密码学报 (中英文) ISSN 2097-4116 CN 10-1900/TN
Journal of Cryptologic Research, 2024, 11(6): 1354–1369

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

低功耗软硬结合 SM2 算法实现*

孔团结, 郑昉昱, 郭 润, 荆继武, 张子昂

中国科学院大学 密码学院, 北京 100049
通信作者: 郭润, E-mail: guorun@ucas.ac.cn

摘 要: 本文提出一种低功耗软硬结合 SM2 算法实现方案, 利用低功耗处理器 Cortex-M0 调用硬件中 SM2 协处理器的功能模块, 可以在资源受限的 RFID 设备中实现. 为减少 SM2 域运算层的功耗和资源, 本文使用 KOM 算法的串行计算和并行计算设计低功耗乘法器, 并且优化模约减, 合并模加减. 在多倍点运算层中采用了固定窗口 NAF 标量乘算法, 并对其进行部分改进, 该算法在面对 SPA 攻击时既保持了良好的安全性, 又通过减少计算量的方式提高了性能, 并降低了功耗. 本文采用软硬结合的方式来实现 Barrett 算法、SM2 签名和验签算法, 达到提升整体的性能并降低资源消耗的目标. 实验测试结果表明, SM2 协处理器在时钟频率为 50 MHz 的条件下, 计算任意标量点的速度可达到 0.869 ms/次, 且仅消耗了 4.9 μJ 的能量. Cortex-M0 和 SM2 协处理器通过软硬件结合方式, 签名速度可达 0.98 ms/次, 验签速度为 1.74 ms/次.

关键词: RFID; SM2 算法; 软硬件结合; 低功耗

中图分类号: TP309.7 **文献标识码:** A DOI: 10.13868/j.cnki.jcr.000741

中文引用格式: 孔团结, 郑昉昱, 郭润, 荆继武, 张子昂. 低功耗软硬结合 SM2 算法实现[J]. 密码学报 (中英文), 2024, 11(6): 1354–1369. [DOI: 10.13868/j.cnki.jcr.000741]

英文引用格式: KONG T J, ZHENG F Y, GUO R, JING J W, ZHANG Z A. Implementation of low-power software and hardware combination for SM2 algorithm[J]. Journal of Cryptologic Research, 2024, 11(6): 1354–1369. [DOI: 10.13868/j.cnki.jcr.000741]

Implementation of Low-Power Software and Hardware Combination for SM2 Algorithm

KONG Tuan-Jie, ZHENG Fang-Yu, GUO Run, JING Ji-Wu, ZHANG Zi-Ang

School of Cryptology, University of Chinese Academy of Sciences, Beijing 100049, China
Corresponding author: GUO Run, E-mail: guorun@ucas.ac.cn

Abstract: This study proposes a low-power software-hardware combined SM2 algorithm implementation scheme. Utilizing a low-power Cortex-M0 processor to call the functionality module of the SM2 coprocessor in the hardware, it could be realized in RFID of restricted resources. To reduce the power consumption and resource usage in the SM2 field arithmetic layer, this study designs low-power multipliers using both serial and parallel computations based on the KOM algorithm. Additionally, optimizations such as modular reduction and merging of modular additions and subtractions are applied. In the multiple-point arithmetic layer, a width-w NAF scalar multiplication algorithm

* 基金项目: 国家重点研发计划 (2022YFB3103301)
Foundation: National Key Research and Development Program of China (2022YFB3103301)
收稿日期: 2023-08-12 定稿日期: 2024-01-09

is adopted and partially improved, which maintains good security against SPA attacks while improving performance and reducing power consumption by reducing computational complexity. This study adopts a software-hardware combined approach to implement the Barrett algorithm, SM2 signature, and verification algorithm, aiming to improve overall performance and reduce resource consumption. Experimental test results show that the SM2 coprocessor can compute arbitrary scalar points at a speed of 0.869 ms per calculation with a clock frequency of 50 MHz, consuming only 4.9 μJ of energy. Through the software-hardware combination of Cortex-M0 and the SM2 coprocessor, the signature speed reaches 0.98 ms per calculation, and the verification speed is 1.74 ms per calculation.

Key words: RFID; SM2; hardware and software; low power

1 引言

无线射频识别技术 (radio frequency identification, RFID) 是一种利用射频方式进行非接触双向通信的自动识别技术. RFID 系统一般由电子标签 (tag)、阅读器 (reader) 和数据交换与管理系统 (processor) 三大部分组成 [1]. RFID 系统的工作流程大致如下: 当带有标签的物体进入阅读器的通信区域时, 由阅读器发射的调制射频信号被标签接收. 标签解调射频信号, 根据解调后的信号执行相应的功能, 随后发送响应给阅读器. 阅读器将标签的响应发送到数据管理系统, 在数据管理系统中执行相应的功能. 标签工作过程中消耗的能量往往由阅读器发出的射频脉冲所提供.

随着 RFID 技术在各个领域的普及, 对 RFID 的安全性提出了越来越高的要求. 在 RFID 标签芯片中集成 SM2 椭圆曲线公钥密码算法标准, 可增强其安全性并扩展 RFID 的应用范围, 比如在物流与供应链管理、物品识别与管理、工业控制、智能家居与物联网等领域. 当标签中含有财务或者隐私等数据时, SM2 算法可以保证标签与阅读器之间的通讯是安全的. 同时, 带有私钥的 RFID 标签芯片, 可以实现物品的防伪功能. 如果每一个 RFID 标签芯片都有唯一、不可修改且攻击者无法获取的私钥, 造假者难以获得私钥来伪造产品. 椭圆曲线密码 (elliptic curve cryptography, ECC) 是一种高效的公钥密码算法. 在相同的安全级别下, ECC 比 RSA 需要的密钥参数更短. 例如, 256 位公钥长度的 ECC 可以提供与 3072 位公钥长度的 RSA 相当的安全性 [2]. 所以, ECC 很适合应用到 RFID 标签芯片上. SM2 算法由国家密码管理局发布, 是基于 ECC 的公钥密码算法, 并被确立为中国商业应用的 ECC 标准. 关于基于 SM2 的签名, 公钥加密和密钥建立方案的更多详细信息, 请参见文献 [3–7].

虽然使用 SM2 算法可以增加 RFID 标签芯片的安全性, 让 RFID 标签拥有更广泛的应用场景, 但是将 SM2 集成到资源有限的 RFID 标签芯片, 有三个关键问题需要考虑. 一是 SM2 算法中涉及的椭圆曲线点运算以及相关的大整数模运算较多, 导致其计算量相对较大、消耗能量多, 因此在实现 SM2 算法时需要考虑低功耗和性能优化. 二是 RFID 标签芯片具有便携性、低成本和应用场景广泛的优势, 所以在设计 SM2 协处理器时需要考虑降低资源消耗. 三是加密设备在运行过程中会泄露功耗、电磁辐射等物理信息, 容易造成密钥被窃取 [8]. 简单功耗攻击 (simple power analysis, SPA) 可以通过直接观察 ECC 处理器的功耗轨迹来推断密钥, 在 RFID 标签芯片中实现 SM2 算法时需要考虑抗 SPA 攻击.

目前, 关于 RFID 标签芯片中 ECC 加速器的研究多数局限于二元扩域上的椭圆曲线 [9–13], 且密钥长度较短. 虽然短密钥可以减少资源和能量消耗, 但其密钥的安全性较低. 同时, RFID 标签芯片中关于素域 SM2 协处理器的研究很少. 本文旨在进行低功耗素域 SM2 设计研究, 以实现满足 RFID 标签芯片需求的低功耗 SM2 协处理器.

SM2 算法是一种基于椭圆曲线密码学的安全算法, 它分为四个层次: 域运算层、点运算层、多倍点运算层和 SM2 协议层. 域运算层是 SM2 的基础组成部分, 由于其中模乘的计算量较大, 所以是优化的重点. 文献 [14] 在 1985 年提出蒙哥马利乘法, 其避免计算带余的除法从而提高了对模乘计算效率. 此外, 文献 [15] 还提出梅森素数法, 可以在特定素数 p 的情况下提高计算效率. Karatsuba-Ofman (KOM) 算法 [16] 和 Toom-Cook 算法 [17] 都是高效的多项式乘法算法, 对乘法模块进行了一定的性能优化. 现有的一些文献都关注 ECC 处理器的高性能 [18–21], 使用 Karatsuba-Ofman 算法或者梅森素数来对模乘进行加速. 然而, 这些硬件加速器大多数都只是做了标量乘的性能测试和资源消耗评估, 并没有注意到能量的

消耗, 且大多数文献都没有完成整个签名和验签的功能测试.

为了应对 ECC 算法面临的 SPA 攻击的威胁, 在算法层面上, 蒙哥马利标量乘算法 [22] 和 double-and-add-always 算法 [23] 被广泛应用于 ECC 中, 无论密钥位是 0 或 1, 它们规定标量乘算法中的倍点和点加操作总是执行. 尽管这些算法被广泛应用, 但它们带来了计算量增加的挑战. 文献 [24] 提出一种固定窗口非邻接形式 (non-adjacent form, NAF) 标量乘算法, 它可以让点加与倍点按照固定执行顺序来执行, 以此来实现抗 SPA 攻击, 同时还会减少一定的点加计算量.

1.1 本文贡献

本文通过实验探究 KOM 算法并行计算中最小乘法器的位数与其能量消耗和资源消耗之间的关系, 来设计实现低功耗 64 位乘法器. 利用实现的低功耗 64 位乘法器采用 KOM 算法的串行计算, 来实现低功耗 256 位乘法器. 此外, 通过优化实现模约减, 合并实现模加减, 来实现降低功耗、减少资源消耗的目标.

在硬件中实现固定窗口 NAF 标量乘算法, 为低功耗且需要抗 SPA 攻击的环境提供了高效计算标量乘的解决方案. 此外, 本文对该算法进行了一定的改进, 优化掉最后一次的点加计算, 从而进一步减少了资源消耗和能量消耗.

本文采用软硬件结合的方法来实现 SM2 的签名以及验签, 这种设计将会发挥软件和硬件各自的优势, 提高了系统性能, 并节约了硬件资源的使用.

在 130 nm CMOS 标准单元库上进行实验验证, SM2 协处理器计算标量乘的速度达到了 0.869 ms/次, 且仅消耗了 4.9 μJ 的能量. 此外, 本文在搭载 Cortex-M0 处理器的开发板上采用软硬结合的方式实现 SM2 签名和验签功能, 签名速度达到了 0.98 ms/次, 验签速度达到了 1.74 ms/次.

1.2 本文组织结构

第 2 节介绍背景知识, 包括 SM2 椭圆曲线群和软硬件开发平台介绍; 第 3 节介绍 SM2 所需的功能模块, 包括模乘、模逆、模加减、标量乘算法以及点加与倍点的实现; 第 4 节介绍 SM2 软硬结合的整体设计方案, 包括 SM2 整体设计架构、主控接口; 第 5 节对 SM2 算法的签名与验签进行软硬结合实现, 并与其他方案在性能与功耗方面进行比较; 第 6 节总结全文.

2 背景知识

2.1 SM2 椭圆曲线群

SM2 定义在素域 F_p 上的椭圆曲线方程可表示为

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

其中, $a, b \in F_p$ 且 $(4a^3 + 27b^2) \bmod p \neq 0$. 本文将国家密码管理局给出的推荐参数中 n 和 p 的取值设为 N_{SM2} 与 P_{SM2} [7]. 椭圆曲线上的点集记为 $E(F_p) = \{(x, y)|x, y \in F_p \text{ 且 } y^2 = x^3 + ax + b\} \cup \{O\}$, 其中 O 是椭圆曲线的无穷远点. 在仿射坐标系下椭圆曲线群的加法操作具有以下性质:

(1) $O + O = O$;

(2) $\forall P = (x, y) \in E(F_p)\backslash\{O\}$, $P + O = O + P = P$;

(3) $\forall P = (x, y) \in E(F_p)\backslash\{O\}$, P 的逆元 $-P = (x, -y)$, 且 $P + (-P) = O$;

(4) $P_0 = (x_0, y_0) \in E(F_p)\backslash\{O\}$, $P_1 = (x_1, y_1) \in E(F_p)\backslash\{O\}$, 那么 $P_2 = (x_2, y_2) = P_0 + P_1 \neq O$, 则

$$\begin{cases} x_2 = \lambda^2 - x_0 - x_1 \\ y_2 = \lambda(x_0 - x_2) - y_0 \end{cases}, \quad \text{其中} \lambda = \begin{cases} \frac{y_1 - y_0}{x_1 - x_0}, & x_0 \neq x_1 \\ \frac{3x_0^2 + a}{2y_0}, & x_0 = x_1 \end{cases}.$$

当 $P_0 \neq P_1$ 时, 也就是对应公式中 $x_0 \neq x_1$ 时, $P_0 + P_1$ 称为椭圆曲线上的点加运算; 当 $P_0 = P_1$ 时, 也就是对应公式中 $x_0 = x_1$ 时, $P_0 + P_1 = 2P_0$ 称为椭圆曲线上的倍点运算.

标量乘也称为多倍点运算, 是决定椭圆曲线密码体制运算速度的关键. 椭圆曲线的标量乘运算是指曲线上的某一基点 P 与整数点 k 进行乘法运算, 也就是 k 个 P 相加, 即 $kP=\sum_1^k P=P+P+\cdots+P$.

为便于理解, 本文将椭圆曲线的点加表示为 PointAdd(Q,P), 其中 Q 和 P 为椭圆曲线上的点, 而椭圆曲线的倍点表示为 PointDouble(Q,w), 其中 w 表示将点 Q 连续进行倍点的次数.

2.2 软件开发平台

Cortex-M 系列处理器是由 ARM 公司推出的一种低功耗、高性能的 32 位 RISC 处理器, 广泛应用于嵌入式系统、智能设备、物联网等领域. 在 Cortex-M 系列中, Cortex-M0 是最基础的型号, 它的主要特点是低功耗、低成本、高性能和易于集成. 相比于其他型号的 Cortex-M 内核, Cortex-M0 的面积和功耗都较小, 所以可以在资源受限的环境中得到广泛应用. Cortex-M0 处理器核所具备的低功耗和高性能特点恰好满足了 RFID 标签芯片的需求.

Keil uVision5 是一款由 ARM 公司开发的嵌入式系统开发工具套件. 它提供了全面的软件开发环境, 用于嵌入式系统的设计、开发和调试. Keil 支持多种微控制器架构, 包括 ARM Cortex-M 系列和 8051 系列等. Keil 的编译器和调试器能够针对不同的微控制器架构进行优化, 并提供强大的调试功能, 包括代码跟踪、变量监视和断点设置等.

2.3 硬件开发平台

Design Compiler 是由 Synopsys 公司开发的一款逻辑合成工具. 它能够根据硬件描述语言和约束条件进行设计, 并针对特定的工艺库自动综合出一个经过优化的门级电路. 它可以接受多种输入格式, 如硬件描述语言、原理图和网表等, 并产生多种性能报告, 在缩短设计时间的同时提高设计性能. 此外, 它还可以通过使用门控时钟技术, 有效降低芯片的功耗.

PrimePower 是一种针对复杂门级设计的动态全芯片功耗验证工具, 具备门级功耗分析的能力. 它能够准确而高效地测量 ASIC/SOC 设计中的平均功耗和峰值功耗. PrimePower 的全面功耗验证功能帮助设计人员选择合适的封装方案, 确定散热需求, 并确保设计的正确性.

Vivado 是由 Xilinx 公司开发的一款集成电路设计工具套件, 其提供了全面的硬件描述语言设计、综合、实现和验证功能, 适用于可编程逻辑器件的设计. Vivado 具有先进的设计自动化和优化功能, 能够帮助设计人员快速、高效地完成复杂的数字电路设计. 它还具有高级综合和优化功能, 可以针对目标设备和设计约束进行智能综合和布局布线, 以实现最佳的时序性能和资源利用率. 同时, 为确保设计人员设计的正确性和稳定性, 它还支持多种验证方法, 包括仿真、时序分析和逻辑等静态验证.

3 SM2 功能模块

3.1 软硬件功能模块划分

为加速 SM2 算法整体的性能和减少资源消耗, 本文采用软硬结合的方式来实现 SM2 算法中所需的功能, 功能之间的依赖关系以及软硬的功能划分如图 1 所示. 其中标量乘及底层运算完全由硬件实现, SM2 中协议层的签名和验签以及一些辅助函数由软件实现.

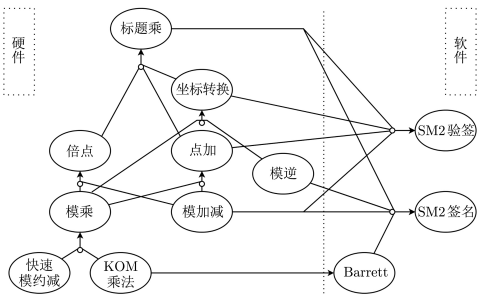


图 1 SM2 算法功能模块拓扑结构图

Figure 1 Topology of functional modules of SM2 algorithm

3.2 模乘

本节将介绍两种求模乘的算法, 第一个方法是在硬件中, 针对素数 P_{SM2} 采用 KOM 算法和快速模约减算法相结合的方式来实现模乘, 该方法的优势是模乘的计算量小、消耗的能量少、计算效率高, 劣势是对素数有限制要求; 第二种方法是采用软硬结合的方式来实现 Barrett 算法, 应用到 SM2 协议层中针对 N_{SM2} 的模乘运算, 该方法的优势是对参数没有限制要求, 劣势是需要多次进行乘法计算, 消耗的能量相对较多.

3.2.1 低功耗乘法

本小节旨在优化 SM2 算法中经常使用且功耗占比较大的乘法模块. 为此, 本文采用 Karatsuba-Ofman 算法 (KOM) [16], 该算法采用分治方法降低了乘法的计算复杂度, 从而达到降低功耗且提升性能的目的, 对于低功耗 RFID 系统具有重要意义. 对于 n 位整数 A 乘以 n 位整数 B, KOM 算法的核心思想如下:

$$\begin{aligned} A \times B &= \{a_1, a_0\} \times \{b_1, b_0\} \\ &= (a_1 2^{\frac{n}{2}} + a_0)(b_1 2^{\frac{n}{2}} + b_0) \\ &= a_1 b_1 2^n + (a_1 b_0 + a_0 b_1) 2^{\frac{n}{2}} + a_0 b_0 \\ &= a_1 b_1 2^n + ((a_0 + a_1)(b_0 + b_1) - a_1 b_1 - a_0 b_0) 2^{\frac{n}{2}} + a_0 b_0. \end{aligned}$$

如上述公式, KOM 算法将一个 n 位的乘法计算转换成 2 个 $\frac{n}{2}$ 位和 1 个 $\frac{n}{2}+1$ 位的乘法计算以及一些加法计算. 与相比传统的移位乘法算法的 $O(n^2)$ 计算复杂度, KOM 算法可以将乘法的算法计算复杂度降低到 $O(n^{\log 3})$, 从而具有更快的计算性能, 以及更少的能量消耗, 详细内容见算法 1.

算法 1 KOM 算法

Input: A, B, 乘法器位宽 n
Output: $C = A \times B$

if $n == 16$ **then**
  return $C = A \times B$
end
$A = \{a_1, a_0\} = a_1 \times 2^{\frac{n}{2}} + a_0$;
$B = \{b_1, b_0\} = b_1 \times 2^{\frac{n}{2}} + b_0$;
$As = a_1 + a_0$;
$Bs = b_1 + b_0$;
$C_0 = \mathrm{KOM}(a_1, b_1, \frac{n}{2})$;
$C_1 = \mathrm{KOM}(a_0, b_0, \frac{n}{2})$;
$C_2 = \mathrm{KOM}(As[\frac{n}{2}:1], Bs[\frac{n}{2}:1], \frac{n}{2})$; //$As[\frac{n}{2}:1]$ 表示 As 二进制展开后第 1 位到第 $\frac{n}{2}$ 位构成的值
$C_{2\mathrm{adj}} = (As[0]\&!Bs[0])?Bs :$ //$As[0]$ 表示 As 二进制展开后第 0 位的值
  $(!As[0]\&Bs[0])?As :$
  $(As[0]\&Bs[0])?As + Bs - 1 : 0$;
$\mathrm{mid} = \{C_0, C_1[n-1:\frac{n}{2}]\} - (C_0 + C_1)$;
$\mathrm{mid1} = C_2 \ll 2 + \mathrm{mid} + C_{2\mathrm{adj}}$;
return $\{\mathrm{mid1}, C_1[\frac{n}{2} - 1 : 0]\}$

本文采用 KOM 算法串行计算和并行计算相结合的方案来实现低功耗乘法器, 其中两者之间最大的差异是对乘法器是否进行了复用, 如图 2 和图 3 所示. 串行计算方案只用到了一个乘法器 M0, 这样做虽然会增加计算的周期数, 但是会减少资源的消耗, 适合乘法器位数 n 较长的情况. 而并行计算方案会用到多个乘法器, 虽然会增加一部分资源消耗, 但是计算完成的延时很短, 适合乘法器位数 n 较短的情况.

本文在实现 256 位乘法器的实现方案时, 当 $n > 64$ 选择串行计算方案, 当 $n \leq 64$ 选择并行计算方案, 这样不仅可以减少资源消耗, 还可以提高计算的性能. 同时, 串行计算方案和并行计算方案还有一点差异, 串行计算方案中将 $\frac{n}{2}+1$ 位乘法器用 $\frac{n}{2}$ 位的乘法器和一个中间值计算值 $C_{2\mathrm{adj}}$ 来合成计算. 这将有利于提高代码复用, 并且进一步减少资源消耗.

为了采用 KOM 算法的并行计算实现低功耗 64 位乘法器, 本文通过实验探究 KOM 算法中最小乘

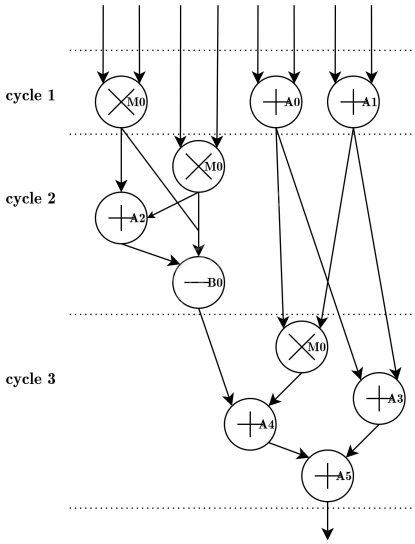


图 2 KOM 算法串行计算
Figure 2 Serial computation of KOM algorithm

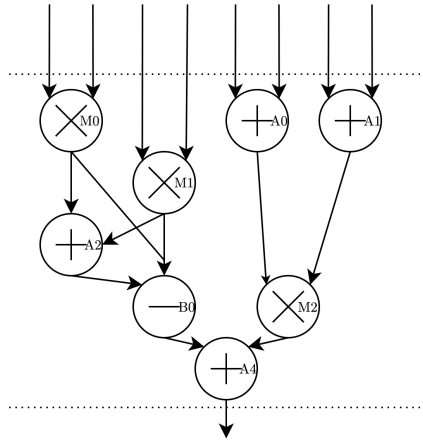


图 3 KOM 算法并行计算
Figure 3 Parallel computation of KOM algorithm

法器的位数与其能量消耗和资源消耗之间的关系, 来确定 KOM 算法递归最佳 n 值, 如图 4 所示. 图 4 中乘法器的能量消耗和资源消耗并不是一直随着 n 的降低而降低的, 原因是 KOM 算法在降低乘法的计算复杂度时, 也增加了一部分中间值的计算. 乘法器位数 n 特别短时增加的中间值计算量大于所降低的计算量, 整体的能量消耗和资源消耗反而会增加. 本文实验结果表明, 当递归最小 n 值为 16 时, 能够实现资源消耗与能量消耗的最低值.

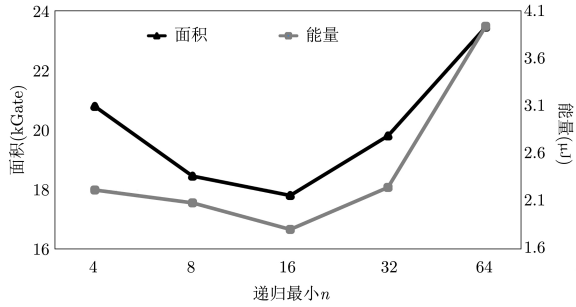


图 4 乘法器的资源和能量消耗情况
Figure 4 Resource and energy consumption of multiplier

3.2.2 快速模约减

传统的取模运算通常需要使用除法来计算余数, 而除法的计算复杂度较高, 导致能量消耗增加. 另一种方法是通过反复做减法来实现取模运算, 但是当商是一个很大的数时, 需要多次重复减法操作, 造成性能的下降. 在 SM2 的推荐参数中, P_{SM2} 是一种特殊的素数 — 扩展梅森素数, 可以通过快速模约减来实现快速取模. 如算法 2 所示, 快速模约减将除法变为加减法运算, 从而大大降低模约减的复杂度与能量消耗. 设 $c = \mathrm{KOM}(a, b, 256)$, 且 $0 \le a < P_{\mathrm{SM2}}$, $0 \le b < P_{\mathrm{SM2}}$, 则 $0 \le c \le P_{\mathrm{SM2}}^2$.

参考文献 [19] 由于 $\mathrm{sum} \in [0, 15p]$, 可以通过预计算提前给出 preP, 再根据 sum 的高 4 位选择对应的 preP$[k]$, 最后计算 2 次减法完成模约减. 算法 2 中 ele0 – ele7 无依赖关系是并行计算的, 而 tele0 – tele7 存在依赖关系是串行计算的.

算法 2 基于素数 P_{SM2} 的快速模约减

Input: $c=\{c_{15},c_{14},\cdots,c_0\}=c_{15}\times 2^{480}+c_{14}\times 2^{448}+\cdots+c_0$ 且 $0\le c\le P_{\mathrm{SM2}}^2$,
$\mathrm{preP}=\{0,P_{\mathrm{SM2}},\cdots,15P_{\mathrm{SM2}}\}$

Output: $c \bmod P_{\mathrm{SM2}}$

$m_{89}=c_8+c_9$; $m_{101}=c_{10}+c_{11}$; $m_{123}=c_{12}+c_{13}$; $m_{811}=m_{89}+m_{101}$;
$m_{145}=c_{14}+c_{15}$; $m_{135}=c_{13}+m_{145}$; $m_{125}=m_{123}+m_{145}$;
$m_{115}=c_{11}+m_{125}$; $m_{105}=m_{101}+m_{125}$; $m_{815}=m_{811}+m_{125}$;
$\mathrm{ele0}=c_0+m_{815}+m_{135}$; $\mathrm{tele0}=\mathrm{ele0}$;
$\mathrm{ele1}=c_1+m_{105}+m_{145}+c_9$; $\mathrm{tele1}=\mathrm{ele1}+\mathrm{ele0}[35:32]$;
$\mathrm{ele2}=c_2$; $\mathrm{tele2}=\mathrm{ele2}+\mathrm{ele1}[35:32]$;
$\mathrm{ele3}=c_3+m_{115}+c_8+c_{13}$; $\mathrm{tele3}=\mathrm{ele3}$;
$\mathrm{ele4}=c_4+m_{125}+c_9+c_{14}$; $\mathrm{tele4}=\mathrm{ele4}$;
$\mathrm{ele5}=c_5+m_{135}+c_{10}+c_{15}$; $\mathrm{tele5}=\mathrm{ele5}+\mathrm{elej}[99:96]$;
$\mathrm{ele6}=c_6+m_{145}+c_{11}$; $\mathrm{tele6}=\mathrm{ele6}+\mathrm{ele5}[35:32]$;
$\mathrm{ele7}=c_7+c_{15}+m_{815}+m_{125}$; $\mathrm{tele7}=\mathrm{ele7}+\mathrm{ele6}[35:32]$;
$\mathrm{ele}j=\mathrm{tele2}+(\mathrm{tele3}\ll 32)+(\mathrm{tele4}\ll 64)-(c_{14}+c_{13}+m_{89})$;
$\mathrm{sum}=\{\mathrm{tele7},\mathrm{tele6}[31:0],\mathrm{tele5}[31:0],\mathrm{ele}j[95:0],\mathrm{tele1}[31:0],\mathrm{tele0}[31:0]\}$;
$k=\mathrm{sum}[259:256]$;
$kP_{\mathrm{SM2}}=\mathrm{preP}[k]$;
$\{\mathrm{carry}_1,\mathrm{result}_1[259:0]\}=\mathrm{sum}-kP_{\mathrm{SM2}}$;
$\{\mathrm{carry}_2,\mathrm{result}_2[259:0]\}=\mathrm{result}_1-P_{\mathrm{SM2}}$;
return $\mathrm{carry}_1?\mathrm{result}_2[255:0]:\mathrm{result}_1[255:0]$

传统的快速模约减算法 [25] 在计算 sum 时往往需要计算多次 256 位加法, 但其加法输入参数的二进制表示中存在很多 0 比特, 导致 256 位加法器无法被有效利用, 同时也存在中间值反复计算的过程. 本文对于反复计算的中间值进行预计算, 并多次复用, 以减少重复计算的次数. 其次, 将 256 位加法拆分为多个短位加法操作, 从而更好地利用硬件资源, 以此来降低功耗和减少资源消耗. 算法 2 整体只使用 54 个 32 位加法器, 而传统的快速模约减算法则需要 120 个 32 位加法器. 整个快速模约减模块主要用到的资源就是加法器, 所以算法 2 的资源理论上是传统的快速模约减算法消耗资源的 45%. 本文通过减少加法器的数量, 达到减少加法运算的次数. 最终改进后的方案经过实验验证, 该模块整体的资源消耗方面仅为传统的快速模约减算法方案的 66.1%, 而整体的功耗仅为传统的快速模约减算法方案的 58.2%.

3.2.3 Barrett 算法

本文在硬件中采用 256 位 KOM 乘法器和快速模约减的方案实现模乘运算. 但是 SM2 协议层中模运算的模数是 N_{SM2}, 它并不是一个梅森素数, 不能直接调用快速模约减的硬件来进行计算. Barrett 算法 [19,26] 可以计算任意参数模乘的计算. 该算法利用乘法和模约减运算代替高成本的除法运算来实现取模运算, 从而高效地计算任何参数的模乘. 对于 $0\le a<N_{\mathrm{SM2}}$, $0\le b<N_{\mathrm{SM2}}$, 则 $a\times b<N_{\mathrm{SM2}}^2$. 为计算 $c=a\times b \bmod N_{\mathrm{SM2}}$, $0\le c<N_{\mathrm{SM2}}$, 令 $d=a\times b$, 如果存在 e 使 $d=e\times N_{\mathrm{SM2}}+c$, 则可求得 c.

Barrett 算法首先计算 e 的近似值 e', 令 $u=\lfloor\frac{2^{2n}}{N_{\mathrm{SM2}}}\rfloor$, $e'=\lfloor\frac{d}{N_{\mathrm{SM2}}}\rfloor$, 则

$$e'=\left\lfloor\frac{d}{N_{\mathrm{SM2}}}\right\rfloor\approx\left\lfloor d\frac{u}{2^{2n}}\right\rfloor\approx\left\lfloor\left\lfloor\frac{d}{2^n}\right\rfloor\frac{u}{2^n}\right\rfloor.$$

通过 $c'=d-e'\times N_{\mathrm{SM2}}$ 得到一个近似 c 值, 并且 c' 的取值范围为 $[0,2P_{\mathrm{SM2}})$, 最后对 c' 最多做一次减法就可以得到 c 的值, 具体内容如算法 3 所示.

为降低能量消耗和资源消耗, 本文采用软硬结合的方式实现 Barrett 算法, 其中 256 位的乘法由低功耗硬件乘法器来计算, 其余部分由软件来计算. 当 SM2 算法使用推荐参数时, SM2 协议层中的模运算模数都是 N_{SM2}, 因此 u 可以通过预计算得到且只需计算一次.

与纯硬件实现 Barrett 算法相比, 软硬结合实现的优势是减少用于存储中间变量的寄存器数量, 代价是会增加软硬件之间的数据传输. 由于 Barrett 算法只是为了计算协议层模乘运算, 使用的次数较少, 对功耗和性能影响比较小, 而纯硬件实现方案中寄存器的消耗一直存在, 所以软硬结合实现的 Barrett 算法

在 RFID 标签芯片中更具优势.

算法 3 Barrett 模乘算法

Input: a, b, $n=256$, N_{SM2}, $u=\lfloor\frac{2^{2n}}{N_{\mathrm{SM2}}}\rfloor$
Output: $c=a\times b \bmod N_{\mathrm{SM2}}$
$d=\mathrm{KOM}(a,b,n)$; //硬件
$e_1=\mathrm{KOM}(u[n-1,0],d[511:256],n)$; //硬件
$e_2=\{d[511:256],256'd0\}$;
$e'=(e_1+e_2)\gg n$;
$d'=\mathrm{KOM}(N_{\mathrm{SM2}},e',n)$; //硬件
$c=d-d'$;
if $c>N_{\mathrm{SM2}}$ **then**
  $c=c-N_{\mathrm{SM2}}$;
end
return c

3.3 模逆

通常在素数域上求逆的算法有扩展欧几里得、费马小定理. 费马小定理需要求次幂, 这就意味着需要计算大量的乘法, 造成大量的能量消耗. 而二进制扩展欧几里得算法采用辗转相除法, 将除法全部改成加减运算, 并以二进制移位完成除 2 运算, 有利于减少能量消耗, 具体如算法 4 所示.

3.4 模加减

为了降低资源消耗, 本文将模加减操作合并到一个模块中. 该模块根据不同的模加减模式选择不同的初始值进行计算, 其中加法的结果可能超过 P, 因此结果需要再减去 P. 而减法的结果可能为负数, 因此最后需要加上 P, 并根据两次的溢出标志位判断最终的输出结果, 如算法 5 所示.

算法 4 二进制扩展欧几里得算法

Input: a, b, 且 $a,b\in[0,P-1]$
Output: $c=b\times a^{-1} \bmod P$
$u=P$; $v=a$; $r=0$; $s=b$;
while $u\neq 1$ and $v\neq 1$ **do**
  if $u[0]==0$ **then**
    $r=r/2 \bmod P$; $u=u/2$;
  end
  else if $v[0]==0$ **then**
    $s=s/2 \bmod P$; $v=v/2$;
  end
  else if $u>v$ **then**
    $r=r-s$; $u=u-v$;
  end
  else
    $s=s-r$; $v=v-u$;
  end
end
if $u==1$ **then**
  return r
end
else
  return s
end

算法 5 模加减

Input: a, b, P, SEL, 且 $a,b\in[0,P-1]$, SEL$\in\{+,-\}$
Output: $c=a\pm b \bmod P$
$u=P$; $v=a$; $r=0$; $s=b$;
if SEL $==+$ **then**
  $b_1=b$; $m=-P$;
end
else
  $b_1=-b$; $m=P$;
end
$\{c_0,s_0\}=a+b_1$;
$\{c_1,s_1\}=\{c_0,s_0\}+m$;
if ($c_1==1$ and SEL $==+$) or ($c_0==1$ and SEL $==-$) **then**
  return s_0
end
else
  return s_1
end

3.5 固定窗口 NAF 标量乘算法

多倍点运算层是 SM2 密码算法中的一个重要层次, 它定义了如何对多个点进行标量乘运算. 关于标量乘的计算方法, 常见的包括蒙哥马利标量乘算法、NAF 标量乘算法、wNAF 标量乘算法、固定窗口标

量乘算法等. 相比于其他算法, 固定窗口 NAF 标量乘算法在实现抗 SPA 攻击的同时还能减少点加的计算量. 该算法的核心思想是将二进制的标量转换为:

$$|0\cdots 0x_m|0\cdots 0x_{m-1}|\cdots|0\cdots 0x_0|, \qquad x_i \in \{\pm 1, \pm 3, \cdots, \pm(2^w-1)\} \tag{1}$$

的形式, 其中 x_i 是奇数, 所以只需要存储固定窗口 w 内奇数标量点. 根据转换后的标量, 执行的点加与倍点序列图为 $|D\cdots DA|D\cdots DA|\cdots|D\cdots DA|$, 其中 D 代表倍点, A 代表点加. 无论密钥的值是什么, 通过 SPA 攻击检测的点加与倍点序列都是一样的, 因此可以实现抗 SPA 攻击.

转换成公式 (1) 的形式需要满足转换前的标量是一个奇数的条件. 为了获得计算偶数标量的能力, 往往都需要额外的处理, 首先将标量转变为奇数. 文献 [24] 提出的固定窗口 NAF 标量乘算法中, 如果遇到偶数标量 k, 会把其转换成 $k'=k+1$, 最后的标量乘结果为 $kP=k'P-P$. 为了让奇数标量与偶数标量的点加与倍点的序列图一致, 也需要有额外的点加操作, 需要将奇数标量转换为 $k'=k+2$, 最后的标量乘结果为 $kP=k'P-2P$. 这就意味着需要进行额外的点加, 且需要多存储 $2P$ 这个标量点, 从而增加了计算的复杂度和硬件资源的开销. 本文提出一种改进方案, 可以获得计算偶数标量的能力, 同时避免在最后进行冗余的点加操作和存储 $2P$ 标量点的值. 改进方案的关键是利用椭圆曲线中的参数 n 将标量 k 转换成奇数 d. 根据椭圆曲线的定义 [3], 有 $n\cdot G=O$, 且 n 是一个素数. 这意味着, 如果 k 是偶数, 则将其加上 n 后进行标量乘, 最终的标量乘结果将会保持不变, 且将标量变成了奇数, 这样就可以延续使用固定窗口 NAF 标量乘算法. 为了保证计算的平衡性, 当 k 是奇数时, 改进方案会给其加上 $2n$. 通过这种方式, 本文的改进方案可以实现对奇偶标量的高效处理, 从而进一步提高标量乘计算的效率, 并减少硬件资源的开销, 详细内容如算法 6 和算法 7 所示.

算法 6 固定窗口 NAF 标量表示

Input: w, k, $l=\lceil\frac{m+2}{w}\rceil-1$ //k 为初始标量, m 代表标量 k 的位数
Output: $d_w[lw:0]$ //d 为转换后标量

if $k\%2==1$ **then**
  $d=k+2\times n$;
end
else
  $d=k+n$;
end
$u[0]=d \bmod 2^w$;
$d=d-u[0]$;
$d=d/2^w$;
for i **from** 1 **to** l **do**
  $u[i]=d \bmod 2^w$;
  if $d\%2==0$ **then**
    $b=\text{sign}(u[i-1])$; $u[i]=u[i]+b$;
    $u[i-1]=u[i-1]-b2^w$;
  end
  $d_w[(i-1)w+w-1:(i-1)w+1]=0$;
  $d_w[(i-1)w]=u[i-1]$;
  $d=d-u[i]$;
  $d=d/2^w$;
end
$d_w[lw+w-1:lw+1]=0$;
$d_w[lw]=u[l]$;
return $d_w[lw:0]$.

算法 7 固定窗口 NAF 标量乘算法

Input: $d_w[lw:0]$, P, 窗口长度 w, $d_w[i]\in\{0,\pm1,\pm3,\cdots,\pm(2^w-1)\}$
Output: $Q=d_wP$

$P_1=P$;
$P_2=\text{PointDouble}(P_1,1)$;
for i **from** 3 **to** 2^w-1 **step** 2 **do**
  $P_i=\text{PointAdd}(P_{i-2},P_2)$;
end
$l=\lceil\frac{m+2}{w}\rceil-1$;
$i=lw$;
$Q=P_{d_w[i]}$;
for i **from** $(l-1)w$ **to** 0 **step** $-w$ **do**
  $Q=\text{PointDouble}(Q,w)$;
  $Q=\text{PointAdd}(Q,P_{d_w[i]})$;
end
return Q

注 l: 窗口操作 (w 次倍点 +1 次点加的操作) 的执行次数.

蒙哥马利标量乘法也可实现抗 SPA 攻击, 但其计算量为 $mC_A+mC_D+C_C$, 其中 m 代表密钥长度, C_A 代表一次点加的计算量, C_D 代表一次倍点的计算量, C_C 代表坐标转换所需要的计算量. 表 1 中记录

固定窗口 NAF 标量乘算法在优化前后的计算量对比, 当密钥长度 m 为 256 位、窗口 $w=4$ 时, 其点加的计算次数只有蒙哥马利标量乘算法的 28.5%, 相比优化前的固定窗口 NAF 标量乘算法存储预计算点的资源优化了 11.1%, 且计算量减少了一次点加操作.

表 1 固定窗口 NAF 标量乘算法优化前后计算量对比

Table 1 Comparison of computational complexity before and after optimization of width-w NAF scalar multiplication algorithm

算法	公式	当 $m=256, w=4$ 时
优化前	$(\lceil \frac{m}{w} \rceil + 2^{w-1})C_A + (w\lceil \frac{m}{w} \rceil + 1)C_D + C_C$	$72C_A + 257C_D + C_C$
优化后	$(\lceil \frac{m+2}{w} \rceil + 2^{w-1} - 2)C_A + (w(\lceil \frac{m+2}{w} \rceil - 1) + 1)C_D + C_C$	$71C_A + 257C_D + C_C$

3.6 点加与倍点的实现

F_p 上的椭圆曲线常用的坐标系有仿射坐标系、标准射影坐标系、Jacobian 加重射影坐标系. 在仿射坐标系下, 每次计算点加和倍点时都需要求模逆. 但是模逆计算周期长, 难于优化. 如果采用 Jacobian 加重射影坐标或者标准射影坐标系, 可以在标量乘迭代过程中消除模逆运算, 进而提高运算效率. 在整个标量乘运算过程中, 模逆运算仅在最后的坐标转换过程中使用. 根据文献 [21] 提供的不同坐标系下点加与倍点的计算复杂度, 对比 Jacobian 加重射影坐标系下和标准射影坐标系下标量乘的计算量, 发现 Jacobian 加重射影坐标下标量乘的计算量相对较少. 因此本文选择 Jacobian 加重射影坐标系完成计算, 并在最后进行坐标转换.

在 Jacobian 加重射影坐标系下, 有:

若 $P_0 = P_1$,

$$\begin{cases} x_2 = (3x_0^2 + az_0^4)^2 - 8x_0y_0^2 \\ y_2 = (3x_0^2 + az_0^4)(4x_0y_0^2 - x_2) - 8y_0^4 \\ z_2 = 2y_0z_0 \end{cases};$$

若 $P_0 \neq P_1$,

$$\begin{cases} \lambda_1 = x_0z_1^2 - x_1z_0^2 \\ \lambda_2 = x_0z_1^2 + x_1z_0^2 \\ \lambda_3 = y_0z_1^3 - y_1z_0^3 \\ \lambda_4 = y_0z_1^3 + y_1z_0^3 \\ \lambda_5 = \lambda_3^2 - \lambda_2\lambda_1^2 \\ \lambda_6 = \lambda_2\lambda_1^2 - 2\lambda_5 \end{cases}, \quad \begin{cases} x_2 = \lambda_5 \\ y_2 = (\lambda_6\lambda_3 - \lambda_4\lambda_1^3)/2 \\ z_2 = z_0z_1\lambda_1 \end{cases}. \tag{2}$$

在 Jacobian 加重射影坐标下, 有两种条件的点加方案, 分别是 $z_1=1$ 的条件点加和 z_1 不做要求的无条件点加. 通常来说, 当条件越多时, 其计算量就越小. 根据上述公式 (2) 可以得到无条件点加需要 16 次模乘, 而 $z_1=1$ 的条件点加只需要 12 次模乘. 为了减少资源消耗, 本文实现应用范围较广但只针对推荐参数下 SM2 的签名与验签. 椭圆曲线密码系统的安全性不依赖对这些参数的保密, 因此椭圆曲线系统的参数是可以公开的. 当计算固定点标量乘, 即 SM2 推荐参数 G 的标量乘时, 本文通过储存窗口内 $z_1=1$ 的标量点并且使用条件点加来优化计算过程, 从而减少模乘的计算量. 当需要计算任意点的标量乘时, 需要预计算获得窗口内的标量点的值. 然而, 由于本文采用的是 Jacobian 加重射影坐标系, 预计算得到的结果往往 $z_1 \neq 1$, 所以只能使用无条件标量乘.

4 SM2 软硬结合设计

4.1 SM2 整体架构

本节介绍一种采用软硬结合的 SM2 算法架构的设计方案, 具体实现如图 5 所示. 该架构整体最终由 ASIC 芯片实现, 内部包含低功耗处理器 Corte-M0 和 SM2 协处理器, 两者通过 AHB 总线进行数据交互. 其中, Cortex-M0 主要负责签名、验签以及一些辅助函数的计算, SM2 协处理器则主要负责加速计算, 包括计算标量乘、点加、坐标转换等功能.

4.2 主控接口

如图 5 所示, 主控接口的设计是实现软硬结合功能的关键. 主控接口由指令区域、状态区域、数据区域和主控状态机构成, 其构成元素之间的关系如图 6 所示. 在 Cortex-M0 为 SM2 协处理器的指令区域、状态区域和数据区域分配外设地址之后, Cortex-M0 就可直接访问主控接口中的状态区域信息, 并且可以向指令区域写入数据, 还可以通过数据区域进行数据交互. 在执行 SM2 算法的签名或验签过程中, 当需要调用 SM2 协处理器时, Cortex-M0 会结合状态区域的信息, 将相应的控制指令输入到指令区域. 在这个过程中, Cortex-M0 会与数据区域进行信息交互, 以完成相应的操作.

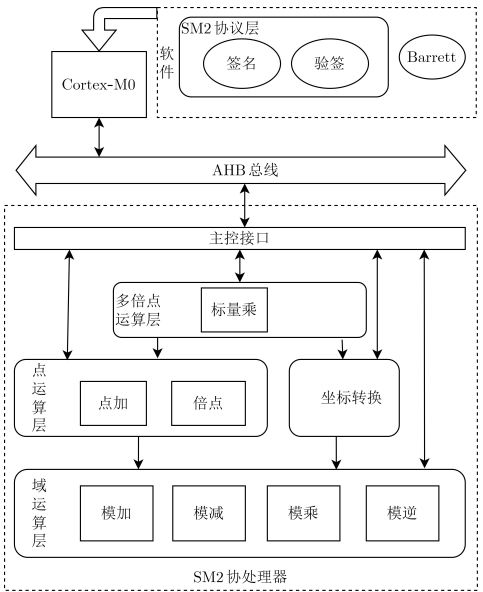


图 5 SM2 整体架构图
Figure 5 SM2 overall architecture diagram

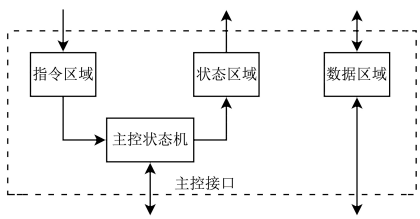


图 6 主控接口
Figure 6 Control interface

主控状态机由 8 种状态构成, 分为 IDLE (闲置态)、INPUT_G (任意点标量乘参数输入态)、INPUT_K (固定点或任意点标量乘参数 k 输入态)、INPUT_U (单元功能模块参数输入态)、PCAL (预计算态)、CAL (标量乘计算态)、UCAL (单元功能模块计算态)、OUTPUT (输出态). 图 7 记录了 8 种状态之间的相互转换. 主控状态机的状态转换通常由指令和反馈电路所控制, 指令区域的指令集如表 2 所示, 反馈电路是指当底层模块计算完成后会给主控状态机计算已经完成的信号, 例如 pcal_done 等.

SM2 协处理器的主要功能包括任意点标量乘、固定点标量乘和单元功能模块的计算, 分别由 idle_tog、idle_tok 和 idle_tou 控制启动. 从图 7 中可以发现固定点与任意点标量乘的差距是任意点标量乘多了预计算的过程. 单元功能模块的计算包括调用硬件中点加、256 位乘法、模加减和模逆功能, 如表 2 和图 1 所示.

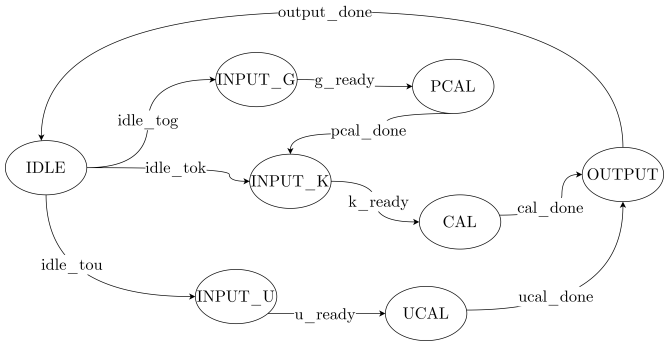


图 7 主控状态机
Figure 7 Control state machine

表 **2** 指令区域的指令集
Table 2 Instruction set of instruction area

指令	状态机影响	作用	指令	状态机影响	作用
0ff1	idle_tou＝true	将调用硬件的点加功能	0fff	idle_tok＝true	将调用硬件的固定点标量乘功能
0ff3	idle_tou＝true	将调用硬件的低功耗乘法功能	0f1e	g_ready＝true	任意点标量参数输入完成
0ff4	idle_tou＝true	将调用硬件的模逆功能	0f1f	k_ready＝true	任意点或固定点标量乘的 k 值输入完成
0ff5	idle_tou＝true	将调用硬件的模加功能	0f11	u_ready＝true	单元功能模块的输入完成
0ff6	idle_tou＝true	将调用硬件的模减功能	3100	output_done＝true	输出结果结束恢复初始状态
0ffe	idle_tog＝true	将调用硬件的任意点标量乘功能	0000		空指令, 无任何操作

5 实验结果

5.1 标量乘性能对比

本文设计的 SM2 协处理器是由 Verilog 语言实现的, 该架构由 Synopsys Design Compiler 基于 130 nm CMOS 标准单元库实现, 并且由 Synopsys PrimePower 来进行门级功耗分析.

将本文方案与其他硬件方案实现的标量乘算法进行性能、能量消耗和资源消耗方面的对比, 如表 3 所示. 文献 [25, 27, 28] 通过提高协处理器的时钟频率, 可以获得较好的性能. 但在 RFID 标签芯片中, 将协处理器的频率调整到很高的水平会导致功耗增加, 与 RFID 设备低功耗的需求冲突. 这就意味着低功耗和高性能之间存在着矛盾的关系. 本文通过功耗与计算时间的乘积—标量乘所消耗能量客观反映了功耗和算法性能之间的关系. 消耗能量值越低, 表明在低功耗条件下与性能之间取得的平衡越好. 尽管本文只针对 SM2 推荐参数进行标量乘计算, 但在能量消耗和性能方面具有明显的优势. 在相同的密钥长度下, 本文方案具有较好的性能, 其速度可达每秒 1150 次. 此外, 本文方案在计算一次标量乘时消耗的能量最少.

本文还探索了基于不同频率下, 计算任意点标量乘所消耗的计算时间、功耗以及能量之间的关系, 如表 4 所示. 降低时钟频率可以降低系统的功耗, 但也会增加计算所花费的时间. 因此, 针对不同的 RFID 环境条件需要权衡功耗和性能, 来调节时钟频率选择适合的设置.

5.2 签名验签性能对比

为了在流片前验证芯片功能的完整性, 需要在 FPGA 板上测试运行 SM2 的签名与验签. 其中硬件平台是 Xilinx FPGA 板, 芯片型号为 Kintex-7 xc7k325t, 并选择 Vivado 来进行综合 (synthesis), 布局 (placement) 和布线 (routing). 软件平台是 Keil, 其作用是将软件编译成可执行的二进制文件, 通过调试器 JlinkV9 下载到 Cortex-M0 内核中.

表 5 列出了在 Cortex-M0 和 SM2 协处理器的时钟频率都设置为 50 MHz 条件下, 计算辅助函数、

表 **3** 不同 ECC 协处理器在计算标量乘时, 在性能、资源消耗和能量消耗方面的比较
Table 3 Comparison of performance, resource consumptionand energy consumption of different ECC coprocessors in scalar multiplication calculations

文献	工艺 (nm)	有限域	位宽 (bit)	频率 (MHz)	周期数 (k)	硬件资源	标准化面积 (mm^2)	速度 (ms)	消耗能量 (μJ)	抗 SPA 攻击
[25]	130	F_p	256	214	45.5	208 kGate	1.73	0.211	8.5	是
[27]	130(1.0v)	F_p	160	141	-	1.35 mm^2	1.35	0.385	31.0	否
[28]	65(1.2v)	F_p	256	500	-	1.1 mm^2	1.1	0.32	38.4	是
[29]	130	F_{2^m}	163	13.56	170	21.8 kGate	0.18	12.5	2.6	是
[30]	130	F_p	192	1	-	0.17 mm^2	0.17	525	20.6	是
本文 ^{f}	130(1.3v)	F_p	256	50	38.8	1.29 mm^2	1.29	0.775	4.5	是
本文 ^{r}	130(1.3v)	F_p	256	50	43.4	1.29 mm^2	1.29	0.869	4.9	是

注 ^{f}: 固定点, ^{r}: 任意点. 在 130 nm 工艺下, $1mm^2 \approx (\sum_{i=1}^{n} \frac{T_i}{S_i})/n \approx 120$ kGate [27,31], 其中 T_i 和 S_i 分别代表第 i 个项目中消耗的门的数量 (kGate) 和综合后的面积 (mm^2).

表 **4** 不同频率下任意点标量乘计算所需的时间、功耗以及能量
Table 4 Time, power consumption and energy required for scalar multiplication calculations of arbitrary points at different frequencies

时钟频率 (MHz)	计算时间 (ms)	功耗 (mw)	能量 (μJ)
10	4.344	1.41	6.112
20	2.172	2.46	5.348
30	1.451	3.51	5.088
40	1.086	4.57	4.966
50	0.869	5.63	4.889

SM2 签名和验签的性能. 由于 SM3 模块并不是本文关注的重点, 所以 SM2 签名与验签并未统计 SM3 模块所消耗的时间. 将本文与其他架构实现的 SM2 签名及验签的性能比较, 如表 6 所示, 与文献 [31, 32] 相比, 签名与验签的性能有明显的提升. 文献 [19] 由于多处采用并行计算并且采用大整数乘法器, 其性能卓越, 但也会造成大量的资源消耗, 而本文相对优势是通过软硬结合等方式来减少资源消耗, 来满足 RFID 标签芯片小巧、便于携带的需求.

表 **5** 通过软硬结合实现 SM2 的功能模块的性能
Table 5 Performance of functionality module of SM2 achieved through integration of software and hardware

实现方式	模块	周期数	时间 (ms)
软硬件	签名	48 862	0.977 24
	验签	87 075	1.741 50
	Barret (模乘)	3224	0.064 48
软硬件	标量乘 ^{f}	38 761	0.775 22
	标量乘 ^{r}	43 442	0.868 84
	点加 ^{f}	151	0.003 02
	点加 ^{r}	200	0.004 00
	倍点	102	0.002 04
	模加减	1	0.000 02
	模逆	600	0.012 00
	模乘	12	0.000 24
	KOM 乘法	11	0.000 22
	快速模约减	1	0.000 02

注 ^{f}: 固定点, ^{r}: 任意点.

表 6 不同架构下 SM2 签名与验签的性能对比

Table 6 Performance comparison of SM2 signature and verification under different architectures

模块	实现	时钟频率 (MHz)	器件	硬件资源		速度 (次 · s^{-1})		速度比	
				消耗	消耗比	签名	验签	签名	验签
[32]	软件	72	STM32F 103ZET6	-	-	4.9	2.6	208.84	220.85
[19]	硬件	27	Xcku-060	{30823LUT+288DSP}(签名) + {32543LUT+288DSP} (验签) $\langle a \rangle$	11.80	7165	3556	0.14	0.16
[31]	软硬件	50	ASIC 0.13 μm	0.6 $mm^2\langle p \rangle$	0.47	64.1	25.6	15.96	22.42
本文	软硬件	50	ASIC 0.13 μm	1.29 $mm^2\langle p \rangle$	1	1023.3	574.2	1	1
			xc7k325t	34507 LUT$\langle a \rangle$	1	1023.3	574.2	1	1

注 1 DSP ≈ 597 LUT [20]; $\langle a \rangle$: 完整实现 SM2 签名与验签所消耗的硬件资源, 考虑其他模块的资源消耗; $\langle p \rangle$: SM2 协处理器的资源消耗, 不考虑其他模块的资源消耗.

6 结束语

本文针对资源受限的 RFID 标签芯片设计了一种低功耗的软硬结合方案, 用于实现 SM2 算法的签名与验签. 首先通过结合 KOM 算法的串行计算和并行计算来设计低功耗乘法器, 并且通过优化实现模约减, 来实现降低功耗、减少资源消耗的目标. 同时, 本文还采用软硬结合的方式实现 Barrett 算法和 SM2 的签名与验签功能, 充分发挥软件和硬件的优势, 提高系统性能, 并节约资源的使用. 此外, 为满足抗 SPA 攻击和减少计算量, 本文用硬件实现固定窗口 NAF 标量乘算法. 实验结果显示, 在计算任意点标量乘时, 速度达到了 0.869 ms/次, 能量消耗仅为 4.9 μJ. 同时, 采用软硬结合的方式实现 SM2 签名与验签功能的速度分别达到 0.98 ms/次和 1.74 ms/次. 本文实现的 SM2 算法主要为了降低功耗应用到 RFID 标签芯片中, 只考虑抗 SPA 而没有考虑抵抗差分功耗攻击 (differential power analysis, DPA) [8]. 为提高 RFID 标签芯片中 SM2 算法的安全性, 未来研究可以集中于实现低功耗抗 DPA 的 SM2 算法, 以此来更好保护 RFID 标签芯片中隐私数据.

参考文献

[1] JIANG H S, ZHANG C, LIN J Y. Radio frequency identification technology and its application and development trend[J]. Application of Electronic Technique, 2005, 31(5): 1–4. [DOI: 10.3969/j.issn.0258-7998.2005.05.001]
蒋皓石, 张成, 林嘉宇. 无线射频识别技术及其应用和发展趋势 [J]. 电子技术应用, 2005, 31(5): 1–4. [DOI: 10.3969/j.issn.0258-7998.2005.05.001]

[2] HANKERSON D, MENEZES A, VANSTONE S. Guide to Elliptic Curve Cryptography[M]. Springer New York, NY, 2006. [DOI: 10.1007/b97644]

[3] State Cryptography Administration. Public key cryptographic algorithm SM2 based on elliptic curves-Part 1: General: GM/T 0003.1-2012[S]. 2012.
国家密码管理局. SM2 椭圆曲线公钥密码算法第 1 部分: 总则: GM/T 0003.1-2012[S]. 2012.

[4] State Cryptography Administration. Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves-Part 2: Digital Signature Algorithm: GM/T 0003.2-2012[S]. 2012.
国家密码管理局. SM2 椭圆曲线公钥密码算法第 2 部分: 数字签名算法: GM/T 0003.2-2012[S]. 2012.

[5] State Cryptography Administration. Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves-Part 3: Key Exchange Protocol: GM/T 0003.3-2012[S]. 2012.
国家密码管理局. SM2 椭圆曲线公钥密码算法第 3 部分: 密钥交换协议: GM/T 0003.3-2012[S]. 2012.

[6] State Cryptography Administration. Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves-Part 4: Public Key Encryption Algorithm: GM/T 0003.4-2012[S]. 2012.
国家密码管理局. SM2 椭圆曲线公钥密码算法第 4 部分: 公钥加密算法: GM/T 0003.4-2012[S]. 2012.

[7] State Cryptography Administration. Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves-Part 5: Parameter Definition: GM/T 0003.5-2012[S]. 2012.
国家密码管理局. SM2 椭圆曲线公钥密码算法第 5 部分: 参数定义: GM/T 0003.5-2012 [S]. 2012.

[8] KOCHER P, JAFFE J, JUN B. Differential power analysis[C]. In: Advances in Cryptology—CRYPTO '99.

Springer Berlin Heidelberg, 1999: 388–397. [DOI: 10.1007/3-540-48405-1_25]

[9] WU C, YANG F, TAN X, et al. An ECC crypto engine based on binary Edwards elliptic curve for low-cost RFID tag chip[C]. In: Proceedings of 2015 IEEE 11th International Conference on ASIC (ASICON). IEEE, 2015: 1–4. [DOI: 10.1109/ASICON.2015.7517207]

[10] LUO P, WANG X N, FENG J, et al. Low-power hardware implementation of ECC processor suitable for low-cost RFID tags[C]. In: Proceedings of 2008 9th International Conference on Solid-State and Integrated-Circuit Technology (ICSICT). IEEE, 2008: 1681–1684. [DOI: 10.1109/ICSICT.2008.4734876]

[11] TAN X, DONG M X, WU C, et al. An energy-efficient ECC processor of UHF RFID tag for banknote anti-counterfeiting[J]. IEEE Access, 2016, 5: 3044–3054. [DOI: 10.1109/ACCESS.2016.2615003]

[12] LIU Z L, LIU D S, SUN X C, et al. Implementation of a resource-constrained ECC processor with power analysis countermeasure[C]. In: Proceedings of 2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS). IEEE, 2016: 206–209. [DOI: 10.1109/APCCAS.2016.7803934]

[13] AHMADI H R, AFZALI-KUSHA A. Low-power flexible GF(p) elliptic-curve cryptography processor[C]. In: Proceedings of 2008 3rd International Design and Test Workshop. IEEE, 2008: 182–186. [DOI: 10.1109/IDT.2008.4802493]

[14] MONTGOMERY P L. Modular multiplication without trial division[J]. Mathematics of Computation, 1985, 44(170): 519–521. [DOI: 10.2307/2007970]

[15] SOLINAS J A. Generalized Mersenne numbers[R]. Faculty of Mathematics, University of Waterloo, 1999.

[16] FENG X, LI S G. A high performance FPGA implementation of 256-bit elliptic curve cryptography processor over GF(p)[J]. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2015, 98.A(3): 863–869. [DOI: 10.1587/transfun.E98.A.863]

[17] DING J N, LI S G, GU Z. High-speed ECC processor over NIST prime fields applied with Toom-Cook multiplication[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2018, 66(3): 1003–1016. [DOI: 10.1109/TCSI.2018.2878598]

[18] CHUNG S C, LEE J W, CHANG H C, et al. A high-performance elliptic curve cryptographic processor over GF(p) with SPA resistance[C]. In: Proceedings of 2012 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2012: 1456–1459. [DOI: 10.1109/ISCAS.2012.6271521]

[19] LI B, ZHOU Q L, CHEN X J, et al. Optimization of reconfigurable SM2 algorithm over prime filed[J]. Journal on Communications, 2022, 43(3): 30–41. [DOI: 10.11959/j.issn.1000-436x.2022043]
李斌, 周清雷, 陈晓杰, 等. 可重构的素域 SM2 算法优化方法 [J]. 通信学报, 2022, 43(3): 30–41. [DOI: 10.11959/j.issn.1000-436x.2022043]

[20] DING J N, LI S G. A reconfigurable high-speed ECC processor over NIST primes[C]. In: Proceedings of 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, 2017: 1064–1069. [DOI: 10.1109/Trustcom/BigDataSE/ICESS.2017.353]

[21] ZHANG D, BAI G Q. Ultra high-performance ASIC implementation of SM2 with power-analysis resistance[C]. In: Proceedings of 2015 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC). IEEE, 2015: 523–526. [DOI: 10.1109/EDSSC.2015.7285166]

[22] MONTGOMERY P L. Speeding the Pollard and elliptic curve methods of factorization[J]. Mathematics of Computation, 1987, 48(177): 243-264. [DOI: 10.2307/2007888]

[23] CORON J S. Resistance against differential power analysis for elliptic curve cryptosystems[C]. In: Cryptographic Hardware and Embedded Systems—CHES '99. Springer Berlin Heidelberg, 1999: 292–302. [DOI: 10.1007/3-540-48059-5_25]

[24] OKEYA K, TAKAGI T. The width-w NAF method provides small memory and fast elliptic scalar multiplications secure against side channel attacks[C]. In: Topics in Cryptology—CT-RSA 2003. Springer Berlin Heidelberg, 2003: 328–343. [DOI: 10.1007/3-540-36563-X_23]

[25] ZHANG D, BAI G Q. Ultra high-performance ASIC implementation of SM2 with SPA resistance[C]. In: Information and Communications Security—ICICS 2015. Springer Cham, 2015: 212–219. [DOI: 10.1007/978-3-319-29814-6_17]

[26] ZHAO Z W, BAI G Q. Exploring the speed limit of SM2[C]. In: Proceedings of 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems. IEEE, 2014: 456–460. [DOI: 10.1109/CCIS.2014.7175778]

[27] LAI J Y, HUANG C T. Energy-adaptive dual-field processor for high-performance elliptic curve cryptographic applications[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2010, 19(8): 1512–1517. [DOI: 10.1109/TVLSI.2010.2048134]

[28] LI W, ZENG X Y, FENG X, et al. A high-throughput processor for dual-field elliptic curve cryptography with power analysis resistance[C]. In: Proceedings of 2015 IEEE 12th International Conference on Ubiquitous Intelligence and Computing and 2015 IEEE 12th International Conference on Autonomic and Trusted Computing and

2015 IEEE 15th International Conference on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom). IEEE, 2015: 570–577. [DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.114]

[29] LIU D S, LIU Z L, YONG Z Q, et al. Design and implementation of an ECC-based digital baseband controller for RFID tag chip[J]. IEEE Transactions on Industrial Electronics, 2015, 62(7): 4365–4373. [DOI: 10.1109/TIE.2014.2387333]

[30] AHMADI H R, AFZALI-KUSHA A. Low-power low-energy prime-field ECC processor based on Montgomery modular inverse algorithm[C]. In: Proceedings of 2009 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools. IEEE, 2009: 817–822. [DOI: 10.1109/DSD.2009.140]

[31] XIE T Y, HUANG K, XIU S W, et al. VLSI implementation of elliptic curve cryptographic accelerator over GF(p)[J]. Computer Engineering and Applications, 2016, 52(1): 89–94. [DOI: 10.3778/j.issn.1002-8331.1410-0277]
谢天艺, 黄凯, 修思文, 等. 素数域椭圆曲线密码加速器的 VLSI 实现 [J]. 计算机工程与应用, 2016, 52(1): 89–94. [DOI: 10.3778/j.issn.1002-8331.1410-0277]

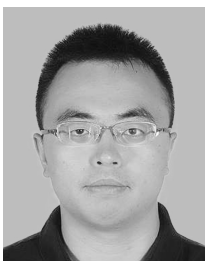
[32] LIU G Q, LI H, ZHU H, et al. Public key cryptographic algorithm SM2 optimized implementation on low power embedded platform[J]. Chinese Journal of Network and Information Security, 2022, 8(6): 29–38. [DOI: 10.11959/j.issn.2096-109x.2022080]
刘赣秦, 李晖, 朱辉, 等. 低功耗嵌入式平台的 SM2 国密算法优化实现 [J]. 网络与信息安全学报, 2022, 8(6): 29–38. [DOI: 10.11959/j.issn.2096-109x.2022080]

作者信息

孔团结 (1998–), 河南商丘人, 硕士研究生. 主要研究领域为密码算法的安全与优化实现.
kongtuanjie21@mails.ucas.ac.cn

郭润(1987–), 北京人, 硕士, 工程师. 主要研究领域为密码设备研发.
guorun@ucas.ac.cn

张子昂(1999–), 河南驻马店人, 博士研究生. 主要研究领域为密码工程与高安全等级硬件安全模块技术研究.
ziangdotz@163.com



郑昉昱 (1988–), 福建三明人, 副教授. 主要研究领域为应用密码学、高性能计算、计算机算术.
zhengfy1028@hotmail.com

荆继武 (1964–), 湖南洪江人, 教授. 主要研究领域为身份管理与网络信任技术、移动终端安全、系统安全理论与技术.
jwjing@ucas.ac.cn