GF(2^m)上椭圆曲线标量乘的硬件结构实现

邬贵明 郑 方 谢向辉 吴 东 严忻恺
(数学工程与先进计算国家重点实验室 无锡 214125)

摘 要 基于Reyhani-Masoleh提出的GF(2^m)高斯正规基乘法实现了三拍非流水的正规基乘法器,并基于该乘法器实现了一种高性能López-Dahab标量乘硬件结构。Reyhani-Masoleh算法利用乘法矩阵的对称性降低了乘法的复杂度;而López-Dahab标量乘算法由于采用投影坐标,计算速度快且可以有效降低存储需求。基于Reyhani-Masoleh乘法器的López-Dahab标量乘结构可以有效利用两种算法的优势,可以达到目前最好的标量乘硬件结构的性能。

关键词 正规基乘法器,标量乘,椭圆曲线,有限域算术

中图法分类号 TP302 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.018

Hardware Implementation of Scalar Multiplication on Elliptic Curves over GF(2^m)

WU Gui-ming ZHENG Fang XIE Xiang-hui WU Dong YAN Xin-kai
(State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi 214125, China)

Abstract A three-stage non-pipelined normal base multiplier was implemented based on an algorithm for GF(2^m) multiplication using Gaussian normal base proposed by Reyhani-Masoleh. On basis of the Gaussian normal base multiplier, we presented a high-performance hardware implementation for the López-Dahab algorithm of scalar multiplication over GF(2^m). The Reyhani-Masoleh algorithm can reduce the computation complexity of the multiplication through exploiting the symmetry of the multiplication matrix, and the memory requirement of the López-Dahab algorithm can be reduced by using projective coordinate. Our architecture can benefit from the combination of the two algorithms, making its performance be equivalent to the best architecture to date.

Keywords Normal base multiplier, Scalar multiplication, Elliptic curve, Finite field arithmetic

1 概述

对于二元扩域GF(2^m)上的非超奇异椭圆曲线

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

其中,曲线参数a、$b\in GF(2^m)$,且$b\neq 0$,其上的标量乘运算kP最常用的是López和Dahab提出的快速算法,即López-Dahab标量乘算法[1]。标量乘乘运算kP为一个正整数k与椭圆曲线上的一个点P相乘,即点P自身累加k次。椭圆曲线构成一个Abel加法群,若3个点处于一条直线上,则3个点之和为零点。

目前有很多相关工作都是基于López-Dahab标量乘算法来进行硬件结构的研究。Kim等[2]采用一种高斯正规基乘法器,该乘法器使用字级乘法,具有3级非流水线结构,基于该结构实现了一个高性能的标量乘结构。Azarderakhsh等[3]基于一种流水化的高斯正规基乘法器,面向特殊曲线设计了标量乘结构。Rebeiro等[4]基于一个四级流水的混合位级并行Karatsuba乘法器[5]进行设计,将标量的连续两位的计算进行高效调度,充分利用了乘法器四级流水的特点和点乘算法中数据的相关性,是目前性能最高的椭圆曲线标量乘运算器。

本文基于Reyhani-Masoleh提出的正规基乘法算法,实现了一种可扩展的乘法硬件架构,通过开发乘法矩阵的对称性来降低实现代价,通过长数位的计算粒度来提升性能。基于设计的正规基乘法器结构,提出了López-Dahab标量乘算法的高效硬件实现,与目前性能最高的椭圆曲线标量乘运算器达到相当的性能。

2 López-Dahab算法

López-Dahab标量乘算法由于采用投影坐标,不需要额外的存储需求,每次迭代的过程都相同,只需要在最后的坐标变换中进行求逆计算,计算速度较快。

标准投影坐标(X,Y,Z)通过简单的计算便可以转换成线性坐标(x,y):

$$x=X/Z, y=Y/Z$$

López-Dahab标量乘算法如图1所示。

Input: $k=(k_{t-1},\cdots,k_1,k_0)_2$ with $k_{t-1}=1$, $P=(x,y)\in GF(2^m)$
Output: $kP=(x_3,y_3)$
1. $X_1=x, Z_1=1, X_2=x^4+b, Z_2=x^2$ //初始化,计算P、2P

到稿日期:2013-12-26 返修日期:2014-03-15 本文受中国博士后科学基金(2013M532179),国家高技术研究发展计划(2013AA010105)资助。
邬贵明(1981-),男,博士,工程师,主要研究方向为高性能计算机体系结构、可重构计算,E-mail:wu.guiming@meac-skl.cn;郑 方(1984-),男,博士生,助理研究员,主要研究方向为高性能计算机体系结构、高性能处理器;谢向辉(1958-),男,博士,研究员,主要研究方向为高性能计算机体系结构、高性能处理器;吴 东(1971-),男,博士,副研究员,主要研究方向为高性能计算机体系结构、可重构计算;严忻恺(1989-),男,硕士生,主要研究方向为高性能计算机体系结构、可重构计算。

2. for i=t−2 downto 0 do

3. $T_1=X_1Z_2, T_2=X_2Z_1, T_3=(T_1+T_2)^2$

4. if ($k_i=1$) then

5. $X_1=xT_3+T_1T_2, Z_1=T_3, X_2=X_2^4+bZ_2^4, Z_2=X_2^2Z_2^2$

6. else

7. $X_2=xT_3+T_1T_2, Z_2=T_3, X_1=X_1^4+bZ_1^4, Z_1=X_1^2Z_1^2$

8. end if

9. end for

10. $x_3=\frac{X_1}{Z_1}, y_3=\frac{1}{x}(x+x_3)[(x+x_3)(x+\frac{X_2}{Z_2})+x^2+y]+y$//坐标变换

图 1 López-Dahab 标量乘算法

算法首先进行初始化，计算 P 和 2P 的投影坐标，然后扫描 k 的每一位，对每一位都进行一次点加(第 3 行)和倍点(第 5 行或第 7 行)操作，共有 6 次乘法。根据 k_i 的值，每次倍点和点加的操作数和目标不同。最后一步将投影坐标变换成线性坐标，涉及到 3 次求逆和 5 次乘法。从该算法可以看出，除了最后一步，其它操作仅需要投影坐标中的 X 和 Z 两个坐标参与运算。

3 López-Dahab 标量乘结构

3.1 López-Dahab 标量乘算法

López-Dahab 标量乘算法的第 5 行和第 7 行具有相同的操作，输入和输出不同。为了将两条不同路径的操作合并在一起，使其更适合硬件实现，Ansari 和 Hasan[6] 提出统一寻址的 López-Dahab 标量乘算法，如图 2 所示。在进行点加和倍点之前，先判断 k_{t-2} 是否为 1，为 1 时将 X_1 和 X_2 之间、Z_1 和 Z_2 之间的值进行交换；每次点加和倍点计算完后检测 k 连续两位是否相等，若不相等则进行值交换。

Input: $k=(k_{t-1},\cdots,k_1,k_0)_2$ with $k_{t-1}=1$, $P(x,y)\in GF(2^m)$

Output: $kP=(x_3,y_3)$

1. $X_1=x, Z_1=1, X_2=x^4+b, Z_2=x^2$//初始化，计算 P、2P

2. if ($k_{t-2}=1$) then

3. Swap(X_1,X_2), Swap(Z_1,Z_2)

4. end if

5. for i=t−2 downto 0 do

6. $T_1=X_1Z_2, T_2=X_2Z_1, T_3=(T_1+T_2)^2$

7. $X_2=xT_3+T_1T_2, Z_2=T_3, X_1=X_1^4+bZ_1^4, Z_1=X_1^2Z_1^2$

8. if ($i\neq0$ and $k_i\neq k_{i-1}$) or ($i=0$ and $k_i=1$) then

9. Swap(X_1,X_2), Swap(Z_1,Z_2)

10. end if

11. end for

12. $x_3=\frac{X_1}{Z_1}, y_3=\frac{1}{x}(x+x_3)[(x+x_3)(x+\frac{X_2}{Z_2})+x^2+y]+y$//坐标变换

图 2 统一寻址的 López-Dahab 标量乘算法

3.2 点加和倍点算术结构

点加和倍点算术结构的基础是高斯正规基乘法器，本文基于 Reyhani-Masoleh 提出的正规基乘法算法[7]，实现了一种可扩展的乘法硬件架构，通过开发乘法矩阵的对称性来降低实现代价，通过长数位的计算粒度来提升性能。该结构是一种参数化的非流水结构，可以根据具体情况调节位宽大小和吞吐率，具有较好的可扩展性和灵活性。

图 3 为图 2 算法中第 6 行和第 7 行的数据流图。如果采用非流水 M 拍的单个乘法器，一次迭代需要 $6M$ 个时钟节拍，这里不考虑指数为 2^s 的求幂和加法代价，在 GNB 下指数为 2^s 的求幂只是简单的循环移位，加法只是简单的一级异或。采用 3 个非流水的乘法器就可以足够开发并行性，这时一次迭代需要 $2M$ 个时钟节拍。在采用 3 个非流水乘法器时，可以将操作分成两个阶段调度在这 3 个乘法器上运行。图 3 的虚线将操作划分成两部分调度到两个阶段分别完成，这里采用了 Kim 等[2] 提出的调度方法。基于图 3 得到了一种完成点加和倍加操作的硬件结构，如图 4 所示，这样两个阶段的操作可以分时复用同一个数据通路。

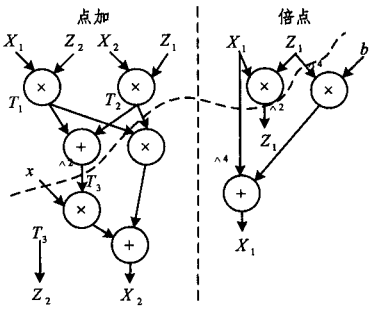

图 3 点加和倍点操作的数据流图

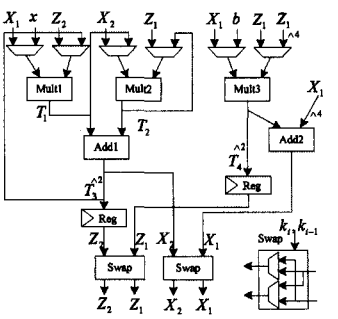

图 4 实现点加和倍点的运算单元

该结构由 3 个乘法部件、2 个加法部件和 2 个 Swap 部件组成。两个寄存器用于缓冲数据，数据延迟拍数等于乘法器完成一次乘法的拍数，这样，输出的 X_1、X_2、Z_1 和 Z_2 同时有效。实际上，点加的计算(T_3 和 X_2 的计算)与 X_1 和 X_2 的交换、Z_1 和 Z_2 的交换无关，交换和不交换的计算结果是一致的，X_1 和 X_2、Z_1 和 Z_2 在点加计算中位置是对称的，因此点加计算的输入 X_1、X_2、Z_1 和 Z_2 不需要采用 Swap 部件交换后的数据，这样可以减少驱动，提升性能。图 4 结构可以在 $2M$ 拍内完成一次迭代，因此完成图 2 算法中的主要循环需要 $2M(t-1)$ 拍。

3.3 坐标转换结构

3.3.1 求逆算术结构

坐标转换过程中最费时的操作是求逆操作，这里采用 Itoh-Tsujii(IT)算法来完成求逆。Itoh-Tsujii 算法基于费马小定理来实现对 $GF(2^m)$ 域上的元素进行求逆运算。费马小定理可描述如下：若 p 是素数，a 是正整数且不能被 p 整除，则

$$a^{p-1}\equiv1(\mathrm{mod}\ p) \tag{1}$$

于是

$$a^{-1}\equiv a^{p-2}(\mathrm{mod}\ p) \tag{2}$$

设 $A\in GF(2^m)$，有 $A^{2^m-1}=1$，于是

$$A^{-1}=A^{2^m-2}=A^{2(2^{m-1}-1)}$$

Itoh-Tsujii 算法基于以下关系式将 $A^{2(2^{m-1}-1)}$ 的计算进行展开：

$$2^s-1=\begin{cases}(2^{\frac{s}{2}}-1)(2^{\frac{s}{2}}+1), & s\text{为偶数}\\ 2(2^{\frac{s-1}{2}}-1)(2^{\frac{s-1}{2}}+1)+1, & s\text{为奇数}\end{cases} \quad (3)$$

根据使用 Itoh-Tsujii 算法计算 $A\in GF(2^{163})$ 的逆的过程，可知算法的每一步都具有相关性，只能串行执行。因此，使用一个串行的乘法器即可，乘法器的两个输入通过选择器进行选择。图 5 所示结构为 $GF(2^{163})$ 上求逆的硬件结构，该结构也与 Kim 等[2]提出的结构类似。

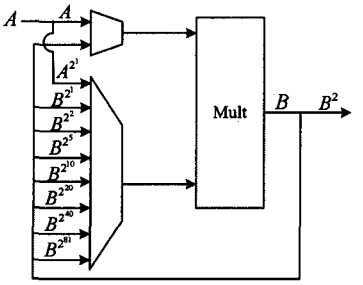

图 5 Itoh-Tsujii 求逆算术结构

3.3.2 坐标转换结构

坐标转换的表达式为：

$$x_3=\frac{X_1}{Z_1},y_3=\frac{1}{x}(x+x_3)[(x+x_3)(x+\frac{X_2}{Z_2})+x^2+y]+y \quad (4)$$

坐标转换结构在求逆结构的基础上进行搭建。坐标变换过程中，每次求逆之后的结果需要乘以一个元素，在具体实现时，可以将求逆和求逆之后的乘法封装在一起，组成一个有限域除法的硬件结构，每次除法需要进行 10 次乘法操作。这里，完成坐标转换的乘法操作需要和除法共用一个乘法器。在这样的约束下，可以将式(4)分解成如下步骤：

$(1)x_3=\frac{X_1}{Z_1}$；$(2)t_1=x+x_3$；$(3)t_2=\frac{X_2}{Z_2}$；

$(4)t_3=x+t_2$；$(5)t_4=t_1*t_3$；$(6)t_5=t_4+x^2+y$；

$(7)t_6=\frac{t_5}{x}$；$(8)t_7=t_6*t_1$；$(9)y_3=t_7+y$。

我们采用了 FSM 来控制以上 9 个步骤的完成。其中的除法只是在求逆的基础上多了一个乘操作，除法的结构与求逆的结构基本相同。

4 实验结果

针对有限域 $GF(2^{163})$ 实现了 López-Dahab 标量乘算法，有限域乘法器采用的是根据 Reyhani-Masoleh 算法设计的结构，选择字宽为 55 的 3 拍非流水线结构。为优化时序，将最后的乘法器的结果又加一次寄存，因此变成了 4 拍的非流水线结构。这里还可以采用字宽为 41 的 4 拍非流水线结构来提升设计的性能。

本节的结构采用 Verilog HDL 进行描述，以 Xilinx Virtex-4 XC4VLX80 为目标器件进行综合。主要基于 ModelSim6.5d 工具进行实验评测，综合结果由 Xilinx ISE 13.3 给出。表 1 给出了两种结构的综合结构，结构 1 中，点加和倍点算术结构使用了 3 个乘法器，求逆部件使用了一个乘法，共使用了 4 个乘法器；结构 2 中，求逆部件与点加和倍点算术部件共用了一个乘法器，共使用了 3 个乘法器。可以看出，结构 1 和结构 2 最大频率是相同的，而结构 2 的 Slice 数目为结构 1 的 83%。从逻辑实现上，结构 2 相比结构 1 逻辑稍微复杂(有共用带来的切换逻辑)，结构 2 的运行频率应低于结构 1，但由于结构 2 比结构 1 占用 FPGA 资源更少，稍微减少了布局布线的压力，使得最大频率比结构 1 略有提升。

表 1 目标器件为 Xilinx XC4VLX80 的综合结果

	结构 1(4 个乘法器)	结构 2(3 个乘法器)
Slice 数目	29736(82%)	24791(69%)
Slice Flip Flops 数目	7948(11%)	7116(9%)
Slice LUT 数目	56337(78%)	45620(63%)
最大频率(综合后)	242.082	206.453
最大频率(布局布线后)	151.561	151.653

表 2 将我们的设计与最近的 3 个相关工作进行比较，不同工作的 López-Dahab 标量乘算法性能的区别在于所采用的有限域乘法器结构和调度方法。Kim 等人[2]采用一种高斯正规基乘法器，该乘法器使用字级乘法，具有 3 级非流水线结构，倍点和点加的调度采用两级分时复用的方法，本文也采取了相同的调度方法，该结构也采用了基 2 的 Itoh-Tsujii 算法。Azarderakhsh 等人[3]采用了一种流水化的高斯正规基乘法器，面向 Binary Edwards 曲线和 Generalized Hessian 曲线等特殊曲线进行了优化设计。Rebeiro 等人[4]提出的乘法器采用了具有 4 拍流水线结构的多项式基混合位级并行 Karatsuba 有限域乘法器[5]，基于该乘法器，同时将标量的连续两位的操作进行流水化调度，充分利用了乘法器，在采用一个乘法器的情况下高效地实现了 López-Dahab 标量乘算法，是目前最好的结构之一，同时实现代价也最低。Chelton 等[8]基于 ASIP(专用指令集处理器)方法，提出 3 种复杂指令来实现倍点和点加，它主要基于一种亚流水的位级 Mastrovito 并行乘法器。

从表 2 可以看出，提出的结构与 Kim 和 Rebeiro 的结构时钟周期接近，但提出的结构由于运行频率更高，与其他结构相比计算时间最短。结构 2 和 Kim 的结构均采用 3 个乘法器，因此硬件实现代价相近，Rebeiro 结构仅采用一个流水化的乘法器，硬件实现代价最低。

表 2 不同结构性能比较

结构	FPGA	Slice 数目	最大频率	时钟周期数	计算时间(us)
Kim[2]	XC4VLX80	24363	143	1446	10.1
Azarderakhsh[3]	XC4VLX100	12834	196	3372	17.2
Rebeiro[4]	XC4VLX80	8070	147	1429	9.7
Chelton[8]	XC4VLX200	16209	153	3010	19.5
结构 1(4 个乘法器)	XC4VLX80	29736	151	1432	9.5
结构 2(3 个乘法器)	XC4VLX80	24791	151	1432	9.5

结束语 本文基于一种 3 拍非流水高斯正规乘法器设计并实现了 López-Dahab 标量乘结构。设计中为了提升运行频率和性能，采用了多个乘法器，并对 3 个乘法器的结构和 4 个乘法器的结构的性能进行了对比，发现 3 个乘法器的结构在性能和实现代价上优于 4 个乘法器的结构。本文设计的结构是目前性能最好的结构之一，完成一次标量乘需要 9.5us，即在 1s 内可以完成 105263 次标量乘。下一步工作是采用流水化的乘法器和更好的调度方法来减小实现代价。

(下转第 89 页)

解及平均解均优于GA算法和PSO算法，而且任务规模越大越明显。GA算法和PSO算法容易出现早熟收敛，陷入局部最优。而CS算法稳定性更好，收敛精度高；对于求解时间，CS算法也明显快于两种对比算法，收敛速度更快，执行时间较两种对比算法缩短超过60%。

由此可知，虽然3种算法均为随机搜索算法，但是CS算法的稳定性更好，求解质量更高，执行速度更快，体现了布谷鸟搜索算法优良的进化机制。

结束语 为了提高多处理器系统中的任务调度效率，本文基于布谷鸟搜索算法提出了一种新的任务调度算法。该算法能充分利用布谷鸟搜索算法求解的精确性与高效性特点，以全部任务的最晚完成时间最小为目标，利用基于任务优先权的编码方案使连续的布谷鸟搜索算法适用于离散的多处理器任务调度问题。通过与遗传算法及粒子群算法性能测试比较结果可知，CS算法的求解质量更高且执行速度更快，能有效缩短任务的完成时间，提高多处理器系统的性能。

参 考 文 献

[1] Kwok Y K, Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors[J]. ACM Computing Surveys (CSUR), 1999, 31(4): 406-471

[2] Daoud M I, Kharma N. A high performance algorithm for static task scheduling in heterogeneous distributed computing systems [J]. Journal of Parallel and Distributed Computing, 2008, 68 (4): 399-409

[3] 陈华平，黄刘生. 启发式任务调度中的处理器选择策略[J]. 软件学报，1999，10(11)：1194-1198

[4] 石威，郑纬民. 相关任务图的均衡动态关键路径调度算法[J]. 计算机学报，2001，24(9)：991-997

[5] Corrêa R C, Ferreira A, Rebreyend P. Scheduling multiprocessor tasks with genetic algorithms[J]. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(8): 825-837

[6] Gupta S, Agarwal G, Kumar V. An Efficient and Robust Genetic Algorithm for Multiprocessor Task Scheduling[J]. International Journal of Computer Theory and Engineering, 2013, 5(2): 377-382

[7] 叶春晓，陆杰. 基于改进遗传算法的网格任务调度研究[J]. 计算机科学，2010，37(7)：233-235

[8] Omara F A, Arafa M M. Genetic algorithms for task scheduling problem[J]. Journal of Parallel and Distributed Computing, 2010, 70(1): 13-22

[9] Hwang R, Gen M, Katayama H. A comparison of multiprocessor task scheduling algorithms with communication costs[J]. Computers & Operations Research, 2008, 35(3): 976-993

[10] 王成昌，陈闵中，方钰，等. 基于混合粒子群算法的网格任务调度[J]. 计算机科学，2012，39(2)：18-21

[11] Al Badawi A, Shatnawi A. Static scheduling of directed acyclic data flow graphs onto multiprocessors using particle swarm optimization[J]. Computers & Operations Research, 2013, 40 (10): 2322-2328

[12] Yin P Y, Yu S S, Wang P P, et al. A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems[J]. Computer Standards & Interfaces, 2006, 28(4): 441-450

[13] Sivanandam S N, Visalakshi P, Bhuvaneswari A. Multiprocessor Scheduling Using Hybrid Particle Swarm Optimization with Dynamically Varying Inertia[J]. IJCSA, 2007, 4(3): 95-106

[14] 高尚，杨静宇. 多处理机调度问题的粒子群优化算法[J]. 计算机工程与应用，2005，41(27)：72-73

[15] Yang X S, Deb S. Engineering optimisation by cuckoo search [J]. International Journal of Mathematical Modeling and Numerical Optimisation, 2010, 1(4): 330-343

[16] Gandomi A H, Yang X S, Alavi A H. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems[J]. Engineering with computers, 2013, 29(1): 17-35

[17] Ouaarab A, Ahiod B, Yang X S. Discrete cuckoo search algorithm for the travelling salesman problem[J]. Neural Computing and Applications, 2014, 24(7/8): 1659-1669

[18] 邱卫东，陈燕，李洁萍，等. 一种实时异构嵌入式系统的任务调度算法[J]. 软件学报，2004，15(4)：504-511

[19] Yang X S, Deb S. Cuckoo search via Lévy flights[C]// World Congress on Nature & Biologically Inspired Computing, 2009 (NaBIC 2009). IEEE, 2009: 210-214

[20] Standard Task Graph Set [OL]. http://www.kasahara.elec.waseda.ac.jp/schedule/index.html

(上接第81页)

参 考 文 献

[1] López J, Dahab R. Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation[C]//CHES. 1999: 316-327

[2] Kim C H, Kwon S, Hong C P. FPGA Implementation of High Performance Elliptic Curve Cryptographic processor over GF (2^{163})[J]. Journal of Systems Architecture - Embedded Systems Design, 2008, 54(10): 893-900

[3] Azarderakhsh R, Reyhani-Masoleh A. Efficient FPGA Implementations of Point Multiplication on Binary Edwards and Generalized Hessian Curves Using Gaussian Normal Basis[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2011, 19(1): 1453-1466

[4] Rebeiro C, Roy S S, Mukhopadhyay D. Pushing the Limits of High-Speed GF (2^m) Elliptic Curve Scalar Multiplication on FPGAs[C]//CHES. 2012: 494-511

[5] Rebeiro C, Mukhopadhyay D. Power Attack Resistant Efficient FPGA Architecture for Karatsuba Multiplier[C]// VLSID. 2008: 706-711

[6] Ansari B, Anwar M. High performance architecture of elliptic curve scalar multiplication[J]. IEEE Transactions on Computers, 2008, 57(11): 1443-1453

[7] Reyhani-Masoleh A. Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases[J]. IEEE Transactions on Computers, 2006, 55(1): 34-47

[8] Chelton N, Benaissa M. Fast Elliptic Curve Cryptography on FPGA[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2008, 16(2): 198-205

[9] Lai Y, Hung Y, Yang H, et al. High-Performance Architecture for Elliptic Curve Cryptography over Binary Field[C]//ISCAS. 2010: 3933-3936

[10] Shu C. Hardware Architectures of Elliptic Curve Based Cryptosystems over Binary Fields[D]. George Mason University, 2008