椭圆曲线

$$y^2 = x^3 + x + 1 \quad (2-19)$$

是 $GF(23)$ 上的一个方程。该椭圆曲线方程在 $GF(23)$ 上的解为:

$(0, 1), (0, 22), (1, 7), (1, 16), (3, 10), (3, 13), (4, 0), (5, 4), (5, 19), (6, 4), (6, 19), (7, 11), (7, 12), (9, 7), (9, 16), (11, 3), (11, 20), (12, 4), (12, 19), (13, 7), (13, 16), (17, 3), (17, 20), (18, 3), (18, 20), (19, 5), (19, 18)$ 。

则上面的27个解加上无穷远点 O 构成 $E(GF(23))$ 。

2.3.2.2 $GF(2^m)$ 上的椭圆曲线

如果 j 不变量 $j(E) \neq 0$, 其中 $j(E) = 1/(-(1+4a)^2 \times b - 432b^2)$, 则由参数 $a, b \in GF(2^m), b \neq 0$ 定义的 $GF(2^m)$ 上的非超奇异椭圆曲线为:

$$y^2 + xy = x^3 + ax^2 + b \quad (2-20)$$

方程(2-20)上的点 $P = (x, y) (x, y \in GF(2^m))$ 和无穷远点 o 构成集合 $E(GF(2^m))$ 。

如果 j 不变量为 0, 则由参数 $a, b \in GF(2^m), c \neq 0$ 定义的 $GF(2^m)$ 上的非超奇异椭圆曲线为:

$$y^2 + cy = x^3 + ax^2 + b \quad (2-21)$$

方程(2-21)上的点 $P = (x, y) (x, y \in GF(2^m))$ 和无穷远点 o 构成集合 $E(GF(2^m))$ 。

例 2.2 令椭圆曲线

$$y^2 + xy = x^3 + \alpha^4 x^2 + 1 \quad (2-22)$$

是 $GF(2^4)$ 上的一个方程, 其中, α 是不可约多项式 $f(x) = x^4 + x + 1$ 的根。该椭圆曲线方程在 $GF(2^4)$ 上的解为:

$$\begin{aligned} &(0, 1), (1, \alpha^6), (1, \alpha^{13}), (\alpha^3, \alpha^8), (\alpha^3, \alpha^{13}), (\alpha^5, \alpha^3), \\ &(\alpha^5, \alpha^{11}), (\alpha^6, \alpha^8), (\alpha^6, \alpha^{14}), (\alpha^9, \alpha^{10}), (\alpha^9, \alpha^{13}), \\ &(\alpha^{10}, \alpha^1), (\alpha^{10}, \alpha^8), (\alpha^{12}, 0), (\alpha^{12}, \alpha^{12}) \end{aligned}$$

则上面的15个解加上无穷远点 O 构成 $E(GF(2^4))$ 。

2.3.3 椭圆曲线有理点群

定义 2.12 设 $E(K)$ 表示所有 K -有理点组成的集合, o 是无穷远点, 如果:

- (1) 对 $\forall P \in E(K)$, $o + P = P + o = P$, 即 o 是单位元;
- (2) $-o$ 表示 o 的逆元, 且有: $-o = o$;
- (3) 对 $\forall P \in E(K)$, $P = (x_1, y_1) \neq o$, $-P$ 表示 P 的逆元, 有

$$-P = (x_1, -y_1 - a_1 x_1 - a_3), \quad P + (-P) = o;$$

- (4) 假设 $P = (x_1, y_1) \neq o$, $Q = (x_2, y_2) \neq o$, $Q \neq -P$, 定义 $P + Q = (x_3, y_3)$ 为:

$$\begin{cases} x_3 = k^2 + a_1 k - a_2 - x_1 - x_2 \\ y_3 = -(k + a_1)x_3 - l - a_3 \end{cases} \quad (2-23)$$

其中

$$k = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & P \neq Q \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}, & P = Q \end{cases}, \quad l = y_1 - kx_1 \quad (2-24)$$

以上定义的椭圆曲线上的点“+”运算，使 $\langle E(K), + \rangle$ 成为 Abel 群，无穷远点 O 为该群的单位元，该群的阶记为 $\#E(K)$ 。点加运算的几何描述见图 2-3。

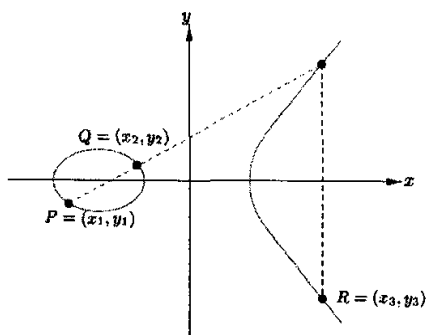


图2-3 椭圆曲线上 $P + Q = R$ 的几何描述

ECC 就是建立在 Abel 群 $\langle E(K), + \rangle$ 上的密码体制。 $\#E(K)$ 对于密码体制的安全性至关重要，然而，精确计算 $\#E(K)$ 是很困难的问题。

例 2.3 假设椭圆曲线方程为(2-19)。令 $P = (x_1, y_1) = (3, 10)$ ， $Q = (x_2, y_2) = (9, 7)$ ， $P + Q = (x_3, y_3)$ ， $2P = (x_4, y_4)$ ，计算 x_3 ， y_3 和 x_4 ， y_4 。

解：比较(2-17)和(2-19)知， $a_1 = a_2 = a_3 = 0$ ， $a_4 = a_6 = 1$ ，于是公式(2-23)和(2-24)简化为：

$$\begin{cases} x_3 = k^2 - x_1 - x_2 \\ y_3 = k(x_1 - x_3) - y_1 \end{cases}, \quad \text{其中 } k = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & P \neq Q \\ \frac{3x_1^2 + 1}{2y_1}, & P = Q \end{cases}$$

于是有: $k_1 = (y_2 - y_1)/(x_2 - x_1) = (7 - 10)/(9 - 3) = -1/2 = 11 \in GF(23)$;

$$x_3 = k_1^2 - x_1 - x_2 = 11^2 - 3 - 9 = -6 = 17;$$

$$y_3 = k(x_1 - x_3) - y_1 = 11(3 - 17) - 10 = 89 = 20;$$

$$k_2 = (3x_1^2 + 1)/2y_1 = (3 \times 3^2 + 1)/(2 \times 10) = 5/20 = 6 \in GF(23);$$

$$x_4 = k_1^2 - 2x_1 = 6^2 - 6 = 30 = 7;$$

$$y_3 = k_2(x_1 - x_3) - y_1 = 6(3 - 7) - 10 = -34 = 12.$$

2.3.4 椭圆曲线有理点群结构

记 $\#E(GF(q))$ 表示 $E(GF(q))$ 上有理点的个数。

定理 2.14^[76] (Hasse 定理) 令 $\#E(GF(q)) = q + 1 - t$, 则 $|t| \leq 2\sqrt{q}$ 。

由 Hasse 定理, 可求得椭圆曲线上点数 $\#E(GF(q))$ 的范围。

定义 2.13^[76] 称 $t = q + 1 - \#E(GF(q))$ 为椭圆曲线 E 的 Frobenius 迹, 记为 $Tr(Frq)$ 。

设 p 是有限域 $GF(q)$ 的特征, 如果 p 整除 t , 则称椭圆曲线 E 是超奇异的, 否则, 是非超奇异的。

定理 2.15^[10] (椭圆曲线有理点群结构定理) $E(GF(q))$ 是一个秩为 1 或 2 的交换 (Abel) 群, 即:

$$E(GF(q)) \cong Z_{n_1} \oplus Z_{n_2}$$

并且 n_1, n_2 满足性质: $n_2 | n_1, n_2 | q - 1$ 。

设 E 是定义在 $GF(q)$ 上的椭圆曲线, 如果把它看作 $GF(q)$ 的 k 次扩域 $GF(q^k)$ 上的曲线, 则下面的定理:

定理 2.16^[10] (Weil 定理) 设 E 是定义在 $GF(q)$ 上的椭圆曲线, $t = q + 1 - \#E(GF(q))$, 则 $\#E(GF(q^k)) = q + 1 - \alpha^k - \beta^k$, 其中, α, β 为方程 $x^2 - tx + q$ 的两个复数根。

定理 2.16 说明, 对定义在 $GF(q)$ 上的椭圆曲线 E , 如果已知它在 $GF(q)$ 上的有理点群的阶, 则可以容易的计算出它在 $GF(q)$ 的 k 次扩域上的阶, 其中, k 为任意正整数。

推论 2.17^[10] 设 E 是定义在 $GF(q)$ 上的椭圆曲线, $t = q + 1 - \#E(GF(q))$, 令 $c_0 = 2$, $c_1 = t$, 对 $k > 1$, 定义序列:

$$c_k = c_1 c_{k-1} - q c_{k-2}$$

则 $\#E(GF(q^k)) = q + 1 - c_k$ 。

2.4 小结

本章主要从两个方面介绍了论文后续部分将涉及的数学基础。第一部分主要详细介绍了群和域的基本概念，其中重点介绍了有限域上的多项式，有限域的性质，有限域的类型和有限域的基，并给出了在不同有限域上各种运算的计算规则；第二部分详细介绍了 ECC 的基本概念，相关性质，以及椭圆曲线有理点群的点加和倍加公式，并给出了相关计算实例。本章涉及的概念和基本性质将作为论文后续工作的基础。

第三章 素数域上的取模运算

公钥密码学,特别是 ECC 的发展,重新点燃了研究者们研究多项式计算的热情^[44,77]。我们知道,如果令 $f(x)$ 是 $GF(p)$ 上次数为 n 的不可约多项式,则 $\{1, x, x^2, \dots, x^{n-1}\}$ 形成有限扩域 $GF(p^n)$ 的多项式基, $GF(p^n)$ 中的每一个元素均可表示为 $GF(p)$ 上次数不超过 $n-1$ 的多项式。当使用多项式基表示域元素时,域元素的乘法就转换为多项式乘法,然后再以不可约多项式 $f(x)$ 为模数作取模运算;加法则是两个域元素对应位的取模加法。显然,有限扩域上的乘法和加法运算都涉及取模运算,前者是以一个多项式为模数,而后者是以一个素数为模数。因此,加快取模运算的计算速度对有限域运算,特别是对有限域乘法运算性能的提高具有十分重要的意义。

本章首先给出经典取模算法, Barrett 取模算法和 Montgomery 算法等三种算法,并从其优缺点两个方面进行分析比较;然后针对具有特殊形式的模数(如 Mersenne 数,伪 Mersenne 数,广义 Mersenne 数等)深入分析了其取模运算规律,得到如下结果:对 Mersenne 数和伪 Mersenne 数,设计了取模运算转换为模加或模减运算公式;对模数为任意不可约首一三项式和首一五项式产生的广义 Mersenne 数,推导了相应的取模计算复杂度解析式,并给出了详细分析。利用该公式,对任意给定的不可约首一三项式和五项式,可以根据该多项式的系数分别计算出以对应广义 Mersenne 数为模数的取模计算所需要的模加(或模减)的次数。

3.1 取模运算

3.1.1 以正整数为模数

对正整数 n , 如果存在某个整数 k , 使得 $a=b+kn$, 那么 $b \equiv a \pmod{n}$, 其中, $a, b \in \mathbb{Z}$, $0 \leq b < n$, \mathbb{Z} 表示整数的集合, 即 b 可看作是 a 整除 n 后的余数。有时, b 又被叫做 a 模 n 的余数(residue), 或称为 a 与 b 模 n 同余(congruent) (“ \equiv ”表示同余)。

例如: $11 \equiv 23 \pmod{12}$, 则 11 就是 23 模 12 的余数, 或者说 23 和 11 模 12 同余。

3.1.2 以多项式为模数

设 $A(x), f(x) \in GF[p](x)$, 如果存在多项式 $K(x), B(x) \in GF[p](x)$, 且 $\deg(B(x)) < \deg(f(x))$, 使得 $A(x) = B(x) + K(x)f(x)$, 那么 $B(x) \equiv A(x) \pmod{f(x)}$ 。即 $B(x)$ 可看作是 $A(x)$ 整除 $f(x)$ 后的余式。有时, $B(x)$ 又叫做 $A(x)$ 模 $f(x)$ 的余式, 或者叫做 $A(x)$ 与 $B(x)$ 模 $f(x)$ 同余。

例如: $x+1 \equiv x^4 \pmod{x^4+x+1}$, 则 $x+1$ 就是 x^4 模 x^4+x+1 的余式, 或者说 x^4 和 $x+1$ 模 x^4+x+1 同余。

3.1.3 取模运算的运算律

取模运算就像普通的运算一样, 它是可交换的、可结合的、可分配的^[9]。设 $a, b \in Z$, 则以正整数 n 为模数的取模运算满足如下运算规律:

交换律: $(a+b) \bmod n = (b+a) \bmod n$;

结合律: $((a+b) \bmod n + c) \bmod n = (a + ((b+c) \bmod n)) \bmod n$;

分配律: $(a \times (b+c)) \bmod n = ((a \times b) \bmod n + ((a \times c) \bmod n)) \bmod n$ 。

同时, 在取模运算中, 每一个中间结果的模 n 运算, 其作用与先进行全部运算, 然后再进行模 n 运算是一样的。即具有如下的运算公式:

$$(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n;$$

$$(a-b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n;$$

$$(a \times b) \bmod n = ((a \bmod n) \times (b \bmod n)) \bmod n。$$

类似地, 如果将 a, b, n 分别用 $A(x), B(x)$ 和 $f(x)$ 代替, 上面的运算定律仍然成立。

从取模运算的定义可以看出, 以一个正整数或多项式为模数的取模运算通过整数除法或者多项式除法来完成的, 而除法运算在所有的运算中开销最大。因此, 为了减少计算开销, 通常利用预计算和一些开销较小的移位计算来取代除法运算, 从而达到加快取模运算速度的目的。

3.2 现有算法及其分析

在一些签名机制里, 取模运算被广泛涉及, 而且模数并不具有某种特殊形式, 为此, 这一小节我们介绍三种通用算法: 经典取模算法(classical algorithm), Barrett

取模算法^[3,78]和 Montgomery 取模算法^[79]。

首先假设模数 $p = \sum_{i=0}^{k-1} p_i b^i, 0 < p_{k-1} < b, 0 \leq p_i < b (i=0,1,\dots,k-2)$, 待取模的数 $z = \sum_{i=0}^{l-1} z_i b^i, 0 < z_{l-1} < b, 0 \leq z_i < b (i=0,1,\dots,l-2)$ 均为 b 进制表示, 且 $z < p^2, b \geq 2$ 是任意整数。 b 的取值可根据计算机的结构和所使用的编程语言来决定, 恰当选取 b 的值可加快乘法、除法和模 b^{2k} 计算的速度。如果选取编程语言可获得的整数值作为 b 的值, 上面提到的三种计算可分别向左移 k 位, 向右移 k 位和保留最右边的 k 位完成, 而且, b 越大, 执行 b 进制计算的次数越少, 当然计算速度越快。

3.2.1 经典取模算法

经典取模算法是通常 $(l-k)$ 步笔算除法的形式化算法, 每一步都是 z 的 $(k+1)$ 位除以模数 p 的 k 位, 产生商 q 的 1 位和余数 r 的 k 位, 且 r 小于 p , 随后 r 的 k 位加上被除数 z 的下一位构成新的 $(k+1)$ 位, 进行下一步除法。

D. Knuth^[29]描述的经典取模算法分两步完成: 尽可能精确地估计商 q 的值, 用 p_{k-1} 除 z 中最左边的两位就可得出一个估计值, 如果 $p_{k-1} \geq \lfloor b/2 \rfloor$, 则估计值最多不超过 2 倍。如果使用 z 最左边的三位和 p 最左边的两位进行估计的话, 估计值几乎总是准确值, 而且最多不超过 1 倍, 其发生的概率大约为 $2/b$ 。具体算法如下:

算法 3.1 经典取模算法

输入: 正整数 $z = (z_l \dots z_1 z_0)_b, p = (p_k \dots p_1 p_0)_b$, 其中 $l \geq k \geq 1, p_k \neq 0$

输出: 商 $q = (q_l \dots q_1 q_0)_b$ 和余数 $r = (r_k \dots r_1 r_0)_b$, 使得 $z = qp + r, 0 \leq r < p$

1 For $j=0$ to $(l-k)$ do: $q_j \leftarrow 0$

2 While $(z \geq p b^{l-k})$ do

2.1 $q_{l-k} \leftarrow q_{l-k} + 1$

2.2 $z \leftarrow z - p b^{l-k}$

3 For $i=l$ downto $(k+1)$ do

3.1 If $z_i = p_k$ then set $q_{i-k-1} \leftarrow b-1$

else set $q_{i-k-1} \leftarrow \lfloor (z_i b + z_{i-1}) / p_k \rfloor$

3.2 While $(q_{i-k-1}(p_k b + p_{k-1}) > z_i b^2 + z_{i-1} b + z_{i-2})$ do: $q_{i-k-1} \leftarrow q_{i-k-1} - 1$

3.3 $z \leftarrow z - q_{i-k-1} p b^{i-k-1}$

3.4 If $z < 0$ then set $z \leftarrow z + p b^{i-k-1}$ and $q_{i-k-1} \leftarrow q_{i-k-1} - 1$

4 $r \leftarrow z$

5 Return (q, r)

一般地,校正值(normalization) $p^* = \lfloor b/p_{k-1} \rfloor \times p$ 将确保 $p_{k-1}^* \geq \lfloor b/2 \rfloor$ 。在二进制计算机中, b 通常取 2 的幂次, 这时, 校正过程仅需执行简单移位即可。最后正确的余数 r 通过进行与校正过程相反的移位即可获得。

3.2.2 Barrett 取模算法

Barrett 取模算法由 P.D.Barrett 在 1987 年提出^[78], 该算法的基本思想也是基于估计商的值。与经典取模算法比较, Barrett 取模算法在估计商的值时开销比经典取模算法少, 而且还可以通过预计算完成, 即是说, 如果恰当选择一个基 b , 则可以使用较少的开销估计商 $\lfloor z/p \rfloor$ 。具体算法描述如下:

算法 3.2 Barrett 取模算法

输入: $p, b \geq 3, k = \lfloor \log_b p \rfloor + 1, 0 \leq z < b^{2k}, \mu = \lfloor b^{2k}/p \rfloor$

输出: $z \bmod p$

1 $\hat{q} \leftarrow \lfloor \lfloor z/b^{k-1} \rfloor \cdot \mu/b^{k+1} \rfloor$

2 $r \leftarrow (z \bmod b^{k+1}) - (\hat{q} \cdot p \bmod b^{k+1})$

3 If $r < 0$ then $r \leftarrow r + b^{k+1}$

4 While $r \geq p$ do: $r \leftarrow r - p$

5 Return (r)

从算法 3.2 可以看出, 算法必须预计算一个与模数 p 有关的项 $\mu = \lfloor b^{2k}/p \rfloor$, 这使得该算法更适合单一模数(single modulus)的取模情形。

另外, 从方便计算的角度考虑, 在使用 Barrett 取模算法时, 应注意如下几点:

1. 基 b 的选取通常取为 $b=2^L$ 的形式, 其中, L 尽量接近机器的字长。
2. 与 μ 的计算不同的是, 被要求的两个除法(第 1 步)仅是 b 进制表示的一个简单移位。

3. 令 $z' = \lfloor z/b^{k-1} \rfloor$, 显然 z' 和 μ 至多含 $(k+1)$ 个 b 进制字节。第 1 步中 \hat{q} 的计算丢弃了乘积 $z' \cdot \mu$ 中最右边的 $(k+1)$ 位。即, 给定 z' 和 μ 的 b 进制表示:

$$z' = \sum_{i=0}^k z'_i b^i, \quad \mu = \sum_{j=0}^k \mu_j b^j, \quad \text{则}$$

$$z' \mu = \sum_{l=0}^{2k} \underbrace{\sum_{i+j=l} (z'_i \mu_j)}_{w_l} b^l \quad (3-1)$$

其中, w_l 可能超过 $b-1$ 。如果 $b \geq k-1$, 则 $\sum_{l=0}^{k-2} w_l b^l < b^{k+1}$, 所以

$$0 \leq \frac{z' \mu}{b^{k+1}} - \sum_{l=k-1}^{2k} \frac{w_l b^l}{b^{k+1}} = \sum_{l=0}^{k-2} \frac{w_l b^l}{b^{k+1}} < 1 \quad (3-2)$$

式(3-2)说明如果 $b \geq k-1$, 则用 $\sum_{l=k-1}^{2k} \frac{w_l b^l}{b^{k+1}}$ 代替 \hat{q} 最多只差 1, 且找到 \hat{q} 的估计值最多需要 $(1+2+\cdots+(k+1))+k = 0.5k^2 + 2.5k + 1$ 个一位数(single-precision)乘法(即小于 b 值的乘法)。

4. 在第 2 步仅需要 $\hat{q} \cdot p$ 的 $(k+1)$ 个最小的数字, 因为 $p < b^k$, 经过 $(1+2+\cdots+k) + (k-1) = 0.5k^2 + 1.5k - 1$ 个一位数乘法即可得到最右边的 $(k+1)$ 位。

事实上, Barrett 取模算法的最大开销在于第一步中的 $(k+1)$ 位的大数乘法和第二步中 k 位的大数乘法, 即计算 $z \bmod p$ 共需要 $(k^2 + 4k)$ 个一位数乘法。

3.2.3 Montgomery 取模乘法

Montgomery 取模算法由 Montgomery^[79]提出。与 Barrett 取模算法一样, Montgomery 取模乘法也是用一些开销较小的运算来取代取模运算中开销最大的除法运算。该方法对单个取模乘法效果不好, 但是能够被有效地应用到诸如模幂运算(对给定的输入, 需要执行多次乘法)中, 因为 Montgomery 取模乘法所要求的数据转换开销, 可通过存储中间结果来抵消, 从而使得该算法有效, 具体可参见文献[2,20,80-85]。

假设 p 是奇数, $R > p$, 且 $\gcd(R, p) = 1$, $0 \leq z < pR$, 则计算 $c = zR^{-1} \bmod p$ 的 Montgomery 取模乘法的算法如下:

算法 3.3 Montgomery 取模乘法算法

输入: 正整数 z , p 和 R (因为 p 是奇数, 所以可以选择 $R = 2^k$, 这样的 R 使得其做除数时开销相对较小)。

输出: c

1 预计算 $p' = -p^{-1} \bmod R$ (即, 预计算 p 的负逆元 p')

2 计算 $c \leftarrow (z + (zp' \bmod R)p) \bmod R / R \quad (3-3)$

3 If $c \geq p$ then $c \leftarrow c - p \quad (3-4)$

该算法的复杂性主要在于第 2 步中的两个乘法: zp' 和 $(zp' \bmod R)p$ 。由于 $R = 2^k$, 因此这两个乘法仅需计算其最右边的 k 个位。从而, 该算法需要 $k(k+1)$ 个一位数乘法。

3.2.4 三种算法的性能比较

Barrett 取模算法的优点在于：①模数的选取是任意的；②去掉了取模运算中开销最大的除法运算，取而代之的是一些开销较小的移位运算。缺点：该算法要求预计算一个与模数 p 有关的除法运算。因此，它比较适合对同一个模数进行多次取模的情形。该算法可以看成是对经典取模算法的一个直接代替。

与 Barrett 取模算法一样，Montgomery 取模乘法也是用一些开销较小的运算来代替取模运算中开销最大的除法运算。该方法对单个取模乘法效果不好，但是能够被有效地应用到诸如取模幂运算(对给定的输入，需要执行多次乘法)中。

下面我们从加法、减法和一位数乘法和多位数乘法的次数几个方面比较三种计算 $z \bmod p$ 的算法，除了一位除法外，还包含除以 b 的幂次和对 b 的幂次取模。几种算法具体的比较结果见表 3-1，其中 z 和 p 的长度分别是 $2k$ 和 k 。

表 3-1 三种取模算法的复杂度分析

算法	经典算法	Barrett 算法	Montgomery 算法	乘法
乘法	$k(k+2.5)$	$k(k+4)$	$k(k+1)$	k^2
除法	k	0	0	0
预计算	无	b^{2k}/p	$-p_0^{-1} \bmod b$	0
数据转换	无	无	p -剩余	无
事后计算	无	无	约简	无
限制条件	无	$z < b^{2k}$	$z < p^{bk}$	无

从表 3-1 可以看出，若仅考虑取模运算，且 z 是模数 p 的 2 倍时，Montgomery 算法显然快于另外两种算法，而且几乎和乘法一样快，但 Barrett 算法又稍慢于经典取模算法。在最后一行的“限制条件”中，Barrett 算法和 Montgomery 算法分别要求 z 小于 b^{2k} 和 pb^k 。但是，如果这三个算法被用于约简小于模数 p 的两个整数的乘积，即如果 $x < p^2 < pb^k < b^2$ ，这个限制条件对它们的适用性没有任何影响。

3.3 特殊模数的取模运算

Barrett 取模算法和 Montgomery 取模乘法的共同点都是去掉了开销较大的除法运算，使得取模运算的速度得到较大提高。但是当模数 p 具有某种特殊形式，如

Mersenne 数, 伪 Mersenne 数和广义 Mersenne 数时, 取模计算不需任何预计算, 直接用模加和模减代替取模计算中的除法运算, 从而使得取模计算的速度得到大大提高。

3.3.1 Mersenne 数的取模运算

形如 (2^n-1) 的数称为 Mersenne 数。

设 $p=2^n-1$, 对任意正整数 $A(<p^2)$, 令 $A=A_1+A_2 \times 2^n$, 其中 $0 \leq A_1, A_2 < 2^n-1$, 则

$$A \bmod p = (A_1 + A_2) \bmod p \quad (3-5)$$

定理 3.1 令 $S=A_1+A_2$, A_1 和 A_2 的值如上所述, 则 $S < 2p$, 且 S 的平均值 $\bar{S} < p$ 。

证明 由(3-5)的假设知, $0 \leq A_1, A_2 < 2^n-1$, 于是,

$$S = A_1 + A_2 < 2^n-1 + 2^n-1 = 2(2^n-1) = 2p \quad (3-6)$$

因为 $0 \leq A_1, A_2 < 2^n-1$, 所以 A_1 和 A_2 可能取 $[0, 2^n-2]$ 中的任何值, 根据等概率原则, 有

$$\bar{S} = 2 \times \frac{1}{2^n-1} \times (0+1+\dots+2^n-2) = 2^n-2 < 2^n-1 = p \quad (3-7)$$

根据定理 3.1 和式(3-5)的假设, 我们得到模数为 Mersenne 数的取模算法。

算法 3.4 模数为 Mersenne 数的取模算法

输入: $p, A < p^2$, p 是 Mersenne 数, $A = (a_{2n-1}, a_{2n-2}, \dots, a_1, a_0)$

输出: $A \bmod p$

1 $A_1 = (a_{2n-1}, a_{2n-2}, \dots, a_{n+1}, a_n)$

2 $A_2 = (a_{n-1}, a_{n-2}, \dots, a_1, a_0)$

3 $S = A_1 + A_2$

4 If $S > p$ then $S = S - p$

5 Return (S)

从算法 3.4 可以看出, 计算 $A \bmod p$ 的值最坏情况下仅需要一个加法和一个减法。显然, 如果 p 是 Mersenne 数, 则取模计算的速度非常快。因此, 如果 ECC 的基域选择 Mersenne 素数域, 则 ECC 的具体实现速度会得到很大提高。但遗憾的是, 2005 年 2 月, 人们才发现第 42 个 Mersenne 素数。因此 D.V.Bailey 和 C.Paar^[55]以及 P.Mihăilescu^[86]提出了伪 Mersenne 数的概念。

3.3.2 伪 Mersenne 数的取模运算

形如 $(2^n - c)$ 的数称为伪 Mersenne 数, 其中 c 是正整数, 且 $\log_2 c \leq \lfloor 0.5n \rfloor$ 。

如果 $p = 2^n - c$ 是伪 Mersenne 数, 则可将 c 表示成如下的二进制形式:

$$c = 2^{n_k} + 2^{n_{k-1}} + \dots + 2^{n_1} + 1 \quad (3-8)$$

其中 $n_k > n_{k-1} > \dots > n_1 > 0$ 。于是 p 具有如下形式:

$$p = 2^n - 2^{n_k} - 2^{n_{k-1}} - \dots - 2^{n_1} - 1 \quad (3-9)$$

对任意正整数 $A (< p^2)$, 令

$$A = A_1 + A_2 \times 2^n \quad (0 \leq A_1 \leq 2^n - 1) \quad (3-10)$$

$$A_2 = A_3 + A_4 \times 2^{n-n_k} \quad (0 \leq A_3 \leq 2^{n-n_k} - 1) \quad (3-11)$$

$$A_{2i} = A_{2i+1} + A_{2i+2} \times 2^{n_{k-i} - n_{k-i-1}} \quad (0 \leq A_{2i+1} \leq 2^{n_{k-i} - n_{k-i-1}} - 1) \quad (3-12)$$

则

$$\begin{aligned} A \bmod p = & A_1 + \sum_{i=1}^{k+1} A_{2i} + \\ & (A_3 + \sum_{i=2}^{k+1} A_{2i}) \sum_{i=1}^k 2^{n_i} + 2^{n-k} \sum_{i=2}^k \sum_{j=1}^{w-i} A_{2i+1} 2^{n_j} \pmod{p} \end{aligned} \quad (3-13)$$

事实上, 伪 Mersenne 数可看作广义 Mersenne 数。下面我们研究广义 Mersenne 数的取模运算。

3.3.3 广义 Mersenne 数的取模运算

令 s 是机器字长, 如果 $f(x)$ 是首一 d 次整系数多项式, 且 $f(x)$ 只有少数几个非零项, 则称 $f(2^s)$ 是广义 Mersenne 数。

例如: 设 $s=8$, $f(x) = x^{14} - x^7 - 1$, 则 $p = (2^8)^{14} - (2^8)^7 - 1$ 是广义 Mersenne 数。

因为广义 Mersenne 数具有形式 $2^n - 2^m - 1$, 所以以其为模数的取模计算可采用与伪 Mersenne 数类似的方法进行转化。

3.3.3.1 基于多项式的取模运算原理

假设 $p = f(2^s)$, 计算 $x^d \bmod f(x)$, $x^{d+1} \bmod f(x)$, \dots , $x^{2d-1} \bmod f(x)$, ($f(x)$ 的描述见 3.3.3 节第一自然段)如下:

$$x^d \bmod f(x) = c_{0,0} + c_{0,1}x + \dots + c_{0,d-1}x^{d-1}$$

$$x^{d+1} \bmod f(x) = c_{1,0} + c_{1,1}x + \cdots + c_{1,d-1}x^{d-1}$$

.....

$$x^{2d-1} \bmod f(x) = c_{d-1,0} + c_{d-1,1}x + \cdots + c_{d-1,d-1}x^{d-1}$$

设 $A = A_0 + A_1 2^s + A_2 2^{2s} + \cdots + A_{2d-1} 2^{(2d-1)s} < p^2$, 则

$$\begin{aligned} A \bmod p &= (A_0 + A_1 2^s + A_2 2^{2s} + \cdots + A_{2d-1} 2^{(2d-1)s}) \bmod p \\ &= A_0 + A_1 2^s + \cdots + A_{d-1} 2^{(d-1)s} + A_d 2^{ds} \bmod p + \cdots + A_{2d-1} 2^{(2d-1)s} \bmod p \\ &= (A_0 + A_1 2^s + \cdots + A_{d-1} 2^{(d-1)s} + A_d(c_{0,0} + c_{0,1} 2^s + \cdots + c_{0,d-1} 2^{(d-1)s}) \\ &\quad + \cdots + A_{2d-1}(c_{d-1,0} + c_{d-1,1} 2^s + \cdots + c_{d-1,d-1} 2^{(d-1)s})) \bmod p \end{aligned} \quad (3-14)$$

观察(3-14), 可将其右端 $2^0, 2^s, \dots, 2^{(d-1)s}$ 的系数表示为矩阵 $C = (c_{ij})$:

$$C = \begin{pmatrix} \begin{matrix} 2^0 & 2^s & 2^{2s} & \cdots & 2^{s(d-1)} \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{matrix} \\ \begin{matrix} A_0 & A_1 & A_2 & \cdots & A_{d-1} \\ A_d c_{00} & A_d c_{01} & A_d c_{02} & \cdots & A_d c_{0,d-1} \\ A_{d+1} c_{10} & A_{d+1} c_{11} & A_{d+1} c_{12} & \cdots & A_{d+1} c_{1,d-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ A_{2d-1} c_{d-1,0} & A_{2d-1} c_{d-1,1} & A_{2d-1} c_{d-1,2} & \cdots & A_{2d-1} c_{d-1,d-1} \end{matrix} \end{pmatrix}$$

于是, 根据矩阵 C , 容易得到如下结论。

引理 3.1^[62] 设 $p=f(2^s)$, 正整数 $A < p^2$, $c_{i,j}(i, j=0, \dots, d-1)$ 的定义如上所述,

令

$$y_j = \sum_{i, c_{i,j} > 0} c_{i,j}, \quad z_j = \sum_{i, c_{i,j} < 0} -c_{i,j},$$

$$y_{\max} = \max\{y_j | 0 \leq j \leq d-1\}, \quad z_{\max} = \max\{z_j | 0 \leq j \leq d-1\}, \quad (3-15)$$

则 $A \bmod p$ 运算可转换成做 y_{\max} 次模加法运算和 z_{\max} 次模减法运算。

引理 3.1 虽然给出了 $A \bmod p$ 运算可转换成模加法和模减法次数的公式, 但公式(3-15)并没有揭示出模加法和模减法次数与多项式 $f(x)$ 的系数之间的关系, 为此, 我们将在下面两个小节里就首一三项式和首一五项式分别给出 $A \bmod p$ 运算需要的模加法和模减法的次数与多项式系数之间的关系。

3.3.3.2 首一三项式的取模运算

设 $f(x) = x^d + ax^l + b$, $p = f(2^s)$, 即 $p = f(2^s) = 2^{sd} + a2^{sl} + b$ 。又设 $A = A_0 + A_1 2^s + A_2 2^{2s} + \dots + A_{2d-1} 2^{(2d-1)s} < p^2$, 则 $A \bmod p$ 的结果与 (3-14) 相同, 此处略。于是有如下的定理。

定理 3.2 设 $f(x) = x^d + ax^l + b, a, b \in \mathbb{Z}$, “Num” 表示计算 $A \bmod p$ 所需要的模加法(模减法)的次数, 则我们有如下的结论(此处不考虑加法和减法的区别):

(1) 当 $d \geq 2l$ 时,

(i) 如果 $b \in \mathbb{Z}^-$, $\forall a \in \mathbb{Z}$, 则

$$\text{Num} = |ab| - b + a^2 \quad (3-16)$$

(ii) 如果 $b \in \mathbb{Z}^+$ 并且 $a \in \mathbb{Z}^+$, 则

$$\text{Num} = b + a + ab \quad (b \geq a) \quad (3-17)$$

$$\text{Num} = b + a + a^2 \quad (a > b) \quad (3-18)$$

(iii) 如果 $b \in \mathbb{Z}^+$ 并且 $a \in \mathbb{Z}^-$, 则

$$\text{Num} = a^2 - a - ab + b \quad (3-19)$$

(2) 当 $d < 2l$ 时,

(i) 如果 $b \in \mathbb{Z}^-$, $\forall a \in \mathbb{Z}$, 则

$$\text{Num} = |a^{i+1}| - b - b \sum_{j=1}^i |a^j| \quad (3-20)$$

(ii) 如果 $b \in \mathbb{Z}^+$, $\forall a \in \mathbb{Z}$, 则

$$\text{Num} = b + b \sum_{j=1}^i |a^j| + \sum_{j=1}^{i+1} |a^j| \quad (3-21)$$

其中 $i = \lfloor d/(d-l) \rfloor$ 。

证明 首先, 讨论 $d > 2l$ 的情形。

由以多项式为模数的取模运算定义知:

$$x^{d+t} \bmod (x^d + ax^l + b) = \begin{cases} -ax^{l+t} - bx^t, & t = 0, 1, \dots, (d-l-1) \\ -bx^t + a^2 x^{2l+t-d} + abx^{l+t-d}, & t = (d-l), \dots, (d-1) \end{cases} \quad (3-22)$$

利用式(3-22), 可以得出如图 3-1 所示的矩阵, 其中行元素依次为 $(x^d \bmod f(x), x^{d+1} \bmod f(x), \dots, x^{2d-1} \bmod f(x))$, 列元素依次为 $(x^0, x^1, \dots, x^{d-1})$ 。

将正整数 $A(<p^2)$ 表示为:

$$A = (A_0 \parallel A_1 \parallel A_2 \parallel A_3 \parallel \dots \parallel A_d \parallel A_{d+1} \parallel \dots \parallel A_{2d-2} \parallel A_{2d-1})$$

根据图 3-1, 令

$$T = (A_0 \parallel A_1 \parallel A_2 \parallel A_3 \parallel \dots \parallel A_{d-2} \parallel A_{d-1})$$

$$S_1 = (A_d \parallel A_{d+1} \parallel \dots \parallel A_{2d-2} \parallel A_{2d-1})$$

$$S_2 = (0 \parallel 0 \parallel \dots \parallel 0 \parallel A_{d+l} \parallel A_{d+l+1} \parallel \dots \parallel A_{2d-2} \parallel A_{2d-1})$$

$$S_3 = (0 \parallel 0 \parallel \cdots \parallel 0 \parallel A_{d+l} \parallel A_{d+l+1} \parallel \cdots \parallel A_{d+2l-1} \parallel 0 \parallel \cdots \parallel 0),$$

$$S_4 = (A_d \parallel A_{d+1} \parallel \cdots \parallel A_{d+l-1} \parallel 0 \parallel \cdots \parallel 0),$$

则

$$A \bmod p = (T + |-b| S_1 + |-a| S_2 + a^2 S_3 + |ab| S_4) \bmod p. \quad (3-23)$$

$$\begin{pmatrix} x^d \\ x^{d+1} \\ x^{d+2} \\ x^{d+3} \\ \cdots \\ x^{2d-l-1} \\ x^{2d-l} \\ \cdots \\ x^{2d-1} \end{pmatrix} \bmod f(x) = \begin{pmatrix} -b & \cdots & -a & \cdots & \cdots & 0 \\ & -b & \cdots & -a & \cdots & 0 \\ & & -b & \cdots & -a & \cdots & 0 \\ & & & -b & \cdots & -a & \cdots & 0 \\ & & & & \cdots & \cdots & \cdots & \cdots \\ & & & & & -b & \cdots & -a \\ ab & \cdots & a^2 & \cdots & \cdots & -b & \cdots & 0 \\ & & & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & ab & \cdots & a^2 & \cdots & \cdots & -b \end{pmatrix}_{(d \times d)}$$

$\begin{matrix} \uparrow & & \uparrow & & \uparrow \\ (l-1)th\ C & & (2l-1)th\ C & & (d-1)th\ C \end{matrix}$

图 3-1 利用式(3-22)得到的矩阵

根据(3-23)，我们可以得出下面的结论：

(i) 当 $b \in Z^-$ 时，对 $\forall a \in Z$ ， $-a$ 和 ab 总是同号，因此 S_2 和 S_4 二者可以拼接在一起。此时

$$A \bmod p = (T + |-b| S_1 + |-a| S_2 + a^2 S_3 + |ab - a| S_4) \bmod p$$

从而，计算 $A \bmod p$ 的加法次数为：

$$Num = |-b| + |-a| + a^2 + |ab - a| = |ab| - b + a^2$$

即，公式(3-16)成立。

(ii) 当 $b \in Z^+$ ，并且 $a \in Z^+$ 时， a^2 和 ab 同号，因此 S_3 和 S_4 可以拼接在一起。此时

$$A \bmod p = (T + |-b| S_1 + |-a| S_2 + a^2 S_3 + |ab - a^2| S_4) \bmod p \quad (b \geq a)$$

或者
$$A \bmod p = (T + |-b| S_1 + |-a| S_2 + (a^2 - ab) S_3 + ab S_4) \bmod p \quad (a > b)$$

从而, 计算 $A \bmod p$ 的加法次数为:

$$Num = |-b| + |-a| + a^2 + |ab - a^2| = b + a + ab \quad (b \geq a)$$

或者

$$Num = |-b| + |-a| + |a^2 - ab| + ab = b + a + a^2 \quad (b < a)$$

即, 公式(3-17)和(3-18)成立。

(iii) 当 $b \in Z^+$, $a \in Z^-$ 时, $-a \in Z^+$ 并且 $a^2 \in Z^+$, 但是 $ab \in Z^-$, 因此 S_4 既不能和 S_3 拼接在一起, 也不能和 S_2 拼接在一起。此时

$$A \bmod p = (T + |-b|S_1 + |-a|S_2 + a^2S_3 + |ab|S_4) \bmod p$$

从而, 计算 $A \bmod p$ 的加法次数为:

$$Num = b + |a| + a^2 + |ab|$$

即, 公式(3-19)成立。

其次, 我们讨论 $d = 2l$ 时的情形。

同样地, 根据以多项式为模数的取模运算定义知

$$x^{d+t} \bmod (x^d + ax^l + b) = \begin{cases} -a_{l,j+t}x^{l+t} - bx^t, & t = 0, 1, \dots, (l-1) \\ (a^2 - b)x^t + abx^{t-l}, & t = l, \dots, (2l-1) \end{cases} \quad (3-24)$$

根据式(3-24), 可以得出如图 3-2 所示的矩阵如下:

$$\begin{pmatrix} x^d \\ x^{d+1} \\ \vdots \\ x^{d+l-1} \\ x^{d+l} \\ \vdots \\ x^{2d-1} \end{pmatrix} \bmod f(x) = \begin{pmatrix} -b & \dots & & -a & & \dots & 0 \\ & -b & \dots & & -a & \dots & 0 \\ & \dots & \dots & \dots & \dots & \dots & \dots \\ & & -b & \dots & & \dots & -a \\ ab & \dots & & \dots & a^2 - b & \dots & 0 \\ & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & ab & \dots & & \dots & a^2 - b \end{pmatrix}$$

\uparrow $(l-1)th \ C$ \uparrow $(d-1)th \ C$

图 3-2 利用式(3-24)得到的矩阵

观察图 3-1 和 3-2, 容易看出, $d = 2l$ 与 $d > 2l$ 对应矩阵的不同之处在于: 图

3-1 中 a^2 所在斜线上的元素与主对角线上的元素 $(-b)$ 重合, 从而所得矩阵中从第 l 行开始, 主对角线上的元素变为 $a^2 - b$ 。但是, 这并不影响取模所需要的加法次数。因此, 当 $d = 2l$ 时, 结论与 $d > 2l$ 时相同。

最后, 我们讨论 $d < 2l$ 的情形, 与前两种情况相比, 相对复杂一些。

设存在正整数 i , 使得 $d \leq li/(i-1)$ 。根据 d 和 l 的值, 我们有

$$d \leq li/(i-1) \Rightarrow d(i-1) \leq li \Rightarrow (d-l)i \leq d \Rightarrow i \leq d/(d-l)$$

根据上式, 取 $i = \lfloor d/(d-l) \rfloor$ 即可。

同样地, 由以多项式为模数的取模运算定义知:

$$x^{d+t} \bmod f(x) = \begin{cases} -ax^{t+l} - bx^t, & t = 0, \dots, (d-l-1) \\ a^2 x^{2l-d+t} - bx^t + abx^{t+l-d}, & t = (d-l), \dots, 2(d-l)-1 \\ -a^3 x^{3l-2d+t} - bx^t + abx^{t+l-d} - a^2 bx^{t+2l-2d}, & t = 2(d-l), \dots, 3(d-l)-1 \\ \dots & \dots \\ (-1)^{(i+1)} a^{(i+1)} x^{(i+1)l-id+t} + b \sum_{j=0}^i (-1)^{j+1} a^j x^{t+j(l-d)}, & t = i(d-l), \dots, d-1. \end{cases} \quad (3-25)$$

利用式(3-25), 可以得出对应矩阵如图 3-3。

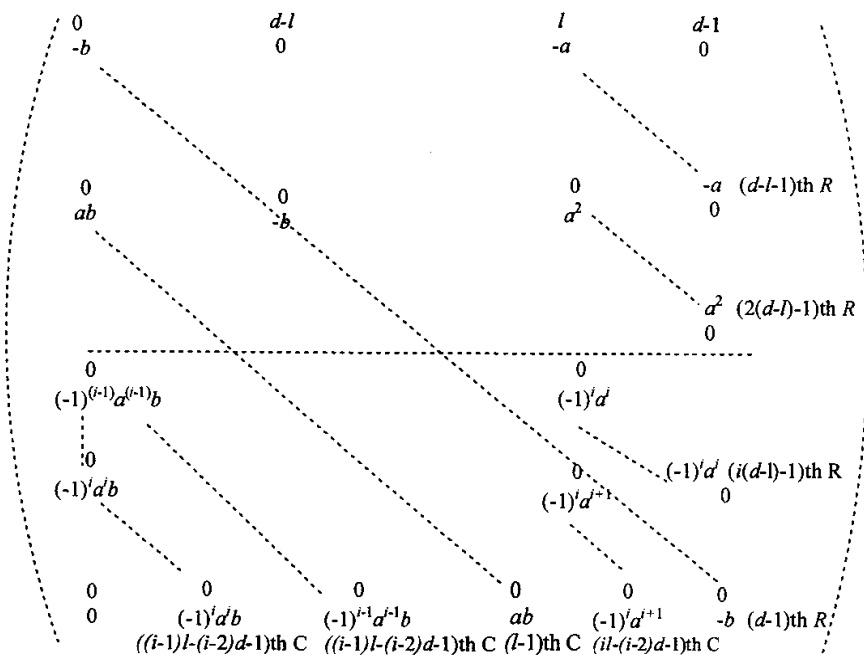


图 3-3 利用式 (3-25)得到的矩阵

下面我们讨论各矩阵元素在矩阵中的放置。

由下取整定义知, $i(d-l) = \lfloor d/(d-l) \rfloor (d-l) \leq (d/(d-l))(d-l) = d$, 从而

$$(i-1)(d-l) = i(d-l) - (d-l) \leq d - (d-l) = l$$

另外, 我们有

$$l+l/i = l+l/\lfloor d/(d-l) \rfloor \leq l+l/(d/(d-l)-1) = l+(d-l) = d$$

于是, 有

$$(i-1)(d-l) \leq l \leq i(d-l)。$$

根据上式, 从第一行到第 $i(d-l)-1$ 行, 均有 $-b$ 一定在 $(-1)^j a^j (j=1, \dots, i)$ 的左边, 但是从第 $i(d-l)$ 行到第 $(d-1)$ 行, $-b$ 一定在 $(-1)^{i+1} a^{i+1}$ 的右边。当 $(d-l)|d$ 时, $-b$ 和 $(-1)^j a^j (j=1, \dots, i+1)$ 重合, 均在主对角线上, 此时, 主对角线上的元素为 $(-b + (-1)^j a^j) (j=1, \dots, i+1)$, 具体见图 3-3。

同样地, 将任意小于 p^2 的正整数 A 表示为:

$$A = (A_0 \| A_1 \| A_2 \| A_3 \| \dots \| A_d \| A_{d+1} \| \dots \| A_{2d-2} \| A_{2d-1})$$

则根据图 3-3, 令

$$\begin{aligned} T &= (A_0 \| A_1 \| A_2 \| A_3 \| \dots \| A_{d-2} \| A_{d-1}) \\ S_0 &= (A_d \| A_{d+1} \| A_{d+2} \| A_{d+3} \| \dots \| A_{2d-1}) \\ S_j &= (A_d \| A_{d+1} \| \dots \| A_{d+jl-(j-1)d-1} \| 0 \| \dots \| 0), j=1, \dots, i \\ D_j &= (0 \| 0 \| \dots \| 0 \| A_{d+l} \| A_{d+l+1} \| \dots \| A_{2d-2} \| A_{2d-1}), j=1, 2, \dots, i \\ D_{i+1} &= (0 \| 0 \| \dots \| 0 \| A_{d+l} \| A_{d+l+1} \| \dots \| A_{d+il-(i-2)d-1}) \end{aligned}$$

从而,

$$A \bmod p = (T + |-b| S_0 + \sum_{j=1}^i |(-1)^{j+1} a^j b| S_j + \sum_{j=1}^{i+1} |(-1)^j a^j| D_j) \quad (3-26)$$

根据(3-26), 我们可以得出下面的结论:

(i) 如果 $b \in \mathbb{Z}$, 则对 $\forall a \in \mathbb{Z}$, $(-1)^j a^j$ 和 $(-1)^{j+1} a^j b$ ($j=1, \dots, i$) 总是同号, 因此 S_j 和 D_j 二者恰好可以拼接在一起。此时计算 $A \bmod p$ 的加法次数为:

$$Num = |b| + \sum_{j=1}^i |a^j b| + |a^{i+1}|$$

即, 公式(3-20)成立。

(ii) 如果 $b \in \mathbb{Z}^+$, 则对 $\forall a \in \mathbb{Z}$, $(-1)^j a^j$ 和 $(-1)^{j+1} a^j b$ 总是异号, 因此 S_j 和 D_j 不能拼接在一起。此时计算 $A \bmod p$ 的加法次数为:

$$Num = |b| + \sum_{j=1}^i |a^j b| + \sum_{j=1}^{i+1} |a^j|$$

即，公式(3-21)成立。

综上所述，定理 3.2 得证。

在 ECC 中，通常用到的多项式各项的系数均为 1，因此就 a, b 的绝对值均为 1 的情形，我们给出如下的推论。

推论 3.3 设 $f(x) = x^d + ax^l + b, a, b \in Z$ ，“Num”表示计算 $A \bmod p$ 所需要的模加法(模减法)的次数，则我们有如下的结论(此处不考虑加法和减法的区别)：

- (1) 当 $d \geq 2l$ 时，
 - (i) 如果 $b = -1, a = \pm 1$ ，则 $Num = 3$ ；
 - (ii) 如果 $b = 1, a = 1$ ，则 $Num = 3$ ；
 - (iii) 如果 $b = 1, a = -1$ ，则 $Num = 4$ ；
- (2) 当 $d < 2l$ 时，
 - (i) 如果 $b = -1, a = \pm 1$ ，则 $Num = i + 2$ ；
 - (ii) 如果 $b = 1, a = \pm 1$ ，则 $Num = 2(i + 1)$ 。

由定理 3.2，可直接得出推论 3.3 的结论，证明略。

表 3-2 定理 3.2 的相关实例

	$p-192$	$p-224$	$p-448$	$p-480$	$p-512$	$p-544$
$f(x)$	$x^6 - x^2 - 1$	$x^7 - x^3 + 1$	$x^{14} - x^7 - 1$	$x^{15} - 2x^4 - 1$	$x^{16} - x - 1$	$x^{17} - x + 1$
P	$2^{192} - 2^{64} - 1$	$2^{224} - 2^{96} + 1$	$2^{448} - 2^{224} - 1$	$2^{480} - 2^{128} - 1$	$2^{512} - 2^{32} - 1$	$2^{544} - 2^{32} - 1$
Num	3	4	3	7	3	4

表 3-2 给出了实际例子进一步验证我们的结论，其中 $p-192, p-224$ 是 FIPS 186-2 标准推荐的两种素数域。

显然，表 3-2 的结论与文献[3]和[62]中的运算结果一致。

3.3.3.3 首一五项式的取模运算

对首一五项式 $f(x) = x^d + ax^m + bx^l + cx^k + e (d > m > l > k)$ ，仅考虑 $d > 2m$ ， $m > 2l$ ， $l > 2k$ 这种较简单的情形，对应得到下面的定理。

定理 3.4 假设 $f(x) = x^d + ax^m + bx^l + cx^k + e (d > m > l > k)$ ，“Num”表示计算 $A \bmod p$ 所需要的模加法(模减法)的次数，则我们有如下的结论(此处不考虑加法和减法的区别)：

- (1) 如果 $e \in Z^-$ ，并且 a, b, c 同号，则

$$\begin{aligned} \text{Num} &= -e + |a+b+c|(|c|-e) + bc + b^2 & (|c| \geq |b| \geq |a|) \\ \text{Num} &= -e + |a+b+c|(|a|-e) + ac + ab & (|a| \geq |b| \geq |c|) \end{aligned} \quad (3-27)$$

(2) 如果 $e \in Z^-$ ，并且 a 仅与 b, c 之一异号，则

$$\begin{aligned} \text{Num} &= -e + (|a|+|b|+|c|)(|b|+|c|-e) & (|c| \geq |b| \geq |a|) \\ \text{Num} &= -e + (|a|-e)(|a|+|b|+|c|) + |ac| + |ab| + |bc| & (|a| \geq |b| \geq |c|) \end{aligned} \quad (3-28)$$

(3) 如果 $e \in Z^-$ ，并且 a 与 b, c 异号，则

$$\begin{aligned} \text{Num} &= -e + |a-b-c|(|b|+|c|-e) - a(c+b) & (|b| \geq |a|) \\ \text{Num} &= -e + |a-b-c|(|a|+|c|-e) - ab + bc & (|a| \geq |b|) \end{aligned} \quad (3-29)$$

(4) 如果 $e \in Z^+$ ，并且 a, b, c 同号，则

$$\begin{aligned} \text{Num} &= e + |a+b+c| \cdot (1+|c|+e) + b^2 + bc & (|c| \geq |b| \geq |a|) \\ \text{Num} &= e + |a+b+c| \cdot (1+|a|+e) + ac + ab & (|a| \geq |b| \geq |c|) \end{aligned} \quad (3-30)$$

(5) 如果 $e \in Z^+$ ，并且 a 仅与 b, c 之一异号，则

$$\begin{aligned} \text{Num} &= e + (|a|+|b|+|c|)(1+|b|+|c|+e) & (|c| \geq |b| \geq |a|) \\ \text{Num} &= e + (|a|+|b|+|c|)(1+|a|+e) + |ac| + |ab| + |bc| & (|a| \geq |b| \geq |c|) \end{aligned} \quad (3-31)$$

(6) 如果 $e \in Z^+$ ，并且 b, c 与 a 异号，则

$$\begin{aligned} \text{Num} &= e + (|c|+|b|+|a|)(1+|b|+|c|+e) + |ac| + |ab| & (|b| \geq |a|) \\ \text{Num} &= e + (|c|+|b|+|a|)(1+|a|+|c|+e) + |ab| + |bc| & (|a| \geq |b|) \end{aligned} \quad (3-32)$$

证明：由以多项式为模数的取模运算定义知：

$$x^{d+l} \bmod f(x) = \begin{cases} -ax^{m+l} - bx^{l+l} - cx^{k+l} - ex', & t=0,1,\dots,(d-m-1) \\ -bx^{l+l} - cx^{k+l} - ex' + a^2x^{2m+l-d} \\ + abx^{m+l+l-d} + acx^{m+k+l-d} + aex^{m+l-d} & t=(d-m),\dots,(d-l-1) \\ -cx^{k+l} - ex' + a^2x^{2m+l-d} + 2abx^{m+l+l-d} + acx^{m+k+l-d} + aex^{m+l-d} \\ + b^2x^{2l+l-d} + bcx^{l+k+l-d} + bex^{l+l-d}, & t=(d-l),\dots,(d-k-1) \\ -ex' + a^2x^{2m+l-d} + 2abx^{m+l+l-d} + 2acx^{m+k+l-d} + aex^{m+l-d} + b^2x^{2l+l-d} \\ + 2bcx^{l+k+l-d} + bex^{l+l-d} + c^2x^{2k+l-d} + cex^{k+l-d}, & t=(d-k),\dots,(d-1) \end{cases} \quad (3-33)$$

利用式(3-33)，可以得出对应矩阵如图 3-4 或者 3-5。其中对应行元素依次为 $(x^{d+l} \bmod f(x), x^{d+l-1} \bmod f(x), \dots, x^{d-l} \bmod f(x))$ ，列元素依次对应 (x^0, \dots, x^{d-1}) 。

首先说明这两个矩阵都是正确的。

利用已知不等式 $d > m > l > k$ ， $d > 2m$ ， $m > 2l$ ， $l > 2k$ ，有

$$k-1 < 2k-1 < l-1 < l+k-1 < 2l-1 < m-1 < m+k-1 < m+l-1 < 2m-1 < d-1 \quad (3-34)$$

该不等式组对应着图 3-4 和图 3-5 最后一行各元素的位置关系。

$$\begin{array}{cccccccccccccccccccc}
 0 & k & l-k & l & \underline{2l-k} & \underline{m-l} & m+k-l & m-k & m & m+l-k & 2m-l & \underline{2m-k} & \underline{d-m} & d-m+k & d-l & d-m+l & d-l+k & d-k & d-1 \\
 \downarrow & \downarrow & & \downarrow & & & & & \downarrow & & & & & & & & & & \\
 \begin{pmatrix}
 -e & -c & & -b & & & & & & -a & & & & & & & & & \\
 ae & ac & & ab & & & & & a^2 & & & & -e & -c & & -b & & & -a \\
 & & & & & & & & & & & & & & & & & 0 \\
 be & bc & & b^2 & \underline{ae} & ac & & 2ab & a^2 & & & & & & -e & -c & & -b & -a \\
 & & & & & & & & & & & & & & & & & 0 \\
 ce & c^2 & be & 2bc & \underline{b^2} & & ae & 2ac & 2ab & & \underline{a^2} & & & & & & -e & -c & -b \\
 & & & & & & & & & & & & & & & & & 0 \\
 0 & ce & c^2 & be & 2bc & b^2 & & ae & 2ac & 2ab & & a^2 & & & & & & -e & -a \\
 \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
 k-1 & 2k-1 & l-1 & & l+k-1 & 2l-1 & & m-1 & m+k-1 & m+l-1 & & 2m-1 & & & & & & d-1 & d-1
 \end{pmatrix}
 \end{array}$$

图 3-4 利用式(3-33)得到的矩阵

$$\begin{array}{cccccccccccccccccccc}
 0 & k & l-k & l & \underline{m-l} & \underline{2l-k} & m+k-l & m-k & m & m+l-k & 2m-l & \underline{d-m} & \underline{2m-k} & d-m+k & d-l & d-m+l & d-l+k & d-k & d-1 \\
 \downarrow & \downarrow & & \downarrow & & & & & \downarrow & & & & & & & & & & \\
 \begin{pmatrix}
 -e & -c & & -b & & & & & & -a & & & & & & & & & \\
 ae & ac & & ab & & & & & a^2 & & & & -e & -c & & -b & & & -a \\
 & & & & & & & & & & & & & & & & & 0 \\
 be & bc & & b^2 & \underline{ae} & ac & & 2ab & a^2 & & & & & & -e & -c & & -b & -a \\
 & & & & & & & & & & & & & & & & & 0 \\
 ce & c^2 & be & 2bc & & \underline{b^2} & ae & 2ac & 2ab & & \underline{a^2} & & & & & & -e & -c & -b \\
 & & & & & & & & & & & & & & & & & 0 \\
 0 & ce & c^2 & be & 2bc & b^2 & & ae & 2ac & 2ab & & a^2 & & & & & & -e & -a \\
 \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
 k-1 & 2k-1 & l-1 & & l+k-1 & 2l-1 & & m-1 & m+k-1 & m+l-1 & & 2m-1 & & & & & & d-1 & d-1
 \end{pmatrix}
 \end{array}$$

图 3-5 利用式(3-33)得到的矩阵

如果再添加条件 $m > 3l - k$ 和 $d > 3m - k$, 则有

$$\begin{aligned}
 0 &< k < l - k < l < \underline{2l - k} < \underline{m - l} < m + k - l < m - k < m < m + l - k < 2m - l \\
 &< \underline{2m - k} < \underline{d - m} < d - m + k < d - m + l < d - l < d - l + k < d - k < d - 1
 \end{aligned} \quad (3-35)$$

此时, 式(3-35)对应着图 3-4。

反过来, 如果添加条件 $m < 3l - k$ 和 $d < 3m - k$, 则有

$$\begin{aligned} 0 < k < l - k < l < m - l < 2l - k < m + k - l < m - k < m \\ < m + l - k < 2m - l < \underline{d - m} < \underline{2m - k} < d - m + k \\ < d - m + l < d - l < d - l + k < d - k < d - l \end{aligned} \quad (3-36)$$

此时, 式(3-36)对应着图 3-5。

从图 3-4 和图 3-5 可以看出, 是否增加 $m < 3l - k$ 和 $d < 3m - k$ 或者 $m > 3l - k$ 和 $d > 3m - k$ 并不影响将要证明的结论。因此, 我们将图 3-4 和图 3-5 简化为同一个矩阵图, 见图 3-6。

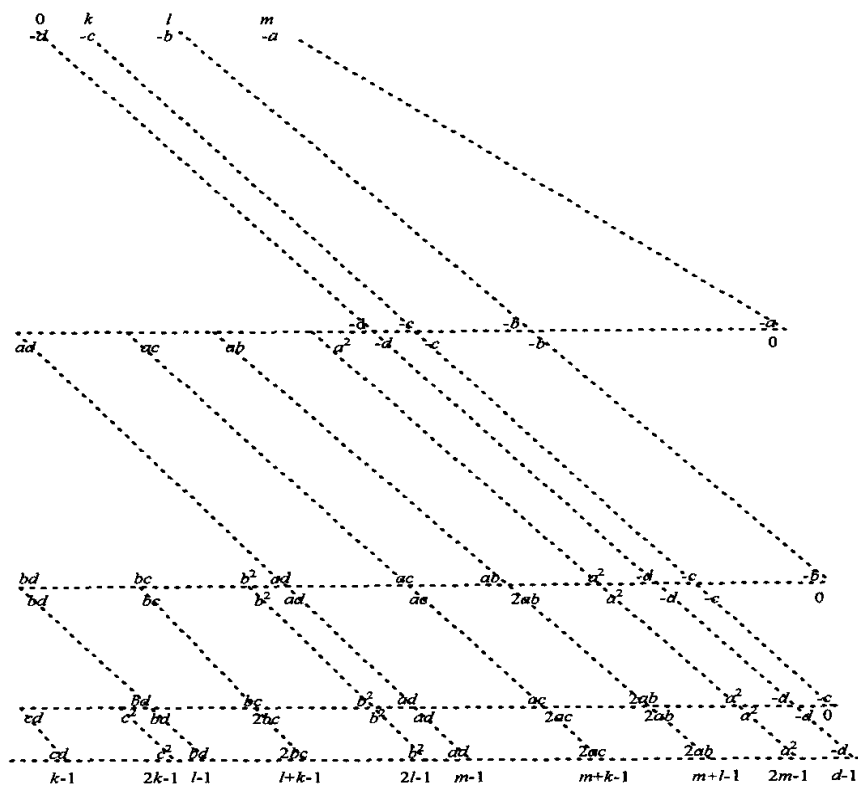


图 3-6 图 3-4 和图 3-5 的简化图

于是, 将任意小于 p^2 的正整数 A 表示为:

$$A = (A_0 \parallel A_1 \parallel A_2 \parallel A_3 \parallel \cdots \parallel A_d \parallel A_{d+1} \parallel \cdots \parallel A_{2d-2} \parallel A_{2d-1})$$

则根据图 3-6, 令

$$\begin{aligned}
 T &= (A_0 \| A_1 \| A_2 \| A_3 \| \cdots \| A_{d-2} \| A_{d-1}) \\
 S_1 &= (A_d \| A_{d+1} \| \cdots \| A_{2d-2} \| A_{2d-1}) \\
 S_2 &= (0 \| 0 \| \cdots \| 0 \| A_{d+k} \| A_{d+k+1} \| \cdots \| A_{2d-2} \| A_{2d-1}) \\
 S_3 &= (0 \| 0 \| \cdots \| 0 \| A_{d+l} \| A_{d+l+1} \| \cdots \| A_{2d-2} \| A_{2d-1}) \\
 S_4 &= (0 \| 0 \| \cdots \| 0 \| A_{d+m} \| A_{d+m+1} \| \cdots \| A_{2d-2} \| A_{2d-1}) \\
 S_5 &= (A_d \| A_{d+k+1} \| \cdots \| A_{d+m-1} \| 0 \| \cdots \| 0) \\
 S_6 &= (0 \| \cdots \| 0 \| A_{d+k} \| A_{d+k+1} \| \cdots \| A_{d+m+k-1} \| 0 \| \cdots \| 0) \\
 S_7 &= (0 \| 0 \| \cdots \| 0 \| A_{d+m} \| A_{d+m+1} \| \cdots \| A_{d+m+k-1} \| 0 \| \cdots \| 0) \\
 S_8 &= (0 \| \cdots \| 0 \| A_{d+l} \| \cdots \| A_{d+m+l-1} \| 0 \| \cdots \| 0) \\
 S_9 &= (0 \| \cdots \| 0 \| A_{d+m} \| \cdots \| A_{d+m+l-1} \| 0 \| \cdots \| 0) \\
 S_{10} &= (0 \| \cdots \| 0 \| A_{d+m} \| \cdots \| A_{d+2m-1} \| 0 \| \cdots \| 0) \\
 S_{11} &= (A_d \| \cdots \| A_{d+l-1} \| 0 \| \cdots \| 0) \\
 S_{12} &= (0 \| \cdots \| 0 \| A_{d+k} \| \cdots \| A_{d+l+k-1} \| 0 \| \cdots \| 0) \\
 S_{13} &= (0 \| \cdots \| 0 \| A_{d+l} \| \cdots \| A_{d+l+k-1} \| 0 \| \cdots \| 0) \\
 S_{14} &= (0 \| \cdots \| 0 \| A_{d+l} \| \cdots \| A_{d+2l-1} \| 0 \| \cdots \| 0) \\
 S_{15} &= (A_d \| \cdots \| A_{d+k-1} \| 0 \| \cdots \| 0) \\
 S_{16} &= (0 \| \cdots \| 0 \| A_{d+k} \| \cdots \| A_{d+2k-1} \| 0 \| \cdots \| 0)
 \end{aligned}$$

从而,

$$\begin{aligned}
 A \bmod p &= T + |-e|S_1 + |-c|S_2 + |-b|S_3 + |-a|S_4 + |ae|S_5 \\
 &\quad + |ac|S_6 + |ac|S_7 + |ab|S_8 + |ab|S_9 + |a^2|S_{10} + |be|S_{11} \\
 &\quad + |bc|S_{12} + |bc|S_{13} + |b^2|S_{14} + |ce|S_{15} + |c^2|S_{16} \bmod p
 \end{aligned} \tag{3-37}$$

根据(3-37), 我们可以得到下面的结论:

当 $e \in Z^-$ 时, 对 $\forall a, b, c \in Z$, $-a$ 和 ae 、 $-b$ 和 be 、 $-c$ 和 ce 总是分别同号, 因此 S_4 和 S_5 、 S_3 和 S_{11} 、 S_2 和 S_{15} 分别可以拼接在一起。

另外, 因为

$$\begin{aligned} d+k-1 &< d+k < d+2k-1 < d+l-1 < d+l < d+l+k-1 \\ &< d+2l-1 < d+m-1 < d+m < d+m+k-1 < d+m+l-1 < d+d-1 \end{aligned}$$

所以

(1) 当 a, b, c 同号时, c^2S_{16} 和 abS_8, bcS_{12} 和 acS_7, bcS_{13} 和 abS_9, b^2S_{14} 和 a^2S_{10} 分别可以拼接在一起。于是, 式(3-37)可以简化为:

$$\begin{aligned} A \bmod p = & T + |-e|S_1 + |ae|S_5 + |ac|S_6 + |be|S_{11} + |bc|S_{12} \\ & + |bc|S_{13} + |b^2|S_{14} + |ce|S_{15} + |c^2|S_{16} \bmod p \quad (|c| \geq |b| \geq |a|) \end{aligned}$$

或者

$$\begin{aligned} A \bmod p = & T + |-e|S_1 + |ae|S_5 + |ac|S_6 + |ac|S_7 + |ab|S_8 \\ & + |ab|S_9 + |a^2|S_{10} + |be|S_{11} + |ce|S_{15} \bmod p \quad (|a| \geq |b| \geq |c|) \end{aligned}$$

从而, 计算 $A \bmod p$ 的加法次数为:

$$\begin{aligned} Num = & -e + |a+b+c|(|c|-e) + bc + b^2 \quad (|c| \geq |b| \geq |a|) \\ Num = & -e + |a+b+c|(|a|-e) + ac + ab \quad (|a| \geq |b| \geq |c|) \end{aligned}$$

即, 公式(3-27)成立。

(2) i) 当 $e \in \mathbb{Z}^-$, a, b 与 c 异号时, c^2S_{16} 和 abS_8, bcS_{12} 和 acS_7, b^2S_{14} 和 a^2S_{10} 分别可以拼接在一起, 但 bcS_{13} 和 abS_9 不能拼接在一起, 此时, 根据(3-37), 计算 $A \bmod p$ 的加法次数为:

$$\begin{aligned} Num = & |e| + |ae| + |ac| + |ab| + |be| + 2|bc| + |b^2| + |ce| + |c^2| \quad (|c| \geq |b| \geq |a|) \\ Num = & |e| + |ae| + 2|ac| + 2|ab| + |a^2| + |be| + |bc| + |ce| \quad (|a| \geq |b| \geq |c|) \end{aligned}$$

ii) 当 $e \in \mathbb{Z}^-$, a, c 与 b 异号时, 仅有 c^2S_{16} 和 acS_7, bcS_{13} 和 abS_9, b^2S_{14} 和 a^2S_{10} 分别可以拼接在一起。于是根据(3-37), 计算 $A \bmod p$ 的加法次数为:

$$\begin{aligned} Num = & |e| + |ae| + |ac| + |ab| + |be| + 2|bc| + |b^2| + |ce| + |c^2| \quad (|c| \geq |b| \geq |a|) \\ Num = & |e| + |ae| + 2|ac| + 2|ab| + |a^2| + |be| + |bc| + |ce| \quad (|a| \geq |b| \geq |c|) \end{aligned}$$

显然, 上面的两组等式完全相同, 于是, 化简上面的式子, 即得公式(3-28)。

(3) 当 $e \in \mathbb{Z}^-$, b, c 与 a 异号时, 仅有 b^2S_{14} 和 a^2S_{10} 可以拼接在一起。于是根据(3-37), 计算 $A \bmod p$ 的加法次数为:

$$Num = |e| + |ae| + 2|ac| + 2|ab| + |be| + 2|bc| + |b^2| + |ce| + |c^2| \quad (|b| \geq |a|)$$

$$Num = |e| + |ae| + 2|ac| + 2|ab| + |a^2| + |be| + 2|bc| + |ce| + |c^2| \quad (|a| \geq |b|)$$

化简上面的式子, 即得公式(3-29)。

但是, 当 $e \in Z^+$ 时, 对 $\forall a, b, c \in Z$, $-a$ 和 ae 、 $-b$ 和 be 、 $-c$ 和 ce 总是分别异号, 此时 S_4 和 S_5 、 S_3 和 S_{11} 、 S_2 和 S_{15} 都不能拼接在一起。于是, 我们有

(4) 当 $e \in Z^+$, a, b, c 同号时, c^2S_{16} 和 abS_8 、 bcS_{12} 和 acS_7 、 bcS_{13} 和 abS_9 、 b^2S_{14} 和 a^2S_{10} 分别可以拼接在一起。于是根据(3-37), 计算 $A \bmod p$ 的加法次数为:

$$Num = |e| + |c| + |b| + |a| + |ae| + |ac| + |be| + 2|bc| + |b^2| + |ce| + |c^2| \quad (|c| \geq |b| \geq |a|)$$

$$Num = |e| + |c| + |b| + |a| + |ae| + 2|ac| + 2|ab| + |a^2| + |be| + |ce| \quad (|a| \geq |b| \geq |c|)$$

化简上面的式子, 即得公式(3-30)。

(5) i) 当 $e \in Z^+$, a, b 与 c 异号时, c^2S_{16} 和 abS_8 、 bcS_{12} 和 acS_7 、 b^2S_{14} 和 a^2S_{10} 分别可以拼接在一起, 但 bcS_{13} 和 abS_9 不能拼接在一起。于是根据(3-37), 计算 $A \bmod p$ 的加法次数为:

$$Num = |e| + |c| + |b| + |a| + |ae| + |ac| + |ab| + |be| + 2|bc| + |b^2| + |ce| + |c^2| \quad (|c| \geq |b| \geq |a|)$$

$$Num = |e| + |c| + |b| + |a| + |ae| + 2|ac| + 2|ab| + |a^2| + |be| + |bc| + |ce| \quad (|a| \geq |b| \geq |c|)$$

ii) 当 $e \in Z^+$, a, c 与 b 异号时, 仅有 c^2S_{16} 和 acS_7 、 bcS_{13} 和 abS_9 、 b^2S_{14} 和 a^2S_{10} 分别可以拼接在一起。于是根据(3-37), 计算 $A \bmod p$ 的加法次数为:

$$Num = |e| + |c| + |b| + |a| + |ae| + |ac| + |ab| + |be| + 2|bc| + |b^2| + |ce| + |c^2| \quad (|c| \geq |b| \geq |a|)$$

$$Num = |e| + |c| + |b| + |a| + |ae| + 2|ac| + 2|ab| + |a^2| + |be| + |bc| + |ce| \quad (|a| \geq |b| \geq |c|)$$

即, 上面两种情形有相同的结论, 化简上面两组公式即得式(3-31)。

(6) 当 $e \in Z^+$, b, c 与 a 异号时, 仅有 b^2S_{14} 和 a^2S_{10} 可以拼接在一起。于是根据(3-37), 计算 $A \bmod p$ 的加法次数为:

$$Num = |e| + |c| + |b| + |a| + |ae| + 2|ac| + 2|ab| + |be| + 2|bc| + |b^2| + |ce| + |c^2| \quad (|b| \geq |a|)$$

$$Num = |e| + |c| + |b| + |a| + |ae| + 2|ac| + 2|ab| + |a^2| + |be| + 2|bc| + |ce| + |c^2| \quad (|a| \geq |b|)$$

即, 公式(3-32)成立。

综上所述, 定理 3.4 得证, 证毕。★

注意, 当 a, b, c 中任意两个为 0 时, 首一五项式退化为首一三项式。如果将 0 看作与非零数 c (或 a, b) 同号时, 可以得到与定理 3.2 相同的结论。

例如对 $p=512$: $f(x)=x^{16}-x-1$, $p=2^{512}-2^{32}-1$, 令 $a=b=0$, $c=e=-1$, 则由定理 3.4 (1)知:

$$Num=1+|0+0-1|\times(1-(-1))=3$$

与前面的结论一致。

又例如对 $p=544$: $f(x)=x^{17}-x+1$, $p=2^{544}-2^{32}+1$, 令 $a=b=0$, $c=-1$, $e=1$, 则由定理 3.4 (4)知:

$$Num=1+|0+0-1|\times(1+1+1)+0+0=4$$

与前面的结论一致。

3.4 小结

本章主要研究素数域 $GF(p)$ 上的取模运算。我们首先分析了通用的经典取模算法, Barrett 取模算法和 Montgomery 乘法; 然后对 ECC 中经常用到的由不可约首一三项式和首一五项式产生的广义 Mersenne 素数的取模运算进行了深入研究, 得出仅与不可约首一三项式和五项式的系数有关的取模运算计算量公式, 并对公式的正确性进行了理论证明和实例验证。利用所得公式, 对任意给定的由不可约首一三项式和五项式的广义 Mersenne 数, 根据其对应的多项式系数, 可容易计算出 $A \bmod p$ 运算需要的模加法次数。

第四章 有限扩域上的乘法运算

本章主要讨论最优扩域(OEFs), 最优塔域(QTFs)和广义最优扩域(GOEFs)上的乘法运算, 其中广义最优扩域(GOEFs)上的乘法运算是本章讨论的重点。

在开始本章的内容之前, 首先对有限域 $GF(q)(q=p^m)$ 做如下假设: p 是大于 2 的素数, $m \geq 1$ 。

4.1 最优扩域(OEFs)

4.1.1 OEFs 的性质

最优扩域(Optimal Extension Fields, OEFs)分别由D.V.Bailey和C.Paar^[55]以及P.Mihăilescu^[86]在1998年和1997年独立提出的。具体定义如下:

定义 4.1 令 c 是一个正整数, 伪-Mersenne 素数是指形如

$$2^n \pm c, \text{ 其中 } \log_2 c \leq \lfloor 0.5n \rfloor$$

的素数。

定义 4.2 有限域 $GF(p^m)$ 是最优扩域(Optimal Extension Fields, OEFs), 如果满足:

- (1) p 是伪-Mersenne素数;
- (2) $GF(p)$ 上存在不可约多项式 $f(x) = x^m - w$ 。

定义 4.3 如果 $c=1$, 则称OEFs为I型OEFs; 如果 $w=2$, 则称OEFs为II型OEFs。

下面的表4-1是OEFs的一些实际例子。

引理 4.1 令 $f_1(x), f_2(x), \dots, f_N(x) \in GF(p)[x]$ 是次数为 m , 阶为 e 的不同的首一不可约多项式, $t \in \mathbb{Z}$ 且 $t \geq 2$ 。若满足:

- (1) 如果 r 是 t 的素因数, 则 $r | e$, 但 $r \nmid ((p^m - 1)/e)$;
- (2) 如果 $t \equiv 0 \pmod{4}$, 则 $p^m \equiv 1 \pmod{4}$ 。

则 $f_1(x^t), f_2(x^t), \dots, f_N(x^t) \in GF(p)[x]$ 是次数为 mt , 阶为 et 的首一不可约多项式。

引理 4.2 令 $f_1(x), f_2(x), \dots, f_N(x) \in GF(q)[x]$ 是次数为奇数 m , 阶为 e 的不同首一不可约多项式。令 $q = 2^a u - 1$, $t = 2^b v$ ($a, b \geq 2$), u, v 是奇数。如果 r 是 t 的素因数,

则 $r|e$, 但 $r \nmid (s^m - 1)/e$ 。令 $k = \min(a, b)$, 则每一个多项式 $f_j(x')$ 可以分解为 2^{k-1} 个次数为 mt^{1-k} 的首一不可约多项式 $g_{ij}(x) \in GF(q)[x]$ 的乘积, 这 $2^{k-1}N$ 个 $g_{ij}(x)$ 是 $GF(q)[x]$ 中不同的首一不可约多项式, 其次数均为 mt^{1-k} , 阶均为 et 。

表4-1 OEFs实例

P	f	参数	类型
$2^7 + 3$	$x^{13} - 5$	$n=7, c=-3, m=13, w=5$	-
$2^{13} - 1$	$x^{13} - 2$	$n=13, c=1, m=13, w=2$	I, II
$2^{31} - 19$	$x^6 - 2$	$n=31, c=19, m=6, w=2$	II
$2^{31} - 1$	$x^6 - 7$	$n=31, c=1, m=6, w=7$	I
$2^{32} - 5$	$x^5 - 2$	$n=32, c=5, m=5, w=2$	II
$2^{37} - 13$	$x^3 - 2$	$n=57, c=13, m=3, w=2$	II
$2^{61} - 1$	$x^3 - 37$	$n=61, c=1, m=3, w=37$	I
$2^{81} - 1$	$x^3 - 3$	$n=89, c=1, m=2, w=3$	I

引理4.3^[7] 令 $m \geq 2$ 是整数, $w \in GF(q)^*$, 则二项式 $f(x) = x^m - w$ 在 $GF(q)[x]$ 上不可约, 当且仅当下面的两个条件满足:

- (1) m 的每个素因数整除 w 在 $GF(q)^*$ 中的阶 e , 但不整除 $(q-1)/e$;
- (2) 如果 $m \equiv 0 \pmod{4}$, 则 $q \equiv 1 \pmod{4}$ 。

具体证明过程请参见参考文献[7]。

推论 4.4 如果 w 是 $GF(q)^*$ 的本原元(primitive), 且 $m|(p-1)$, 则二项式 $f(x) = x^m - w$ 在 $GF(q)[x]$ 上不可约。

4.1.2 OEFs 的性能优势

OEFs 是一簇具有特殊性质的有限域, 在其上能够利用软件有效执行域运算, 其主要动机在于使被选出的域参数与机器更匹配, 从而使得域运算操作能够更有效执行。特别地, 如果素数 p 恰好可由目标处理机的一个寄存器表示, 则这种 OEFs 上的域运算尤其有效^[56]。

Smart^[87] 分别对一般素域、广义 Mersenne 素域、特征为 2 的非素域和 OEFs 等四种域上的域运算性能进行了实验分析和比较后认为: 在 ECC 中, 无论是最终性能, 还是存储要求, OEFs 都具有明显优势。

4.1.3 OEFs 上的乘法运算^[88]

对任意 $A \in GF(p^m)$, 其在多项式基下的表示形式为

$$A = \sum_{i=0}^{m-1} a_i x^i = a_0 + a_1 x + \cdots + a_{m-1} x^{m-1}, a_i \in GF(p) \quad (4-1)$$

任意给定 $A, B \in GF(p^m)$, 设 $A = \sum_{i=0}^{m-1} a_i x^i, B = \sum_{i=0}^{m-1} b_i x^i, a_i, b_i \in GF(p)$, 则计算 $C = A \times B$ 分两步进行:

1、多项式乘法

$$C' = A \times B = \sum_{i=0}^{2m-2} c'_i x^i, \text{ 其中, } c'_i = \sum_{k+j=i} a_k b_j \bmod p \quad (4-2)$$

2、以多项式为模数的取模运算

$$C = C' \bmod f(x) = \sum_{i=0}^{2m-2} c'_i x^i \bmod (x^m - w) = \sum_{i=0}^{m-1} (c'_i + w c'_{i+m}) x^i, c'_{2m-1} = 0 \quad (4-3)$$

第一步是普通的多项式乘法, 第二步是以多项式为模数的取模运算。在第二步中, 因为多项式 $f(x)$ 为二项式, 所以模多项式运算可简化为 $(m-1)$ 个常系数的乘法和 $(m-1)$ 个模 p 加法。

4.2 最优塔域(OTFs)^[89]

4.2.1 OTFs 的基本性质

定义4.4 有限域 $GF(q^t)$ 是最优塔域(Optimal Tower Fields(OTFs)), 如果

- (1) q 是一个伪-Mersenne素数;
- (2) $GF(q^t)$ 是由一系列 $GF(q^{t^{i-1}})$ 上的不可约多项式 $P_i(x) = x^t - \alpha_{i-1}$ 构造而成, 其中 $P_i(\alpha_i) = 0, 0 < i \leq k$ 。

从定义 4.4 可以看出, 一个塔域就是不断重复扩展具有一系列相同次数的不可约二项式而得到的有限域, 其中这些二项式是相关的。

性质4.1 如果 $\alpha_i \in GF(q^{t^{i-1}}), 0 \leq r < t$, 则 $r = 0$ 。

因为 $P_i(x)$ 是 α_i 在 $GF(q^{t^{i-1}})$ 上的极小多项式, 所以

$$1, \alpha_i, \alpha_i^2, \dots, \alpha_i^{t-1}$$

在 $GF(q^{t^{i-1}})$ 上线性独立的。

于是, 存在 $\beta \in GF(q^{t^{i-1}})$, 由 $\alpha_i^r - \beta = 0$ 可得出 $r = 0$ 。

性质4.2 设 $GF(q^t)$ 由二项式 $P_i(x) = x^t - \alpha_{i-1} (P_i(\alpha_i) = 0, 0 < i \leq k)$ 构造而成, 这些二项式的根 α_i 的关系如下:

$$\text{ord}(\alpha_i) = t \cdot \text{ord}(\alpha_{i-1}) = t^i \cdot \text{ord}(\alpha_0)$$

其中 $\text{ord}(\alpha)$ 表示域元素 α 的阶。

性质4.3 如果在有限域 $GF(q)$ 上存在不可约二项式 $Q(x) = x^t - \alpha$ ，则 $t \mid ((q^t - 1)/(q - 1))$ 。

性质4.4 令 $P(x) = x^t - \alpha$ 是 $GF(q)$ 上的不可约二项式， t' 表示 t 的所有素因数的乘积，则 $q^{t'} \equiv 1 \pmod{t'}$ 。

性质4.5 (OTFs 存在的充要条件) 给定 $GF(q)$ 上的不可约二项式 $P_1(x) = x^t - \alpha_0$ ，且在 $GF(q')$ 上有 $P_1(\alpha_1) = 0$ 。如果 t 的任何素因数均不整除 $(q' - 1)/(t(q - 1))$ ，则所有 $GF(q^{t'})$ 上具有形式为 $P_i(x) = x^t - \alpha_{i-1}$ ，且在 $GF(q^{t'})$ 上有 $P_i(\alpha_i) = 0$ 的二项式也是不可约的。

下面的表4-2和表4-3给出了塔域 $GF(q^{2^k})$ 和 $GF(q^{3^k})$ 。

表4-2 $GF(q^{2^k})$ OTFs，其中， $q=2^n+c$ ， $8 \leq n \leq 16$ ， $-5 \leq c \leq 5$ ， $-5 \leq \alpha_0 \leq 5$

n	c	α_0	n	c	α_0	n	c	α_0	n	c	α_0
8	1	-5	9	-3	2	11	5	2	14	-3	-2
8	1	-3	9	-3	3	11	5	5	14	-3	2
8	1	3	10	-3	-2	12	-3	-5	16	1	-5
8	1	5	10	-3	2	12	-3	-2	16	1	-3
9	-3	-3	11	5	-5	12	-3	2	16	1	3
9	-3	-2	11	5	-2	12	-3	5	16	1	5

表4-3 $GF(q^{3^k})$ OTFs，其中， $q=2^n+c$ ， $7 \leq n \leq 16$ ， $-5 \leq c \leq 5$ ， $-5 \leq \alpha_0 \leq 5$

n	c	α_0	n	c	α_0	n	c	α_0	n	c	α_0
7	-1	3	11	5	-4	12	3	2	14	-3	-4
7	1	3	11	5	5	12	3	-2	14	3	5
10	-3	5	12	-3	5	12	3	-4	16	3	4
10	3	5	12	3	4	12	3	5	16	3	3
11	5	5	12	-3	2	14	-3	5	16	3	2
11	5	4	12	-3	-2	14	-3	4	16	3	-2
11	5	3	12	-3	-1	14	-3	3	16	3	-3
11	5	2	12	-3	-5	14	-3	2	16	3	-4
11	5	-2	12	3	5	14	-3	-2			
11	5	-3	12	3	4	14	-3	-3			

4.2.2 OTFs 和 OEFs 间的转换

当 $m=t^k$ 时，最优塔域 $GF(q^{t^k})$ 和 $GF(q^m)$ 是同构的。在介绍怎样从一个 OTFs 获得其相关的 OEFs 之前，首先介绍下面的定理。

引理4.5 对给定的最优塔域 $GF(q^{t^k})$ ，如果 $t \equiv 2 \pmod{4}$ ，则 $q \equiv 1 \pmod{4}$ 。

引理4.6 对给定 $GF(q^{t^k})$ 的 OTFs 表示，其相应的不可约二项式为 $P_i(x) = x^t - \alpha_{i-1}$ ($0 < i \leq k$)，则存在一个 OEFs 表示，其相应的不可约二项式为

$Q(x) = x^m - w$, 使得 $Q(\alpha_k) = 0, m = t^k, w = \alpha_0$ 。

引理 4.7 对给定 $GF(q^m)$ 的 OEFs 表示, 其相应的不可约二项式为 $Q(x) = x^m - w$ 。如果 $m = tk$, 且 t 的所有素因数均不整除 $(q^t - 1)/(t(q - 1))$, 则存在一个 OTFs 表示, 使得 $P_1(x) = x^t - \alpha_0, \alpha_0 = w$ 。

引理 4.8 给定相关的 OTF/OEF, 二者之间的转换仅是系数之间的简单置换, 该置换映射元素 $A \in GF(q^{t^k})/GF(q^m)$ 的系数 a_i 到相应的 $GF(q^m)/GF(q^{t^k})$ 的 a_s , 其中如果 a_i 的指标 i 写成 k 位长的 t 进制串, 则 a_s 的指标 s 则为 k 位长的逆 t 进制串。

例 4.1 令 $GF(q^{2^3})$ 表示一个 OTF, 构建最优塔域的二项式为 $P_1(x) = x^2 - \alpha_0$, $P_2(x) = x^2 - \alpha_1, P_3(x) = x^2 - \alpha_2$ 分别定义在有限域 $GF(q), GF(q^2)$ 和 $GF(q^{2^2})$ 上, 且 $P_i(\alpha_i) = 0, i = 1, 2, 3$ 。元素 $A \in GF(q^{2^3})$ 具有如下 OTF 表示:

$$A = ((a_{000} + a_{001}\alpha_1) + (a_{010} + a_{011}\alpha_1)\alpha_2) + ((a_{100} + a_{101}\alpha_1) + (a_{110} + a_{111}\alpha_1)\alpha_2)\alpha_3$$

根据引理 4.8 有

$$A = a_{000} + a_{100}\alpha_3 + a_{010}\alpha_3^2 + a_{110}\alpha_3^3 + a_{001}\alpha_3^4 + a_{101}\alpha_3^5 + a_{011}\alpha_3^6 + a_{111}\alpha_3^7$$

4.2.3 OTFs 和 OEFs 的复杂性比较

下面的表 4-4^[89] 给出了 OTFs 和 OEFs 上运算的复杂性比较。其中 $M(m), S(m)$ 和 $I(m)$ 分别表示 $GF(q^m)$ 上的乘法、平方和求逆运算; OTF2 表示 $m = 2^k$ 时的塔域, OTF3 表示 $m = 3^k$ 时的塔域。

表 4-4 OEF 和 OTF 的复杂性比较 ($\delta = \lfloor \log_2(m-1) \rfloor + HW(m-1)$)

运算	$GF(q)$ 上的乘法运算 M_0	$GF(q)$ 上的加法运算 A_0
$M(m)$ (OEF)	m^2	$m^2 - 1$
$M(m)$ (OTF2)	m^2	$1/3(4m^2 - 3m - 1)$
$M(m)$ (OTF3)	m^2	$1/4(5m^2 - 4m - 1)$
$S(m)$ (OEF)	$1/2(m^2 + m)$	$1/2(m^2 + 5m - 8), m$ 是偶数 $1/2(m^2 + 3m - 2), m$ 是奇数
$S(m)$ (OTF2)	$1/2(m^2 + m)$	$1/6(4m^2 + 3m \log_2 m - 4)$
$S(m)$ (OTF3)	$1/2(m^2 + m)$	$1/8(5m^2 + 8m \log_3 m - 5)$
$I(m)$ (OEF)	$(\delta - 1)m^2 + \delta(m - 1) + 2m$	$(\delta - 1)(m^2 - 1) + m - 1$
$I(m)$ (OTF2)	$m^2 + m - 2$	$1/3(4m^2 + 3m \log_2 m - 9m + 5)$
$I(m)$ (OTF3)	$1/16(21m^2 + 12m - 33)$	$1/64(105m^2 - 176m - 8 \log_3 m + 96m \log_3 m + 71)$

从表4-4, 我们可以看出, OTFs在乘法和平方运算上不具有任何优势, 但是对求逆运算来说, 无论它所需要的乘法次数还是加法次数, OEFs都明显高于OTFs, 并且, 即使对刻意选择的 m , 使得其汉明(Hamming)权重为1, 但 $\log_2(m-1)$ 也会随着 m 的增大而增大, 而OTFs对应的复杂度中 m^2 的系数却是常数, 不会因为 m 的增大而改变。因此, OTFs的优势在于其上的求逆运算复杂度较低。

从前面的分析我们可以看出, OEFs的使用大大提高了乘法运算的速度。但是, 通常情况下, c 较小时, 形如 $p = 2^n - c$ 的素数较少。例如, 李^[62]令机器字长是32位, $n = 32 \times d, 5 \leq d \leq 16, c = 1, 3, 5, 7$, 经过搜索发现, 不存在这种形式的素数。于是, 我们提出广义最优扩域(GOEFs)的概念。

4.3 广义最优扩域(GOEFs)

4.3.1 GOEFs 的基本概念

定义 4.5^[90] 令 t 是机器字长, 如果 $f(x)$ 是首一 m 次整系数多项式, 且 $f(x)$ 只有少数几个非零项, 则称 $f(2^t)$ 是广义 Mersenne 数。

注意: $f(x)$ 应满足下面的条件:

(1) $f(x)$ 的次数尽可能高, 但不超过 p 的字长。通常, $f(x)$ 的次数 $\deg(f)$ 满足

$$\text{word}(p) \approx lw, \text{word}(t) \cdot \deg(f) \approx \text{word}(p) \quad (4-4)$$

其中 $\text{word}(\cdot)$ 表示一个整数的字长, w 是处理机的字长, $l > 0$ 是一任意正整数。如果条件(4-4)不能满足, 则提出的方法可能慢于经典算法。

(2) 避免使用大于1的系数。

(3) 多项式的项数尽可能少, 设 m 次多项式 $f(x)$ 的项数为 l , 则多项式转换步数为 $2m-2$, 需要的加法或者减法数目是 $(l-1)(m-1)$ 。

(4) $f(x)$ 是不可约多项式, 因为这样的 $p = f(2^t)$ 才可能是素数。

定义 4.6 有限域 $GF(p^m)$ 是GOEFs, 如果满足:

(1) p 是广义 Mersenne 素数;

(2) 在 $GF(p)$ 上存在不可约三项式 $f(x) = x^m - x^k - w(m > k)$ 。

例如, $p = 2^{192} - 2^{64} - 1$ 是素数, $f(x) = x^{14} - x^7 - 1$ 是 $GF(p)$ 上的不可约三项式, 则 $GF((2^{192} - 2^{64} - 1)^{14})$ 就是GOEFs。

下面给出多项式在 $GF(p)$ 上不可约的充要条件。

引理 4.9^[14] 令 l_1, l_2, \dots, l_k 是 m 的所有素因数, $m_i = m/l_i (1 \leq i \leq k)$ 。次数为 m 的

多项式 $g(x) \in GF(p)[x]$ 是不可约多项式当且仅当

$$(1) g(x) \mid (x^{p^m} - x);$$

(2) $GCD(g(x), (x^{p^i} - x) \bmod g(x)) = 1, (1 \leq i \leq k)$, $GCD(f(x), g(x))$ 表示 $f(x)$ 和 $g(x)$ 的最大公因式。

证明 假设 $g(x)$ 不可约, 则 $g(x)$ 的每一个根 $\alpha \in GF(p^m)$, 由定理 2.6 可得, $\alpha^{p^m} - \alpha = 0$, 且 $(x - \alpha) \mid (x^{p^m} - x)$ 。因为 $g(x)$ 无重根, 所以(1)成立。

又因为 $g(x)$ 是次数为 m 的不可约多项式, 则当 $h < m$ 时, 其在 $GF(p^h)$ 上无根, 从而, 说明(2)成立。

反过来, 如果(2)和(3)成立, 则由(2)可知, 使得 $g(x)=0$ 的根属于 $GF(p^m)$ 。

另外, 假设 $g(x)$ 有一个次数为 $h(h < m)$ 的不可约因式 $g_1(x)$, 则 $g_1(x)$ 的所有根属于 $GF(p^h)$, 从而 $GF(p^h) \subseteq GF(p^m)$, 并且有 $h \mid m$ 和 $h \mid m_i$, m_i 是 m 最大的一个除数。于是 $g_1(x)$ 的所有根属于 $GF(p^m)$, 即 $GCD(g(x), (x^{p^m} - x)) = g_1(x)$, 与(2)矛盾。从而说明 $g(x)$ 是不可约的。

4.3.2 GOEFs 的乘法运算

对任意 $A, B \in GF(p^m)$, 其在多项式基下的表示形式为

$$A = \sum_{i=0}^{m-1} a_i x^i, B = \sum_{i=0}^{m-1} b_i x^i, a_i, b_i \in GF(p) \quad (4-5)$$

则 $C = A \times B$ 分两步进行: 第一步多项式乘法运算, 即令

$$C' = A \times B = \sum_{i=0}^{2m-2} c_i' x^i \quad (4-6)$$

第二步是以多项式为模数的取模运算, 我们将在下一小节讨论。

显然, 按照多项式乘法规则, 计算 C' 需要 m^2 个乘法, $(m-1)^2$ 个加法。因为一个域元素乘法的计算开销大于一个域元素加法的计算开销, 所以通过增加加法次数来减少乘法次数, 可以有效地减少计算 C' 的计算开销。

定理 4.1 设 $D_i = a_i b_i (0 \leq i \leq m-1)$, $d_{ij} = D_i + D_j (0 \leq i \leq m-2, j = i+1)$, $D_j = (a_i + a_j)(b_i + b_j) (0 \leq i \leq m-2, i+1 \leq j \leq m-1)$, 则 AB 所需乘法次数为 $m(m+1)/2$, 加法次数 Num 至多为:

$$Num = \begin{cases} (3m^2 + 5m - 12)/2 & (m > 3) \\ (m-1)(5m-2)/2 & (m = 1, 2, 3) \end{cases} \quad (4-7)$$

证明 设按照多项式乘法规则所得 x^i 的系数为 c_i' , 则

$$c_i' = \begin{cases} \sum_{k=0}^i a_k b_{i-k}, & 0 \leq i \leq m-1 \\ \sum_{k=i-m+1}^{m-1} a_k b_{i-k}, & m \leq i \leq 2(m-1) \end{cases}$$

根据对称性, 仅考虑 $0 \leq i \leq m-1$ 时, c_i' 的计算。

用 D_i, D_{ij} ($0 \leq i \leq m-1$) 的值代替 $a_k b_{i-k}$ ($0 \leq k \leq i$), 得

$$\begin{aligned} c_i' &= \sum_{k=0}^i a_k b_{i-k} \\ &= \begin{cases} (D_{0i} - D_0 - D_i) + \cdots + (D_{i/2-1, i/2+1} - D_{i/2-1} - D_{i/2+1}) + D_{i/2, i/2} & (i \text{ 是偶数}) \\ (D_{0i} - D_0 - D_i) + \cdots + (D_{(i-1)/2, (i+1)/2} - D_{(i-1)/2} - D_{(i+1)/2}) & (i \text{ 是奇数}) \end{cases} \quad (4-8) \end{aligned}$$

显然, 根据公式(4-8)计算 c_i' ($0 \leq i \leq 2(m-1)$) 时, 不需要域运算的乘法。从而, 计算 c_i' ($0 \leq i \leq 2(m-1)$) 所需要的域元素乘法次数可通过计算 D_i, D_{ij} 中出现的域元素乘法次数而得到。根据 D_i, D_{ij} 的假设, 计算 D_i 需要 m 次乘法, 计算 D_{ij} 需要 $m(m-1)/2$ 次乘法, 于是, 计算 c_i' ($0 \leq i \leq 2(m-1)$) 所需要的域元素乘法次数为 $m(m+1)/2$ 。

又由 D_i, D_{ij} 的假设和(4-8)知, 当 i 是偶数时, 计算 c_i' ($0 \leq i \leq 2(m-1)$) 所需加法数为 $5i/2$; 当 i 是奇数时, 计算 c_i' ($0 \leq i \leq 2(m-1)$) 所需加法数为 $(5i+3)/2$ 。另外, 当 m 是偶数时, 偶数项比奇数项多一项, 即偶数项有 $(m-1)/2$ 项, 奇数项有 $(m-3)/2$ 项; 当 m 是奇数时, 偶数项和奇数项一样多, 都为 $(m-1)/2$ 项。

于是计算所有 c_i' ($0 \leq i \leq 2(m-1)$) 所需加法次数 Num_1 为

$$Num_1 = \begin{cases} 2 \sum_{i=0}^{(m-3)/2} [5i + (5(2i+1)+3)/2] + 5(m-1)/2 = (m-1)(5m-2)/2 & (m \text{ 是奇数}) \\ 2 \sum_{i=0}^{(m-2)/2} [5i + (5(2i+1)+3)/2] - (5m-2)/2 = (m-1)(5m-2)/2 & (m \text{ 是偶数}) \end{cases}$$

(1) $m > 3$ 时, 利用 d_{ij} 的假设, 为了简单, 仅考虑 d_{ij} 被使用一次的情形, 可以再减少的加法次数 Num_2 为:

$$Num_2 = 2[m(m-1)/2 - (2m-3)] - (m-1) = m^2 - 6m + 7$$

(2) $m=1, 2, 3$ 时, 不会使用 d_{ij} , 因此加法数与 Num_2 无关, 从而其加法数为 $(m-1)(5m-2)/2$ 。

综上所述, 定理得证。◆

4.3.3 GOEFs 的取模运算

定理 4.2 令 $C = C' \bmod f(x) = \sum_{i=0}^{2m-2} c_i' x^i \bmod (x^m - x^k - a) = \sum_{i=0}^{m-1} c_i x^i$, $c_i \in GF(p)$,

则

(I) $m > 2k$,

$$\begin{cases} c_i = c_i + a(c_{m+i} + c_{2m-k+i}), & i = 0, 1, \dots, k-1 \\ c_{k+i} = c_{k+i} + c_{m+i} + c_{2m-k+i} + ac_{m+k+i}, & i = 0, 1, \dots, k-1 \\ c_{2k-1+i} = c_{2k+i} + c_{m+k+i} + ac_{m+2k+i}, & i = 0, 1, \dots, m-2k-1 \\ c_{2m-1} = 0 \end{cases} \quad (4-9)$$

(II) $m = 2k$,

$$\begin{cases} c_i = c_i + a(c_{m+i} + c_{2m-k+i}) \\ c_{k+i} = c_{k+i} + c_{2k+i} + (a+1)c_{3k+i} & i = 0, 1, \dots, k-1 \\ c_{2m-1} = 0 \end{cases} \quad (4-10)$$

(III) $m < 2k$,

$$\begin{cases} c_i = c_i + a \sum_{j=0}^{n-1} c_{(j+1)m-jk+i} & i = 0, 1, \dots, k-1 \\ c_{k+i} = c_{k+i} + \sum_{j=0}^{n-1} c_{(j+1)m-jk+i} + ac_{m+k+i} & i = 0, 1, \dots, m-k-1 \\ c_i = 0 & i \geq 2m-1 \\ n = \lfloor m/(m-k) \rfloor \end{cases} \quad (4-11)$$

证明 因为

$$C = \sum_{i=0}^{2m-2} c_i x^i \bmod (x^m - x^k - a) = \left(\sum_{i=0}^{2m-2} c_i x^i (\bmod (x^m - x^k - a)) \right) \bmod (x^m - x^k - a)$$

所以, 我们可以先求出 $c_{m+i} x^{m+i} \bmod (x^m - x^k - a), t = 0, 1, \dots, (m-1)$, 然后将各项的结果求和, 合并同类项而得到结果。

首先, 讨论 $m > 2k$ 的情形。

根据以多项式为模数的取模运算定义知:

$$\begin{aligned} & c_{m+i} x^{m+i} \bmod (x^m - x^k - a) \\ &= \begin{cases} c_{m+i} x^{k+i} + ac_{m+i} x^i, & t = 0, 1, \dots, (m-1-k) \\ ac_{m+i} x^i + c_{m+i} x^{2k+i-m} + ac_{m+i} x^{k+i-m}, & t = (m-k), \dots, (m-1) \end{cases} \end{aligned} \quad (4-12)$$

利用公式(4-12), 可以得出对应矩阵如图 4-1, 其中该矩阵的行元素依次为 $(x^m \bmod f(x), x^{m+1} \bmod f(x), x^{m+2} \bmod f(x), \dots, x^{2m-1} \bmod f(x))$, 列元素依次为 $(x^0, x^1, \dots, x^{m-2}, x^{m-1})$ 的系数。

$$\begin{pmatrix} x^m \\ x^{m+1} \\ \dots \\ x^{2m-k-1} \\ x^{2m-k} \\ x^{2m-k+1} \\ \dots \\ x^{2m-1} \end{pmatrix} \bmod f(x) = \begin{pmatrix} ac_m & \dots & c_m & \dots & 0 \\ & ac_{m+1} & \dots & c_{m+1} & \dots & 0 \\ & & \dots & \dots & \dots & \dots \\ & & & ac_{2m-k-1} & \dots & c_{2m-k-1} \\ ac_{2m-k} & \dots & c_{2m-k} & \dots & ac_{2m-k} & 0 \\ & ac_{2m-k+1} & \dots & c_{2m-k+1} & \dots & ac_{2m-k+1} & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & & & ac_{2m-1} & \dots & c_{2m-1} & \dots & ac_{2m-1} \end{pmatrix}$$

$x^0 \qquad \qquad \qquad x^k$
 $x^{k-1} \qquad \qquad \qquad x^{2k-1} \qquad \qquad \qquad x^{m-1}$

图 4-1 利用式(4-12)得到的矩阵

在图(4-1)中，同一列的次数相同。从而，根据图(4-1)，有

$$\begin{aligned} C &= \sum_{i=0}^{2m-2} c_i' x^i \bmod (x^m - x^k - a) \\ &= \left(\sum_{i=0}^{2m-2} c_i' x^i (\bmod (x^m - x^k - a)) \right) \bmod (x^m - x^k - a) \\ &= (c_0' + ac_m' + ac_{2m-k}')x^0 + \dots + (c_{k-1}' + ac_{m+k-1}' + c_{2m-1}')x^{k-1} + \dots \\ &\quad + (c_k' + c_m' + ac_{m+k}' + c_{2m-k}')x^k + \dots + (c_{2k-1}' + c_{m+k-1}' + ac_{m-1}' + c_{2m-1}')x^{2k-1} \\ &\quad + (c_{2k}' + c_{m+k}' + ac_{m+2k}')x^{2k} + \dots + (c_{m-1}' + c_{2m-k-1}' + ac_{2m-1}')x^{m-1} \end{aligned}$$

即

$$\begin{cases} c_i = c_i' + a(c_{m+i}' + c_{2m-k+i}'), & i = 0, 1, \dots, k-1 \\ c_{k+i} = c_{k+i}' + c_{m+i}' + c_{2m-k+i}' + ac_{m+k+i}', & i = 0, 1, \dots, k-1 \\ c_{2k-1+i} = c_{2k+i}' + c_{m+k+i}' + ac_{m+2k+i}', & i = 0, 1, \dots, m-2k-1 \\ c_{2m-1} = 0 \end{cases}$$

其次，我们讨论 $m = 2k$ 时的情形。

同样地，根据以多项式为模数的取模运算定义知：

$$\begin{aligned} &c_{m+t}' x^{m+t} \bmod (x^m - x^k - a) \\ &= \begin{cases} c_{m+t}' x^{k+t} + ac_{m+t}' x^t, & t = 0, 1, \dots, (m-1-k) \\ (ac_{m+t}' + c_{m+t}')x^t + ac_{m+t}' x^{k+t-m}, & t = (m-k), \dots, (m-1) \end{cases} \end{aligned} \quad (4-13)$$

利用式(4-13)，可以得出对应矩阵如图 4-2。

$$\begin{pmatrix} x^m \\ x^{m+1} \\ \dots \\ x^{2m-k-1} \\ x^{2m-k} \\ x^{2m-k+1} \\ \dots \\ x^{2d-1} \end{pmatrix} \bmod f(x) = \begin{pmatrix} x^0 & & & x^k & & & \\ ac_m & \dots & & c_m & \dots & & 0 \\ & ac_{m+1} & \dots & c_{m+1} & \dots & & 0 \\ & & \dots & & \dots & & \\ & & & ac_{2m-k-1} & \dots & & c_{2m-k-1} \\ ac_{2m-k} & \dots & & c_{2m-k} & + & ac_{2m-k} & 0 \\ & ac_{2m-k+1} & \dots & c_{2m-k+1} & + & ac_{2m-k+1} & 0 \\ & & \dots & & \dots & & \\ & & & ac_{2m-1} & \dots & & c_{2m-1} + ac_{2m-1} \\ & & & & & x^{k-1} & x^{m-1} \end{pmatrix}$$

图 4-2 利用式(4-13)得到的矩阵

根据图(4-2), 有

$$\begin{aligned} C &= \sum_{i=0}^{2m-2} c_i x^i \bmod (x^m - x^k - a) \\ &= \left(\sum_{i=0}^{2m-2} c_i x^i (\bmod (x^m - x^k - a)) \right) \bmod (x^m - x^k - a) \\ &= (c_0 + ac_m + ac_{2m-k})x^0 + (c_1 + ac_{m+1} + ac_{2m-k+1})x^1 \\ &\quad + \dots + (c_{k-1} + ac_{m+k-1} + c_{2m-1})x^{k-1} \\ &\quad + \dots + (c_k + c_m + (a+1)c_{3k})x^k \\ &\quad + \dots + (c_{m-1} + c_{m+k-1} + (a+1)c_{2m-1})x^{m-1} \end{aligned}$$

即

$$\begin{cases} c_i = c_i + a(c_{m+i} + c_{2m-k+i}) \\ c_{k+i} = c_{k+i} + c_{m+i} + (a+1)c_{3k+i} & i = 0, 1, \dots, k-1 \\ c_{2m-1} = 0 \end{cases}$$

最后, 我们讨论 $m < 2k$ 的情形, 与前两种情况相比, 相对复杂一些。

假设 $m \leq kn/(n-1)$, 根据 m 和 k 的值, 可以求出 n 。即,

$$m \leq kn/(n-1) \Rightarrow m(n-1) \leq kn \Rightarrow (m-k)n \leq m \Rightarrow n \leq m/(m-k)$$

从而, 取 $n = \lfloor m/(m-k) \rfloor$ 。

同样地, 由以多项式为模数的取模运算定义知:

$$c_{m+t}'x^{m+t} \bmod f(x) = \begin{cases} c_{m+t}'x^{t+k} + ac_{m+t}'x^t, & t=0, \dots, (m-k)-1 \\ c_{m+t}'x^{2k-m+t} + ac_{m+t}'x^t + ac_{m+t}'x^{t+k-m}, & t=(m-k), \dots, 2(m-k)-1 \\ c_{m+t}'x^{3k-2m+t} + ac_{m+t}'x^t + ac_{m+t}'x^{t+k-m} + ac_{m+t}'x^{t+2k-2m}, & t=2(m-k), \dots, 3(m-k)-1 \\ \dots & \dots \\ c_{m+t}'x^{nk-(n-1)m+t} + a \sum_{j=0}^{n-1} c_{m+t}'x^{t+j(k-m)}, & t=n(m-k), \dots, m-1. \end{cases} \quad (4-14)$$

利用式(4-14)，可以得出对应矩阵如图 4-3。

下面我们讨论各矩阵元素在矩阵中的放置。

因为 $n(m-k) = \lfloor m/(m-k) \rfloor (m-k) \leq (m/(m-k))(m-k) = m$ (由下取整的定义知)，所以 $(n-1)m \leq nk$ 。即在第 $(j-1)(m-k)$ ($j=1, \dots, n$) 行， c_{m+t}' 一定在 ac_{m+t}' 的右边，当 $(m-k)|m$ 时， c_{m+t}' 和 ac_{m+t}' 重合，均在主对角线上，此时，主对角线上的元素为 $c_{m+t}' + ac_{m+t}'$ 。

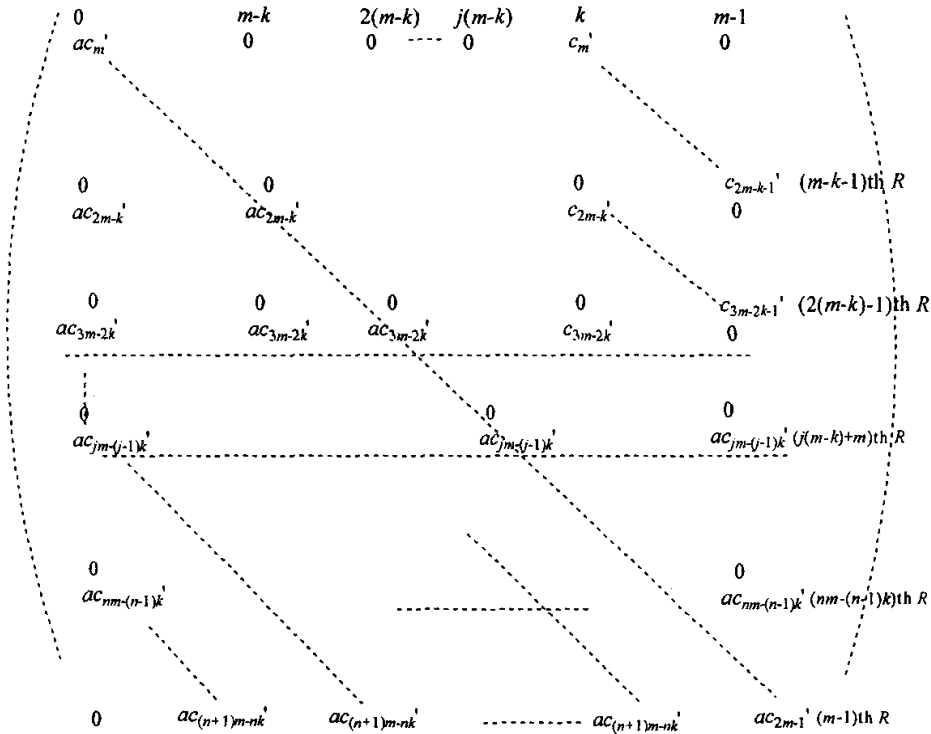


图 4-3 利用式 (4-14)得到的矩阵

根据图(4-3), 有

$$\begin{cases} c_i = c_i' + a \sum_{j=0}^{n-1} c_{(j+1)m-jk+i}' & i = 0, 1, \dots, k-1 \\ c_{k+i} = c_{k+i}' + \sum_{j=0}^{n-1} c_{(j+1)m-jk+i}' + ac_{m+k+i}' & i = 0, 1, \dots, m-k-1 \\ c_i = 0 & i \geq 2m-1 \end{cases}$$

综上所述, 定理 4.2 得证。

由公式(4-9), (4-10), (4-11)可以看出, 在取模运算中, 除法完全被加法和乘法代替, 其所需的加法和乘法次数如表 4-5 所示。

4.3.4 性能比较

4.3.4.1 与多项式乘法规则结果比较

按照多项式乘法规则, 所需乘法和加法次数分别为 m^2 和 $(m-1)^2$, 而采用定理 4.1 的假设方式, 所需乘法次数为 $m(m+1)/2$, 加法次数至多为 $(3m^2 + 5m - 12)/2$ 个。假设一个乘法所需时间是一个加法所需时间的 t 倍, 令

$$tm^2 + (m-1)^2 = tm(m+1)/2 + (3m^2 + 5m - 12)/2$$

则

$$t = (m^2 + 9m - 14)/(m^2 - m) = (1 - 9/m - 14/m^2)/(1 - 1/m) \rightarrow 1, (m \rightarrow \infty) \quad (4-15)$$

显然, 由(4-15)知, 当 $m > 1$ 时, $t > 1$ 。

图 4-4 给出了 m 的取值从 2 到 1000000 时, t 的取值情况。由图 4-4 可以看出, 对不同的 m , 当 t 取对应曲线上方的值时, 本文方法优于传统方法。而事实上乘法运行时间远大于加法运行时间, 因此, 本文方法所需运算时间小于一般多项式乘法法则所需时间。

表 4-5 取模所需加法和乘法次数

	加法次数	乘法次数
$m < 2k$	$5k-4$	$m(n+1)m-k$
$m = 2k$	$4k$	$m-1$
$m > 2k$	$2m+2k$	$m-1$

其中, $n = \lfloor m/(m-k) \rfloor$

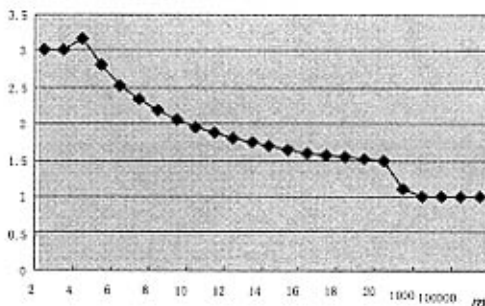


图 4-4 t 和 m 的关系图

4.3.4.2 与 Bailey 结果的比较

Bailey 在文献[88]提出了一种快速乘法方式,但该方法只对 m 是偶数,且 $E_0(x)$, $E_1(x)$, $E_2(x)$ 已有最少乘法次数和加法次数的情形才有效。例如, $m=6$ 时,

$$A(x)=a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x+a_0, B(x)=b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0, \text{ 则}$$

$$A(x)B(x)=E_2(x)x^6+(E_1(x)-E_2(x)-E_0(x))x^3+E_0(x)$$

其中, $A_l(x)=a_5x^2+a_4x+a_3$, $A_h(x)=a_2x^2+a_1x+a_0$; $B_l(x)=b_5x^2+b_4x+b_3$,

$$B_h(x)=b_2x^2+b_1x+b_0; E_2(x)=A_l(x)B_l(x), E_1(x)=(A_l(x)+A_h(x))(B_l(x)+B_h(x)),$$

$$E_0(x)=A_h(x)B_h(x).$$

Bailey 指出乘法次数 $=6 \times 3 = 18$, 加法次数 $=3 \times 13 + (3+3) + (5+5) + 4 = 59$, 但按照本文的方法, 当 $m=6$ 时, 乘法次数 $=21$, 加法次数 $=63$ 。

然而, 当 $m=5$ 时, 该方法失效。因为, 此时,

$$A(x)B(x)=E_2(x)x^4+(E_1(x)-E_2(x)-E_0(x))x^2+E_0(x)$$

其中,

$$A_l(x)=a_4x^2+a_3x+a_2, A_h(x)=a_1x+a_0; B_l(x)=b_4x^2+b_3x+b_2,$$

$$B_h(x)=b_1x+b_0; E_2(x)=A_l(x)B_l(x), E_1(x)=(A_l(x)+A_h(x))(B_l(x)+B_h(x)),$$

$$E_0(x)=A_h(x)B_h(x).$$

乘法次数 $=6 \times 3 = 18$, 加法次数 $=13 \times 3 + (2+2) + (5+5) + 6 = 59$, 但本文方法却得出乘法次数 $=15$, 加法次数 $=44$, 优于文献[88]的方法。

由此, 我们得出结论: 文献[88]的方法仅适合 m 是偶数, 且 $E_0(x)$, $E_1(x)$, $E_2(x)$ 已有最少乘法次数和加法次数的情形, 而本文方法对所有 m 值都适用, 因此比文献[88]的方法更有效, 更适用。

4.4 小结

本章首先介绍了 OEFs 的基本概念, 在 OEFs 下的乘法运算; 接着介绍了一类特殊的 OEFs—OTFs 的基本性质及其与 OTFs 之间的关系和复杂性比较; 最后通过指出 OEFs 的局限性, 提出 GOEFs 的基本概念, 并深入研究了快速乘法运算和取模运算, 给出了快速乘法运算的复杂度公式和取模运算的运算公式, 得到了更一般的结果。通过相关比较, 本章 GOEFs 上的工作推广了 D.V. Bailey 和 C.Paar, P.M ăilescu, D.V.Bailey 和 A.D.Woodbury 等在 OEFs 上所做工作。

第五章 有限域上的求逆运算

由第二章的预备知识知, 当椭圆曲线上的点用仿射坐标表示时, 无论是点加还是倍乘, 都会涉及有限域上的求逆运算。在通常情况下, 求逆运算的开销至少是乘法运算开销的 4 倍。因此, 许多研究者开始考虑使用投影坐标表示椭圆曲线上的点, 因为在这种坐标表示形式下, 椭圆曲线有理点群的点加和倍加计算不需要求逆运算。但是, 此时却需要更多的临时存储空间, 这对存储空间较小的设备来说则是另一个难以解决的问题。

本章将深入分析和研究有限域 $GF(q)$ 和二元扩域 $GF(2^m)$ 上的各种求逆运算算法, 其中重点讨论最优扩域, 最优塔域上的求逆算法, 并对各种求逆算法的性能进行分析比较。

5.1 $GF(p)$ 上的求逆运算

任意 $a \in GF(p), a \neq 0$, 寻找 $a^{-1} \in GF(p)$, 使得

$$aa^{-1} \equiv 1 \pmod{p} \quad (5-1)$$

则 a^{-1} 称为 a 的逆元, 寻找逆元的过程称为求逆运算。

对大素数域, 基本的求逆技术是基于欧氏算法, 或者它的演变体如准求逆算法, 或者 Fermat 小定理。下面给出这几种经典的 $GF(p)$ 上的求逆算法。

5.1.1 扩展欧氏求逆算法

引理 5.1 令 a 和 b 是正整数, 则对任意的整数 c , $GCD(a,b)=GCD(b-ca,a)$, 其中, $GCD(a,b)$ 表示 a 和 b 的最大公约数。

古典欧氏算法计算 $GCD(a,b)$ 的基本思想是: 用 b 除以 a 得到商 q 和余数 r , 即 $b=aq+r(0 \leq r < a)$, 由引理 5.1, $GCD(a,b)=GCD(r,a)$, 从而, 求 $GCD(a,b)$ 简化为求 $GCD(r,b)$, 重复此过程, 直到其中一个为 0, 就得到 $GCD(a,b)$ 的值, 因为 $GCD(0,d)=d$ 。当 a 的二进制长度为 k 时, 所需要的除法步数至多为 $2k$, 即其所需要的除法步数为 $O(2\log a)$ 。

如果将古典欧氏算法扩展到寻找 x 和 y , 使得 $ax+by=d$ 时, 则可得到下面的扩展欧氏求逆算法。

算法 5.1 扩展欧氏算法

输入：正整数 a 和 b ，其中， $a \leq b$

输出： $d = \text{GCD}(a, b)$ 和整数 x ，使得 $ax + by = d$

```

1  $u \leftarrow a, v \leftarrow b$ 
2  $x_1 \leftarrow 1, y_1 \leftarrow 0, x_2 \leftarrow 0, y_2 \leftarrow 1$ 
3 While  $u \neq 0$  do
    3.1  $q \leftarrow \lfloor v/u \rfloor, r \leftarrow v - qu, x \leftarrow x_2 - qx_1, y \leftarrow y_2 - qy_1$ 
    3.2  $v \leftarrow u, u \leftarrow r, x_2 \leftarrow x_1, x_1 \leftarrow x, y_2 \leftarrow y_1, y_1 \leftarrow y$ 
4  $d \leftarrow v, x \leftarrow x_2, y \leftarrow y_2$ 
5 Return( $d, x, y$ )
    
```

如果假设 p 是素数， $a \in [1, p-1]$ ，则 $\text{GCD}(a, p) = 1$ 。在算法 5.1 中，令 a, p 为其输入，则最后的非零剩余 $r = 1$ 在第 3.1 步中即可得到，随后，在 3.2 步中，有 $ax_1 + py_1 = 1$ ，从而有 $ax_1 \equiv 1 \pmod{p}$ ，即 $a^{-1} \equiv x_1 \pmod{p}$ ，显然并未用到 y_1, y_2 ，因此得到下面的扩展欧氏求逆算法。

算法 5.2 扩展欧氏求逆算法

输入：素数 p 和 $a \in [1, p-1]$

输出： $a^{-1} \pmod{p}$

```

1  $u \leftarrow a, v \leftarrow p$ 
2  $x_1 \leftarrow 1, x_2 \leftarrow 0$ 
3 While  $u \neq 1$  do
    3.1  $q \leftarrow \lfloor v/u \rfloor, r \leftarrow v - qu, x \leftarrow x_2 - qx_1$ 
    3.2  $v \leftarrow u, u \leftarrow r, x_2 \leftarrow x_1, x_1 \leftarrow x$ 
4 Return( $x_1 \pmod{p}$ )
    
```

算法 5.1 和算法 5.2 的缺点在于第 3.1 步都需要开销昂贵的除法运算。为了避免除法运算，得到了改进的二进制扩展欧氏算法(Binary GCD Algorithm)和二进制求逆算法(Binary Inversion Algorithm)，见算法 5.3 和 5.4。

算法 5.3 二进制扩展欧氏算法

输入：正整数 a 和 b ，其中， $a \leq b$

输出： $d = \text{GCD}(a, b)$

```

1  $u \leftarrow a, v \leftarrow b, e \leftarrow 1$ 
2 While  $u$  和  $v$  都是偶数 do  $u \leftarrow u/2, v \leftarrow v/2, e \leftarrow 2e$ 
3 While  $u \neq 0$  do
    3.1 While  $u$  是偶数 do  $u \leftarrow u/2$ 
    
```

```

3.2 While  $v$  是偶数 do  $v \leftarrow v/2$ 
3.3 While If  $u \geq v$  then  $u \leftarrow u - v$  else  $v \leftarrow v - u$ 
4 Return( $e \cdot v$ )

```

算法 5.4 二进制求逆算法

输入：素数 p 和 $a \in [1, p-1]$

输出： $a^{-1} \bmod p$

```

1  $u \leftarrow a, v \leftarrow p$ 
2  $x_1 \leftarrow 1, x_2 \leftarrow 0$ 
3 While ( $u \neq 1$  and  $v \neq 1$ ) do
    3.1 While  $u$  是偶数 do
         $u \leftarrow u/2$ 
        If  $x_1$  是偶数 then  $x_1 \leftarrow x_1/2$  else  $x_1 \leftarrow (x_1 + p)/2$ 
    3.2 While  $v$  是偶数 do
         $v \leftarrow v/2$ 
        If  $x_2$  是偶数 then  $x_2 \leftarrow x_2/2$  else  $x_2 \leftarrow (x_2 + p)/2$ 
    3.3 If  $u \geq v$  then  $u \leftarrow u - v, x_1 \leftarrow x_1 - x_2$ 
        else  $v \leftarrow v - u, x_2 \leftarrow x_2 - x_1$ 
4 If  $u = 1$  then return ( $x_1 \bmod p$ ) else return ( $x_2 \bmod p$ )

```

算法 5.3 和 5.4 利用移位和减法运算代替了除法运算，使得计算速度得到较大提高。

5.1.2 Montgomery 求逆算法

在第四章第三节里介绍 Montgomery 取模乘法时我们已经特别指出其优势：用开销较小的移位、减法计算和预计算取代了开销较大的除法运算。在这一小节，我们将介绍基于 Montgomery 方法的求逆算法。利用该算法求出元素 a 的逆元为 $a^{-1}2^n \bmod p$ ，我们称为 Montgomery 逆元，其中 $n = \lceil \log_2 p \rceil$ 。具体算法如下：

算法 5.5 局部 Montgomery 求逆算法

输入：素数 $p > 2$, $a \in [1, p-1]$, $n = \lceil \log_2 p \rceil$

输出：“无逆元”或者 (x, k) ，其中 $n \leq k \leq 2n$, $x = a^{-1}2^k \bmod p$

```

1  $u \leftarrow a, v \leftarrow p, x_1 \leftarrow 1, x_2 \leftarrow 0, k \leftarrow 0$ 
2 While  $v > 0$  do
    2.1 If  $v$  是偶数 then  $v \leftarrow v/2, x_1 \leftarrow 2x_1$ 
        else if  $u$  是偶数 then  $u \leftarrow u/2, x_2 \leftarrow 2x_2$ 

```

```

        else if  $v \geq u$  then  $v \leftarrow (v-u)/2, x_2 \leftarrow x_1 + x_2, x_1 \leftarrow 2x_1$ 
        else  $u \leftarrow (u-v)/2, x_1 \leftarrow x_1 + x_2, x_2 \leftarrow 2x_2$ 
    2.2  $k \leftarrow k+1$ .
    3 If  $u \neq 1$  then return ("not invertible")
    4 If  $x_1 > p$  then  $x_1 \leftarrow x_1 - p$ 
    5  $x \leftarrow x_1$ 
    6 Return( $x_1, k$ )
    
```

算法 5.5 返回的 x 与 k 的关系为 $x = a^{-1}2^k \bmod p$ 。因此，为了求 $a^{-1} \bmod p$ ，此处需重复 k 次如下变换：

$$\text{If } x \text{ 是偶数 then } x \leftarrow x/2 \text{ else } x \leftarrow (x+p)/2 \quad (5-2)$$

算法 5.5 对 x_1 和 x_2 的更新比算法 5.4 更简单，但(5-2)式可能会带来更多开销。

为了运算的有效性，Montgomery 算法常选择 $R = 2^w \geq 2^n$ ，同时还利用了 $\tilde{x} = xR \bmod p$ 。为了使 Montgomery 算法能够直接得到 $a^{-1} \bmod p$ 或者 $a^{-1}R \bmod p$ ，而不是算法 5.5 中的 $a^{-1}2^k \bmod p$ ，于是对(5-2)式作出修改得到下面的算法 5.6。如果椭圆曲线上的点使用仿射坐标的话，该算法可以用于 ECC 中的点加算法中。

算法 5.6 Montgomery 求逆算法

输入：素数 $p > 2$ ， $R^2 \bmod p$ ， $\tilde{a} = aR \bmod p$ ， $\text{GCD}(a, p) = 1$ ， $n = \lceil \log_2 p \rceil$

输出： $a^{-1}R \bmod p$

```

    1 利用算法 5.5 找出  $(x, k)$ ，其中， $x = \tilde{a}^{-1}2^k \bmod p$  且  $n \leq k \leq 2n$ 
    2  If  $k < W_t$  then
        2.1  $x \leftarrow \text{Mont}(x, R^2) = a^{-1}2^k \bmod p$ 
        2.2  $k \leftarrow k + W_t$ 
    3   $x \leftarrow \text{Mont}(x, R^2) = a^{-1}2^k \bmod p$ 
    4   $x \leftarrow \text{Mont}(x, 2^{2W_t-k}) = a^{-1}R \bmod p$ 
    5  Return( $x$ )
    
```

如果需要同时对 $k(k \geq 1)$ 个元素求逆，则可利用下面的同步求逆算法。该算法需要一次域元素求逆运算和 $3k$ 次域元素乘法运算。

算法 5.7 同步求逆算法

输入：素数 $p > 2$ 和非零元素 $a_1, a_2, \dots, a_k \in GF(p)$

输出： $a_1^{-1}, a_2^{-1}, \dots, a_k^{-1}$ ，其中， $a_i^{-1}a_i \equiv 1 \pmod{p} (i=1, \dots, k)$

```

    1  $c_1 \leftarrow a_1$ 
    2 For  $i=2$  to  $k$  do  $c_i \leftarrow c_{i-1}a_i \bmod p$ 
    3  $u \leftarrow c_k^{-1} \bmod p$ 
    
```

```

4 For  $i=k$  downto 2 do
  4.1  $a_i^{-1} \leftarrow u c_{i-1} \bmod p$ 
  4.2  $u \leftarrow u a_i \bmod p$ 
5  $a_1^{-1} \leftarrow u$ 
6 Return( $a_1^{-1}, a_2^{-1}, \dots, a_k^{-1}$ )

```

5.2 二元扩域 $GF(2^m)$ 上的求逆运算

假设任意 $A(x) = x^{m-1} + a_{m-2}x^{m-2} + \dots + a_0 \in GF(2^m)$, $a_i \in GF(2)$, $i = 0, \dots, m-2$, 则寻找 $A(x)^{-1} \in GF(2^m)$, 使得

$$A(x)A(x)^{-1} \equiv 1 \pmod{f(x)} \quad (5-3)$$

其中 $f(x) = x^{m-1} + f_{m-2}x^{m-2} + \dots + f_1x + f_0$, $f_i \in GF(2)$, $i = 0, \dots, m-2$, 则称 $A(x)^{-1}$ 为 $A(x)$ 的逆元, 寻找逆元的过程称为求逆运算。

对二元扩域, 基本的求逆技术仍然是基于欧氏算法的。我们首先介绍三种二元扩域上经典的求逆算法: 欧氏求逆算法(Extended Euclidean Algorithm, *EEA*), 二进制欧氏求逆算法(Binary Euclidean Algorithm, *BEA*), 准求逆算法(Almost Inverse Algorithm, *AlI*), 随后介绍其上的除法运算, 并对两种运算的性能进行分析和比较。

5.2.1 三种经典求逆算法

欧氏算法还可以扩展用来寻找多项式 $g(x)$ 和 $h(x)$, 使得

$$A(x)g(x) + f(x)h(x) \equiv D(x), \quad D(x) = \text{GCD}(A(x), f(x))$$

如果 $f(x)$ 是一个次数为 m 的不可约多项式, $A(x)$ 是次数不高于 $(m-1)$ 的非零多项式, 则有 $\text{GCD}(A(x), f(x)) = 1$ 。于是, 类似于算法 5.2, 得到下面的扩展欧氏求逆算法(Extended Euclidean Algorithm, *EEA*), 具体算法如下:

算法 5.8 扩展欧氏求逆算法(*EEA*)

输入: $f(x), A(x) \in GF(2^m), A(x) \neq 0$

输出: $A(x)^{-1} \bmod f(x)$

```

1  $u \leftarrow A(x), v \leftarrow f(x)$ 
2  $g_1(x) \leftarrow 1, g_2(x) \leftarrow 0$ 
3 While  $u \neq 1$  do

```

```

3.1  $j \leftarrow \deg(u) - \deg(v)$ 
3.2 If  $j < 0$  then  $u \leftrightarrow v, g_1(x) \leftrightarrow g_2(x), j \leftarrow -j$ 
3.3  $u \leftarrow u + x^j v$ 
3.4  $g_1(x) \leftarrow g_1(x) + x^j g_2(x)$ 
4 Return ( $g_1(x)$ )
    
```

从算法 5.8 可以知道, u 和 v 的比特数是从高位到低位去掉的, 当最后只剩下最低位为 1 时, 结束算法, 求得 $A(x)^{-1}$ 。

与算法 5.8 刚好相反, 下面的 BEA 算法则是将 u 和 v 的比特数从低位到高位去掉的。具体算法如下:

算法 5.9 二进制欧氏求逆算法(BEA)

输入: $f(x), A(x) \in GF(2^m), A(x) \neq 0$

输出: $A(x)^{-1} \bmod f(x)$

```

1  $u \leftarrow A(x), v \leftarrow f(x)$ 
2  $g_1(x) \leftarrow 1, g_2(x) \leftarrow 0$ 
3 While ( $u \neq 1 \wedge v \neq 1$ ) do
    3.1 While  $x|u$  do
         $u \leftarrow u/x$ 
        If  $x|g_1(x)$  then  $g_1(x) \leftarrow g_1(x)/x$ ; else  $g_1(x) \leftarrow (g_1(x) + f)/x$ 
    3.2 While  $x|v$  do
         $v \leftarrow v/x$ 
        If  $x|g_2(x)$  then  $g_2(x) \leftarrow g_2(x)/x$ ; else  $g_2(x) \leftarrow (g_2(x) + f)/x$ 
    3.3 If  $\deg(u) > \deg(v)$  then  $u \leftarrow u + v, g_1(x) \leftarrow g_1(x) + g_2(x)$ 
        else  $v \leftarrow u + v, g_2(x) \leftarrow g_2(x) + g_1(x)$ 
4 If  $u = 1$  then return ( $g_1(x)$ ); else return ( $g_2(x)$ )
    
```

显然, 第 3.3 步涉及多项式次数的比较, 该比较可以简单地替换成多项式的二进制表示的比较。

如果首先寻找多项式 $g(x)$ 和正整数 k 使得 $A(x)g(x) \equiv x^k \pmod{f(x)}$, 然后再计算 $A(x)^{-1} \equiv x^{-k}g(x) \pmod{f(x)}$, 则我们可以得到下面的准求逆算法(ALA)^[92]。具体算法如下:

算法 5.10 准求逆算法 (ALA)

输入: $f(x), A(x) \in GF(2^m), A(x) \neq 0$

输出: $A(x)^{-1} \bmod f(x)$


```

1  $u \leftarrow A(x), v \leftarrow f(x)$ 
2  $g_1(x) \leftarrow 1, g_2(x) \leftarrow 0, k \leftarrow 0$ 
3 While ( $u \neq 1$  且  $v \neq 1$ ) do
    3.1 While  $x|u$  do
         $u \leftarrow u/x, g_2(x) \leftarrow xg_2(x), k \leftarrow k+1$ 
    3.2 While  $x|v$  do
         $v \leftarrow v/x, g_1(x) \leftarrow xg_1(x), k \leftarrow k+1$ 
    3.3 If  $\deg(u) > \deg(v)$  then  $u \leftarrow u+v, g_1(x) \leftarrow g_1(x)+g_2(x)$ 
        else  $v \leftarrow u+v, g_2(x) \leftarrow g_2(x)+g_1(x)$ 
4 If  $u=1$  then  $g(x) \leftarrow g_1(x)$  else  $g(x) \leftarrow g_2(x)$ 
5 Return ( $x^{-k}g(x) \bmod f(x)$ )

```

显然, 在 AIA 中, 第 5 步还需要约简 $x^{-k}g(x)$, 其约简过程如下:

令 $l = \min\{i \geq 1 \mid f_i = 1\}$, $s(x)$ 是 $g(x)$ 最右边的 l 个比特形成的多项式, 则 $s(x)/f(x) + g(x)$ 被 x^l 整除, 令 $t(x) = (s(x)/f(x) + g(x))/x^l$ 的次数小于 m , 则 $t(x) = g(x)x^{-l} \bmod f(x)$ 。重复此过程, 直到得出 $x^{-k}g(x) \bmod f(x)$ 。如果 l 在某个门限值(这个门限值依赖于具体的实现, 例如, 当字长是 32 比特时, l 的取值要求大于等于 32)之上, 则这个约简多项式 $f(x)$ 称为是适配(*suitable*)的。当 $f(x)$ 是适配的, 此约简步的开销会相应减少。

为了扩大适配(*suitable*)多项式的范围, 我们采用下面的两种策略。以相对较低的开销^[92], 将上一段的方法扩展到任意的 $l \leq m$ 。令 $q(x) = f_{l-1}x^{l-1} + \dots + f_1x + 1$, 首先预计算 $Q(x) (\deg Q < l)$, 使得 $Q(x)q(x) \equiv 1 \pmod{x^l}$ 。假设 $S(x) \equiv s(x)Q(x) \pmod{x^l}$, 如果 $\deg S(x) < l$, 则 $S(x)f(x) + g(x)$ 被 x^l 整除。如果 $f(x) = x^m + q(x)$, 则除以 x^l 的除法需要 l^2 个多项式乘法。另外, 我们还可以先计算 $A'(x) = x^{2m}A(x) \bmod f(x)$ 和 $C'(x) = x^k / A'(x) \bmod f(x)$, 然后计算 $C(x) = x^{2m-k}C'(x) \bmod f(x) = A^{-1}(x) \bmod f(x)$ 。此时, 计算的是 $x^{2m}A(x)$, 而不是 $A(x)$, 从而第 5 步改为寻找 $x^{2m-k}g(x)$ 即可。

显然, 算法 5.10 中的 3.1 和 3.2 步比算法 5.9 中的相应步骤更简单, 而且, $g_1(x)$ 和 $g_2(x)$ 的增长速度更慢。因此, 如果约简多项式 $f(x)$ 是适配的, 则 AIA 优于 BEA, 反之亦然。至于 BEA, 次数的计算还可以被更简单的比较代替。

5.2.2 除法运算

修改二进制欧氏求逆算法(BEA), 可以容易地得到除法运算, 即:

$$b/a = ba^{-1} \quad (5-4)$$

如果求逆运算与乘法运算的开销比 I/M 较小的话, 则直接计算除法对 ECC 的实现是更有利的, 因为在 ECC 中, 其点加运算公式(见第 2 章)涉及的是除法运算而不是求逆运算。

5.2.2.1 基于 BEA 的除法

为了得到 b/a , 我们可以修改算法 5.9 的第 2 步, 用 $g_1(x) \leftarrow b$ 代替 $g_1(x) \leftarrow 1$ 即可。当 $u=1$ 时, 算法停止, 得到 $g_1(x) = ba^{-1}$ 。

因为算法 5.9 中的 $g_1(x)$ 在第 3.1 步的几个迭代步中都定位于标准长度, 即是说, $g_1(x)$ 的初始值对整个算法的执行时间影响不大, 所以, 改进后的除法算法与算法 5.9 的求逆算法开销是相同的。

假设椭圆曲线上的点用仿射坐标表示。如果用算法 5.9 进行求逆运算, 则计算两个不同点的加法需要一个求逆运算(用 “I” 表示)和 2 个乘法运算(用 “M” 表示), 即需要的计算量是 $I+2M$; 如果用算法 5.9 进行除法运算, 则计算两个不同点的加法需要一个求逆运算(用 “I” 表示)和 1 个乘法运算(用 “M” 表示), 即需要的计算量是 $I+M$ 。当 I/M 的值较小时, 做这样的改进是很有意义的。例如假设 $I/M=3$, 则

$$\frac{(I+2M)-(I+M)}{I+2M} = \frac{M}{5M} = 0.2 \quad (5-5)$$

显然, 如果算法 5.9 用于除法运算, 将会使得点加运算的开销减少 20%。

但是, 当 $I/M > 7$ 时,

$$\frac{(I+2M)-(I+M)}{I+2M} < \frac{M}{8M} = 0.125 \quad (5-6)$$

即, 计算开销的减少量不会超过 12.5%。

因此, 如果 I/M 的比值不是较小的话, 就可能使用减少求逆运算次数的策略, 使得椭圆曲线数乘运算中的点加运算涉及相对较少的域的求逆, 以抵消使用除法运算得到的开销节省量。

5.2.2.2 基于 EEA 的除法算法

用相同的方式, 算法 5.8 可以修改用于除法运算。但是, 初始步骤的变化对除法运算的执行具有较大的影响。影响性能的两个原因主要在第 3.4 步中确定变量的长度和对 $g_1(x)$ 的加法。

在算法 5.8 中, 确定 u 和 v 的有效长度相对容易, 而且, 确定 $g_1(x)$ 和 $g_2(x)$ 的

长度也是可能的。但是,如果改为除法运算后, $g_1(x)$ 一开始就是全长(full-length),这使得其依赖较短长度的优势就不复存在了。

第二个影响性能的因素是对 $g_1(x)$ 的加法。即次数不超过 $(m-1)$ 的多项式 $g_1(x)$ 和 $g_2(x)$ 的加法。因此,如果改进为除法运算,3.4 步实现的有效性欠佳,因为 g_1 在一开始就是全长(full-length)的。

5.2.2.3 基于 AIA 的除法运算

尽管算法 5.10 与算法 5.9 是类似的,但是在有效确定 $g_1(x)$ 和 $g_2(x)$ (包括 u 和 v) 的长度方面,算法 5.10 比 5.9 具有明显的执行优势(如果 $f(x)$ 是适配(suitable)的)。

值得注意的是,算法 5.10 中对 $g_1(x)$ 和 $g_2(x)$ (包括 u 和 v) 长度的有效确定可能会产生巨大的代码扩张(大概有 t^2 个片断(fragment),而不是 BEA 的 t 个片断(fragment))。如果这种扩张是不能容忍的(因为应用或平台特征的限制),则算法 5.10 与其它算法相比,就不再具有任何优势了。

5.3 扩域 $GF(p^m)$ 上的求逆运算

设 $f(x) = x^m - w$, 对 $\forall A(x) \in GF(p^m)$, $A(x) \neq 0$, 寻找 $A^{-1}(x) \in GF(p^m)$, 使得 $A(x)A^{-1}(x) \equiv 1 \pmod{f(x)}$, 称 $A^{-1}(x)$ 为 $A(x)$ 的逆元, 寻找逆元的过程称为求逆运算。

对扩域 $GF(p^m)$, 经典的求逆算法是 Itoh-Tsujii Inversion(ITI)算法。

5.3.1 通用的求逆算法—ITI 算法

引理 5.2^[16,18] 令 $A(x) \in GF(p^m)^*$, $r = (p^m - 1)/(p - 1)$, 则 A 的乘法逆元计算如下:

$$A(x)^{-1} = (A^r(x))^{-1} A^{r-1}(x) \quad (5-7)$$

利用引理 5.2 求元素的逆元需要下面的四步:

- 1 在 $GF(q^m)$ 上求幂, 产生 A^{r-1} ;
- 2 将 A 和 A^{r-1} 相乘, 产生 $A^r \in GF(p)$;
- 3 在 $GF(p)$ 上求逆, 产生 $(A^r)^{-1}$;
- 4 将 $(A^r)^{-1}$ 和 A^{r-1} 相乘, 产生 A^{-1} 。

第 2 步和第 4 步中的计算量可以忽略不计, 因为 A 和 A^{r-1} , $(A^r)^{-1}$ 和 A^{r-1} 相乘都是在子域上进行。在大多数情况下, 子域上的计算量都小于对应扩域上的计算量。第 3 步的子域求逆运算的复杂性主要依赖于子域 $GF(q)$ 的阶和其类型, 此处不做讨论。

但是,在许多实际情况下,例如,在密码学应用中,子域很小,其上的求逆运算非常有效。所以,该算法的主要复杂度还在于第1步。

文献[16]定义的范数(norm)函数指出,任意 $\alpha \in GF(p^m)$, $\alpha^{(p^m-1)/(p-1)} \in GF(p)$,这是 ITI 算法将扩域上的求逆问题退化为子域的求逆问题的依据。

许多作者在应用 ITI 算法时^[16],通常利用查找表完成子域求逆。当 p 的值小于 2^{16} 时,在现代的台式机和工作站上,可以容易的存储这个查找表,但是 p 的值逼近 2^{32} 或者 2^{64} 次方时,这时构造的查找表就难以存储了。

5.3.2 OEFs 上的求逆运算

为了解决查找表的难以存储问题,在 OEFs 上,我们采用二进制扩展欧氏算法(前面的算法 5.3)计算子域的求逆问题,然后再与 ITI 算法结合,进行 OEFs 上的求逆运算。同时利用 Frobenius 映射和二项式 $P(x)$ 所具有的特殊形式可使得 OEFs 上的求逆运算速度得到较大提高。

5.3.2.1 OEF 上的 Frobenius 映射的性质

定义5.1 映射 $\varphi: GF(p^m) \rightarrow GF(p^m)$ 定义为: $\varphi(a) = a^p, \forall a \in GF(p^m)$, 则映射 φ 称为Frobenius映射。

注意: 如果 $a \in GF(p)$, 则 $\varphi(a) = a$ 。

显然Frobenius映射的第 i 次迭代 $\alpha \rightarrow \alpha^{p^i}$ 也是自同构的^[108]。

假设 $A(x) = a_{m-1}x^{m-1} + \dots + a_1x + a_0, a_i \in GF(p)$, 则

$$A^{p^j}(x) = a_{m-1}^{p^j} x^{(m-1)p^j} + \dots + a_1^{p^j} x^{p^j} + a_0^{p^j} \quad (5-8)$$

由Fermat小定理知, $a_i^{p^j} = a_i \bmod p$, 于是(5-7)变形为:

$$A^{p^j}(x) = a_{m-1} x^{(m-1)p^j} + \dots + a_1 x^{p^j} + a_0 \quad (5-9)$$

为了化简 x^{jp^j} ($0 < j < m$), 我们给出下面的定理。

引理 5.3 令 $f(x) = x^m - w \in GF(p)[x]$ 是不可约二项式, e 是非负整数, 并且 $x \in GF(p)[x]$, 则

$$x^e = w^q x^s \bmod f(x) \quad (5-10)$$

其中, $s \equiv e \bmod m$, $q = (e-s)/m$ 。

$$\text{引理 5.4} \quad (x^j)^{p^j} = w^q x^j \bmod f(x) \quad (5-11)$$

其中 $x^j \in GF(p)[x]$, i 是一个任意的正有理数, 其它变量的定义如引理 5.3。

在式(5-8)中, 如果给定 $f(x)$, 则 x^{jp^j} ($0 < i, j < m$) 可以先预计算。由上面的结论知, 计算 $a_j x^{jp^j}$ 仅需要一个子域乘法, 因此如果我们利用引理 5.4 和预计算

$x^{jp} (0 < j < m)$, 则由 $A(x)$ 计算 $A^{p^j}(x)$ 仅需要 $m-1$ 个子域乘法。

例 5.1 设 $p = 2^{31} - 1$, $f(x) = x^6 - 7$, 利用引理 5.4, 我们仅需要预计算 p 和 p^2 这两种情形, 具体结果如下:

$$x^p \bmod f(x) = 1513477736x$$

$$x^{p^2} \bmod f(x) = 1513477735x$$

$$x^{2p} \bmod f(x) = 1513477735x^2$$

$$x^{2p^2} \bmod f(x) = 634005911x^2$$

$$x^{3p} \bmod f(x) = -x^3$$

$$x^{3p^2} \bmod f(x) = x^3$$

$$x^{4p} \bmod f(x) = 634005911x^4$$

$$x^{4p^2} \bmod f(x) = 1513477735x^4$$

$$x^{5p} \bmod f(x) = 634005912x^5$$

$$x^{5p^2} \bmod f(x) = 634005911x^5$$

例 5.2 计算 $A(x)^{r-1}$

假设 $p = 2^{31} - 1$, $f(x) = x^6 - 7$, $r-1 = p^5 + p^4 + \dots + p$, 使用此序列计算 $A(x)^{r-1}$ 的过程见下面的表 5-1, p 和 p^2 的预计算值见例 5.1。

表 5-1 当 $m=3, 5, 6$ 时, $A(x)^{r-1}$ 的计算过程

$m=3$	$m=5$	$m=6$
$T \leftarrow a^p$	$T_1 \leftarrow a^p$	$T_1 \leftarrow a^p$
$T \leftarrow Ta = a^{p+1}$	$T_1 \leftarrow T_1 a = a^{p+1}$	$T_2 \leftarrow T_1 a = a^{p+1}$
$a^{r-1} \leftarrow T^p = a^{p^2+p}$	$T_2 \leftarrow T_1^{p^2} = a^{p^3+p^2}$	$T_3 \leftarrow T_2^{p^2} = a^{p^3+p^2}$
	$T_1 \leftarrow T_1 T_2 = a^{p^3+p^2+p+1}$	$T_2 \leftarrow T_3 T_2 = a^{p^3+p^2+p+1}$
	$a^{r-1} \leftarrow T_1^p = a^{p^4+p^3+p^2+p}$	$T_2 \leftarrow T_2^{p^2} = a^{p^3+p^4+p^3+p^2}$
		$a^{r-1} \leftarrow T_2 T_1$
开销: $1M+2\phi$	开销: $2M+2\phi$	开销: $3M+3\phi$

表 5-1 的最后一行表示计算开销, M 表示 $GF(p^m)$ 上一个乘法的开销, ϕ 表示 Frobenius 映射的迭代开销。

一般地, 如果用 $w(x)$ 表示整数 x 的汉明权重, 则计算 $A(x)^{r-1}$ 需要 $t_1(m)$ 个乘法和 $t_2(m)$ 个 Frobenius 映射的迭代。其中:

$$t_1(m) = \lfloor \log_2(m-1) \rfloor + w(m-1) - 1 \quad (5-12)$$

$$t_2(m) = \begin{cases} t_1(m) + 1, & m \text{ 是奇数} \\ j = \lfloor \log_2(m-1) \rfloor + 1 & m = 2^j \\ \lfloor \log_2(m-1) \rfloor + w(m) - 1 & \text{其他} \end{cases} \quad (5-13)$$

因为 $t_1(m)+1 > t_2(m)$, 因此 $A(x)^{r-1}$ 的计算开销由 $GF(p^m)$ 上的乘法开销决定。

5.3.2.2 OEF 上的 ITI 求逆算法

我们知道, 对任意的 $\alpha \in GF(p^m)$, $\alpha^{(p^m-1)/(p-1)} \in GF(p)$, 并且

$$A^{-1}(x) = (A^r)^{-1}(x)A^{r-1}(x), r = \frac{p^m-1}{p-1} \quad (5-14)$$

因此, 根据式(5-14), 我们可得到下面的算法 5.11。

算法 5.11 OEFs 求逆算法

输入: $A(x), f(x) \in GF(p)[x]$

输出: $A^{-1}(x) \in GF(p^m)$ 使得 $A(x)A^{-1}(x) = 1 \pmod{f(x)}$

- 1 $A^{-1}(x) \leftarrow A(x)$
- 2 使用加法链方式计算 $A^{-1}(x) \leftarrow (A^{-1}(x))^{r-1}$
- 3 $C \leftarrow A^{-1}(x)A(x)$
- 4 通过 $GF(p)$ 上的求逆算法得到 C^{-1} , 使得 $CC^{-1} = 1 \pmod{p}$
- 5 返回 $C^{-1}A^{r-1}(x)$

算法 5.11 的计算量主要在第 2 步, 即需要计算

$$A(x)^{-1} = (A^r(x))^{-1}A^{r-1}(x) \pmod{P(x)} \quad (5-15)$$

其中

$$r = (p^m - 1)/(p - 1) = p^{m-1} + \dots + p^2 + p + 1 \quad (5-16)$$

将 r 的这个表达式(5-16)看成 r 的 p 进制表示, 即 $r-1 = (11\dots 10)_p$ 。当有限域给定后, $r-1$ 也就确定了, 此时算法 5.11 的主要计算量就转化为求幂运算。对求幂运算, 通常采用的方法是加法链方式。

5.4 小结

本章首先介绍了有限域 $GF(p)$ 的各种求逆算法; 然后综述了其扩域 $GF(p^m)$ ($p \geq 2$) 上的各种求逆运算算法, 同时讨论了二元扩域上的求逆算法修改为除法运算时对 ECC 点加算法的性能影响; 最后对一种特殊的扩域—OTFs 上的求逆运算进行了分析研究。本章的进一步工作是对我们提出的 GOTFs 上的求逆运算进行研究, 以期得到较简单的求逆算法, 从而为 GOTFs 上的运算建立完整的体系。

第六章 串、并行乘法器设计

有限域乘法运算极大地影响各种密码算法的加/解密速度,设计时间和空间复杂性低的乘法运算算法和乘法器变得尤其重要。根据域元素的不同表示方法,乘法器可分为多项式基乘法器、对偶基乘法器和正规基乘法器。本章主要讨论基于正规基乘法器的串、并行性设计。

首先讨论并行性设计的基本思想;接着讨论正规基乘法器的各种串、并行算法设计,复杂性分析,通过增加 XG 数量达到减少 AG 数量的方式,提出了一个串行乘法器和一个并行乘法器;最后给出一种 II 型最优正规基串行乘法器算法设计,该乘法器需要 $(2m-2)$ 个 XG, m 个 AG。

6.1 并行性设计

并行性是指问题中具有可同时运算或者操作的特性^[9]。开发并行性的目的是为了能更好地进行并行处理,以提高计算机的处理效率。例如,在相同时间的元器件条件下,采用 n 位运算器进行 n 位并行运算的速度几乎是用 1 位运算器进行 n 位串行运算的 n 倍。

并行性可以有不同的等级,而且从不同的角度看,等级的分法也不一样。

从计算机系统中执行程序的角度来看,并行性从低到高可分为:

- 指令内部的并行
- 指令之间的并行
- 任务之间的并行
- 作业之间的并行

从计算机系统中处理数据的并行性来看,并行性从低到高可分为:

- 位串行字符串行
- 位并行字符串行
- 位片串行字符串行
- 位并行和字符串行

从信息加工的步骤和阶段来看,并行性从低到高可分为:

- 存储器操作并行

- 处理器操作步骤并行
- 处理器操作并行
- 指令、任务、作业并行

另外,并行性在不同的处理级别中可表现为多种形式:如先行方式、流水方式、向量化、并发性和同时性等。

为提高计算机系统的并行性,可以通过下面的三种方式达到:

- (1) 时间重叠 即在并行性概念中引入时间因素,让多个处理过程在时间上相互错开,轮流重叠地使用同一套硬件设备的各个部分,以加快硬件周转而赢得速度;
- (2) 资源重复 资源重复是在并行性概念中引入空间因素,通过重复设置硬件资源来提高可靠性或性能;
- (3) 资源共享 资源共享则是利用软件的方法让多个用户按一定时间顺序轮流地使用同一套资源,以提高其利用率,这样相应地也可以提高整个系统的性能。

为实现并行性,必须综合考虑硬件、软件以及它们之间的匹配等因素。硬件并行性是指机器结构和硬件多样性所决定的并行性类型,它通常是价格和性能折衷的函数,反映同时可执行操作的资源利用方式。软件并行性是由程序的控制和数据的相关性决定的,它是算法、程序设计风格和编译器优化的函数。为解决软件和硬件的并行匹配问题,一种方法是加强编译支持,另一种是重新设计硬件。

软件并行设计时需要依赖一定的计算模型。并行计算的理论模型是从物理模型抽象得到的,它们为开发并行算法提供一种方便的框架而无需顾及实现细节或物理的约束条件。当前最重要和典型的几种并行计算模型包括 PRAM (Parallel Random Access Machine) 模型^[92]、BSP (Bulk Synchronous Parallel) 模型^[93]和 LogP 模型^[94],贯穿其发展的一条主线是如何准确地反映并行机特有的通信开销。这几种模型一般用于同构系统,近年发展的异构系统,包括网格系统,其计算模型一般是基于 BSP 模型的扩展。关于并行计算模型的详细内容还可参见文献[95-99]。

6.2 多项式基乘法器

假设 $A(x)$, $B(x)$, $f(x) \in GF(2)[x]$, $C(x) = A(x) \times B(x) \bmod f(x)$, 则计算 $C(x)$ 的并行多项式基乘法器最早由 Bartee 和 Schneider^[45]提出,其实现需要 $GF(2)$ 上的 $(m^3 - m)$ 个二值输入的 XG ^[46], 其中 m 是其依赖的不可约多项式的次数。因其较高的空间

复杂性和缺乏规则性，在具体实现时，通常不采用这种乘法器。

6.2.1 Mastrovito 并行 PB 乘法器

Mastrovito^[47,48]提出了一种并行多项式基乘法器。该乘法器由 f -network 和 $IP(m)$ 两部分构成，其对应的硬件体系结构见图 6-1。

因为 $f(x) \in GF(2^m)[x]$ ，所以 f -network 的复杂性与使用的不可约多项式有关，因此我们仅根据 BSP 模型分析 $IP(m)$ 部分的空间复杂性和时间复杂性。在 $IP(m)$ 部分，首先是 b_j 和 $f_{i,j}(j=0, \dots, m-1)$ 相乘，需要 m 个 AG 在一个 T_A 时间步内完成，其中 T_A 表示一个 AG 的时间延迟；接着用 $(m-1)$ 个 XG 在 $\lceil \log_2 m \rceil$ 步内完成 m 个乘积的加法，该计算过程的计算时间为 $\lceil \log_2 m \rceil T_X$ ，其中， T_X 表示一个 XG 的时间延迟。

综上所述，该并行乘法器需要 $m(m-1)$ 个 XG 和 m^2 个 AG，其时间复杂性为 $T_A + \lceil \log_2 m \rceil T_X$ 。

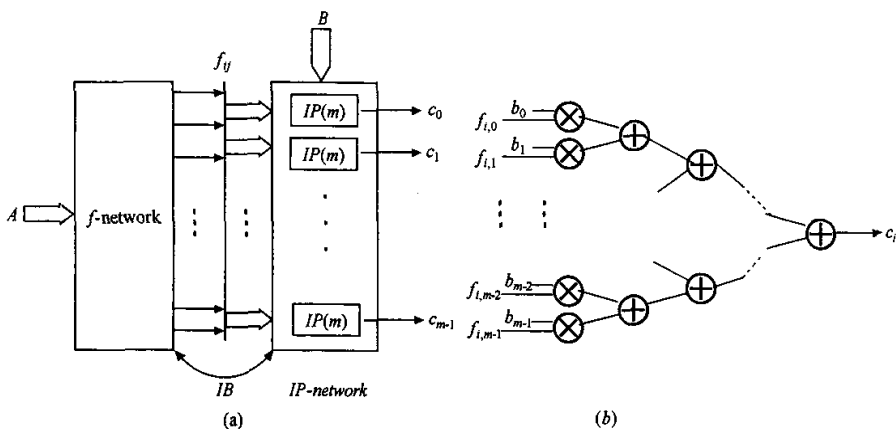


图 6-1 (a) $GF(2^m)$ 上的 Mastrovito 乘法器体系结构 (b) $IP(m)$ 的详细结构

6.2.2 基于特殊多项式的 Mastrovito 并行 PB 乘法器

Sunar 和 Koç^[49]利用三项式，对 Mastrovito 算法提出了一个新的公式，指出其对应的乘法器需要 (m^2-1) 个 XG 和 m^2 个 AG。Halbutogullari 和 Koç^[50]推广了 Sunar 和 Koç 的思想，对任意不可约多项式，找到了构造 Mastrovito 乘法器的方法。迄今为止，对这些特定的多项式，就 XG 的个数和时间延迟来说，Halbutogullari 和 Koç 算法的时间复杂度是最低的。Zhang 和 Parhi^[51]提出一种设计 Mastrovito 乘法器的对称方法，而且，他们将这种方法应用于设计改进的 Mastrovito 乘法方案^[52]。对两类不可约五项式，他们提出了新的 Mastrovito 乘法器的复杂性结果。

与 Mastrovito 乘法器相同, 一个 $GF(2^m)$ 上的乘法可以先直接相乘, 随后再进行取模运算, 许多文献如[53]等就采用这种方式。作者 Wu^[53]使用不可约三项式作为约简多项式, 并指出 $GF(2^m)$ 上的一个取模运算需要 $(w-1)(m-1)$ 个加法, 其中, w 是不可约多项式的汉明权重。在硬件实现乘法运算时, 需要 $((m^2-1) + (w-1)(m-1))$ 个 XG 和 m^2 个 AG。最近, Rodriguez, Henriquez 和 Koç^[54]对特殊的五项式提出了一个 PB 乘法器, 并得出其时间延迟和所需要的 XG 和 AG 的个数。尽管作者都称其乘法器为 Mastrovito 乘法器, 但是他们的体系结构与最初的 Mastrovito 乘法器是完全不同的, 其体系结构是单独使用两步乘法完成的。

6.2.3 Karatsuba 乘法器

大多数有限域上的乘法运算算法都是分两步完成: 第一步是多项式乘法, 第二步是取模运算。即假设 $A(x), B(x) \in GF(2^m)$, 首先计算

$$C(x) = B(x)A(x) = \left(\sum_{i=0}^{m-1} a_i x^i \right) \left(\sum_{i=0}^{m-1} b_i x^i \right) \quad (6-1)$$

然后计算

$$C'(x) = C(x) \bmod P(x) \quad (6-2)$$

第一步通常采用经典(Classic)的多项式乘法算法, Karatsuba 算法, 以及二者的混合运用算法, 具体可参见论文第四章。第二步通常采用的取模多项式有: 等距多项式(Equally-spaced Polynomial), 三项式和五项式。具体可参见论文第 3 章和文献[6]。

Karatsuba 算法由 Karatsuba 在 1962 年提出, 它是第一个能够在低于 $O(m^2)$ 个操作内完成的多项式乘法[6,28], 其时间复杂度为 $O(m^{\log_2 3})$, 其基本思想为:

假设

$$\begin{aligned} A(x) &= \sum_{i=0}^{m-1} a_i x^i = \sum_{i=m/2}^{m-1} a_i x^i + \sum_{i=0}^{m/2-1} a_i x^i \\ &= x^{m/2} \sum_{i=0}^{m/2-1} a_{i+m/2} x^i + \sum_{i=0}^{m/2-1} a_i x^i = x^{m/2} A^H + A^L \\ B(x) &= \sum_{i=0}^{m-1} b_i x^i = \sum_{i=m/2}^{m-1} b_i x^i + \sum_{i=0}^{m/2-1} b_i x^i \\ &= x^{m/2} \sum_{i=0}^{m/2-1} b_{i+m/2} x^i + \sum_{i=0}^{m/2-1} b_i x^i = x^{m/2} B^H + B^L \end{aligned}$$

则

$$\begin{aligned}
C(x) &= x^m A^H B^H + \left(A^H B^H + A^L B^L + (A^H + A^L)(B^H + B^L) \right) x^{\lceil m/2 \rceil} + A^L B^L \\
&= x^m C^H + C^L.
\end{aligned}$$

最好的结果是结合经典算法和 Karatsuba 策略而得到, 该 PB 乘法器的空间复杂度和时间复杂度如下:

$$XOR \text{ gates} \leq \left(\frac{m}{n} \right)^{\log_2^3} (n^2 + 6n - 1) - 8m + 2$$

$$AND \text{ gates} \leq \left(\frac{m}{n} \right)^{\log_2^3} n^2$$

$$Time \text{ Delay} \leq T_{AND} + T_{XOR} (\log_2^n + k)$$

其中 $m = 2^k n$ 。

6.3 正规基乘法器

在域元素的正规基表示下, 域元素的平方运算仅需对该元素的坐标进行简单循环移位即可, 在硬件实现平方运算时, 其开销可以忽略不计, 因此, 基于正规基表示的乘法运算及其乘法器设计得到了大量研究, 具体可参见文献[22,24,25,35,37,38,42,115-120]。

一般地, 记 $A = (a_0, a_1, \dots, a_{m-1})$, a_i 表示 A 的第 i 个坐标。

令 $\bar{a} = [a_0, a_1, \dots, a_{m-1}]$ 和 $\bar{b} = [b_0, b_1, \dots, b_{m-1}]$ 分别对应域元素 $A = (a_0, \dots, a_{m-1})$ 和 $B = (b_0, b_1, \dots, b_{m-1})$ 的行向量, $C = (c_0, c_1, \dots, c_{m-1}) \in GF(2^m)$ 表示它们的乘积, 则

$$C = AB = \left(\sum_{i=0}^{m-1} a_i \gamma^{2^i} \right) \left(\sum_{j=0}^{m-1} b_j \gamma^{2^j} \right) \quad (6-3)$$

$$c_i = \bar{a} M^{(i)} \bar{b}^T \quad (6-4)$$

其中, 矩阵 $M^{(i)}$ 可通过 i 次循环右移和 i 次循环下移矩阵 $M^{(0)} = M$ 得到, 而 M 的计算方法见文献[30, 119]; \bar{b}^T 表示 \bar{b} 的转置。

6.3.1 实例

设 $f(x) = x^5 + x^2 + 1$ 是不可约多项式(正规多项式), 其根为 α , 由 $f(x)$ 产生的有限域是 $GF(2^5)$ 。如果选择 $\gamma = \alpha^5$, 则 $N = \{\gamma, \gamma^2, \gamma^{2^2}, \gamma^{2^3}, \gamma^{2^4}\}$ 就是一个正规基。使用文献[36]的方法, 可得到 M 如下所示:

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

从而, 由(6-4)得

$$c_0 = (a_0b_1 + a_1b_0) + (a_1b_3 + a_3b_1) + (a_2b_4 + a_4b_2) + (a_3b_2 + a_2b_3) + a_4b_4 \quad (6-5)$$

另外, 如果保持 M 不变, 分别将 \bar{a} , \bar{b} 左移 i 次, 记为: $\bar{a} \ll i$ 和 $\bar{b} \ll i$, 则

$$c_i = (\bar{a} \ll i) M (\bar{b} \ll i)^T \quad (6-6)$$

文献[25]的算法 1 就是根据(6-6)式得到。

与其它有限域乘法器一样, 正规基乘法器的硬件执行也分为并行乘法器和串行乘法器。首先分析并行乘法器。

6.3.2 并行正规基乘法器

假设 $A, B \in GF(2^m)$, 且 $A = (a_0, a_1, \dots, a_{m-1})$, $B = (b_0, b_1, \dots, b_{m-1})$ 。当乘法器接收到 A 和 B 的 $2m$ 个比特数据后, 经过各个逻辑门的延迟或者存储器存取延迟, $C = AB$ 的积所对应的 m 个比特数据同时输出。如图 6-2 就是并行乘法器执行 6.3.1 节实例的情形。

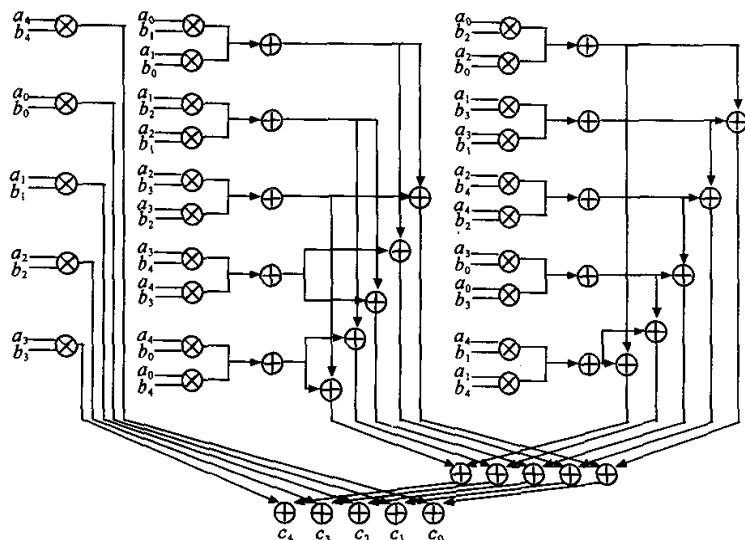


图 6-2 关于 6.3.1 节实例并行正规基乘法器

由图 6-2 可以看出, $GF(2^5)$ 上两个域元素的乘积, 需要 25 个 AG, 30 个 XG。一般地, 对 $GF(2^m)$ 上正规基下两个域元素的乘积, 需要 m^2 个 AG 和 $m(m+1)$ 个 XG, 其时间复杂性为 $(T_A + (2 + \lceil \log_2 m \rceil)T_X)$ 。而 Massey 和 Omura 提出的正规基乘法器却需要 m^2 个 AG 和 $2m(m-1)$ 个 XG, 其时间复杂性为 $(T_A + (1 + \lceil \log_2(m-1) \rceil)T_X)$ 。

可见, 不管是前面的哪一种乘法器, 当 m 很大时, 并行乘法器都需要大量的硬件设备。在密码学应用中, m 的值通常很大, 例如, ECC 要求 $m \geq 160$, RSA 要求 $m \geq 1024$ 。因此, 在硬件条件有限的情况下, 建立并行乘法器是不切实际的。

6.3.3 串行正规基乘法器

6.3.3.1 位-级串行输出的串行乘法器 (SMSO)

在 SMSO 中(如图 6-3 所示^[32,38]), A 和 B 的坐标首先串行加载到移位存储器中, 然后, 在每一个时钟循环里, 通过存储器的循环移位, 由 u 函数产生的乘积 C 的坐标从 c_{m-1} 到 c_0 依次得到。

一般地, SMSO 乘法器中 XG 和 AG 的个数分别是 C_N 和 $C_N - 1$, 其关键路径延迟为 $T_A + \lceil \log_2 C_N \rceil T_X$, 其中 T_A 和 T_X 分别是一个 AG 和一个 XG 的延迟, C_N 表示矩阵 M 中非零元的个数。

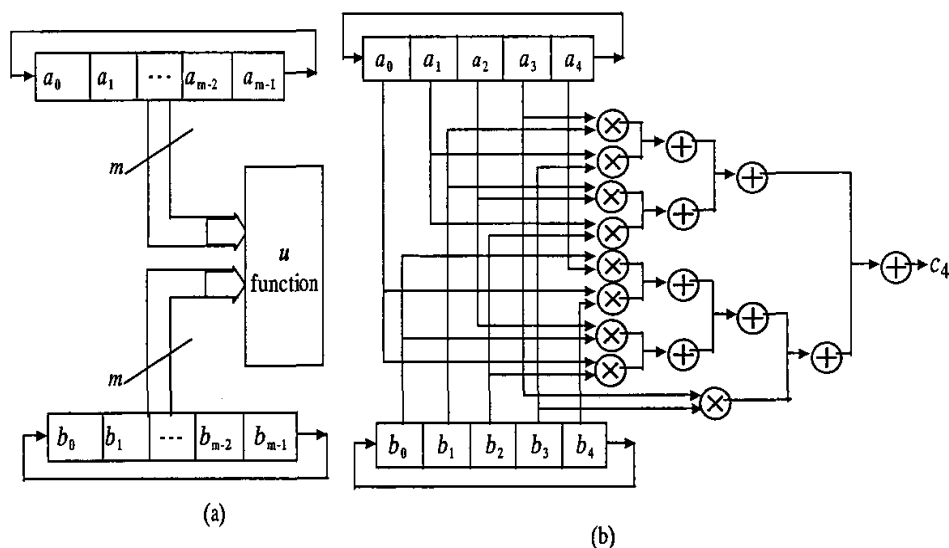


图 6-3 (a) $GF(2^m)$ 上的 Massey-Omura 按位 SMSO
(b) 关于 6.3.1 节实例的 Massey-Omura 乘法器

6.3.3.2 位-级并行输出的串行乘法器 (SMPO)

在文献[100], Agnew 等提出了另一种正规基乘法器。经过 m 个时钟循环后, 该乘法器并行输出 C 的 m 个坐标。就该文献 6.3.1 节中的例子, 图 6-4 给出了相应乘法器体系结构。

一般地, SMPO 乘法器的 XG 和 AG 个数分别为 m 和 C_N , 其关键路径延迟为 $T_A + (1 + \lceil \log_2 \rho \rceil) T_X$, 其中 ρ 是式(6-4)中含 a_i 项的最大个数, 即 ρ 为乘法矩阵 M 的所有行(或列)中含 1 的最大个数。显然 $2 \leq \rho \leq m$ 。对 II 型最优正规基, $\rho = 2$, 所以其关键路径延迟是 $T_A + 2T_X$, 而其它类型的正规基, 其延迟小于等于 $T_A + (1 + \lceil \log_2 \rho \rceil) T_X$ 。

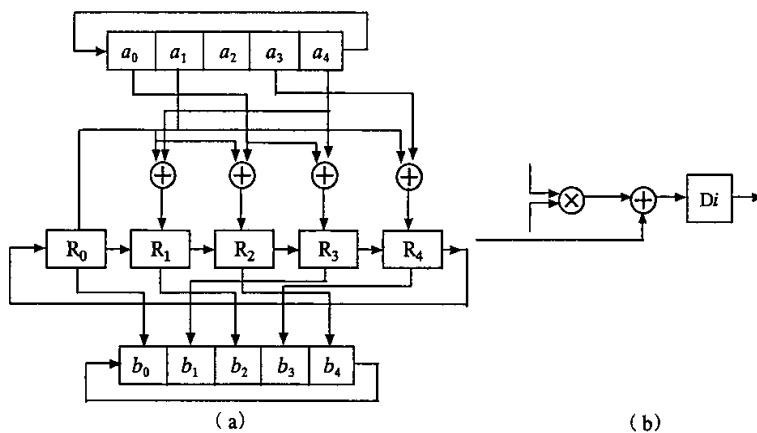


图 6-4 (a) Agnew et al 在 $GF(2^5)$ 上的 SMPO (b) 设计 R_i 的细节

6.3.4 Reyhani-Masoleh 和 Anwar Hasan 所做工作

A, B, C 的假设与前面一致。Reyhani-Masoleh 和 Anwar Hasan 修改了文献[32,36] 的公式得下面的公式:

$$C = \left((F_{m-1}^2 + F_{m-2})^2 + \cdots + F_1 \right)^2 + F_0 \quad (6-7)$$

$$F_i = a_{i-g} b_{i-g} \gamma + \sum_{j=1}^v z_{i,j} \delta_j$$

其中,

$$z_{i,j} = \begin{cases} (a_i + a_{i+j})(b_i + b_{i+j}), & \text{if } g = 0 \\ a_i b_{i+j} + a_{i+j} b_i, & \text{if } g = 1 \end{cases}$$

仍利用 6.3.1 的实例, 但只考虑 $g=0$ 的情形。

已知 $\delta_1 = \gamma + \gamma^{2^3}$, $\delta_2 = \gamma^{2^3} + \gamma^{2^4}$, 则

$$\begin{aligned} F_i &= a_i b_i \gamma + (a_i + a_{i+1})(b_i + b_{i+1})(\gamma + \gamma^{2^3}) + (a_i + a_{i+2})(b_i + b_{i+2})(\gamma^{2^3} + \gamma^{2^4}) \\ &= (a_i b_i + (a_i + a_{i+1})(b_i + b_{i+1}))\gamma + [(a_i + a_{i+1})(b_i + b_{i+1}) + (a_i + a_{i+2})(b_i + b_{i+2})]\gamma^{2^3} + (a_i + a_{i+2})(b_i + b_{i+2})\gamma^{2^4} \end{aligned}$$

即:

$$\begin{aligned} F_i &= (a_i b_i + (a_i + a_{i+1})(b_i + b_{i+1}), 0, 0, (a_i + a_{i+1})(b_i + b_{i+1}) \\ &\quad + (a_i + a_{i+2})(b_i + b_{i+2}), (a_i + a_{i+2})(b_i + b_{i+2})), \quad 0 \leq i \leq 4 \end{aligned}$$

以 $F_4=(D_0, D_1, D_2, D_3, D_4)$ 为例, 得到下面的正规基乘法器, 如图 6-5 所示。

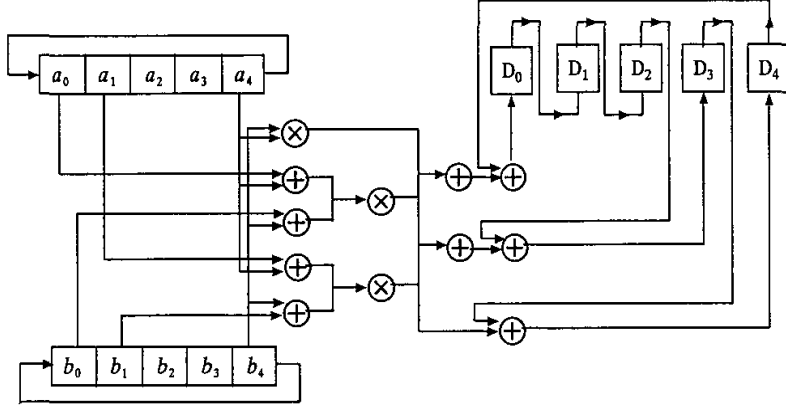


图 6-5 关于 6.3.1 节实例改进的 b -SMPO_I

容易证明, b -SMPO_I 关键路径延迟上界是 $T_A + (1 + \lceil \log_2(m+4) \rceil)T_X$, 令 ρ 是用正规基表示的 m 个坐标中非零项的最大个数, 则其关键路径延迟为 $T_A + (1 + \lceil \log_2 \rho \rceil)T_X$ 。对最优正规基, $\rho=3$, 此时, 延迟为 $T_A + 3T_X$ 。

6.3.5 改进的正规基乘法器

引理 6.1^[100] 令 $A, B \in GF(2^m)$, $C = AB$, 则

$$C = \sum_{i=0}^{m-1} a_i b_i \gamma^{2^i} + \sum_{j=1}^v \sum_{k=1}^{h_j} \left(\sum_{i=0}^{m-1} y_{((i-w_{j,k}), j)} \gamma^{2^i} \right) \quad (6-8)$$

其中,

$$y_{i-w_{j,k}, j} = (a_{i-w_{j,k}} + a_{i-w_{j,k}+j})(b_{i-w_{j,k}} + b_{i-w_{j,k}+j})$$

$h_j (1 \leq j \leq v)$ 是 $\delta_j = \gamma^{1+2^j}$ 表示为正规基形式时非零坐标的个数, $v = \lfloor m/2 \rfloor$ 。

将上面的(6-8)式改写为:

$$C = \sum_{i=0}^{m-1} \left(a_i b_i + \sum_{j=1}^v \left(\sum_{k=1}^{h_j} y_{((i-w_{j,k}), j)} \right) \right) \gamma^{2^i} \quad (6-9)$$

$$\text{即 } c_i = \left(a_i b_i + \sum_{j=1}^v \sum_{k=1}^{h_j} y_{((i-w_{j,k}), j)} \right)$$

仍以 6.3.1 节的例子说明如下: 因为 $\delta_1 = \gamma + \gamma^{2^3}$, $\delta_2 = \gamma^{2^3} + \gamma^{2^4}$, 所以

$$\begin{aligned} c_i &= a_i b_i + y_{i,1} + y_{i-3,1} + y_{i-3,2} + y_{i-4,2} \\ &= a_i b_i + (a_i + a_{i+1})(b_i + b_{i+1}) + (a_{i-3} + a_{i-2})(b_{i-3} + b_{i-2}) \\ &\quad + (a_{i-3} + a_{i-1})(b_{i-3} + b_{i-1}) + (a_{i-4} + a_{i-2})(b_{i-4} + b_{i-2}), (0 \leq i \leq 4) \end{aligned} \quad (6-10)$$

令 $i=4$, 有

$$\begin{aligned} c_0 &= a_0 b_0 + (a_0 + a_1)(b_0 + b_1) + (a_2 + a_3)(b_2 + b_3) \\ &\quad + (a_2 + a_4)(b_2 + b_4) + (a_1 + a_3)(b_1 + b_3) \end{aligned}$$

根据(6-10), 可得到下面的串行输出的串行乘法器(SMSO), 见图 6-6。

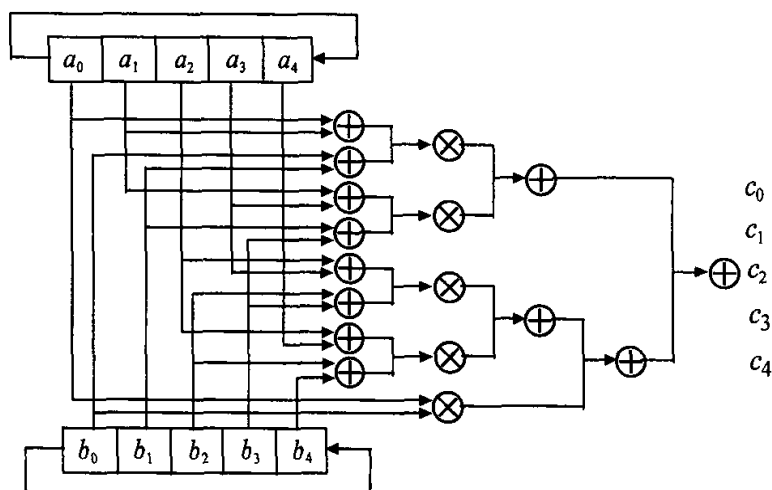


图 6-6 关于 6.3.1 节实例改进的-SMPO₁

在图 6-6 中, 每经过一个时钟循环, 就得到 C 的一个坐标。在整个乘法器中, 共使用了 12 个 XG, 5 个 AG。

另外, 我们还可以得到下面的并行输出的并行乘法器(PMPO), 如图 6-7。

与图 6-2 比较, 可以看出, 修改后的并行正规基乘法器使用加法代替乘法, 仅

需要 15 个 AG, 40 个 XG 就能并行得到结果。对同样的例子, 文献[32]则需要 25 个 AG, 30 个 XG。改进的 PMPO_I 的关键路径延迟上界是 $T_A + (1 + \lceil \log_2(m+3) \rceil) T_X$ 。

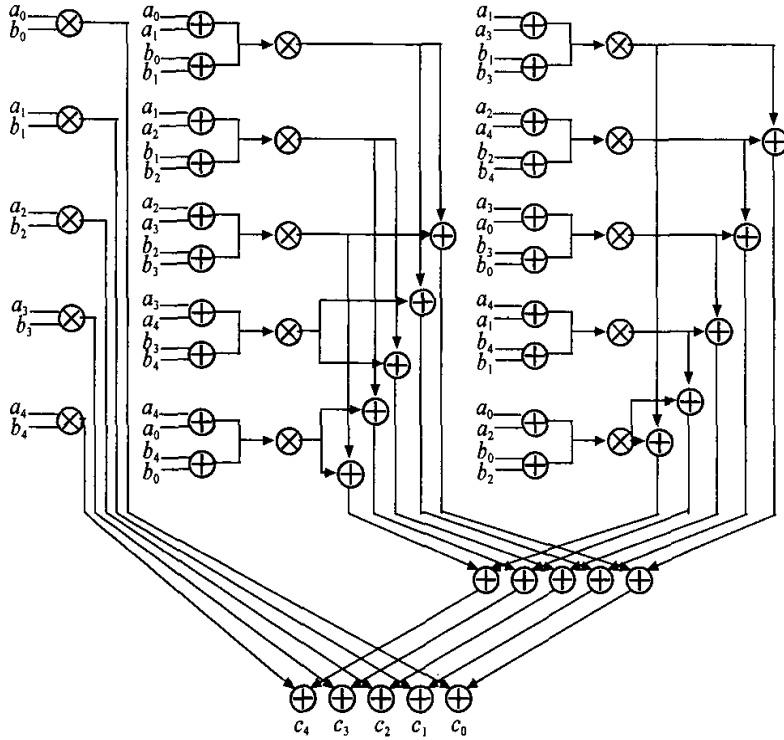


图 6-7 关于 6.3.1 节实例改进的 -PMPO

针对特殊的 II 型最优正规基, 我们还得到了一种改进的串行正规基乘法器的算法设计。

6.4 II 型最优正规基串行乘法器算法设计

6.4.1 相关工作

Yan 在其文献[8]的表格 5.1 中指出, 当 $m \in [2, 2001]$ 时, 有 117 个 m 的值是 I 型最优正规基, 319 个 m 的值是 II 型最优正规基。显然, II 型最优正规基的数量几乎是 I 型最优正规基数量的 3 倍。这个例子说明, 研究 II 型最优正规基是非常有意义的。

近年来, 对特殊的有限域提出了一些有效的正规基乘法器^[34,35], 但是这些乘法器仅对 I 型正规基有效。尽管 Massey-Omura 乘法器^[33]对 I 型和 II 型最优正规基

都有效，但是，其所需要的空间复杂度较高，XG 个数为 $2m(m-1)$ 。基于 Massey-Omura 乘法器，作者 B.Sunar and .K.Ko^[37]在 II 型最优正规基上提出了一个并行正规基乘法器，其所需 XG 数为 $1.5m(m-1)$ ，AG 数为 m^2 。

在这一部分，我们提出了一个 II 型最优正规基串行乘法器，该乘法器要求 $(2m-2)$ 个 XG， m 个 AG。

6.4.2 算法设计

由定理 2.13 知， $M = \{\beta + \beta^{-1}, \beta^2 + \beta^{-2}, \dots, \beta^{2^{m-1}} + \beta^{-2^{m-1}}\}$ 和 $N = \{\beta + \beta^{-1}, \beta^2 + \beta^{-2}, \dots, \beta^m + \beta^{-m}\}$ 是等价的。因此，对任意 $A \in GF(2^m)$ ，

如果用基 M 表示，有

$$A = a_1' \gamma + a_2' \gamma^2 + \dots + a_m' \gamma^{2^{m-1}} \quad (6-11)$$

其中 $\gamma = \beta + \beta^{-1}$ 。

如果用基 N 表示，有

$$A = a_1 \gamma_1 + a_2 \gamma_2 + \dots + a_m \gamma_m \quad (6-12)$$

其中 $\gamma_i = \beta^i + \beta^{-i}$ 。

由定理 2.13，我们还知道 A 关于基 M 和 N 表示的系数 a_j ， a_i' 之间的关系为：

$$a_j = a_i'$$

其中，

$$j = \begin{cases} k & \text{if } k \in [1, m] \\ (2m+1) - k & \text{if } k \in [m+1, 2m] \end{cases} \quad (6-13)$$

$$k = 2^{i-1} \pmod{2m+1} \quad (i=1, 2, \dots, m)$$

显然，利用(6-13)式可将域元素的正规基表示转换为多项式基表示，在多项式基下做乘法运算，然后再利用(6-13)式将结果转换为正规基表示。

$A, B \in GF(2^m)$ 用多项式基 N 表示为：

$$A = \sum_{i=1}^m a_i \gamma_i = \sum_{i=1}^m a_i (\beta^i + \beta^{-i}) \quad (6-14)$$

$$B = \sum_{i=1}^m b_i \gamma_i = \sum_{i=1}^m b_i (\beta^i + \beta^{-i}) \quad (6-15)$$

则二者的乘积 C 为：

$$C = AB = \left(\sum_{i=1}^m a_i (\beta^i + \beta^{-i}) \right) \left(\sum_{j=1}^m b_j (\beta^j + \beta^{-j}) \right) \quad (6-16)$$

进一步,

$$C = \sum_{k=1}^m \left(\sum_{|i \pm j|=k, 1 \leq i, j \leq m} a_i b_j (\beta^k + \beta^{-k}) \right) + \sum_{k=m+1}^{2m} \left(\sum_{i+j=k, 1 \leq i, j \leq m} a_i b_j (\beta^k + \beta^{-k}) \right) \quad (6-17)$$

由于 $\beta^{2m+1} = 1$, 所以 $\beta^j + \beta^{-j} = \beta^{(2m+1)-j} + \beta^{-(2m+1)+j}$, 因此(6-17)可进一步改写为:

$$\begin{aligned} C &= \sum_{i=1}^m a_i \left(\sum_{k=1}^m \left(\left(\sum_{|i \pm j|=k, 1 \leq j \leq m} b_j \right) + \left(\sum_{i+j=2m+1-k, 1 \leq j \leq m} b_j \right) \right) (\beta^k + \beta^{-k}) \right) \\ &= \sum_{i=1}^m a_i \left(\sum_{k=1}^m (b_{|i-k|} + b_{(i+k)}) (\beta^k + \beta^{-k}) \right) = \sum_{k=1}^m \left(\sum_{i=1}^m a_i (b_{|i-k|} + b_{(i+k)}) \right) (\beta^k + \beta^{-k}) \end{aligned} \quad (6-18)$$

其中

$$(i+k) = \begin{cases} i+k & i+k \leq m \\ 2m+1-(i+k) & i+k > m \end{cases}$$

于是根据(6-18)和矩阵乘法知识, 可将(6-18)写成如下矩阵:

$$C = NT(a_1, \dots, a_m)^T = N(T_1 + T_2)(a_1, \dots, a_m)^T \quad (6-19)$$

其中

$$\begin{aligned} T &= \begin{pmatrix} b_2 & b_1+b_3 & \cdots & b_{m-2}+b_m & b_{m-1}+b_m \\ b_1+b_3 & b_4 & \cdots & b_{m-3}+b_m & b_{m-2}+b_{m-1} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ b_{m-2}+b_m & b_{m-3}+b_m & \cdots & b_3 & b_1+b_2 \\ b_{m-1}+b_m & b_{m-2}+b_{m-1} & \cdots & b_1+b_2 & b_1 \end{pmatrix} \\ T_1 &= \begin{pmatrix} b_0 & b_1 & \cdots & b_{m-2} & b_{m-1} \\ b_1 & b_0 & \cdots & b_{m-3} & b_{m-2} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ b_{m-2} & b_{m-3} & \cdots & b_0 & b_1 \\ b_{m-1} & b_{m-2} & \cdots & b_1 & b_0 \end{pmatrix}, \quad T_2 = \begin{pmatrix} b_2 & b_3 & \cdots & b_m & b_m \\ b_3 & b_4 & \cdots & b_m & b_{m-1} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ b_m & b_m & \cdots & b_3 & b_2 \\ b_m & b_{m-1} & \cdots & b_2 & b_1 \end{pmatrix}, \quad b_0 = 0 \end{aligned}$$

由(6-19), 我们可以得到矩阵 T, T_1, T_2 的相关性质:

- 1 T, T_1, T_2 是 m 阶的对称矩阵;
- 2 T_1 的主对角线元素全为 b_0 , 各行以 b_0 为中心, b 的下标依次向两边递增;
- 3 T_2 的副对角线元素和副对角线左上次副对角线元素全为 b_m , 各行以它们为中心, b 的下标依次向两边递减。

根据以上性质, 假设

$$D[1..2m+1] = (b_0, b_1, \dots, b_{m-1}, b_m, b_m, b_{m-1}, \dots, b_1)$$

则我们可以用下面的算法求出乘积 C 的值。

算法 6.1

输入: $A, B \in GF(2^m), D[1..2m+1]$

输出: $C = AB$

1 初始化 $C = (c_1, c_2, \dots, c_m) = (0, 0, \dots, 0)$

2 For $j=1$ to m

{

3 $D_1[1..m] = D[1..m], D_2[1..m] = D[2m..m+1]$

4 $R = A \otimes (D_1[1..m] \oplus D_2[1..m])$

5 $C = C + R$

6 $D[1..2m+1] \gg 1$

}

7 Return C

其中:

$$A \otimes B = (a_1, a_2, \dots, a_m) \otimes (b_1, b_2, \dots, b_m) = (a_1 b_1, a_2 b_2, \dots, a_m b_m),$$

$$A \oplus B = (a_1, a_2, \dots, a_m) \oplus (b_1, b_2, \dots, b_m) = (a_1 + b_1, a_2 + b_2, \dots, a_m + b_m)。$$

6.4.3 算法复杂性分析

改进的算法首先计算 $D_1[1..m] \oplus D_2[1..m]$, 由于 $D_1[1..m]$ 中有一个元素为 0, 所以仅需要 $(m-1)$ 个二值输入的 XG, 因是并行执行的, 因此仅要求单个的 XG 延迟 T_X ; $A \otimes (D_1[1..m] \oplus D_2[1..m])$ 需要 m 个二值输入的 AG, 同样因为是并行执行, 所以仅要求单个的 AG 延迟 T_A ; 最后将 $A \otimes (D_1[1..m] \oplus D_2[1..m])$ 的并行结果相加, 需要 $m-1$ 个二值输入的 XG, 其延迟为 $\lceil \log_2^m \rceil T_X$, 这样, 计算单个 c_i 值的延迟为 $T_A + (1 + \lceil \log_2^m \rceil) T_X$ 。从而, 计算出乘积 C 的时间延迟为 $m(T_A + (1 + \lceil \log_2^m \rceil) T_X)$, 共需要 $(2m-2)$ 个二值输入的 XG, m 个二值输入的 AG。在进行具体的硬件实现时, 所需要的存储空间为 $3m+1$ 。

6.4.4 实例

下面以一个具体的例子来描述实现过程：

为了便于比较，我们仍然使用 6.3.1 节的例子。

对有限域 $GF(2^5)$ ，因为 $2m+1=2 \times 5+1=11$ 是素数，且 2 是模 11 本原根，所以由定理 2.11，在 $GF(2^5)$ 可以构造 II 型最优正规基，即，存在正规基元素 $\gamma \in GF(2^5)$ ，其中， $\gamma = \beta + \beta^{-1}$ ，使得基 $M = \{\gamma, \gamma^2, \gamma^{2^2}, \gamma^{2^3}, \gamma^{2^4}\}$ 是 $GF(2^5)$ 的 II 型最优正规基。利用 $\beta^{11}=1$ ，我们有

$$\begin{aligned}\gamma &= \beta + \beta^{-1} = \beta + \beta^{-1} = \gamma_1 \\ \gamma^2 &= \beta^2 + \beta^{-2} = \beta^2 + \beta^{-2} = \gamma_2 \\ \gamma^{2^2} &= \beta^{2^2} + \beta^{-2^2} = \beta^4 + \beta^{-4} = \gamma_4 \\ \gamma^{2^3} &= \beta^8 + \beta^{-8} = \beta^{-3} + \beta^3 = \gamma_3 \\ \gamma^{2^4} &= \beta^{16} + \beta^{-16} = \beta^{-5} + \beta^5 = \gamma_5\end{aligned}$$

从而得到新的基 $N = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5\}$ 。

将元素 A 分别用基 M 和 N 表示为：

$$A = (a_1', a_2', a_3', a_4', a_5') = a_1' \gamma + a_2' \gamma^2 + a_3' \gamma^{2^2} + a_4' \gamma^{2^3} + a_5' \gamma^{2^4}$$

和

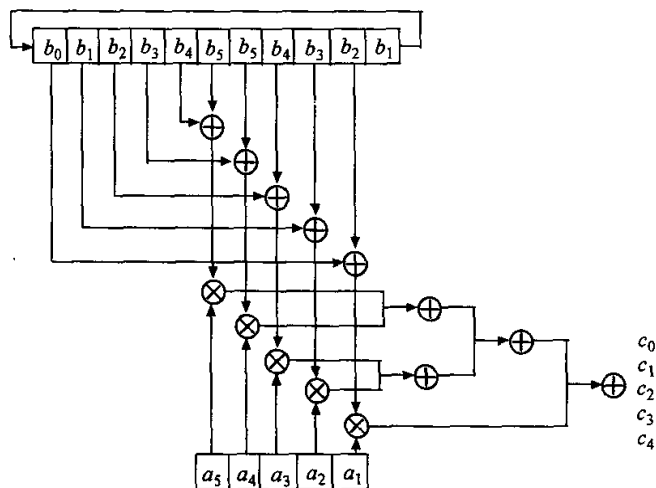
$$A = (a_1, a_2, a_3, a_4, a_5) = a_1 \gamma_1 + a_2 \gamma_2 + a_3 \gamma_3 + a_4 \gamma_4 + a_5 \gamma_5,$$

从而有

$$(a_1, a_2, a_3, a_4, a_5) = (a_1', a_2', a_4', a_3', a_5') \quad (6-20)$$

因此，为了计算按照正规基 M 表示的 $C = AB$ 的乘积，我们首先计算按照基 N 表示的 $C = AB$ 的乘积，然后根据(6-20)对 C 的坐标进行置换，就得到 C 按照基 M 表示的值。图 6-8 就是按照算法 6.1 的具体实现过程。

由图 6-8 看出，经过 5 个时刻便可以得到按照基 N 表示的 C 值，从而得到按照正规基 M 表示的 $C = (c_1', c_2', c_3', c_4', c_5') = (c_1, c_2, c_4, c_3, c_5)$ 。


 图 6-8 $GF(2^5)$ 上乘法运算的具体实现过程

6.5 小结

本章首先讨论了并行性设计的基本思想；然后分析了多项式基乘法器、正规基乘法器的设计思想，综述了各种串、并行乘法器。针对文献[22, 32]的工作进行了一定的改进，以增加 XG 的个数来达到减少 AG 方式，提出了一个串行输出的串行乘法器和一个并行输出的并行乘法器。同时，得到一个 II 型最优正规基乘法器的算法设计。我们得到的乘法器都具有较低的时间和空间复杂性。该乘法器要求 $(2m-2)$ 个 XG， m 个 AG。

第七章 椭圆曲线数乘运算

相同椭圆曲线, 采用不同的坐标表示, 对应不同的点加公式。本章首先分析了各种坐标下点加公式的计算复杂度。然后以此为基础, 分析和比较了数乘运算的点加及倍加方法(add and double methods)、加减方法和窗口方法等三种算法的计算复杂性; 接着比较了点加及倍加方法和 Montgomery 方法的计算开销, 并指出, 在仿射坐标下, 当域元素乘法运算和求逆运算的比值大于等于 2 时, 点加及倍加方法的计算开销少于 Montgomery 方法, 但是, 在投影坐标下, Montgomery 方法则优于点加及倍加方法; 最后对同时计算多个数乘运算的几种算法进行了算法复杂度分析。

本章中出现的 M , S , I 分别表示域元素的乘法, 平方和求逆。

7.1 素数域上椭圆曲线点的表示

7.1.1 仿射坐标

设 E 是素数域 $GF(p)$ 上的椭圆曲线, 其仿射方程为:

$$y^2 = x^3 + ax + b \quad a, b \in GF(p) \quad (7-1)$$

令 $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ 是 E 上的两个点, 且 $P_1 \neq -P_2$, 则 $P_3 = P_1 + P_2 = (x_3, y_3)$ 可由下式计算:

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = -\lambda(x_1 - x_3) - y_1 \end{cases} \quad (7-2)$$

其中

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & P_1 \neq P_2 \\ \frac{3x_1^2 + a}{2y_1}, & P_1 = P_2 \end{cases} \quad (7-3)$$

由式(7-2)和(7-3)知, 计算点加需要 1 个求逆运算, 2 个乘法运算和 1 个平方运算, 即, $I+2M+S$; 计算倍点需要 1 个求逆运算, 2 个乘法运算和 2 个平方运算,

即, $I+2M+2S$, 其中, 常系数与域元素的乘法可看成是相同域元素的加法, 因此, 其计算开销可以忽略不计。

7.1.2 投影坐标

M.Brown, D.Hankerson 等人^[72]粗略估计, 求逆运算与乘法运算(带快速取模运算)的开销比是 80:1, 因此用不求逆运算的投影坐标代替仿射坐标表示坐标点, 存在强大的计算优势。下面给出几种投影坐标表达式形式。

7.1.2.1 常用的几种投影坐标

标准投影坐标(*Standard Projective Coordinates*) 投影点 $(x:y:z)$ ($z \neq 0$)对应仿射点 $(x/z, y/z)$, 即, 将 $(x/z, y/z)$ 代入方程(7-1), 然后两端同时乘以 z^3 , 则得到与投影点 $(x:y:z)$ ($z \neq 0$)对应的椭圆曲线方程

$$y^2z = x^3 + axz^2 + bz^3 \quad (7-4)$$

Jacobian 投影坐标(*Jacobian Projective Coordinates*) 投影点 $(x:y:z)$ ($z \neq 0$)对应仿射点 $(x/z^2, y/z^3)$, 即, 将 $(x/z^2, y/z^3)$ 代入方程(7-1), 然后两端同时乘以 z^6 , 则得到与投影点 $(x:y:z)$ ($z \neq 0$)对应的椭圆曲线投影方程

$$y^2 = x^3 + axz^4 + bz^6 \quad (7-5)$$

Chudnovsky Jacobian 坐标(*Chudnovsky Jacobian Coordinates*) Chudnovsky Jacobian 坐标是指 Jacobian 点 $(x:y:z)$ 用点 $(x:y:z:z^2:z^3)$ 表示。

7.1.2.2 倍加公式和点加公式

Jacobian 坐标下的倍点公式: $2(x_1:y_1:z_1)=(x_3:y_3:z_3)$, 其中

$$\begin{aligned} A &= 4x_1 \cdot y_1^2, B = 8y_1^4, C = 3(x_1 - z_1^2) \cdot (x_1 + z_1^2), D = -2A + C^2 \\ x_3 &= D, y_3 = C(A - D) - B, z_3 = 2y_1 \cdot z_1 \end{aligned} \quad (7-6)$$

混合 Jacobian-affine 坐标的加法公式: $(x_1:y_1:z_1)+(x_2:y_2:1)=(x_3:y_3:z_3)$, 其中

$$\begin{aligned} A &= x_2 \cdot z_1^2, B = y_2 \cdot z_1^3, C = A - x_1, D = B - y_1, x_3 = D^2 - (C^3 + 2x_1C^2), \\ y_3 &= D(x_1 \cdot C^2 - x_3) - y_1 \cdot C^3, z_3 = z_1 \cdot C \end{aligned} \quad (7-7)$$

混合 Jacobian-Chudnovsky 坐标的加法公式: $(x_1:y_1:z_1)+(x_2:y_2:z_2:z_2^2:z_2^3)=(x_3:y_3:z_3)$, 其中

$$\begin{aligned} A &= x_1 \cdot z_2^2, B = y_1 \cdot z_2^3, C = x_2 \cdot z_1^2 - A, D = y_2 \cdot z_1^3 - B, \\ x_3 &= D^2 - 2AC^2 - C^3, y_3 = D(AC^2 - x_3) - B \cdot C^3, z_3 = z_1 \cdot z_2 \cdot C \end{aligned} \quad (7-8)$$

7.1.3 计算复杂性

不同坐标表示，点加与倍点加所需要的计算复杂性不同，对应的计算复杂性见表 7-1，其中， A 表示“Affine”， P 表示“标准投影”， J 表示“Jacobian”， C 表示“Chudnovsky”。

由 M.Brown, D.Hankerson 等人^[72]的粗略估计知， $1I \approx 80M$ ， $1M \approx 1.16S$ ，于是，从表 7-1 可以看出，对倍点加公式，Jacobian 坐标的计算开销最小；对点加公式，Chudnovsky 坐标计算开销最小；对混合坐标公式， $J+A \rightarrow J$ 和 $C+A \rightarrow C$ 的计算开销相同。

表 7-1 各种坐标对应的计算量

	计算公式	计算量	合计(S)
倍点加	$2A \rightarrow A$	$1I + 2M + 2S$	97.12
	$2P \rightarrow P$	$7M + 3S$	11.12
	$2J \rightarrow J$	$4M + 4S$	8.64
	$2C \rightarrow C$	$5M + 4S$	9.8
点加	$A + A \rightarrow A$	$1I + 2M + 1S$	96.12
	$P + P \rightarrow P$	$12M + 2S$	15.92
	$J + J \rightarrow J$	$12M + 4S$	17.82
	$C + C \rightarrow C$	$11M + 3S$	15.76
混合坐标	$J + A \rightarrow J$	$8M + 3S$	12.28
	$J + C \rightarrow J$	$11M + 3S$	15.76
	$C + A \rightarrow C$	$8M + 3S$	12.28

7.2 二元扩域上椭圆曲线点的表示

7.2.1 仿射坐标

设 E 是二元扩域 $GF(2^m)$ 上的椭圆曲线，其仿射方程为：

$$y^2 + xy = x^3 + ax^2 + b \quad a, b \in GF(2^m) \quad (7-9)$$

7.2.1.1 常用仿射坐标公式

令 $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ 是 E 上的两个点, 且 $P_1 \neq -P_2$, 则 $P_3 = P_1 + P_2 = (x_3, y_3)$ 可由下式计算:

1) $P_1 \neq P_2$

$$\begin{cases} \lambda = (y_2 + y_1)/(x_2 + x_1) \\ x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \end{cases} \quad (7-10)$$

2) $P_1 = P_2$

$$\begin{cases} \lambda = y_1 / x_1 + x_1 \\ x_3 = \lambda^2 + \lambda + a \\ y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \end{cases} \quad (7-11)$$

由式(7-10)和(7-11)知, 不管是哪一种情况, 均需要两个域元素乘法, 一个域元素平方和一个域元素求逆, 即, $I+2M+S$ 。

当 $P_1 = P_2$ 时, 结合(7-9)和(7-11), 可以得到下面的公式:

$$\begin{cases} \lambda = y_1 / x_1 + x_1 \\ x_3 = x_1^2 + b / x_1^2 \\ y_3 = x_1^2 + (\lambda + 1)x_3 \end{cases} \quad (7-12)$$

显然, 用公式(7-12)计算倍点 $2P_1$ 的 x 坐标不需要点 P_1 的 y 坐标。如果使用一个临时变量记下 $1/x_1$ 的值, 则利用该公式计算倍点共需要三个域元素乘法, 两个域元素平方, 一个域元素求逆, 即 $3M+2S+I$ 。如果仅计算一次倍点, 则(7-11)更好。但是, 在数乘运算中, 如果计算过程仅计算倍点的 x 坐标, 则(7-12)更好。该公式的具体使用情况见 7.4 节。

7.2.1.2 特殊仿射坐标公式

如果点 P_1 , P_2 和 P 总是满足如下条件:

$$P_2 = P_1 + P \quad (7-13)$$

则我们有下面的定理。

引理 7.1 令 $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ 是 E 上的两个点, 则 $P_1 + P_2 = (x_3, y_3)$ 的 x 坐标的计算公式如下:

$$x_3 = \frac{x_1 y_2 + x_2 y_1 + x_1 x_2^2 + x_2 x_1^2}{(x_1 + x_2)^2} \quad (7-14)$$

引理 7.2^[101] 令 $P = (x, y)$, $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ 是 E 上的三个点, 假设 P_1 , P_2 和 P 满足(7-13), 则 $P_1 + P_2 = (x_3, y_3)$ 的 x 坐标计算公式如下:

$$x_3 = \begin{cases} x + \left(\frac{x_1}{x_1 + x_2} \right)^2 + \frac{x_1}{x_1 + x_2} & P_1 \neq P_2 \\ x_1^2 + b/x_1^2 & P_1 = P_2 \end{cases} \quad (7-15)$$

如果在计算 kP 过程中, 始终满足(7-13), 则计算 kP 的计算开销会大大降低。例如, Montgomery 在文献[70]中提出一种始终满足(7-13)的计算数乘的方法, 该方法仅要求 P_1 , P_2 的 x 坐标, 经过 $(k-1)$ 次迭代, 可获得 kP 和 $(k+1)P$ 的 x 坐标, 然后经过下面的引理 7.3 恢复 kP 的 y 坐标, 从而求出 kP 的值。

引理 7.3^[101] 令 $P = (x, y)$, $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ 是 E 上的三个点, 假设 P_1 , P_2 和 P 满足(7-13), 且 $x \neq 0$, 则 $P_1 + P_2 = (x_3, y_3)$ 的 y 坐标计算公式如下:

$$y_3 = \frac{(x_1 + x)((x_1 + x)(x_2 + x) + x^2 + y)}{x} + y \quad (7-16)$$

7.2.2 投影坐标

$GF(2^m)$ 上的求逆运算开销相对较大, 例如当 $m \geq 128$ 时, 基于 Fermat 定理的一个求逆运算至少相当于 7 个乘法运算。因此, 使用投影坐标避开求逆运算存在计算优势。下面的表 7-2 给出了求逆对乘法运算的开销比 r 。

表 7-2 求逆对乘法的开销比^[102]

作者	m	r
Schroeppel[103]	155	2.48, 3.55
De Win[104]	177	3.13
Hankerson[105]	163, 233, 283	10
Weimerskirch[106]	163	16.17

7.2.2.1 通用投影坐标公式

1 文献[107]中的投影坐标

投影坐标 $(x:y:z) (z \neq 0)$ 对应仿射坐标 $(x/z^2, y/z^3)$, 即, 将 $(x/z^2, y/z^3)$ 代入方程(7-9), 然后两端同时乘以 z^6 , 则得到与投影点 $(x:y:z) (z \neq 0)$ 对应的椭圆曲线方程

$$y^2 + xyz = x^3 + x^2 z^2 + bz^6 \quad (7-17)$$

则 $P_1 + P_2 = (x_3, y_3, z_3)$ 的计算公式如下:

1) $P_1 \neq P_2$

$$\begin{aligned} u_1 &= x_1 z_2^2, & s_1 &= y_1 z_2^3 \\ u_2 &= x_2 z_1^2, & s_2 &= y_2 z_1^3 \\ w &= u_1 + u_2, & r &= s_1 + s_2 \\ l &= z_1 w, & v &= r x_2 + l y_2 \\ z_3 &= l z_2, & t &= r + z_3 \\ x_3 &= z_3^2 + r t + w^3, & y_3 &= t x_3 + v l^2 \end{aligned} \quad (7-18)$$

2) $P_1 = P_2$

$$\begin{aligned} z_3 &= x_1 z_1^2, & x_3 &= (x_1 + \sqrt[4]{b} z_1^2)^4 \\ u &= z_3 + x_1^2 + y_1 z_1, & y_3 &= x_1^4 z_3 + u x_3 \end{aligned} \quad (7-19)$$

2 文献[108]中的投影坐标

投影坐标 $(x:y:z)$ ($z \neq 0$) 对应仿射坐标 $(x/z, y/z^2)$, 即, 将 $(x/z, y/z^2)$ 代入方程(7-9), 然后两端同时乘以 z^4 , 则得到与投影点 $(x:y:z)$ ($z \neq 0$) 对应的椭圆曲线方程为:

$$y^2 + x y z = x^3 z + x^2 z^2 + b z^4 \quad (7-20)$$

则 $P_1 + P_2 = (x_3, y_3, z_3)$ 的计算公式如下:

1) $P_1 \neq P_2, z_1 = 1$

$$\begin{aligned} u &= z_2^2 y_1 + y_2, & s &= z_2 x_1 + x_2 \\ t &= z_2 s, & r &= s^2 (t + z_2^2) \\ z_3 &= t^2, & q &= u t \\ x_3 &= u^2 + r + q, & v &= x_3 + x_1 z_3 \\ w &= x_3 + y_1 z_3, & y_3 &= q v + z_3 w \end{aligned} \quad (7-21)$$

2) $P_1 = P_2$

$$\begin{aligned} z_3 &= z_1^2 x_1^2, & x_3 &= x_1^4 + b z_1^4 \\ y_3 &= b z_1^4 z_3 + x_3 (z_3 + y_1^2 + b z_1^4) \end{aligned} \quad (7-22)$$

3 文献[102]中的投影坐标

E.Al-Daoud 和 R.Mahmod 等人对 $z_1 = 1$ 的情形给出了新的投影坐标公式, 该公式仅需要 8 个域元素乘法, 其中, 投影坐标和仿射坐标的对应方式与[108]相同。具体如下:

假设 $P_1 = (x_1, y_1, z_1)$, $P_2 = (x_2, y_2, z_2)$, 则 $P_3 = P_1 + P_2 = (x_3, y_3, z_3)$ 的计算公式如下:

$$\begin{aligned}
 u &= z_2^2 y_1 + y_2, & s &= z_2 x_1 + x_2 \\
 t &= z_2 s, & z_3 &= t^2 \\
 v &= z_3 x_1, & x_3 &= u^2 + t(u + s^2 + t) \\
 c &= x_1 + y_1, & y_3 &= (v + x_3)(tu + z_3) + cz_3^2
 \end{aligned} \tag{7-23}$$

7.2.2.2 特殊投影坐标公式

引理 7.4 令 P, P_1, P_2 是 E 上的三个点, 假设 P_1, P_2 和 P 满足(7-13), P_i 的 x -坐标表示为 $x_i/z_i, (i=1,2)$, 由引理 7.2, $2P_i$ 的 x -坐标表示为:

$$\begin{cases} X_{2i} = X_i^4 + bZ_i^4 \\ Z_{2i} = Z_i^2 \cdot X_i^2 \end{cases} \tag{7-24}$$

同样地, $P_1 + P_2$ 的 x 坐标计算公式如下:

$$\begin{cases} Z_3 = (X_1 Z_2 + X_2 Z_1)^2 \\ X_3 = x Z_3 + (X_1 \cdot Z_2)(X_2 \cdot Z_1) \end{cases} \tag{7-25}$$

显然, 利用公式(7-24)计算倍加的 x 坐标需要一个域乘法, 一个常数与域元素的乘法, 4 个域元素的平方, 即 $2M+4S$; 利用公式(7-25)计算点加的 x 坐标需要三个域乘法, 一个 x (固定值)与域元素的乘法, 一个域元素的平方, 即 $4M+S$ 。

7.2.3 性能分析和比较

本小节从公式所需要的域乘法、平方、求逆、加法和临时变量等基本方面进行比较。此处仅比较点加公式, 即比较公式(7-10), (7-18), (7-21), (7-23)和(7-25)。相应的倍点公式可采用同样的方式进行比较, 此处略。具体比较结果见表 7-3。此处忽略了加法和平方的计算量, $I=10M$ 的估计值可参见[105]。

表 7-3 点加公式的复杂度

8	公式	域运算 $I=10M$					
		M	S	I	A	临时变量	合计
仿射	(7-10)	2	1	1	9	0	12
投影	(7-18)($z_1 \neq 1$)	14	5	0	7	9	10
	(7-18)($z_1 = 1$)	10	4	0	7	7	10
投影	(7-21)($z_1 = 1$)	9	4	0	8	7	9
投影	(7-23)($z_1 = 1$)	8	5	0	9	5	8
特殊投影	(7-25)	3	1	0	2	0	8

7.3 计算一个数乘运算

ECC 的加密、解密过程，公钥协议如 ECDH, ECDSA 和 ECAES 等的执行都要求计算椭圆曲线数乘运算。即计算如下形式：

$$Q = kP = \underbrace{P + \dots + P}_{k\text{次}}$$

其中 P 是椭圆曲线上的点, $k(0 \leq k \leq n, n \text{ 是点 } P \text{ 的阶})$ 是正整数。根据给定的协议, 点 P 或者是固定的, 可产生 $E(GF(q))$ 素阶子群的点, 或者是这样一个子群中的任意一个点。

7.3.1 典型算法

7.3.1.1 二进制数乘方法 (binary methods)

数乘运算最基本的算法是基于求幂运算的重复平方乘方法的加法形式, 该加法形式是基于 k 的二进制表示的, 具体算法如下:

算法 7.1 (从左到右) 二进制点乘方法

输入: 正整数 $0 < k = (k_{m-1}, k_{m-2}, \dots, k_1, k_0)_2, P \in E(GF(p))$

输出: $Q = kP, Q = kP \in E(GF(p))$

```

1   $Q \leftarrow o$ 
2  For  $i$  from  $m-1$  downto  $0$  do
    2.1  $Q \leftarrow 2Q$ 
    2.2 If  $k_i = 1$  then
         $Q \leftarrow Q + P$ 
3  Return ( $Q$ )
    
```

该算法最坏情况下需要 $(m-1)$ 次倍乘, $(m-1)$ 次点加; 最好情形下需要 $(m-1)$ 次倍乘, 0 次点加。因此该算法平均需要 $(m-1)$ 次倍乘, $(m-1)/2$ 次点加, 记为: $0.5(m-1)A + (m-1)/2D$ 。

7.3.1.2 加-减方法 (addition-substraction methods)

由 2.3.1.2 节知, 对素数域和二元扩域上的椭圆曲线, 点 $P=(x, y)$ 的逆元 $-P$ 分别为 $(x, -y)$ 和 $(x, x+y)$, 因此椭圆曲线有理点减法和加法具有相同的开销。于是得到新的数乘运算算法, 即加-减方法。

每一个整数 k 都有唯一的形如 $k = \sum_{j=0}^{m-1} k_j 2^j, k_j \in \{-1, 0, 1\}$ 的表示形式, 使得任意两个连续的数不同时为零, 这就是大家熟知的 NAF (Non-Adjacent Form)表示形式, 具体可参见文献[30,109,110]。其中有如下结论:

- NAF 的期望权重为 m 的 $1/3$;
- 点 $P = (x, y) \in E(F_q)$ 的逆元 $-P$ 的计算无需额外开销, 因此减法和加法的开销是相同的。

得到 $NAF(k)$ 后, 可利用 $NAF(k)$ 计算 kP 。类似于上面的二进制方法, 该算法对 k 的 NAF 表示从左到右进行扫描(也可以从右到左进行, 这样可不必存储 k 的 NAF 表示, 具体内容可参见[111]), 根据 $NAF(k)$ 的每一位数的正负号进行加或减, 具体见算法 7.2。

算法 7.2 加—减方法(Addition-Subtraction Method)

输入: 正整数 k 和点 $P = (x, y) \in E(GF(p))$

输出: 点 $Q = kP \in E(GF(p))$

1 计算 $NAF(k) = (u_{m-1} \cdots u_1 u_0)$

2 Set $Q \leftarrow o$

3 for j from $m-1$ downto 0 do

Set $Q \leftarrow 2Q$

If $u_j = 1$ then Set $Q \leftarrow Q + P$

If $u_j = -1$ then Set $Q \leftarrow Q - P$

4 Return (Q)

该算法平均需要 $(m-1)$ 次倍乘, $(m-1)/3$ 次加法, 记为: $1/3(m-1)A + (m-1)D$ 。

7.3.1.3 标号窗口方法(signed windows methods)

如果可以获得额外的存储空间, 则使用窗口方法(Windows method)还可将算法 7.2 的运行时间减少。

令 w 是一个大于 1 的整数, 则每个正整数 k 有唯一的宽度为 w 的非相邻形式: $k = \sum_{j=0}^{m-1} u_j 2^j$, 其中

- 每个非零数 u_j 是奇数, 且其绝对值不超过 2^{w-1} ;
- 在任意 w 个相连的数字中, 至多有一个非零。

宽度为 w 的 k 的 NAF 形式记为: $NAF_w(k) = (u_{m-1} \cdots u_1 u_0)$, 计算 $NAF_w(k)$ 的算法可见文献[105]的算法 14。基于 $NAF_w(k)$ 表示的点加运算的计算过程见算法 7.3。

算法 7.3 宽度为 w 的窗口方法

输入: 正整数 k 和点 $P = (x, y) \in E(GF(q))$

输出: 点 $Q = kP \in E(GF(q))$

// 预计算:

// 计算 uP, u 是奇数, 且 $2 < u < 2^{w-1}$

1 Set $P_0 \leftarrow P, T \leftarrow 2P$

2 for i from 1 downto $2^{w-2} - 1$ do

Set $P_i \leftarrow P_{i-1} + T$

//主计算:

3 计算 $NAF_w(k) = (u_{m-1} \cdots u_1 u_0)$

4 Set $Q \leftarrow o$

5 for j from $m-1$ downto 0 do

Set $Q \leftarrow 2Q$

If $u_j \neq 0$ then

Set $i \leftarrow \lceil |u_j| - 1 \rceil / 2$

If $u_j > 0$ then Set $Q \leftarrow Q + P_i$

else Set $Q \leftarrow Q - P_i$

6 Return (Q)

$NAF_w(k)$ 中非零数字的个数平均位 $m/(w+1)$ (原因参见[112]), 所以在预计算步, 需要 $2^{w-2} - 1$ 次点加运算和一次倍点运算; 在主计算部分, 需要 $m/(w+1)$ 次点加, m 次倍乘。注意尽管选择一个合适的宽度 w , 可以减少加法的数目, 但是倍乘的数目并不会减少。可以采用前面的方法进行同样的比较, 但是计算 kP 所需要的有限域运算数目不仅依赖于所使用的算法, 而且还依赖于求逆运算与乘法运算的比率和宽度 w 。该算法的复杂性分析具体见后面的 7.3.2.2。

7.3.1.4 Montgomery 算法

对定义于 $GF(2^m)$ 上的椭圆曲线, 1999 年, J.López 和 R. Dahab 基于 Montgomery 的一个思想提出了一种新的数乘运算方法^[101], 即 Montgomery 方法。该算法很适合软硬件执行, 而且, 由于在每次迭代过程中都具有相同的计算量(加法后面紧跟倍点运算), 这可以有效防止定时攻击(timing attacks)^[1]。该算法的理论依据是引理 7.1 和 7.2, 具体算法如下:

算法 7.4 Montgomery 数乘运算

输入：正整数 k ，点 $P = (x, y) \in E(GF(p))$

输出： $Q = kP \in E(GF(p))$

1 If $k=0$ or $x=0$ then 输出 $(0, 0)$ ，终止；

2 Set $k \leftarrow (k_{m-1} \dots k_1 k_0)_2$;

3 Set $R_1 \leftarrow P, R_2 \leftarrow 2P$.

4 For i from $m-2$ downto 0 do

If $k_i = 1$ then

set $R_1 \leftarrow R_1 + R_2, R_2 \leftarrow 2R_2$

else

set $R_2 \leftarrow R_1 + R_2, R_1 \leftarrow 2R_1$

5 Set $r_1 \leftarrow x_1 + x, r_2 \leftarrow x_2 + x$ // (x, x_1, x_2) 分别是 R_1, R_2 的 x 坐标

6 Set $y_1 \leftarrow r_1(r_1 r_2 + x^2 + y)/x + y$

7 Return($Q = (x_1, y_1)$)

在算法第 3 步，需要 1 次倍加计算 R_2 的 x 坐标，根据公式(7-12)，需要 1 次求逆和 1 次乘法；在第 6 步，需要 1 次求逆，3 次乘法和 1 次平方；在第 4 循环步，需进行 $(m-1)$ 次迭代，即分别需要 $(m-1)$ 次倍乘和 $(m-1)$ 次点加。

7.3.2 算法性能分析

本节根据点使用的坐标形式，给出了四种计算数乘 kP 方法的性能比较和分析。

7.3.2.1 算法 7.1 和算法 7.2 的比较

算法 7.1 平均需要 $(m-1)$ 次倍乘， $(m-1)/2$ 次点加，记为： $0.5(m-1)A + (m-1)D$ 。

如果使用仿射坐标，对素数域可利用公式(7-2)和(7-3)分别计算点加和倍加；对二元扩域则可利用(7-10)和(7-11)计算点加和倍加。根据 7.2.1 节和 7.3.1 节的分析，素数域和二元扩域的区别仅在倍加时多了一个平方运算，而平方运算对最终性能几乎没有影响，因此，此处我们一律利用公式(7-2)和(7-3)的计算量。于是，计算 kP 的时间复杂性为：

$$\begin{aligned} & 0.5(m-1)(1I + 2M + 1S) + (m-1)(1I + 2M + 2S) \\ & = 1.5mI + 3mM + 2.5mS - (1.5I + 3M + 2.5S) \end{aligned} \quad (7-26)$$

其中 $m = \lceil \log k \rceil$ 。

如果使用混合 *Jacobian-Affine* 坐标, 则 Q 使用 *Jacobian* 坐标存储, 而 P 使用仿射坐标存储。此时, 可利用公式(7-6)和(7-7)分别计算倍加和点加, 即算法 7.1 第 2.1 步使用公式(7-6)计算, 第 2.2 步使用公式(7-7)计算。根据表 7-1, 公式(7-6)计算一次倍加需要的计算量为: $4M+4S$, 公式(7-7)计算一次点加需要的计算量为: $8M+3S$, 具体见表 7-4。另外, 投影坐标还原为仿射坐标需要 $1I+3M+1S$ 。此时, 计算 kP 的时间复杂性为:

$$\begin{aligned} & 0.5(m-1)(8M+3S) + (m-1)(4M+4S) + 1I + 3M + 1S \\ & = 8mM + 5.5mS + 1I - 5M - 4.5S \end{aligned} \quad (7-27)$$

其中 $m = \lceil \log k \rceil$ 。

算法 7.2 平均需要 $(m-1)$ 次倍乘, $(m-1)/3$ 次加法, 记为: $1/3(m-1)A + (m-1)D$ 。

与算法 7.1 类似, 如果使用仿射坐标, 则计算 kP 的时间复杂性为:

$$\begin{aligned} & 1/3(m-1)(1I + 2M + 1S) + (m-1)(1I + 2M + 2S) \\ & \approx 1.3mI + 2.6mM + 2.3mS - 1.3I - 2.6m - 2.3S \end{aligned} \quad (7-28)$$

如果使用混合 *Jacobian-Affine* 坐标, 则计算 kP 的时间复杂性为:

$$\begin{aligned} & 1/3(m-1)(8M+3S) + (m-1)(4M+4S) + 1I + 3M + 1S \\ & \approx 6.4mM + 4.9mS + 1I - 3.4M - 3.9S \end{aligned} \quad (7-29)$$

分别比较(7-26)和(7-28), (7-27)和(7-29), 则有

$$\begin{aligned} \frac{(7-26)-(7-28)}{(7-28)} \times 100\% &= \frac{(m-1)(0.2I + 0.4M + 0.2S)}{(m-1)(1.3I + 2.6M + 2.3S)} \times 100\% \\ &= \frac{0.2I + 0.4M + 0.2S}{1.3I + 2.6M + 2.3S} \times 100\% \end{aligned} \quad (7-30)$$

$$\frac{(7-27)-(7-29)}{(7-29)} \times 100\% = \frac{(m-1)(1.6M + 0.6S)}{(m-1)(6.4M + 4.9S) + 1I + 3M + S} \times 100\% \quad (7-31)$$

如果是素数域上的数乘运算, 则我们假设 $1I \approx 80M$, $1M \approx 1.16S^{[72]}$ 。此时, 分别计算公式(7-30)和(7-31)得:

$$\begin{aligned} \frac{0.2I + 0.4M + 0.2S}{1.3I + 2.6M + 2.3S} \times 100\% &= \frac{19.224S}{125.956S} \times 100\% = 15.26\% \\ \frac{(m-1)(1.6M + 0.6S)}{(m-1)(6.4M + 4.9S) + 1I + 3M + S} \times 100\% &< \frac{2.456(m-1)S}{12.324(m-1)S} \times 100\% = 19.93\% \end{aligned}$$

可见, 如果使用仿射坐标, 则算法 7.1 的开销比算法 7.2 的开销大 15.26%; 如果使用混合坐标, 则算法 7.1 的开销比算法 7.2 的开销大 19.93%。

如果是二元扩域, 则忽略平方的计算量, 且假设 $I=10M$ 。此时, 分别计算公式(7-30)和(7-31)简化为:

$$\frac{0.2 \times 10M + 0.4M}{1.3 \times 10M + 2.6M} \times 100\% = \frac{2.4}{15.6} \times 100\% = 15.38\%$$

$$\frac{1.6(m-1)M}{6.4(m-1)M + 10M + 3M} \times 100\% < \frac{1.6}{6.4} \times 100\% = 25\%$$

显然, 如果使用仿射坐标, 则算法 7.1 的开销比算法 7.2 的开销大约大 15.38%; 如果使用混合坐标, 则算法 7.1 的开销比算法 7.2 的开销至多大 25%。

7.3.2.2 算法 7.1 和算法 7.3 的比较

由算法 7.3 知, 该算法需要 $m/(w+1)$ 次点加, m 次倍乘, 另外在预计算步, 需要 $2^{w-2}-1$ 次点加和一次倍点运算, 则计算 kP 的时间复杂性为:

$$\begin{aligned} & \frac{m}{(w+1)}(1I + 2M + 1S) + m(1I + 2M + 2S) + (2^{w-2}-1)(1I + 2M + 1S) + (1I + 2M + 2S) \\ &= \left(\frac{m}{(w+1)} + m + 2^{w-2} \right) I + \left(\frac{2m}{(w+1)} + 2m + 2^{w-1} \right) M + \left(\frac{m}{(w+1)} + 2m + 2^{w-2} + 1 \right) S \quad (7-32) \end{aligned}$$

如果使用混合 *Jacobian-Affine* 坐标, 则计算 kP 的时间复杂性为:

$$\begin{aligned} & \frac{m}{(w+1)}(8M + 3S) + m(4M + 4S) + (2^{w-2}-1)(8M + 3S) + (7M + 5S) + 1I \\ &= \left(\frac{8m}{(w+1)} + 4m + 2^{w+1} - 1 \right) M + \left(\frac{3m}{(w+1)} + 4m + 3 \times 2^{w-2} + 2 \right) S + 1I \quad (7-33) \end{aligned}$$

设是素数域上的数乘运算, 且 $1I \approx 80M$, $1M \approx 1.16S^{[72]}$, 则分别比较(7-26)和(7-32), (7-27)和(7-33)有

$$x = \frac{(7-26)-(7-32)}{(7-32)} = \frac{48.06m - \frac{96.12m}{(w+1)} - 24.03 \times 2^w - 1}{\frac{96.12m}{(w+1)} + 97.12m + 24.03 \times 2^w} \quad (7-34)$$

$$y = \frac{(7-27)-(7-33)}{(7-33)} = \frac{6.14m - \frac{12.28m}{(w+1)} - 3.07 \times 2^w + 3.54}{\frac{12.28m}{(w+1)} + 8.64m + 3.07 \times 2^w + 93.64} \quad (7-35)$$

分别对(7-34)和(7-35)两式求关于 w 的导数, 有

$$x' = \frac{\left(\frac{96.12m}{(w+1)^2} - 24.03 \times \ln 2 \times 2^w \right) (145.18m - 1)}{\left(\frac{96.12m}{(w+1)^2} + 97.12m + 24.03 \times 2^w \right)^2} \quad (7-36)$$

$$y' = \frac{\left(\frac{12.28m}{(w+1)^2} - 3.07 \times \ln 2 \times 2^w \right) (14.78m + 3.54)}{\left(\frac{12.28m}{(w+1)^2} + 8.64m + 3.07 \times 2^w + 93.64 \right)^2} \quad (7-37)$$

令 $x' = 0$, $y' = 0$, 则有

$$\frac{96.12m}{(w+1)^2} - 24.03 \times \ln 2 \times 2^w = 0 \quad \text{和} \quad \frac{12.28m}{(w+1)^2} - 3.07 \times \ln 2 \times 2^w = 0$$

化简上面两个式子, 均为

$$m = 0.25 \times \ln 2 \times 2^w \times (w+1)^2 \quad (7-38)$$

式(7-38)说明算法 7.1 和算法 7.3 计算开销比的极值点与采用的坐标表示形式无关。下面的图 7-1 给出了(7-34)和(7-35)取得极值时对应的 m 和 w 的值。

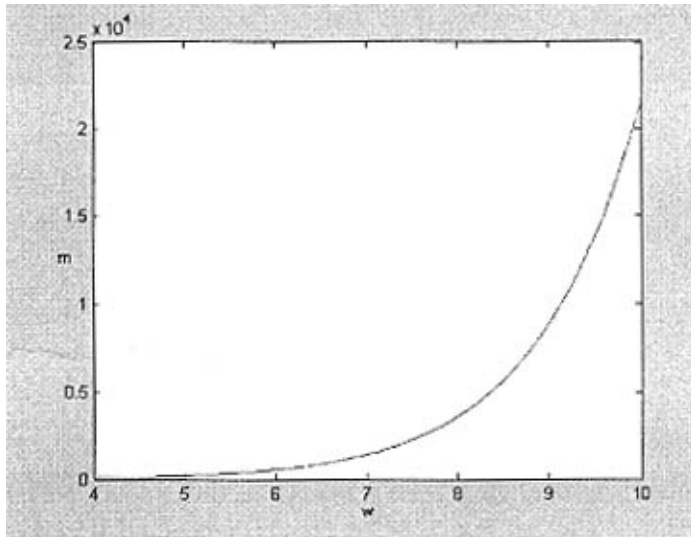


图 7-1 式(7-32)和(7-33)对应的极值点(m, w)

表 7-4 和表 7-5 分别给出了在仿射坐标和混合 *Jacobian-Affine* 坐标下算法 7.1 和算法 7.3 的性能比分析情况, 对应极值点相应的性能比见表 7-4 和表 7-5 中黑斜体数字。

表 7-4 素数域上算法 7.1 和算法 7.3 在仿射坐标下的性能比



	69	199	543	1419	3593
3	0.1713	0.1888	0.1948	0.1970	0.1978
4	0.1907	0.2274	0.2403	0.2449	0.2467
5	0.1680	0.2408	0.2673	0.2771	0.2808
6	0.0903	0.2243	0.2770	0.2970	0.3406
7	-0.0556	0.1652	0.2646	0.3044	0.3199
8	-0.2630	0.0466	0.2187	0.2947	0.3257
9	-0.4907	-0.1387	0.1220	0.2580	0.3179
10	-0.6861	-0.3675	-0.0397	0.1784	0.2881

表 7-5 素数域上算法 7.1 和算法 7.3 在混合 *Jacobian-Affine* 坐标下的性能比

	69	199	543	1419	3593
3	0.2060	0.2409	0.2541	0.2591	0.2609
4	0.2297	0.2925	0.3169	0.3262	0.3297
5	0.2020	0.3105	0.3550	0.3721	0.3787
6	0.1089	0.2882	0.3688	0.4010	0.4136
7	-0.0603	0.2098	0.3512	0.4117	0.4361
8	-0.2885	0.0585	0.2869	0.3976	0.4447
9	-0.5239	-0.1628	0.1565	0.3448	0.4332
10	-0.7144	-0.4133	-0.0483	0.2335	0.3894

从表 7-4 和表 7-5 中看出:

- 当 m 固定时, 不管是哪一种坐标表示, 相应的性能比都是先随着 w 的增加而增加, 当达到极值点后, 随着 w 的增加而减少, 当 w 的值增加到一定数值时, 性能比小于零, 即是说, 当 w 的值超过一定数值后, 算法 7.1 就会优于算法 7.3;
- 当 m 固定时, 不管是哪一种坐标表示, 相应的性能比都是先随着 w 的增加而增加, 当达到极值点后, 随着 w 的增加而减少;
- 同样地, 当 w 固定时, 不管是哪一种坐标表示, 性能比都随着 m 的增加而增大;
- 对相同的 m 和 w , 使用混合 *Jacobian-Affine* 坐标比使用仿射坐标表示点的性能比更大。

在二元扩域上, 假设同 7.3.2.1, 分别比较(7-26)和(7-32), (7-27)和(7-33)有

$$s = \frac{(7-26)-(7-32)}{(7-32)} = \frac{6m - \frac{12m}{(w+1)} - 12 \times 2^{w-2}}{\frac{12m}{(w+1)} + 12m + 12 \times 2^{w-2}} \quad (7-39)$$

$$t = \frac{(7-27)-(7-33)}{(7-33)} = \frac{4m + 4 - \frac{8m}{(w+1)} - 2^{w+1}}{\frac{8m}{(w+1)} + 4m + 2^{w+1} + 9} \quad (7-40)$$

分别对(7-39)和(7-40)两式求关于 w 的导数, 有

$$s' = 18m \times \left(\frac{12m}{(w+1)^2} - 3 \times \ln 2 \times 2^w \right) / \left(\frac{12m}{(w+1)} + 12m + 12 \times 2^{w-2} \right)^2 \quad (7-41)$$

$$t' = \left(\frac{8m}{(w+1)^2} - \ln 2 \times 2^{w+1} \right) (8m + 13) / \left(\frac{8m}{(w+1)} + 4m + 2^{w+1} - 4 \right)^2 \quad (7-42)$$

令 $s' = 0$ 和 $t' = 0$, 均有

$$4m / (w+1)^2 - \ln 2 \times 2^w = 0$$

$$m = 0.25 \times \ln 2 \times 2^w \times (w+1)^2 \quad (7-43)$$

显然(7-38)和(7-43)这两个式子是相同的, 这说明素数域和二元扩域相应的极值点是相同的, 因此, 式(7-39)和(7-40)相应的极值点(m, w)可参见图 7-1。仿射坐标和混合 *Jacobian-Affine* 坐标下算法 7.1 和算法 7.3 的性能比分析情况分别见表 7-6 和表 7-7, 对应极值点相应的性能比见表 7-6 和表 7-7 中黑斜体数字。

表 7-6 二元扩域上算法 7.1 和算法 7.3 在仿射坐标下的性能比

$\begin{matrix} m \\ w \end{matrix}$	69	199	543	1419	3593
3	0.1728	0.1904	0.1965	0.1986	0.1995
4	0.1924	0.2294	0.2424	0.2471	0.2488
5	0.1695	0.2429	0.2697	0.2795	0.2833
6	0.0911	0.2262	0.2795	0.2997	0.3074
7	-0.0559	0.1666	0.2670	0.3071	0.3229
8	-0.2642	0.0470	0.2205	0.2973	0.3287
9	-0.4924	-0.1395	0.1230	0.2603	0.3209
10	-6876	-0.3690	-0.0399	0.1799	0.2907

表 7-7 二元扩域上算法 7.1 和算法 7.3 在混合 *Jacobian-Affine* 坐标下的性能比

$\begin{matrix} m \\ w \end{matrix}$	69	199	543	1419	3593
3	0.2870	0.3167	0.3271	0.3310	0.3324
4	0.3219	0.3891	0.4138	0.4229	0.4263
5	0.2812	0.4149	0.4675	0.4874	0.4950
6	0.1487	0.3831	0.4872	0.5286	0.5448
7	-0.0738	0.2738	0.4621	0.5442	0.5775
8	-0.3417	0.0744	0.3720	0.5238	0.5899
9	-0.5858	-0.1927	0.1972	0.4488	0.5732
10	-0.7629	-0.4646	-0.0577	0.2966	0.5102

表 7-6 和表 7-7 与表 7-4 和表 7-5 相比, 不管采用哪一种坐标表示点, 二元扩域比素数域的性能比大; 而且当 w 固定时, 两种坐标表示相应的性能比都随着 m 的增加而增大。

7.3.2.3 算法 7.1 和算法 7.4 的比较

在算法 7.4 的第 4 循环步中, 如果 $k_i=1$, 则

$$\text{new}R_2 - \text{new}R_1 = 2\text{old}R_2 - (\text{old}R_1 + \text{old}R_2) = \text{old}R_2 - \text{old}R_1 = P。$$

如果 $k_i=0$, 则

$$\text{new}R_2 - \text{new}R_1 = (\text{old}R_1 + \text{old}R_2) - 2\text{old}R_2 = \text{old}R_1 - \text{old}R_2 = -P。$$

因此该算法始终满足(7-13), 根据引理 7.2, 仅要求 P_1, P_2 的 x 坐标, 于是利用公式(7-15)计算新 R_1 和 R_2 的 x 坐标, 经过 $(m-1)$ 次迭代之后, 根据引理 7.3, 利用公式(7-16)恢复 R_1 的 y 坐标, 计算出 kP 的 x 和 y 坐标, 即求出 kP 的值。

由公式(7-15)计算点加和倍加均需要 1 个求逆, 1 个乘法和 2 个平方, 即 $1I+1M+1S$; 利用公式(7-16)恢复 R_1 的 y 坐标时需要 1 个求逆, 1 个平方和 3 个乘法, 即 $1I+3M+1S$ 。从而, 计算 kP 需要的计算开销为:

$$\begin{aligned} & (m-1)(1I+1M+1S) + (m-1)(1I+1M+1S) + 2I+4M+1S \\ &= (m-1)(2I+2M+2S) + 2I+4M+1S \\ &= 2mI + (2m+2)M + (2m-1)S \end{aligned} \quad (7-44)$$

其中, $m = \lfloor \log_2 k \rfloor$ 。

如果使用投影坐标, 则利用公式(7-24)和 (7-25) 计算点加和倍加的 x 坐标分别需要 $4M+S$ 和 $(M+4S)$, 最后再利用公式(7-16)恢复 R_1 的 y 坐标需要 $1I+3M+1S$ 。从而, 计算 kP 需要的计算开销为:

$$\begin{aligned}
 & (m-1)(4M+S) + (m-1)(2M+4S) + 2I + 8M + 6S \\
 & = (m-1)(6M+5S) + 2I + 8M + 6S \\
 & = 2I + (6m+2)M + (5m+1)S
 \end{aligned} \tag{7-45}$$

其中, $m = \lfloor \log_2 k \rfloor$ 。

在二元扩域上, 一个域元素的平方运算仅需一次简单的循环移位, 其开销可以忽略不计。记算法 7.1 在仿射坐标下和投影坐标下的开销分别为 $Cost_{7.1f}$ 和 $Cost_{7.1l}$, 算法 7.4 在仿射坐标下和投影坐标下的开销分别 $Cost_{7.4f}$ 和 $Cost_{7.4l}$, 则公式(7-26), (7-27), (7-44)和(7-45)分别化简为:

$$Cost_{7.1f} = 1.5(m-1)I + 3(m-1)M \tag{7-46}$$

$$Cost_{7.1l} = 8mM + 1I - 5M \tag{7-47}$$

$$Cost_{7.4f} = 2mI + (2m+2)M \tag{7-48}$$

$$Cost_{7.4l} = 2I + (6m+2)M \tag{7-49}$$

其中 $m = \lceil \log k \rceil$ 。

(1) 假设 $Cost_{7.4f} < Cost_{7.1f}$, 则

$$Cost_{7.1f} - Cost_{7.4f} = -0.5mI + mM - 1.5I - 5M > 0$$

即

$$\frac{I}{M} < \frac{m-5}{0.5m+1.5} < \frac{m}{0.5m} = 2$$

上式表明, 当 I/M 的值小于 2 时, $Cost_{7.4f} < Cost_{7.1f}$; 当 I/M 的值大于等于 2 时, $Cost_{7.4f} > Cost_{7.1f}$ 。

事实上, Schroeppe^[103], De Win^[104], D. Hankerson^[105]等指出, $GF(2^m)$ 上的求逆运算开销相对较大, 例如当 $m \geq 128$ 时, 基于 Fermat 定理的一个求逆运算至少相当于 7 个乘法运算。因此, 基于这样的事实, 算法 7.4 的开销大于算法 7.1 的开销。

如果假设 $I=10M$, 则有:

$$\begin{aligned}
 & \frac{(7-44)-(7-26)}{(7-26)} \times 100\% = \frac{(m-1) \times 22M + 24M - (m-1) \times 18M}{(m-1) \times 18M} \times 100\% \\
 & \doteq \frac{4}{18} \times 100\% \doteq 11.1\%
 \end{aligned} \tag{7-50}$$

即在此假设下, 算法 7.4 比算法 7.1 大约慢 11.1%。

(2) 假设 $Cost_{7.4l} < Cost_{7.1l}$, 则

$$Cost_{7.1l} - Cost_{7.4l} = 2mM - 7M - I > 0$$

即

$$\frac{I}{M} < 2m - 7$$

在 ECC 中, m 的取值通常大于 160, 因此根据上式, 总有 $Cost_{7.4t} < Cost_{7.4l}$ 。

如果假设 $I=10M$, 则有:

$$\begin{aligned} \frac{(7-27)-(7-45)}{(7-45)} \times 100\% &= \frac{(8mM+5M)-(6mM+22M)}{6mM+22M} \times 100\% \\ &= \frac{2m-17}{6m+22} \times 100\% \doteq 33.3\% \end{aligned} \quad (7-51)$$

即, 如果使用投影坐标, 则算法 7.1 比算法 7.4 大约慢 33.3%。

7.4 计算多个数乘运算

椭圆曲线数字签名算法(ECDSA)是椭圆曲线密码的重要组成部分。ECDSA 由签名的产生和签名的验证两部分构成。在签名产生过程, 计算量主要在于计算 kP , 在签名验证过程计算量主要在于计算是 $kP + lQ$, 因此研究多个数乘运算的计算对 ECC 的具体实现有重要的意义[71]。

7.4.1 现有算法

为了计算 $kP + lQ$, 通常采用下面的算法 7.5 完成。

算法 7.5 同时计算多个数乘运算

输入: $k = (k_{m-1}, \dots, k_1, k_0)_2$, $l = (l_{m-1}, \dots, l_1, l_0)_2$ 和点 $P, Q \in E$

输出: $kP + lQ$

1 预计算 $P+Q$

2 $R \leftarrow O$

3 For $i = m-1$ downto 0 do

3.1 $R \leftarrow 2R$;

3.2 $R \leftarrow R + (k_i P + l_i Q)$

4 Return ($kP + lQ = R$)

算法 7.5 的算法复杂性分析如下:

首先在第 1 步需预计算 $P+Q$, 计算量分别为: $I+2M+S$ 。

在算法 7.5 的第 4 步中, 根据 k_i 和 l_i 的不同取值, 有四种情况, 当二者均为零时, 没有点加运算, 而其它三种都是点加运算, 按照等概率原则, 出现点加的概

率是 $3/4$ ，而倍点是每一循环步都会出现。根据前面的分析，总的计算量(加上预计算)为：

$$\begin{aligned} & (m-1)(I+2M+2S) + \frac{3}{4}(m-1)(I+2M+S) + (I+2M+S) \\ &= \frac{7m-3}{4}(I+M+S) + \frac{7m-3}{4}M + (m-1)S \end{aligned} \quad (7-52)$$

利用 Montgomery 方法，白国强等人在文献[71]中提出了一种计算 $kP+lQ$ 的新算法，该算法的描述如下。

算法 7.6 基于 Montgomery 方法的多个数乘运算算法

输入： $k = (k_{m-1}, \dots, k_1, k_0)_2$ ， $l = (l_{m-1}, \dots, l_1, l_0)_2$ 和点 $P, Q \in E$

输出： $kP+lQ$

1 Set $R_1 \leftarrow P+Q$, $R_2 \leftarrow 2(P+Q)$, $s=0$

2 For i from $m-2$ downto 0 do

 If $k_i=1, l_i=1$ and $s=0$, set $R_1 \leftarrow R_1+R_2$, $R_2 \leftarrow 2R_2$

 If $k_i=1, l_i=0$ and $s=0$, set $R_1 \leftarrow R_1+R_2$, $R_2 \leftarrow 2(R_2-Q)$, $s=1$

 If $k_i=0, l_i=0$ and $s=0$, set $R_2 \leftarrow R_1+R_2$, $R_1 \leftarrow 2R_1$

 If $k_i=0, l_i=1$ and $s=0$, set $R_2 \leftarrow R_1+R_2$, $R_1 \leftarrow 2(R_1+Q)$, $s=1$

 If $k_i=1, l_i=1$ and $s=1$, set $R_1 \leftarrow R_1+R_2$, $R_2 \leftarrow 2(R_2+Q)$

 If $k_i=1, l_i=0$ and $s=1$, set $R_1 \leftarrow R_1+R_2$, $R_2 \leftarrow 2R_2$

 If $k_i=0, l_i=1$ and $s=1$, set $R_2 \leftarrow R_1+R_2$, $R_1 \leftarrow 2R_1$

 If $k_i=0, l_i=0$ and $s=1$, set $R_2 \leftarrow R_1+R_2$, $R_1 \leftarrow 2(R_1-Q)$

3 If $s=1$, set $R_1 \leftarrow R_1-Q$

4 Return $(kP+lQ=R_1)$

该算法的复杂度为^[71]：

$$3m(I+M+S) + \frac{5}{2}(I+M+S) + (m-1)M \quad (7-53)$$

7.4.2 算法性能分析

白国强等人在文献[71]中指出，其提出的新算法，即此处的算法 7.6 比用算法 7.4 计算多个数乘运算速度至少提高 22.7%。但是该比较是基于分别计算 kP ， lQ ，然后再相加的 Montgomery 方法(即本文的算法 7.4)进行比较而得到的结果。

下面我们比较同时计算多个数乘运算的算法 7.5 和 7.6。仍假设 $I=10M$ ，且忽

略 S 的计算量, 则(7-52) 和(7-53)可简化为:

$$(1.75m - 0.75) \times 12M \quad (7-54)$$

$$(3m + 2.5) \times 11M + (m - 1)M \quad (7-55)$$

于是根据公式(7-54)和(7-55), 有:

$$\frac{(7-55)-(7-54)}{(7-54)} \times 100\% = \frac{13m + 35.5}{21m - 9} \times 100\% < \frac{13}{21} \times 100\% = 61.9\% \quad (7-54)$$

即同样计算多个数乘运算, 算法 7.6 比算法 7.5 至少慢 61.9%。

7.5 其它相关方法

数乘运算的快速运算是 ECC 能够实现的关键^[111-113], 因此, 除了上面提到的一些基本方法外, 还有下面一些最近提到计算方法。

7.5.1 重复计算倍乘

在 $GF(2^m)$ 上的椭圆曲线上重复计算倍乘(即, $2^i P$)的算法见文献[86], 该算法是对[111]中提出的公式的一种改进。改进后, 计算 $2^i P$ 只需要一次逆运算。如果求逆运算和求乘运算的开销比是 2.5 的话, 则该算法快于通常的计算 $2^i P$ 的算法。该算法可以用来加速前面讲述的窗口算法(算法 7.3)。

在 $GF(2^m)$ 上的椭圆曲线上重复计算倍乘(即, $2^i P$)的另一算法见[114]。该算法适合于求逆运算快于乘法运算的情况。

对素数域 $GF(p)$ 上的椭圆曲线, Itoh 等^[16]提出计算投影坐标下的快速重复倍点公式, 该方法可以同时减少域乘法和加法的数目。该技术可以和窗口方法(window methods)结合进行数乘运算计算。

7.5.2 基于折半的数乘计算

计算椭圆曲线数乘运算的点折半运算(即, 已知 P , 计算 Q , 使得 $2Q = P$)代替倍点运算的一种算法由 Knudsen^[114]提出。该算法对 $GF(2^m)$ 上椭圆曲线参数 a 的迹等于 1 的情形是有效的。该算法的执行需要 $GF(2^m)$ 上的运算: 域元素的平方根, 域元素的迹, 和形如 $x^2 + x = s$, $s \in GF(2^m)$ 的二次方程的解等寻找快速路由。因为这些运算在使用正规基时非常有效, 所以这些运算适合硬件执行。使用多项式基

执行 Knudsen 的算法在软硬件执行时, 存储和速度之间存在一种折衷。

7.6 小结

本章主要是基于椭圆曲线点的不同坐标表示方法, 分析和比较了各种点加和倍加计算公式的复杂性。然后, 以此为基础, 分析和比较了数乘运算四种算法的计算复杂性, 并指出了这些算法性能优劣的相关条件。最后, 对同时计算多个数乘运算的几种算法进行了算法复杂度分析。

第八章 全文总结和未来工作

8.1 全文总结

1985 年 N.Koblitz, V.Miller 等人先后将椭圆曲线应用到密码学中, 产生了椭圆曲线密码体制(ECC)。ECC 的有效实现依赖于数乘运算的快速实现, 数乘运算又直接依赖于有限域运算的高效算法, 为此, 本文主要研究了有限域运算算法和椭圆曲线数乘运算算法。纵观全文, 主要创新性工作如下:

- 对 Mersenne 数和伪 Mersenne 数, 给出了取模运算转换为模加或模减运算的公式; 对模数为任意不可约首一三项式和五项式产生的广义 Mersenne 数, 推导了相应取模运算的复杂度计算解析表达式, 并给出了详细推导过程。根据该表达式, 对任意给定的不可约首一三项式和五项式, 可以根据该多项式的系数分别得出以广义 Mersenne 数为模数的取模运算所需要的模加法(或模减法)的次数。
- 利用广义 Mersenne 数代替伪 Mersenne 数, 提出了广义最优扩域的概念, 并研究其上的快速乘法运算和取模运算, 为乘法运算给出了通用的复杂度公式, 为取模运算给出了具体的运算公式, 推广了 Bailey, Mihăilescu 和 Woodbury 等在最优扩域上的相应结果。
- 基于正规基表示有限域 $GF(2^m)$ 上元素的方法, 以增加 XG 数目达到减少 AG 数目的方式, 提出了一个串行乘法器和一个并行乘法器。同时, 得到一个 II 型最优正规基乘法器的算法设计, 该乘法器要求 $(2m-2)$ 个 XG, m 个 AG。
- 分析了不同坐标下点加公式的计算复杂度, 并以此为基础, 首先分析和比较了数乘运算的点加及倍加方法, 加减方法和窗口方法等三种算法的计算复杂性。接着比较了点加及倍加方法和 Montgomery 方法的计算开销, 并指出, 在仿射坐标下, 当域元素求逆运算和乘法运算的比值大于等于 2 时, 点加及倍加方法的计算开销少于 Montgomery 方法; 但是, 在投影坐标下, 点加及倍加方法的计算开销却大于 Montgomery 方法。最后对同时计算多个数乘运算的几种算法进行了计算复杂性分析。

8.2 未来工作

在本文工作的基础上，未来值得继续关注和探索的方向主要包括：

1 进一步完善广义最优扩域体系结构

这方面可研究的内容有：

- 广义最优扩域上的求逆技术；
- 广义最优扩域的构造；
- 为广义最优扩域建立完整的体系结构。

2 乘法器的设计

有限域乘法运算是最基本的有限域运算之一，研究有限域乘法的硬件实现有着重要的现实意义，因此，今后进一步的研究方向是设计更有效的串、并行乘法器。

3 椭圆曲线数乘算法的并行实现

椭圆曲线数乘算法是 ECC 具体实现的关键，本文第七章只是对现有算法做了一些深入的分析和比较，因此，今后将以此为基础，提出新的高效数乘运算算法，并研究其并行实现。

致 谢

在本论文即将完成之际，我谨向我的导师孙世新教授致以真诚的感谢！从选题，论证到论文的撰写都是在孙老师的精心指导下完成的，自始至终都倾注了导师的辛劳、心血和希望。也正是他的远见卓识，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。

我一直认为，博士阶段就人的一生而言相对短暂，但它是人生中非常重要的一个阶段。孙老师德高望重，为我国计算机事业的发展作出过杰出贡献，能成为孙老师的一名博士生，我一直感到莫大的荣幸和自豪。在攻读博士学位的这几年，我在学业和生活上得到了孙老师慈父般的教导和关怀，诸多往事，历历在目，我深为感激。

在学业上，孙老师以他敏锐的洞察力帮助我确定了科学研究方向，以他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，孙老师以他高洁的人格魅力、宽厚待人的师者风范感染着我，潜移默化之中教诲了我许多做事为人的道理，为我树立了工作和生活中学习的楷模。孙老师取得的辉煌成就也激励着我在今后的人生道路上努力奋进。

同时，我深深地感谢范明钰教授，许春香教授在密码学方面所给予的指导和帮助。

真诚地感谢教研室顾小丰博士、刘宴宾博士、卢国明博士、杨浩淼博士和教研室其他成员，和他们的相处和交流是愉快而有益的，他们的勤奋上进和朝气蓬勃时常给我激励和启发。

我要特别感谢我的父母，丈夫，女儿和其它亲朋好友，他(她)们始终不渝的关心和照顾是我前进的动力，每一个前进的脚印，都凝刻着他(她)们无私的支持和爱。

最后，感谢曾经教育和帮助过我的所有老师，衷心地感谢为评审本论文而付出辛勤劳动的教授和专家们！

参考文献

- [1] N. Koblitz, A. Menezes and S. Vanstone. The state of elliptic curve cryptography. Designs, Codes and Cryptography, 2000,19:173-193
- [2] D.Hankerson, A.Menezes and S.Vanstone. Guide to elliptic curve cryptography. Springer-Verlag Professional Computing Series,2004
- [3] John Kerl.Computation in finite fields. Arizona State University and Lockheed Martin Corporation. April, 2004
- [4] 胡予濮, 张玉清, 肖国镇. 对称密码学. 机械工业出版社, 2002
- [5] 周玉洁, 冯登国. 公开密钥密码算法及其快速实现. 国防工业出版社, 2002. 9
- [6] F.Rodríguez-Henríquez and Ç.K.Koç. On fully parallel Karatsuba Multipliers for $GF(2^m)$
<http://security.ece.orst.edu/papers/c29fpkar.pdf>
- [7] R.Lidl and H.Niederreiter. Introduction to finite fields and their applications. Cambridge: Univ. Press, 1994
- [8] A.J.Menezes,I.F.Blake,X.Gao,et al. Applications of finite fields. Kluwer Academic, 1993
- [9] Bruce Schneier 著, 吴世忠, 祝世跃, 张文政等译. 应用密码学—协议, 算法与 C 源程序. 机械工业出版社, 2000
- [10] 裴定一, 祝跃先. 算法数论. 北京: 科学出版社, 2002
- [11] Yan S.Y. Number theory for computing, 2nd Edition. New York: Springer-Verlag, 2002
- [12] A.Menezes, P.van Oorschot and S.Vanstone. Handbook of applied cryptography. CRC Press,1997
- [13] R.Lidl and H.Neiderreiter. Finite fields, volume 20 of Encyclopedia of Mathematics and its Applications. Addison-Wesley, Reading, Massachusetts, 1983
- [14] D.Jungnickel.Finite fields.B.I.Wissenschaftsverlag,Mannheim,Leipzig,Wien,Zürich, 1993
- [15] E.D.Mastrovito. VLSI designs for multiplication over finite fields $GF(2^m)$. In LNCS-357, Proc. AAECC-6, 1988:297-309
- [16] T. Itoh and S. Tsujii. A fast algorithm for computing multiplicative inverses in $GF(2^m)$. Using Normal Basis,Information and Computing, 1988,78:171-177
- [17] J.Goodman and A.Chandrakasan. An energy efficient reconfigurable public-key cryptography processor architecture.Cryptographic Hardware and Embedded Systems-CHES 2000, LNCS-1965, 2000,175-190

- [18] T. Naofumi and T. Kazuyoshi. A fast algorithm for multiplicative inversion in $GF(2^m)$ Using Normal Basis. IEEE Trans. Computers, 2001,50(5):394-398
- [19] S. Chang Shantz. From Euclid's GCD to Montgomery multiplication to the great divide. SML Technical Report SMLI TR-2001-95, Sun Microsystems Laboratories, 2001.
- [20] Nadia Nedjah and Luiza de Macedo Mourelle. Two hardware implementations for the Montgomery modular multiplication: Sequential versus Parallel. Proceedings of the 15 th Symposium on Integrated Circuits and Systems Design,(SBCCI'02),2002
- [21] R. Schroepfel. Automatically solving equations in finite fields. US Patent Application No. 09/834,363, filed 12 April 2001, publication number US 2002/0055962 A1, 2002-5-9
- [22] A.R.Masoleh and M.A.Hasan. Efficient multiplication beyond optimal normal bases.IEEE Trans. Computers, 2003,52(4):428-439
- [23] A. Weimerskirch and C. Paar. Generalizations of the Karatsuba algorithm for efficient implementations.<http://www.crypto.ruhr-uni-bochum.de/imperia/md/content/texte/kaweb.pdf>. 2003
- [24] R.Dahab, D.Hankerson, F.Hu, et al. Software multiplication using normal bases. CORR2004-12, Centre for Applied Cryptographic Research, University of Waterloo, 2004. <http://cacr.uwaterloo.ca/techreports/2004/corr2004-12.pdf>
- [25] A. R.Masoleh. Efficient algorithms and architectures for field multiplication using gaussian normal bases. IEEE Transactions on computers,2006,55(1):34-47
- [26] K.Fong, D.Hankerson, J.López,et al. Field inversion and point halving revisited. Technical Report CORR 2003-18, Centre for Applied Cryptographic Research, University of Waterloo,2003. <http://cacr.uwaterloo.ca/techreports/2003/corr2003-18.pdf>
- [27] A.R.Masoleh and M.A.Hasan. Low complexity bit parallel architectures for polynomial basis multiplication over $GF(2^m)$. IEEE Transactions on computers,2004,53(8):945-959
- [28] P.L.Montgomery. Five, six, and seven-term Karatsuba-like formulae. IEEE Trans. Computers, 2005,54(3):362-369
- [29] D.E.Knuth. The art of computer programming-seminumerical algorithms,vol.2, Addison-Wesley, Reading, Massachusetts,2nd edition, 1981
- [30] IEEE Std 1363-2000.IEEE Standard specifications for public-key cryptography. January 2000.
- [31] National Institute of Standards and Technology (NIST): Digital signature standard (DSS) (Federal Information Processing Standards Publication 186-2, February 2000)

- [32] A.R.Masoleh and M.A.Hasan. A new construction of massey-omura parallel multiplier over $GF(2^m)$. IEEE Trans. Computers, 2002,51(5):511-520
- [33] J.L.Massey and J.K.Omura. Computational method and apparatus for finite field arithmetic. US Patent, 4587627, 1986
- [34] M.A.Hasan, M.Z.Wang and V.K.Bhargava. A modified massey-omura parallel multiplier for a class of finite fields. IEEE Trans. Computers, 1993,42(10):1278-1280
- [35] Ç.K.Koç and B.Sunar. Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields. IEEE Trans. Computers, 1998,47(3):353-356
- [36] R.C.Mullin, I.M.Onyschuk, S.A.Vanstone, et al. Optimal Normal Bases in $GF(p^n)$. Discrete Applied Math., 1988/1989,22:149-161
- [37] B.Sunar and Ç.K.Koç. An efficient optimal normal basis type II multiplier. IEEE Trans. Computers, 2001,50(5):83-87
- [38] A.R.Masoleh and M.A.Hasan. Low complexity word level sequential normal basis multipliers. IEEE Trans. Computers, 2005,54(2):98-109
- [39] 廖群英. 有限域上最优正规基的乘法表. 数学学报, 2005, 48(5):947-954
- [40] H.Wu, M.A.Hasan, I.F.Blake, et al. Finite field multiplier using redundant representation. IEEE Trans. Computers, 2002,51(11):1306-1316
- [41] B.Sunar. A generalized method for constructing sub-quadratic complexity $GF(2^k)$ multipliers. IEEE Trans. Computers, 2004,53(9):1097-1105
- [42] H.Fan and Y.Dai. Key function of normal basis multipliers in $GF(2^n)$. Electronics Letters, 2002,38(23):1431-1432
- [43] 鲁俊生, 张文祥, 王新辉. 一种基于有限域的快速乘法器的设计与实现. 计算机研究与发展, 2004, 41(4):755-760
- [44] H.Wu. On computation of polynomial modular reduction. <http://www.cacr.math.uwaterloo.ca/techreports/2000/corr2000-31.pdf>
- [45] T.C.Bartee and D.I.Schneider. Computation with finite fields. Information and Computers, 1963,6(3):79-98
- [46] E.R.Berlekamp. Algebraic coding theory, McGraw-Hill, 1968
- [47] E.D.Mastrovito. VLSI architectures for computation in Galois fields:[PhD thesis]. Linköping Sweden: Linköping Uni., 1991

- [48] E.D.Mastrovito. VLSI designs for multiplication over finite fields $GF(2^m)$. In LNCS-357, Proc. 1988-AAECC, Vol.6:297-309
- [49] B.Sunar and Ç.K.Koç. Mastrovito multiplier for all trinomials. IEEE Trans. Computers, 1999,48(5):522-527
- [50] A.Halbutogullari and Ç.K.Koç. Mastrovito multiplier for general irreducible polynomials. IEEE Trans. Computers,2000,49(5):503-518
- [51] T.Zhang and K.K.Parhi.Systematic design of original and modified Mastrovito multipliers for General irreducible polynomials. IEEE Trans. Computers, 2001,50(7):734-748
- [52] L.Song and K.K.Parhi. Low complexity modified mastrovito multipliers over finite fields $GF(2^m)$. In ISCAS-99, Proc. IEEE International Symposium on Circuits and Systems, 1999,508-512
- [53] H.Wu. Bit-parallel finite field multiplier and squarer using polynomial basis.IEEE Trans. Computers, 2002,51(7):750-758
- [54] F.R.Henriquez and Ç.K.Koç. Parallel multipliers based on special irreducible pentanomials. IEEE Trans. Computers, 2003,48(5):522-527
- [55] D.V.Bailey and C.Paar.Optimal extension fields for fast arithmetic in public-key algorithms.In Crypto.'98, Springer Lecture Notes in Computer Science. Vol.1462:472-485
- [56] D.V.Bailey.Computation in optimal extension fields:[Master Thesis].Department of Electrical & Computer Engineering, Worcester Polytechnic Institute,2000
- [57] Additional ECC Groups for IKE, Mar. 2001, <http://www.ietf.org/proceedings/01dec/I-D/draft-ietf-ipsec-ike-eccgroups-03.txt>
- [58] ANSI X9.62, Public key cryptography for financial service industry: The elliptic curve digital signature algorithm(ECDSA),1999
- [59] ANSI X9.63, Public key cryptography for financial service industry: Elliptic curve key agreement and key transport protocols. working draft, 2000
- [60] ISO/IEC 14888-3 Information technology-security techniques-digital signatures with appendix-part3:certificate based mechanisms,1998
- [61] ISO/IEC 15946 Information technology-security techniques-cryptographic techniques based on elliptic curves, Committee Draft(CD),1999
- [62] Junquan Li. Design and analysis of elliptic curve cryptosystem:[PhD thesis].BeiJing: Chinese Academy of Sciencer, 2001
- [63] 张方国, 陈晓峰, 王育民. 椭圆曲线离散对数的攻击现状. 西安电子科技大学学报(自然科学版), 2002, 29(3):398-403

- [64] J.Guajardo and C.Paar. Efficient algorithms for elliptic curve cryptosystems. *Advances in Cryptology, Proc. Crypto'97*, 1997, vol. 1294: 342-356
- [65] J.Lopez and R.Dahab. Improved algorithms for elliptic curve arithmetic in $GF(2^n)$. *SAC'98*, LNCS Springer Verlag
- [66] J.Solinas. An improved algorithm for arithmetic on a family of elliptic curves. *Advances in Cryptology, Proc. Crypto'97*, 1997, vol. 1294: 357- 371
- [67] V.Müller. Fast multiplication on elliptic curves over small fields of characteristic two. *Journal of Cryptology*, 1998, vol. 11(4): 219-234
- [68] 胡 磊, 冯登国, 文铁华. 一类 Koblitz 椭圆曲线的快速点乘. *软件学报*, 2003, 14(11): 1907-1910
- [69] 白国强, 周涛, 陈弘毅. 一类安全椭圆曲线的选取及其数乘运算的快速计算. *电子学报*, 2002, 30(11): 1654-1657
- [70] P.Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 1987, 48: 243-264
- [71] 白国强等. 椭圆曲线数字签名算法中的快速验证算法. *清华大学学报(自然科学版)*, 2003, 43(4): 564-568
- [72] M.Brown, D.Hankerson, J.López, et al. Software implementation of the NIST elliptic curves over prime fields. In *Topics in Cryptology-CT-RSA 2001*, Springer LNCS, vol. 2020: 250-265
- [73] V.Miller. Uses of elliptic curves in cryptography. *Advances in Cryptology- CRYPTO '85*. Santa Barbara, Calif: Springer Verlag, 1986: 417-426
- [74] N.Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*. 1987, 48(177): 203-209
- [75] 张险峰. 基于 ECC 的门限密码体制及其应用的研究——在入侵容忍中应用的探索:[博士学位论文]. 成都: 电子科技大学, 2003
- [76] I.F. Blake, G. Seroussi, N.P Smart. *Elliptic curve in cryptography*. Cambridge University Press, 1999
- [77] D. S. Phatak and T. Goff. Fast modular reduction for large wordlengths via one linear and one cyclic convolution, *Proceedings of the 17th IEEE Symposium on Computer Arithmetic (ARITH'05)*
- [78] P. Barrett. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. *Advances in Cryptology-CRYPTO '86. Lecture Notes in Computer Science*, 1987, vol. 263: 311-323

- [79] P.Montgomery. Modular multiplication without trial division.Mathematics of Computation,1985, 44:519-521
- [80] 雷明,叶新,张焕国.Montgomery 算法及其快速实现.计算机工程,2003,29(14):44- 46
- [81] 邓玉良,毛志刚,叶以正.Montgomery 算法及其在加密硬件中的实现.计算机研究与发展,1999,36(4):505-508
- [82] 王许书,王新辉,夏宏.Montgomery 方法及其在伪随机数发生器中的应用.计算机工程与应用,2001,11:52-54
- [83] E.Savas, K.Koç.The Montgomery modular inverse-revisited.IEEE Trans. Computers, 2000,49(7):763-766
- [84] T.Blum and C.Paar. High-Radix Montgomery modular exponentiation on reconfigurable hardware. IEEE Trans. Computers,2001,50(7):759-764
- [85] A. Cilardo, A. Mazzeo et al..Carry-Save Montgomery modular exponentiation on reconfigurable hardware. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Designers' Forum (DATE'04)
- [86] P. Mihăilescu. Optimal Galois field bases which are not normal, Recent Result Session, Fast Software Encryption, 1997
- [87] N.P.Smart.A comparison of different finite fields for use in elliptic curve cryptosystems.Technical Report CSTR-00-007, Department of Computer Science, University of Bristol, June 2000
- [88] D. V. Bailey. Computation in Optimal Extension Fields, Msd, 2000
- [89] B.Selçuk.Efficient algorithms for finite fields with applications in elliptic curve cryptography:[Master Thesis].Department of Electrical & Computer Engineering, Worcester Polytechnic Institute,2003
- [90] A.D.Woodbury. Efficient algorithms for elliptic curve cryptosystems on embedded systems:[Master Thesis]. Department of Electrical & Computer Engineering, Worcester Polytechnic Institute,2001
- [91] 徐甲同,李学干.并行处理技术 [M]. 西安电子科技大学出版社, 1999. 5
- [92] R. Schroeppe, H. Orman, S. O'Malley, et al.. Fast key exchange with elliptic curve systems. Advances in Cryptology—CRYPTO '95, Lecture Notes in Computer Science 1995,963:43-56
- [93] 黄铠,许志伟.可扩展并行计算技术、结构与编程[M]. 北京,机械工业出版社, 2000, 5

- [94] L. G.Valiant. A bridging model for parallel computation. Communications of the ACM. 1990,33(8):103~111
- [95] D. Culler, R. Karp, D. Patterson, et al.. LogP: Towards a Realistic Model of Parallel Computation. Proc. of Fourth ACM SIGPLAN Symp. On Principles and Practices of Parallel Programming, May 1993
- [96] Williams T L., Parsons R J. The heterogeneous bulk synchronous parallel model. In Parallel and Distributed Processing, volume 1800 of Lecture Notes in Computer Science, pages 102--108. Springer-Verlag, Cancun, Mexico, May 2000. 8
- [97] 黄伟民, 陆鑫达, 曾国荪. 异构 BSP 模型及其通信协议. 电子学报, 2000, 28(8):72-75
- [98] 李晓梅, 莫则尧, 胡庆丰, 等. 可扩展并行算法的设计和分析. 北京:国防工业出版社, 2000.7
- [99] 计永昶, 丁卫群, 陈国良, 等. 一种实用的并行计算模型. 计算机学报, 2001, 24(4): 437-441
- [100] G.B.Agnew, R.C.Mullin, I.M.Onyszchuk,et al. An implementation for a fast public-key cryptosystem.Journal of Cryptology,1991,vol 3:63-79
- [101] J.López and R.Dahab. Fast multiplication on elliptic curves over $GF(2^n)$ without precomputation. Proc. Cryptographic Hardware and Embedded Systems—CHES '99, 1999,316-327
- [102] E.Al-Daoud, R.Mahmod et al. A new addition formula for elliptic curves over $GF(2^n)$, IEEE Trans. Computers, 2002,51(8):972-975
- [103] R. Schroeppe, H. Orman, S. O'Malley et al. Fast key exchange with Elliptic Curve Systems, Proc. Advances in Cryptology—Crypto '95,1995, 43-56
- [104] E. De Win, A.Bosselaers,et al. A fast software implementation for arithmetic operations in $GF(2^n)$. Advances in Cryptology, Proc. Asiacrypt '96, K. Kim and T. Matsumoto, eds., 1996,65-76
- [105] D. Hankerson, J.L. Hernandez and A. Menezes. Software implementation of Elliptic Curve Cryptography over binary fields. Proc. Cryptographic Hardware and Embedded Systems-CHES 2000, 2000:1-24
- [106] A. Weimerskirch, C. Paar and S.C. Shantz. Elliptic curve cryptography on a Palm OS Device. Proc. Sixth Australasian Conf. Information Security and Privacy (ACISP 2001), 2001
- [107] IEEE P1363, Standard Specifications for Public Key Cryptography,Draft 9.2001, <http://grouper.ieee.org/groups/1363/>

- [108] J.López and R.Dahab.Improved Algorithms for elliptic curve arithmetic in $GF(2^n)$. Proc. Selected Areas in Cryptography—SAC '98, 1998:201-212
- [109] I.F.Blake, G.Seroussi and N.P.Smart. Elliptic curves in cryptography,1999
- [110] F.Morain and J.Olivos. Speeding up the computations on an elliptic curve using addition-substraction chains. Info. Theory Appl.,1990,vol24:631-543
- [111] J.Pollard. Monte Carlo methods for index computation mod p. Mathematics of Computa- tion.1978,vol.32:918-924
- [112] 唐文,唐礼勇,陈钟.基于 Markov 链的椭圆曲线数乘运算算法性能分析.电子学报,2004,32(11):1778-1781
- [113] 候整风,李岚.椭圆曲线密码系统(ECC)整体算法设计及优化研究.电子学报,2004,32(11):1904-1906
- [114] E.W.Knudsen.Elliptic scalar multiplication using point halving.In Asiacrypt'99. Lecture Notes in Computer Science,1999,vol.1716:135-149
- [115] A. R.Masoleh and M.A.Hasan. Fast normal basis multiplication using general purpose processors. IEEE Trans. Computers, 2003,52(11):1379-1390
- [116] D.J. Yang, C.H. Kim, Y. Park et al. Modified sequential normal basis multipliers for type II optimal normal bases. Proc. Int'l Conf. Computation Science and Its Applications,2005:647-656
- [117] M. Elia and M. Leone.On the inherent space complexity of fast parallel multipliers for $GF(2^m)$. IEEE Trans.Computers, 2002,51(3):346-351
- [118] P. Ning and Y.L. Yin. Efficient software implementation for finite field multiplication in Normal Basis. Proc. Int'l Conf. Information and Comm. Security (ICICS 2001).2001:177-181
- [119] A. R.Masoleh and M.A.Hasan. Efficient digit-serial normal basis multipliers over $GF(2^m)$. ACM Trans. Embedded Computing Systems (TECS), special issue on embedded systems and security, 2004,3(3):575-592
- [120] S. Kwon, K. Gaj, C.H. Kim,et al. Efficient linear array for multiplication in $GF(2^m)$ using a normal basis for elliptic curve cryptography. Proc. Workshop Cryptographic Hardware and Embedded Systems,2004:76-91

攻博期间取得的研究成果

一、科研项目

攻博期间作为主研参与了以下科研项目：

[1] 华为基金项目：数字家庭设备互连规范研究, 2004.6~2005.2。

[2] 华为基金项目：基于 P2P 的网络承载模型及其关键技术研究, 2005.5~2006.5。

二、发表论文

第一作者

[1] 王庆先, 孙世新. 广义最优扩域上的快速运算. 电子与信息学报, 2006, 28(2): 404-407

[2] 王庆先, 孙世新. II 型最优正规基串行乘法器算法设计, 系统工程与电子技术, 2005, 27(8): 1494-1496

[3] 王庆先, 孙世新, 尚明生, 刘宴宾. 并行计算模型研究. 计算机科学, 2004, 31(9): 128-131

[4] Wang Qingxian. The Application of Elliptic Curves Cryptography in Embedded Systems, In Proceedings of the 2nd International Conference on Embedded Software and Systems, ICESS 2005, IEEE Computer Press, 2005, pages: 527-530 (ISTP/EI)

[5] Wang Qingxian, Sun Shixin. A Comparative Study of the Normal Basis Multipliers, In Proceedings of the Fourth International Conference on Optical Internet, COIN'2005, Posts & Telecom Press, 2005, pages: 252-257 (EI)

[6] Wang Qingxian, Shang Mingsheng, Fu Yan. Modular reduction operation based on monic pentanomial. The 6th International Conference on ITS Telecommunication.

[7] Wang Qingxian, Shang Mingsheng, Sun Shixin. Modular reduction for generalized mersenne numbers. Proceedings of The Seventh International Conference on Matrix Analysis and its Applications

[8] Wang Qingxian, Shang Mingsheng, Sun Shixin. Complexity Analysis on Elliptic Scalar Multiplication. International Computer Conference 2006 on Wavelet Active Media Technology and Information Processing

第二和第三作者

- [9]Shang Mingsheng, Wang Qinxian.Optimal sequences for scheduling divisible load on heterogeneous system in non-blocking model of communication. In Proceedings of the 11th Joint International Computer Conference, JICC'2005,World Scientific Publishing Co., 2005, pages:43-46.
- [10]Shang Mingsheng, Liu Yanbin, Wang Qingxian.Scheduling Divisible Loads on Heterogeneous Networks with Arbitrary Topology. In Proceedings of the Fourth International Conference on Optical Internet, COIN'2005, Posts & Telecom Press, 2005, pages:252-257(EI)
- [11] Shang Mingsheng, Sun Shixin and Wang Qingxian.An efficient parallel scheduling algorithm of dependent task graphs. In Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT'2003, IEEE Press, 2003, pages:595-598 (ISTP/EI Compledex:04158108124)
- [12] 尚明生, 王庆先, 孙世新. 集群系统中 BSP 模型上的并行 FFT 设计. 计算机应用, 2002, 22(7):34-36
- [13] 尚明生, 王庆先. 一种新的基于分裂法的矢量量化算法. 四川师范学院学报(自然科学版), 2002, 23(1):71-74

三、教学实践

- [1] 组合设计与组合优化理论, 教学和辅导