

低功耗嵌入式平台的 SM2 国密算法优化实现

刘赣秦, 李晖, 朱辉, 黄煜坤, 刘兴东

(西安电子科技大学网络与信息安全学院, 陕西 西安 710071)

摘要: 随着无线通信技术和智能终端的普及, 越来越多的密码算法被应用到物联网设备中以保障通信安全和数据安全, 其中, 由国家密码管理局提出的 SM2 椭圆曲线公钥密码算法作为我国自主研发的椭圆曲线公钥密码算法具有安全性高、密钥短的优点, 已在通信系统中广泛部署, 应用于身份认证、密钥协商等关键环节。然而, 由于算法涉及有限域上的大整数运算, 计算开销较大, 在低功耗嵌入式平台下的执行严重影响用户体验。因此, 面向 ARM-m 系列处理器提出了一种低功耗嵌入式平台的 SM2 算法的高效实现方案。具体来说, 通过 Thumb-2 指令集提供的支持处理进位和节省寻址周期, 对大整数的模加、模减等基础运算进行优化, 并结合平台可用寄存器的数量构建高效的基础运算模块; 基于 ARM-m 系列处理器乘累加指令周期短的特点, 优化蒙哥马利乘法实现, 并结合 CIOS 算法设计高效的模乘方案, 方案不再局限于梅森素数, 极大地提高了模乘计算的速度和灵活性; 在理论分析和实验测试的基础上, 给出了嵌入式平台上多倍点标量乘法 wNAF 滑动窗法的窗长选取方法。实验测试结果表明, 可有效提升资源受限的低功耗嵌入式平台中 SM2 算法的计算效率, 不做预计算的情况下在 Cortex-M3 处理器上测试签名速度可达 0.204 秒/次, 验签速度 0.388 秒/次, 加密速度 0.415 秒/次, 解密速度 0.197 秒/次。

关键词: 信息安全; 椭圆曲线密码体制; SM2; 嵌入式平台; 优化

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2022080

Public key cryptographic algorithm SM2 optimized implementation on low power embedded platform

LIU Ganqin, LI Hui, ZHU Hui, HUANG Yukun, LIU Xingdong

School of Cyber Engineering, Xidian University, Xi'an 710071, China

Abstract: With the development of wireless communication technology and the popularization of intelligent terminals, more and more cryptographic algorithms are applied to IoT devices to ensure the security of

收稿日期: 2022-05-16; 修回日期: 2022-07-05

通信作者: 朱辉, zhuhui@xidian.edu.cn

基金项目: 国家自然科学基金 (61972304, 61932015), 陕西省重点产业创新链项目 (2020ZDLGY08-04)

Foundation Items: The National Natural Science Foundation of China (61972304, 61932015), Natural Science Foundation of Shaanxi Province (2020ZDLGY08-04)

引用格式: 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.

Citation Format: 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.

communication and data. Among them, the SM2 elliptic curve public key cryptography proposed by the State Cryptography Administration is an elliptic curve public key cryptography algorithm developed domestically, which has the advantages of high security and short key. SM2 has been widely deployed in various communication systems and is used in essential parts such as identity authentication and key negotiation. However, since SM2 involves large integer operations on finite fields, the computational cost is high, and its execution on a low-power embedded platform seriously affects the user experience. Therefore, an efficient implementation scheme of SM2 algorithm for low-power embedded platform was proposed for ARM-m series processors. Specifically, Thumb-2 instruction set was adopted to handle carry and save addressing cycles, basic operations such as modulo addition and subtraction of large integers were optimized, and the number of available registers on the platform was combined to build efficient basic operations. Besides, based on the short multiplication and accumulation instruction cycle of ARM-m series processors, the implementation of Montgomery multiplication was optimized, and an efficient modular multiplication scheme was designed in combination with the CIOS algorithm. The scheme was no longer limited to Mersenne primes, and greatly improved the speed and flexibility of modular multiplication. Based on the theoretical analysis and experimental test, the window length selection method of the multiple point-scalar multiplication wNAF sliding window method on the embedded platform was given. The experimental test results show that the proposed scheme can effectively improve the computational efficiency of the SM2 algorithm on the resource-constrained low-power embedded platform. Without pre-calculation, the test signature speed can reach 0.204s/time, the signature verification speed is 0.388s/time, the encryption speed is 0.415s/time, and the decryption speed is 0.197s/time.

Keywords: information security, elliptic curve cryptosystem, SM2, embedded platform, optimization

0 引言

物联网技术出现后,因其具有使用成本低、部署难度小的特点,被广泛应用于各种领域,但在技术迅速得到发展和应用的同时,其安全问题逐渐成为关注重点。在低功耗嵌入式设备中,ARM-m 系列处理器在包括微控制器、智能汽车系统、工业控制系统、智能家居以及无线网络和传感器等领域应用广泛,其中 Cortex-M3 处理器部署的 IoT 设备数量可达数十亿。为保障信息安全,需利用加密技术对信息进行保护,而通过公钥密码系统(PKC, public key cryptosystem)对数据信息进行加密或签名保护是保障网络信息安全的主要方法之一。椭圆曲线公钥密码(ECC, elliptic curve cryptography)算法的安全性基于椭圆曲线离散对数问题难以求解,在相同安全程度要求下密钥规模小于其他公钥密码系统,具有密钥短、运算快、传输效率高的特点,更适合各类资源受限的 IoT 设备。国家密码管理局于 2010 年 12 月 17 日提出了 SM2 椭圆曲线公钥密码算法^[1],不仅满足了各种商用密码的应用需要,也对我国

信息安全体系建设有着重大意义。

然而,椭圆曲线底层运算较为复杂,执行签名、加密计算时将对资源受限的 IoT 设备产生大量运算压力,设计一种高效的椭圆曲线计算方案可以有效地降低服务商的设备部署成本,提升用户体验。近年来,研究人员对 SM2 算法针对不同应用场景需要提出了许多解决方案,大多侧重于应用方面^[2-4],对于优化实现关注较少。针对以上问题,本文基于 ARM-m 系列处理器特点设计并实现了一种 SM2 国密算法的高效计算方案。

SM2 算法主要可分为 4 个层次,域运算层、点运算层、多倍点标量乘运算层、SM2 协议层,如图 1 所示,其优化手段与传统 ECC 算法基本一致。域运算的优化实现研究主要针对计算量较大的模乘与模逆。数学家蒙哥马利(Montgomery)于 1985 年提出了蒙哥马利乘法^[5],极大地提高了大整数计算模乘的速度,Koc 等^[6]在此基础上对蒙哥马利约减算法的步骤进行整合,提出了 5 种优化的蒙哥马利约减算法,进一步提高了效率;文献[7]提出了梅森素数法,对于特定素数 p 效率有所提高。而近年来对于模乘的优化研究偏重于

硬件层面, Aashish 等^[8]设计了 Montgomery 乘法器, 有效地提高了模乘吞吐量, 文献^[9]设计了基于硬件的交叉模乘; 面向嵌入式平台, 王腾飞等^[10]提出一种基于协处理器指令的优化方案, 同样加快了计算速度; 但基于硬件的方法存在灵活性缺陷, 将现有的各类嵌入式设备进行硬件升级成本较大。两乘数相等计算大整数平方时, 在一些算力较强的平台上表现较好的一种思路是利用 Karatsuba 算法和 ToomCook 算法^[11]对平方计算进行加速; 针对支持 64 位运算的平台, 李斌等^[12]提出了结合硬件 KOA 乘法和梅森素数法的模乘方案。对于大整数模逆的优化计算, Hossain 等^[13]提出了基于扩展 Euclid 算法的模逆算法, 面向 GPU 等平台 Zhou 等^[14]设计了利用费马小定理计算模逆的方案, 文献^[15]在此基础上进一步缩短了所需加法链。对于 ECC 的上层运算, 文献^[16]针对 ECC 的点运算进行优化, 分析了各类坐标系下计算的优劣, 取得了约 30% 的计算速度提升。混合坐标法^[17]由 Cohen 提出, 讨论了求逆与模乘时间比不同时各类混合坐标系的优劣性。文献^[18]提出了 ECC 标量乘法对使用预处理及窗口技术提高点乘计算效率的思想。由 Longa 提出的 m-DBL 算法^[19]基于模乘与模平方效率不同, 将重复倍点运算中的模乘转化为模平方来加速运算; 文献^[20]介绍了用预计算的查表法对固定点标量乘法进行加速; 文献^[21]利用可并行的蒙哥马利点乘加速了多倍点标量乘法运算。但此类方法大多针对强算力平台, 对于各种资源受限的嵌入式平台, 由于对平台的一些特点缺乏针对性, 如存储受限、并行效率差, 实测时计算缓慢, 效果欠佳。



图1 SM2 算法层次

Figure 1 Structure of algorithm SM2

本文针对 SM2 国密算法, 充分利用低功耗嵌入式平台特点, 对椭圆曲线密码体制的底层运算进行优化实现研究, 对算法的一些关键步骤进行改进, 主要贡献包括以下几个方面。

1) 结合 ARM-m 系列 MCU 访存开销大的特点, 构建了基于 Thumb-2 指令集的大整数模加、模减基础运算模块, 结合平台可用寄存器的数量进行高效分配, 相比 LibTomMath 开源库实现的模加方案性能提升 2.42 倍。

2) 考虑平台并行效率不高以及访存较慢的特点, 优化了蒙哥马利乘法, 并结合 CIOS 算法设计了高效的模乘方案, 效率超过了在大部分平台上表现较好的梅森素数法和其他各种公开算法, 同时避免了梅森素数法对于特定素数 p 的限制问题, 提高了算法灵活性, 签名效率相比梅森素数法提升约 6.13%。

3) 为存储受限环境高效计算 ECC 多倍点标量乘法提供了解决方案, 利用 wNAF 算法计算多倍点标量乘法, 分析了使得预计算和乘法阶段时间总和最短的窗长选取, 并通过实验测试验证了结论。

4) 在搭载 Cortex-M3 处理器的开发板上应用优化方案实现了使用 256 位素域椭圆曲线的 SM2 签名、验签、公钥生成以及加解密算法, 并进行了各项对比测试。实验测试证明提出的优化方案效果显著, 达到了 0.205 次/秒签名, 0.415 次/秒加密速度, 相比 LibTomMath 开源库实现的版本签名性能提升 1.62 倍, 加密性能提升 1.59 倍。

1 背景知识

1.1 SM2 数字签名算法

SM2 椭圆曲线密码算法包括数字签名生成算法及验证算法、加密及解密算法, 设待签名消息为 M , 求其签名 (r, s) , 公钥 P , 私钥 d , 定义有限素域 p 上的椭圆曲线方程为 $y^2 = x^3 + ax + b$, 基点 G , 运算过程如下。

- 1) 置 $\overline{M} = Z_A \parallel M$ 。
- 2) 计算 $e = H_v(\overline{M})$ 。
- 3) 用随机数发生器产生随机数 $k \in [1, n-1]$ 。
- 4) 计算多倍点标量乘法 $(x_1, y_1) = k \cdot G$ 。

5) 计算 $r = (e + x_1) \bmod n$, 若 $r = 0$ 或 $r + k = n$, 返回 3)。

6) 计算 $s = ((1 + d_A)^{-1} \cdot (k - r \cdot d_A)) \bmod n$, 若 $s = 0$, 返回 3)。

其中, $\parallel$ 表示串拼接, Z_A 为用户 A 身份信息及部分椭圆曲线参数, 与消息拼接得到 $\bar{M}$, H_v 为密码杂凑函数, 这里使用了国家密码管理局发布的 SM3 杂凑算法^[22], 将计算的字符串 e 转化为整数用于后续计算 r 。验签运算过程如下。

1) 置 $\bar{M}' = Z_A \parallel M'$ 。

2) 计算 $e' = H_v(\bar{M}')$ 。

3) 计算 $t = (r' + s') \bmod n$, 若 $t = 0$ 则验证不通过。

4) 计算多倍点标量乘法 $(x_1', y_1') = s' \cdot G + t \cdot P_A$ 。

5) 计算 $R = (e' + x_1') \bmod n$, 检验 $R = r'$ 是否成立, 成立则验证通过, 否则不通过。

1.2 SM2 加密算法

设需发送的消息为比特串 M , 串长为 $klen$, SM2 加密算法运算过程如下。

1) 用随机数发生器产生随机数 $k \in [1, n-1]$ 。

2) 计算多倍点标量乘法 $C_1 = k \cdot G = (x_1, y_1)$, 将 C_1 转换为比特串。

3) 计算多倍点标量乘法 $k \cdot P_B = (x_2, y_2)$, 将 x_2 、 y_2 转换为比特串。

4) 计算 $t = KDF(x_2 \parallel y_2, klen)$, 如 t 为全 0 比特串, 则返回 1)。

5) 计算 $C_2 = M \oplus t$ 。

6) 计算 $C_3 = H_v(x_2 \parallel M \parallel y_2)$ 。

7) 输出密文 $C = C_1 \parallel C_2 \parallel C_3$ 。

其中, KDF 为文献[1]中定义的密钥派生函数, 用于将密钥与消息位数对齐, 同样需要调用一次 SM3 杂凑算法, $\oplus$ 表示异或运算, 解密算法运算过程如下。

1) 从密文 C 中取高 512 位 C_1 , 并将其转换为椭圆曲线上点, 验证是否满足椭圆曲线方程, 如不满足, 则解密失败。

2) 计算多倍点标量乘法 $d_B C_1 = (x_2, y_2)$, 将 x_2 、 y_2 转换为比特串。

3) 计算 $t = KDF(x_2 \parallel y_2, klen)$, 如 t 为全 0 比

特串, 则解密失败。

4) 从密文 C 中再取 $klen$ 位 C_2 , 计算 $M' = C_2 \oplus t$ 。

5) 计算 $u = H_v(x_2 \parallel M' \parallel y_2)$, 取密文 C 中低 256 位 C_3 , 若 $u \neq C_3$, 则解密失败。

6) 输出明文 M' 。

1.3 椭圆曲线定义及坐标表示

定义在素域 F_p 上的椭圆曲线方程可表示为

$$\begin{aligned} y^2 &= x^3 + ax + b \\ a, b &\in F_p, (4a^3 + 27b^2) \bmod p \neq 0 \end{aligned} \quad (1)$$

椭圆曲线上的点可根据文献[1]中定义的椭圆曲线加法构成 abel 群, jacobian 加重射影坐标系可以有效加速点加及倍点运算。设 $z \neq 0$, 记 $X = x/z^2$, $Y = y/z^3$, 则仿射坐标下的椭圆曲线方程 $Y^2 = X^3 + aX + b$ 可化为 $y^2 = x^3 + axz^4 + bz^6$ 。点加及倍点运算的底层实现可参考文献[23]。以 SM2 签名为例, 在多倍点标量乘法计算开始和结束时进行坐标转换, 计算多倍点乘法前取 $z = 1$, 结束时求 z^2 模逆恢复仿射坐标计算签名, 在整个签名算法中只需进行一次求逆, 可显著提升算法效率。实测中, 模逆运算的时间开销约为模乘的 80 倍, 由文献[17]中的结论可知, 使用混合坐标系代替 jacobian 加重射影坐标系加速点运算需要更小的逆乘比。

1.4 蒙哥马利乘法

蒙哥马利乘法可避免计算带余除法从而对模乘进行高效计算, 其有效地结合了 32 位 cpu, 利用剩余系性质, 借助构造一个模 p 的剩余系, 将普通求模运算转化成移位和加法运算。记 $\text{reduce}(T) = TR' \bmod p$, 其中 $\gcd(p, R) = 1$, 对于 256 位椭圆曲线取 $R = 2^{256}$, T 为 512 位的 256 位乘法结果。 R' 为 R 在有限域 p 上的逆元, p' 为 $-p$ 在有限域 R 上的逆元, 蒙哥马利约减定义如下。

设 $0 \leq T < Rp$, 有

$$T = TRR' - Tpp' \quad (2)$$

$$T + Tpp' = TRR' \quad (3)$$

令 $m = Tp'$, 有

$$T + pm = TRR' \quad (4)$$

$$TR' = (T + pm) / R \quad (5)$$

取 $R = 2^{256}$ 时, 右移 256 位很好实现, 注意到 $\forall k$

$$\begin{aligned} & ((Tp' + kR)p + T) / R \bmod p = \\ & (Tp'p + T) / R \bmod p \end{aligned} \quad (6)$$

故计算 m 时,可直接取 T 的低 256 位与 p' 相乘得到 512 位 m 。

欲求 $z = xy \bmod p$, 可先计算 $x' = xR \bmod p$ 、 $y' = yR \bmod p$, 有

$$x' = \text{reduce}(xR^2) \quad (7)$$

$$y' = \text{reduce}(yR^2) \quad (8)$$

$$\begin{aligned} z' &= zR \bmod p = xyR \bmod p = \\ (x'y')R' \bmod p &= \text{reduce}(x'y') \end{aligned} \quad (9)$$

$$\begin{aligned} z &= z'R' \bmod p = \text{reduce}(z') = \\ \text{reduce}(\text{reduce}(x'y')) & \end{aligned} \quad (10)$$

可以通过 4 次蒙哥马利约减和 5 次 256 位乘法 (第 4 次乘数为 1) 来计算 $z = xy \bmod p$, 由于算法中 p 不变, 可对 p' 和 $R^2 \bmod p$ 进行预计算。

1.5 扩展欧几里得算法

对于大整数的模除运算, 直接进行模除往往比较复杂, 通常转化为模逆和模乘的形式, 模逆可以通过扩展欧几里得算法或费马小定理计算, 实测 Cortex-M3 平台上扩展欧几里得算法求模逆性能约为费马小定理 3 倍。由数论中欧几里得定律可知, 必定存在 x, y , 使得 $\gcd(a, b) = ax + by$ 。设 x_1 为 a 在素域 p 中模逆, 可表示为 $a \cdot x_1 \pmod{p} = 1$, 必定满足 $\gcd(a, p) = 1$ 。求 a 在素域 p 中模逆时, 可通过递归流程求解, 满足 $p=0$ 时, 得到的 x_1 即所求模逆结果。

1.6 Cortex-M3 及 Thumb-2 指令集

ARM 公司为了更好适应微控制器和受限的节能应用需求, 设计了 ARM-m 系列处理器, 其中 Cortex-M3 处理器应用广泛, 采用 ARMv7-M 架构, 包括所有的 16 位 Thumb 指令集和基本的 32 位 Thumb-2 指令集架构。Thumb-2 在 Thumb 指令集架构 (ISA, instruction set architecture) 上进行了大量的改进, 它与 Thumb 相比, 具有更高的代码密度并提供 16/32 位指令的更高性能。部分 Thumb-2 指令集如表 1 所示。

2 域运算优化

SM2 算法中基础四则运算涉及的操作数均为 256 位大整数, 运算在有限域中进行。根据 Gura

等^[24]的研究, MCU 平台开销最大的操作为访存, 因此算法优化时应尽可能减少访存次数, 将常用字段放在寄存器中。本节介绍面向低功耗嵌入式平台的域运算构建和优化方法, 对资源有限的寄存器进行合理利用。

表 1 部分 Thumb-2 指令集
Table 1 Part of the instruction set Thumb-2

指令	说明
LDMIA/STMIA Rn <reg list>	从指定存储器位置读取/写入多个字到 reg list 给出的寄存器列表中
CMP Ra,Rb	比较两寄存器内容并修改标志位
ADDS/SUBS Ra,Rb,Rc	Ra=Rb+/-Rc, 修改标志位
ADD/SUB Ra,Rb,Rc	Ra=Rb+/-Rc, 不修改标志位
ADCS/SBCS Ra,Rb,Rc	Ra=Rb+/-Rc+/-进位/借位, 修改标志位
LDR Ra,[Rb,#immed]	将 Rb 中地址偏移量为 immed 的字加载到 Ra
UMLAL RdLo,RdHi,Rn,Rm	{RdHi,RdLo} += Rn*Rm
EOR Rd,Rn	将 Rd 与 Rn 按位异或结果保存在 Rd

2.1 快速模加减

模加减所得结果为有限域中的元素, 在采用标准进制表示时, 被存储在一组机器字中, 以 Cortex-M3 平台的 32 位架构为例, 机器字可表示为 16、32 位整型或 64 位浮点型。由文献[25]可知, 浮点型对字的利用率低, 仅有 31.25%。而 16 位及 32 位带符号整型对字的利用率均不及 32 位无符号整型。因此, 本文存储 256 位整数时均使用 32 位无符号整型。有限域 p 中 256 位模加计算过程如算法 1 所示。

算法 1 256 位模加

输入 加数 x , 加数 y , 素域 p

输出 $z = (x + y) \bmod p$

1) $(\varepsilon, S[0]) \leftarrow x[0] + y[0]$

2) 对 j 从 1 到 7, 重复执行:
 $(\varepsilon, S[j]) \leftarrow x[j] + y[j] + \varepsilon$

3) 对 j 从 7 到 0, 重复执行

3.1 if $p[j] < S[j]$ goto 4

3.2 if $p[j] > S[j]$ return S

4) $(\varepsilon, z[0]) \leftarrow T[0] - p[0]$

5) 对 j 从 1 到 7, 重复执行:

$$(\varepsilon, z[j]) \leftarrow T[j] - p[j] - \varepsilon$$

6) return z

计算模减时需在模加之前进行一次 256 位减法, 设 $z = (x - y) \bmod p$, 令素域 p 中 y 的加法逆元 $t = p - y$, 等价于计算模加 $z = (x + t) \bmod p$ 。

Thumb-2 指令集对 32 位带进位的加减法的支持使得 SM2 算法中大整数的加减可以被优化实现, 如图 2、图 3 所示。利用修改标志位的加法指令 (ADDS、ADCS) 将计算产生的进位 ε 存储在 xPSR 寄存器的 C 标志位上, 并传递到下一位, 处理借位减法时利用 SUBS、SBCS 指令同理, 如图 3 所示。ARM32 位处理器架构中有 13 个通用寄存器, 令 $r0 \sim r2$ 存储 x 、 y 、 p 的地址。由于 256 位变量在内存中连续存储, 为了尽可能减少访存次数, 利用 LDMIA、STMIA 指令将 x 一次性加载到 $r4 \sim r11$ 中, 再从低到高每次取 32 位 y 、 p 存入 $r3$, 加减法结果则覆盖在 $r4 \sim r11$ 中以减少访存, 提高效率, 13 个通用寄存器的分配如表 2 所示。

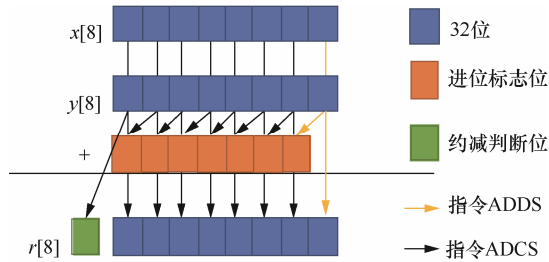


图 2 指令集处理 256 位模加示意
Figure 2 The instruction set handles 256 bit modules plus

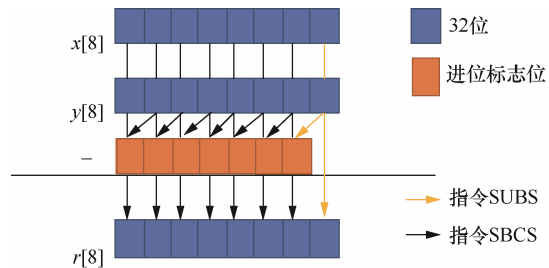


图 3 指令集处理 256 位减法示意
Figure 3 The instruction set handles 256-bit subtraction

表 2 通用寄存器的分配
Table 2 The register allocation

寄存器	功能
$r0 \sim r2$	x 、 y 、 p 地址
$r3$	当前操作数
$r4 \sim r11$	指令 LDMIA 加载的 256 位变量
$r12$	临时存放 $r3$ 操作数, 模减时使用

2.2 改进的蒙哥马利模乘

模乘的执行过程通常可分为两个阶段, 蒙哥马利乘法^[5]和梅森素数法^[7]。首先计算 256 位乘法, 再计算模约减。考虑 MCU 平台通常搭载嵌入式实时操作系统, 并行效率低; 并且, 测试中模平方的按位二次展开法效果不佳, 因此舍弃了蒙哥马利乘法中的相关步骤。由 1.4 节介绍的蒙哥马利乘法, 注意到

$$\begin{aligned} z &= \text{reduce}(z') = \text{reduce}(xyR \bmod p) = \\ &\text{reduce}(x'y) = \text{reduce}(\text{reduce}(xR^2)y) \end{aligned} \quad (11)$$

使用式(11)计算时, 相比 1.4 节中介绍的蒙哥马利乘法, 仅计算两次蒙哥马利约减和两次 256 位乘法。进一步, 可将乘法和约减步骤结合考虑, 本文基于 CIOS (coarsely integrated operand scanning) 算法^[6]对蒙哥马利约减算法进行优化, 调整蒙哥马利乘法的循环体和乘法计算顺序, 整合了乘法和约减过程。整个计算过程分为一层外循环和两层内循环, 基于大整数运算的分段思想, 外循环拆解乘数 b 至 32 位, 第一层每次取 a 的 32 位在结果数组 t 中乘累加, 第二层计算取约减数再次乘累加, 由于结果数组 t 的部分可常驻寄存器, 相比先相乘后约减的模乘计算方式有效降低了访存次数。蒙哥马利约减算法如算法 2 所示。

算法 2 蒙哥马利约减算法

输入 256 位乘法结果 $a \cdot b$

输出 $\text{cios_reduce}(a \cdot b)$

对 i 从 0 到 7, 重复执行

1) $C \leftarrow 0$

2) 对 j 从 0 到 7, 重复执行

2.1) $(C, S) = t[j] + a[j]b[i] + C$

2.2) $t[j] = S$

3) $(C, S) = t[8] + C$

4) $t[8] = S$

5) $t[9] = C$

6) $C \leftarrow 0$

7) $m = t[0]p'[0]$

8) $(C, S) = t[0] + mp[0]$

9) 对 j 从 1 到 7, 重复执行

9.1) $(C, S) = t[j] + mn[j] + C$

9.2) $t[j-1] = S$

① $(C, S) = t[8] + C$

② $t[7]=S$

12) $t[8]=t[9]+C$

其中，步骤 7)为文献[26]提出的改进策略，即使用 $p'[0]$ (p' 的低 32 位)而不是整个 256 位 p' ，实现时对 $p'[0]$ 和 $R^2 \bmod p$ 进行预计算，而不是 p' 和 $R^2 \bmod p$ ，可节省 7 个字的全局存储。最终，改进的蒙哥马利模乘算法如算法 3 所示。

算法 3 改进的蒙哥马利模乘算法

输入 乘数 x ，乘数 y ，素域 p ， $\gcd(p,b)=1$ 且 $R=b^n$ ， $p'=-p^{-1} \bmod b$

输出 $z=xy \bmod p$

1) $T_x = \text{cios_reduce}(x, R^2)$

2) $z = \text{cios_reduce}(T_x, y)$

3) return z

3 多倍点标量乘法优化

考虑低功耗嵌入式平台存储受限的特点，若使用查表法^[20]计算已知点的多倍点标量乘法需占用大量的全局存储空间。本文基于 wNAF 算法^[27]计算多倍点标量乘法，核心思想为对标量 k 进行重编码，并执行一些预计算，占用较少的存储空间对算法进行加速。与同类标量表示算法相比，二进制展开法使用字符集 $\{0,1\}$ ，NAF 使用 $\{0,\pm 1\}$ ，滑动窗 $\{0,1,3,\dots,2^w-1\}$ ，其中 w 为窗口长度。wNAF 算法具有包含更多元素的字符集 $\{0,\pm 1,\pm 3,\dots,\pm 2^w-1\}$ ，由于椭圆曲线域中求加法逆开销相对较低，只需对 y 求素域 p 上的加法逆，那么，wNAF 算法相比滑动窗可在占用相同存储情况下获得双倍字符集长度。算法执行过程可分为标量表示阶段和乘法阶段，如算法 4 和算法 5 所示。

算法 4 wNAF 标量表示算法

输入 标量 k

输出 标量 k 的 wNAF 展开形式 $\text{NAF}_w(k)$

1) $j \leftarrow 0$

2) 当 $k \geq 1$ 时

2.1) 若 k 为奇数， $k_j \leftarrow k \bmod 2^w$ ， $k \leftarrow k - k_j$ ，否则， $k_j \leftarrow 0$

2.2) $k \leftarrow k/2$ ， $j \leftarrow j+1$

3) 返回整数 k 的 wNAF 展开形式 $\text{NAF}_w(k)$

算法 5 wNAF 标量乘法算法

输入 点 P ，整数 k 的 wNAF 展开形式 $\text{NAF}_w(k)$

输出 $Q=k \cdot P$

1) 根据选取的窗长 w 预计算点 $3P, 5P, \dots, (2^w-1)P$

2) $Q \leftarrow O$

3) 对 j 从 $l_{\text{wNAF}}-1$ 到 0，重复执行

3.1) $Q \leftarrow 2Q$

3.2) 若 $k_j > 0, Q \leftarrow Q + k_j P$

3.3) 若 $k_j < 0, Q \leftarrow Q - k_j P$

4) 返回 Q

显然，算法效率与标量 k 的汉明重量（非 0 元素个数）有关，标量表示算法可以有效降低标量 k 的汉明重量。其他标量乘法算法亦可分为预计算和乘法执行两阶段，即通过预计算临时存储得到的结果对乘法执行过程进行加速。一般地，预计算得到结果越多，乘法执行时越快，预计算时间也会增长，所以考虑两阶段开销总和作为算法整体性能标准。计算固定点 G 的乘法时，预计算所得结果可在多次乘法中重复使用，此时预计算所耗时间可忽略不计。考虑低功耗嵌入式设备中存储资源有限，存储一张预计算的乘法表会占用较多存储资源，因此假设预计算结果仅用于一次乘法。忽略对点求加法逆的开销，标量 k 长度为 256 时，考虑 k 最坏情形下（即标量二进制表示汉明重量为 256）的 wNAF 标量乘法计算量如表 3 所示。

表 3 256 位标量计算量
Table 3 256 bit scalar calculated amount

窗口长度 w	预计算阶段 计算量	乘法阶段 计算量	总计算量
2	A+D	86A+256D	87A+257D
3	3A+D	64A+256D	67A+257D
4	7A+D	52A+256D	59A+257D
5	15A+D	43A+256D	58A+257D
6	31A+D	37A+256D	68A+257D
7	63A+D	32A+256D	95A+257D

注：A 代表点加运算，D 代表倍点运算

预计算阶段开销随 w 增大呈指数型增长，同

时乘法阶段计算量的减少量逐渐下降。算法实质上是通过 wNAF 重编码算法获得的标量中的汉明重量减少,从而减少点加计算次数,提高了多倍点标量乘法的整体性能。

4 实验分析

本节给出优化实现的 SM2 国密算法的性能数据,使用了基于 Tom StDenis 编写的 LibTomMath 开源大数运算库实现的 SM2 国密算法作为对比。测试时使用椭圆曲线及有限域参数均为文献[28]中给定参数。

本文进行测试的硬件环境为搭载 cortex-M3 的 STM32F103ZET6 开发板和 ST-Link 下载器,系统时钟频率为 72 MHz (STM32F103ZET6 有 5 个时钟源,最大频率为 72 MHz),存储 flash 为 512 kB;软件环境为 keil5.0 编译器。编译参数使用了 -O3、-Otime、--apcs=interwork,用于开启编译器优化和 thumb 指令集和 ARM 指令集混合调用。以运行时间作为评价指标,将设计的方案与一些其他公开算法进行对比评测。

对于基础运算模块,为避免次数较少、特殊数据的影响,通过重复执行 10 次公钥生成过程在 keil5.0 中的 Simulator 对本文实现的模加模块和 LibTomMath 开源大数运算库中的模加模块进行对比测试,由于库中大数结构体定义较为复杂,对其进行了精简,限定为 256 位整数,结果如表 4 所示,共执行 44 200 次,可见本文设计的基于 Thumb-2 指令集的优化方案使模加算法的执行效率提升明显,性能提升 2.42 倍。

表 4 素数域模加测试结果

Table 4 Test result of prime field modulus plus

方法	约占运行 时间总比例	耗时/ms	平均每次 耗时/ μ s
本文	2%	51.972	1.175 8
LibTomMath 库	6%	177.628	4.018 7

同样地,在开发板上实现设计的模乘方案和一些其他公开算法,进行对比评测。由于模乘在整个 SM2 算法中开销占比约 70%,以随机生成的相同公私钥对整个 SM2 协议运行时间进行测试,通过 JTAG 调试模块记录时间,结果如表 5 所示。

表 5 不同模乘算法下 SM2 算法性能测试结果

Table 5 Test result of SM2 algorithm performance under different modulus multiplication algorithms

模乘算法	SM2 签名时间/s	SM2 验签时间/s	SM2 加密时间/s	SM2 解密时间/s
蒙哥马利乘法 ^[5]	0.652 8	1.293 8	1.276 3	0.649 0
文献[29]	0.412 9	0.810 5	0.807 0	0.410 3
梅森素数法 ^[12]	0.219 9	0.422 0	0.429 2	0.218 2
SOS ^[6]	0.258 1	0.498 8	0.503 9	0.256 1
FIOS ^[6]	0.369 6	0.722 4	0.722 1	0.367 1
本文	0.207 2	0.396 4	0.404 4	0.205 5

由于平台本身不支持模平方的加速算法,同时结合 MCU 平台访存开销大的特点,设计的模乘方案优于强算力平台中表现较好的梅森素数法,且避免了对素数 p 的限定,更广泛适用于实际环境中各种需要更换椭圆曲线参数来提高安全性的需求,同时兼顾 SM2 国密算法中的模 n 运算和模 p 运算,亦可在其他需计算大整数模乘的密码算法中推广。

为了测评多倍点标量乘法中的标量表示算法,对 1 000 次未知点多倍点标量乘法计算时间进行测试对比,结果如图 4 所示,其中预计算开销如表 6 所示。

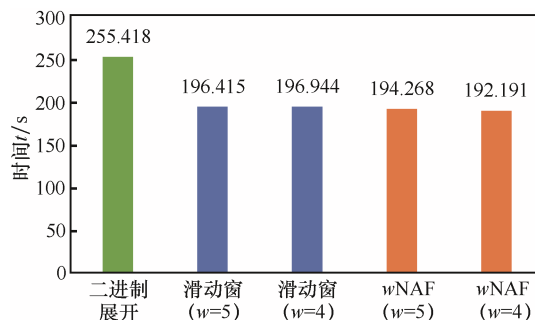


图 4 多倍点标量乘法测试结果

Figure 4 Result of multiples scalar multiplication

表 6 预计算开销分析

Table 6 Analysis of pre-computed overhead

多倍点标量 乘法算法	预计算 时间占比	预计算 耗时/s	标量乘法 耗时/s
二进制展开	0%	0	255.418
滑动窗 (w=5)	7.6%	14.976	196.451
滑动窗 (w=4)	3.7%	7.253	196.944
wNAF (w=5)	7.7%	14.951	194.268
wNAF (w=4)	3.8%	7.272	192.161

结果表明,取 $w=4$ 时的 $wNAF$ 算法效果最佳。由表 3 可知,最坏情形时 w 取 5 较取 4 少计算一次点加,但实际情况中标量的汉明重量通常少于最坏情形,故考虑预计算结果仅用于一次乘法时,应取 $w=4$,适用于未知点乘法,对于固定点乘法预计算时间可忽略不计,此时窗长取值可视存储情况而定。

与基于 LibTomMath 库构建基础运算的实现的 SM2 算法进行对比测试,同时对本文设计的 SM2 国密算法优化方案的整体性能表现进行评估。对签名、验签、公钥生成、加密、解密运算各执行 1 000 次取平均值,结果如图 5 所示,单次签名速度可达 0.205 秒/次,加密速度 0.415 秒/次,显著优于基于开源库 LibTomMath 实现的 SM2 国密算法,签名性能提升 1.62 倍,加密性能提升 1.59 倍。

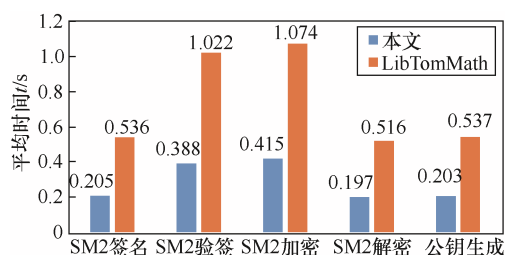


图5 SM2 算法测试结果
Figure 5 Result of algorithm SM2

5 结束语

本文面向低功耗嵌入式平台设计了 SM2 算法的优化实现方案,其中通过 Thumb-2 指令集的支持构建了高效的模加减运算模块,对上层运算提供支持;结合低功耗嵌入式平台访存慢、并行效率低的特点,基于蒙哥马利乘法进行改良,并结合 CIOS 算法设计了高效的模乘计算方案,相比梅森素数法提高了灵活性和效率。实验结果表明,本文给出的优化实现能有效地提升 SM2 算法的计算速度,实验测试中达到了 0.205 秒/次签名,0.415 秒/次加密速度,可以更好地满足在低功耗嵌入式设备上高效计算签名与加密的需求,具有非常高的实际应用价值。

参考文献:

[1] 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]. 北京:中国标准出版社, 2012:8.

State Cryptography Administration. SM2 elliptic curve public key cryptography algorithm part 1: general provisions: GM/T 0003.1-2012[S].8 Beijing: China Standard Press, 2012: 8.

[2] 罗玮榕, 曹进, 李晖, 等. 基于 SM2 联合签名的电子发票公开验证方案[J]. 网络与信息安全学报, 2022, 8(2): 122-131.

LUO Y R, CAO J, LI H, et al. Electronic invoice public verification scheme based on SM2 joint signature [J]. Journal of Network and Information Security, 2022, 8(2): 122-131.

[3] 陈锋, 邹洪, 吴亚楠, 等. 基于 SM2 密码体系的电力信息安全监控系统设计[J]. 电子设计工程, 2022, 30(5):100-103, 108.

CHEN F, ZOU H, WU Y N, et al. Design of power information security monitoring system based on sm2 cipher system[J]. Electronic Design Engineering, 2022, 30(5):100-103, 108.

[4] 程朝辉. 基于 SM2 的无证书加密算法[J]. 密码学报, 2021, 8(1): 87-95.

CHENG C H. Certificateless encryption algorithm based on SM2 [J]. Journal of Cryptography, 2021, 8(1): 87-95.

[5] MONTGOMERY P L. Modular multiplication without trial division[J]. Mathematics of Computation, 1985, 44(170): 519-521.

[6] KOC C, ACAR T, JR B. Analyzing and comparing montgomery multiplication algorithms[J]. Micro, IEEE, 1996, 16(3): 26-33.

[7] SOLINAS J A. Generalized mersenne numbers[M]. Faculty of Mathematics, University of Waterloo, 1999.

[8] PARIHAR A, NAKHATE S. High-speed high-throughput VLSI Architecture for RSA montgomery modular multiplication with efficient format conversion[J]. Journal of the Institution of Engineers (India) Series B, 2019, 100(2): 217-222.

[9] RAHMAN M S, HOSSAIN M S, RAHAT E H, et al. Efficient hardware implementation of 256-bit ECC processor over prime field[C]//2019 International Conference on Electrical, Computer and Communication Engineering (ECCE). 2019: 1-6.

[10] 王腾飞, 张海峰, 许森. SM2 专用指令协处理器设计与实现[J]. 计算机工程与应用, 2022, 58(2): 102-109.

WANG T F, ZHANG H F, XU S. Design and implementation of SM2 special instruction coprocessor[J]. Computer Engineering and Applications, 2022, 58(2): 102-109.

[11] 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, 2019, 66(3): 1003-1016.

[12] 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.

LI B, ZHOU Q L, CHEN X J, et al. Optimization of reconfigurable SM2 algorithm over prime field[J]. Journal of Communication, 2022, 43(3): 30-41.

[13] HOSSAIN M S, KONG Y. High-performance FPGA implementation of modular inversion over F_{256} for elliptic curve cryptography[C]//2015 IEEE International Conference on Data Science and Data Intensive Systems. 2015: 169-174.

[14] ZHOU L, SU C, HU Z, et al. Lightweight implementations of NIST P-256 and SM2 ECC on 8-bit resource-constraint embedded de-

- vice[J]. ACM Transactions on Embedded Computing Systems (TECS), 2019, 18(3): 1-13.
- [15] 朱辉, 黄煜坤, 王枫为, 等. 一种基于图形处理器的高吞吐量 SM2 数字签名计算方案[J]. 电子与信息学报, 2022, 44(10): 1-10.
- ZHU H, HUANG Y K, WANG F W, et al. A high throughput SM2 digital signature computing scheme based on graphics processor[J]. Journal of Electronics and Information Technology, 2022, 44(10): 1-10.
- [16] LONGA P, GEBOTYS C. Efficient techniques for high-speed elliptic curve cryptography[C]//International Workshop on Cryptographic Hardware and Embedded Systems. Heidelberg, 2010: 80-94.
- [17] COHEN H, MIYAJI A, ONO T. Efficient elliptic curve exponentiation using mixed coordinates[C]//International Conference on the Theory and Application of Cryptology and Information Security. 1998: 51-65.
- [18] SETIADI I, KISTIANTORO A I, MIYAJI A. Elliptic curve cryptography: Algorithms and implementation analysis over coordinate systems[C]//2015 2nd International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA). 2015: 1-6.
- [19] LONGA P, MIRI A. Fast and flexible elliptic curve point arithmetic over prime fields[J]. IEEE Transactions on Computers, 2008, 57(3): 289-302.
- [20] HANKERSON D, MENEZES A J, VANSTONE S. Guide to elliptic curve cryptography[M]. Springer Science & Business Media, 2006.
- [21] ISLAM M M, HOSSAIN M S, HASAN M K, et al. FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field[J]. IEEE Access, 2019, 7: 178811-178826.
- [22] 国家密码管理局. SM3 密码杂凑算法: GMT 0004.4-2012[S]. 北京: 中国标准出版社, 2012:8.
- National Cryptography Administration. SM3 password hashing algorithm: GMT 0004.4-2012[S]. Beijing: Standards Press of China, 2012:8.
- [23] RIVAIN M. Fast and regular algorithms for scalar multiplication over elliptic curves[J]. Cryptology ePrint Archive, 2011.
- [24] GURA N, Patel A, Wander A, et al. Comparing elliptic curve cryptography and RSA on 8-bit CPUs[C]//International Workshop on Cryptographic Hardware and Embedded Systems. 2004: 119-132.
- [25] Advanced Micro Devices, Inc. AMD graphic core next[R]. Advanced Micro Devices, Inc, 2011.
- [26] DUSSE S R, KALISKI B S. A cryptographic library for the Motorola DSP56000[C]//Workshop on the Theory and Application of Cryptographic Techniques. 1990: 230-244.
- [27] MUIR J, STINSON D. Minimality and other properties of the width- w nonadjacent form[J]. Mathematics of Computation, 2006, 75(253): 369-384.
- [28] 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: GMT 0003.4-2012[S]. 北京: 中国标准出版社, 2012.
- National Cryptography Administration. SM2 Elliptic curve public key cryptography algorithm Part 5: Parameter definition: GMT 0003.4-2012[S]. Beijing: Standards Press of China, 2012.
- [29] 闫闯. 基于 GPU 的 SM2 数字签名和验证算法的快速实现与优化[D]. 上海: 上海交通大学, 2020.
- YAN M. Fast implementation and optimization of SM2 digital signature and verification algorithm based on GPU [D]. Shanghai: Shanghai Jiaotong University, 2020.

[作者简介]



刘赣秦 (1998-), 男, 江西莲花人, 西安电子科技大学硕士生, 主要研究方向为密码算法的高性能优化。



李晖 (1968-), 男, 河南灵宝人, 博士, 西安电子科技大学教授、博士生导师, 主要研究方向为密码学、无线网络安全、云计算安全、信息论与编码理论。



朱辉 (1981-), 男, 河南周口人, 博士, 西安电子科技大学教授、博士生导师, 主要研究方向为数据安全与隐私保护、安全方案及协议设计、网络及应用安全。



黄煜坤 (1998-), 男, 湖北襄阳人, 西安电子科技大学硕士生, 主要研究方向为密码算法的高性能优化。



刘兴东 (1992-), 男, 湖南衡阳人, 西安电子科技大学硕士生, 主要研究方向为网络空间安全、可信执行环境、密码算法优化。