面向多椭圆曲线的高速标量乘法器设计与实现

于斌，黄海，刘志伟，赵石磊，那宁
（哈尔滨理工大学软件与微电子学院，黑龙江 哈尔滨 150080）

摘 要：针对现有标量乘法器不能适用于多椭圆曲线且运算开销较大的问题，设计了一种能应用于两类素数域椭圆曲线的高速标量乘法器。首先，在标量乘算法上，对 secp256r1 曲线的基点采用 Comb 算法，对普通点采用 Shamir 算法，对 Curve25519 曲线使用蒙哥马利阶梯算法；然后，优化了点加和倍点运算的操作步骤，并对点加中 Z=1 的情况进行简化设计，有效减少计算周期数；最后，采用快速模约简实现模乘，设计了 Curve25519 的快速模约简算法。整个设计充分考虑复用，在 55 nm CMOS 工艺下需 1 022×10^3 个等效门，在 secp256r1 和 Curve25519 上计算普通点标量乘，运算速度分别为 15.3 万次/秒和 15.8 万次/秒，其中 secp256r1 上的运算速度是现有设计的 1.9 倍。

关键词：椭圆曲线密码学；标量乘；快速模约简；硬件实现
中图分类号：TN918；TP309
文献标识码：A
doi: 10.11959/j.issn.1000-463X.2020226

Design and implementation of high-speed scalar multiplier for multi-elliptic curve

YU Bin, HUANG Hai, LIU Zhiwei, ZHAO Shilei, NA Ning
School of Software and Microelectronics, Harbin University of Science and Technology, Harbin 150080, China

Abstract: Aiming at the problem that the existing scalar multiplier cannot be applied to multi-elliptic curve and the cost is expensive, a high-speed scalar multiplier was designed, applicable to two types of elliptic curves over prime fields. Firstly, in terms of the scalar multiplication, secp256r1 base points were processed with the comb algorithm, and the Shamir algorithm for ordinary points, and the Montgomery ladder algorithm for Curve25519. Secondly, the operation of point addition and point doubling was optimized, and the condition of Z=1 in point addition was simplified, thereby effectively reducing the number of calculation cycles. Lastly, a fast modular reduction algorithm of Curve25519 was designed for modular multiplication. Multiplexing was an important factor in the entire designing process. A 1022K equivalent gate was selected for the 55 nm CMOS process. This allowed ordinary point scalar multiplications performed on secp256r1 and Curve25519 respectively, calculating at the speeds of 153 000 times per second and 158 000 times per second, with the speed for secp256r1 1.9 times that of the existing designed one.

Key words: ECC, scalar multiplication, fast modular reduction, hardware implementation

1 引言

随着处理器算力的不断增强，传统加密算法的安全性正受到日益严峻的考验，需要更加复杂的加密算法来保证数据的安全。椭圆曲线加密算法是近年来比较活跃的一种非对称加密算法，与对称加密

收稿日期：2020−05−06；**修回日期**：2020−09−17
通信作者：黄海，ic@hrbust.edu.cn
基金项目：黑龙江省自然科学基金资助项目（No.YQ2019F010）；黑龙江省普通本科高等学校青年创新人才培养计划基金资助项目（No.UNPYSCT-2017081）；国家重点研发计划基金资助项目（No.2018YFB2202100）
Foundation Items: The Natural Science Foundation of Heilongjiang (No.YQ2019F010), The University Nursing Program for Young Scholars with Creative Talents in Heilongjiang (No.UNPYSCT-2017081), The National Key Research and Development Program of China (No.2018YFB2202100)

算法中的高加密标准（AES, advanced encryption standard）一样在密码系统中得到广泛使用[1]。在最新的传输层安全性协议 1.3 版（TLS1.3, transport layer security protocol version 1.3）中，使用多种椭圆曲线进行数据加密，无论是在数量上还是在实际使用频率上，都凸显了椭圆曲线加密日益重要的位置[2]。

椭圆曲线的运算速度和对多种曲线的兼容一直是研究的重点。Kudithi 等[3]利用可编程逻辑器件（FPGA, field programmable gate array）完成了适用于物联网的加密算法结构，利用较低面积完成了对 P224 和 P256 椭圆曲线的支持；Hossain 等[4]设计了同时适用于专用集成电路和 FPGA 的标量乘操作步骤，同样支持了 2 条曲线；Liu 等[5]优化了非相邻形式（NAF, non-adjacent-form）算法并完成了 P256 位的标量乘设计；Lee 等[6]采用双处理单元改进了自右向左的标量乘算法；Chung 等[7]改进了蒙哥马利模乘的流水结构，提升了标量乘的整体速度。

本文设计了椭圆曲线硬件加密中使用的一种标量乘法器，选取最常用的 Curve25519 和 secp256r12 曲线，使其尽可能复用乘法单元完成标量乘，使用了 256 位乘法器得到乘法结果并采用快速模约简来达到提高模乘速度的目的。此外，考虑标量乘的使用环境，尤其是在签名和验签中的使用情况，本文设置了对于基点和普通点不同的算法，同时支持多标量乘，使设计的标量乘法器能够满足椭圆加密的各种应用需求。

2　研究背景

椭圆曲线有很多分类，使用较早的一类曲线是 Weierstrass 曲线，满足式(1)所示的曲线方程。

$$y^2 = x^3 + ax + b \tag{1}$$

TLS1.3 所选用由美国国家标准与技术研究所（NIST, National Institute of Standards and Technology）提出的 secp256r1、secp384r1 和 secp521r1 这 3 条曲线，就属于 Weierstrass 曲线。

椭圆曲线上的运算采用模运算，包括模加减、模乘和模逆运算，最核心的运算方式为模乘和模逆。点加和倍点运算以模运算为基础，并进一步用来实现标量乘。标量乘的性能也是衡量椭圆曲线加密性能的一个重要的指标，为了得到更快的加密速度，需要尽可能快速地完成标量乘。根据标量乘的运算过程，大致可把提高标量乘速度的优化方法分为 3 类：优化标量乘算法[8-13]、转换坐标系[14-16]和优化模运算算法[17-20]。

第一类优化方法是改变标量乘 kP 的运算过程，降低总体运算量，其中，P 是椭圆曲线上的点，k 是待乘的倍数。优化的方法包括：使用自左向右和自右向左 2 种算法，对于 n 位的 k 值需要计算 n 次倍点和 $\frac{n}{2}$ 次点加，其中自左向右算法的点加和倍点可以并行运算，但是需要额外的模运算单元[8]；把 k 做相对简单的预处理，实现 NAF 算法，把点加的次数降低为 $\frac{n}{3}$ 次[9]；对 k 做 m 位的分段处理，实现窗口 NAF 法，预计算需要计算窗口内的 2^m 个点运算结果，预计算量较大[10]；使用 Comb 算法来快速完成标量乘，但需要的预计算量庞大[11]；采取适用于多标量乘的 Shamir 算法，用一次标量乘的时间完成 2 个标量乘的相加[12]；采用多基链算法，增加三倍点、五倍点形成多基链来加速标量乘[13]。

第二类方法采用转换坐标系，目的是优化标量乘中的点加和倍点运算。原曲线方程使用二元坐标系，点加运算需要一次模逆和 3 次模乘（模平方记为模乘，忽略模加减运算，下文相同），倍点运算需要一次模逆和 4 次模乘，而模逆运算需要消耗大量时钟周期，因此可以通过坐标系的转换改变点加和倍点运算计算式，消除模逆运算，例如射影坐标系，把坐标(x,y)按转换式 $x=\frac{X}{Z}$，$y=\frac{Y}{Z}$ 转换为(X,Y,Z)，点加运算需要 14 次模乘，倍点运算需要 12 次模乘[14]。雅可比坐标系使用比较普遍，转换式为 $x=\frac{X}{Z^2}$，$y=\frac{Y}{Z^3}$，这需要 16 次模乘来完成点加运算，10 次模乘完成倍点运算[15]。此外还有修正的雅可比坐标系[16]、混合坐标系[16]等变换，其目的都是在开始时做一次变换，随后在计算点加和倍点过程中消除模逆运算，最后的计算结果再转换为二元坐标，整个标量乘只使用一次模逆运算。

第三类优化方法是从基本的模运算入手，提高模运算的速度，主要是模乘和模逆运算。模乘运算使用较多的是蒙哥马利模乘算法，通过将多位的模乘分解为 2^m 位的模乘，利用二进制的特点，进行多次迭代运算得到最后的模乘结果[17]。IEEE 给出对于 NIST 中所使用的模 p 快速约简公式，把 2 个数据的乘法结果直接做快速的模约简，得到模乘结果[18]。模逆运算

的计算式一般会采用费马小定理，如式(2)所示。

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

通过简单整理即可得到模逆运算的计算式为

$$a^{p-2} \equiv a^{-1} (\mathrm{mod}\ p) \tag{3}$$

这样可以不需要模逆运算，只需要使用模乘即可完成模逆运算，也可以采用专门的模逆模块来完成模逆运算。蒙哥马利模逆算法与蒙哥马利模乘配合使用，不需要经过转换也可以完成模逆运算[19]，也可以使用二进制右移算法，通过多次简单比较和加减来计算模逆运算，资源开销较小[20]，独立设计模逆运算主要是为了释放模乘单元，达到并行的目的，同时减少计算周期数。

还有一类椭圆曲线是蒙哥马利曲线，其方程为

$$y^2 = x^3 + Ax^2 + x \tag{4}$$

Curve25519 曲线属于蒙哥马利曲线，由 Bernstein[21]于 2005 年提出，并在 2016 年与 Curve448 一起成为国际标准[22]，这 2 种曲线也被 TLS1.3 采用。此类曲线提出较晚，在近年开始得到广泛使用，其标量乘本身具有抗 SPA（simple power analysis）特性，所以一般直接使用蒙哥马利阶梯算法而不会做修改[23]，只是在模乘部分做优化，Düll 等[24]计算给出了标量乘所需的模乘和模平方数。

上述算法虽各有优点，但都是仅考虑单一使用而优化的。在椭圆曲线加密中，标量乘有很多使用场合，涉及基点 G 或者普通点 P 的标量乘，在验签时需要计算多标量乘 $\lambda P+\mu G$，同时服务器端签名验签频繁，对标量乘速度的要求极高，需要一个能在各种使用场合下都具备较高性能的标量乘法器。

本文全面考虑标量乘法器在椭圆加密算法中的使用情况，完成了如下工作。

1) 选择适用于不同情况的标量乘算法，使标量乘计算普通点 P、基点 G 和多标量乘 $\lambda P+\mu G$ 时都达到较快的运算速度。

2) 推导了 Curve25519 的快速模约简公式，结合已有的 secp256r1 的快速模约简公式，用快速模约简实现模乘。

3) 结合利用模约简计算模乘的运算特点，排布了 secp256r1 和 Curve25519 这 2 条曲线的点加倍点操作步骤，根据 Comb 算法特点简化点加操作，并排布了雅可比坐标系下特殊点加（有一个点的 Z 值为 1 时）的操作步骤。

4) 在保证模乘速度的前提下，设计充分考虑复用以降低面积，完成适用于 secp256r1 和 Curve25519 曲线的标量乘法器。

3 标量乘法器设计

标量乘法器的设计也同优化一样，需要完成标量乘算法的设计、点加倍点的设计和模运算的设计。在实际使用情况中，secp256r1 曲线和 Curve25519 曲线使用频繁且具有相近的位数，故将这 2 条曲线的标量乘集成为一个单元进行设计。由于采用了 3 种标量乘算法来适应不同需求，整个设计采用了 3 个独立的子控制模块来实现不同算法的控制功能，在工程中可以根据不同的使用场合来切换算法，调用运算单元，达到最快的速度。标量乘法器的整体结构如图 1 所示。

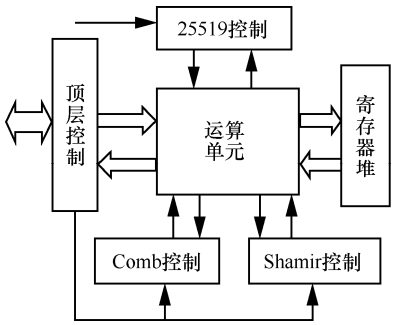


图 1 标量乘法器的整体结构

运算单元完成所有点加倍点运算，除必需的控制逻辑之外，主要运算部分是一个 256 位的乘法器，通过快速模约简单元完成模乘功能，所有预计算数值和中间计算结果都存储至临时寄存器堆中，最大限度地复用了硬件资源。由于标量乘计算结果需使用模逆运算转换至二元坐标系，考虑到椭圆加密的运算过程，将模逆端口一并引出，可供外部调用，使整个设计可以满足椭圆加密中所有关键步骤的使用需求。

3.1 标量乘算法

使用 secp256r1 曲线时，加密系统会在签名和公钥生成时使用基点 G 的标量乘，在密钥交换时使用普通点 P 的标量乘，在验签时使用 $\lambda P+\mu G$ 的多标量乘，所以选择了 Comb 算法和 Shamir 算法来应对不同使用场合。算法 1 为编码长度为 4 的 Comb 算法[11]。

算法 1 编码长度为 4 的 Comb 算法

输入 256 位二进制数 $\lambda=\{\lambda_{255}\lambda_{254}\lambda_{253}\cdots\lambda_1\lambda_0\}$和椭圆曲线 secp256r1 的基点 G。

输出　标量乘 $Q=\lambda G$

1) 构造梳状算法预计算表$\{0000\}G\rightarrow\{1111\}G$，共 16 个点坐标

2) 提取编码系数 $\alpha_i=\{\lambda_{i+192},\lambda_{i+128},\lambda_{i+64},\lambda_i\}$

3) 令 $Q=0$，为无穷远点

4) for i=63:0 do

$Q=2Q$

$Q=Q+\alpha_iG$

end for

5) 返回 Q 值

Comb 算法需要的预计算量非常大，例如$\{1111\}G$ 就需要计算$(2^{192}+2^{128}+2^{64}+2^0)G$，在计算普通点时该算法并不占优势。但是由于基点 G 已知，Comb 算法的预计算表可以提前完成并存储，这样在计算时直接使用该表，再使用 64 次倍点和最多 64 次点加即可完成基点 G 的标量乘，需要额外付出的代价是 15 个点坐标的存储阵列，其中，$\{0000\}G$ 不需要存储。

Shamir 算法可以计算 $\lambda P+\mu G$ 的多标量乘，经过预计算后可以用一次标量乘的时间完成 $\lambda P+\mu G$ 的计算。算法 2 为窗口为 2 的 Shamir 算法[12]。

算法 2　窗口为 2 的 Shamir 算法

输入　256 位二进制数 $\lambda=\{\lambda_{255}\lambda_{254}\lambda_{253}\cdots\lambda_1\lambda_0\}$，$\mu=\{\mu_{255}\mu_{254}\mu_{253}\cdots\mu_1\mu_0\}$，椭圆曲线 secp256r1 上的普通点 P 和基点 G。

输出　$Q=\lambda P+\mu G$

1) 构造预计算表$(00)P+(00)G\rightarrow(11)P+(11)G$，共 16 个点坐标

2) 令 $Q=0$，为无穷远点

3) for i=127:0 do

$Q=4Q$

$Q=Q+\{(\lambda_{2i+1}\lambda_{2i})P+(\mu_{2i+1}\mu_{2i})G\}$

end for

4) 返回 Q 值

当 $\lambda\mu\neq0$ 时，对于输入的 P、G 点坐标，需要 3 次倍点和 4 次点加来计算预计算表，预计算结束后进入循环，4 倍点不设置专用的计算式，复用倍点计算式 2 次来完成 4 倍点功能，这样可与 Comb 算法复用倍点运算单元，所以在窗口为 2 的 Shamir 算法中，依然会计算 256 次倍点和最多 128 次点加，同时存储除了$(00)P+(00)G$ 之外的 15 个点坐标。如果只算普通点乘的 λP，则令 $\mu=0$ 即可，此时进行一次点加和倍点即可完成预计算，标量乘循环计算时所需次数与 $\lambda\mu\neq0$ 时相同。

对于 Curve25519 曲线采用蒙哥马利阶梯算法来完成标量乘，维持抗 SPA 特性，其算法步骤如算法 3 所示[23]。

算法 3　Curve25519 标量乘算法

输入　255 位二进制数 k，点坐标 $P_1=(x_1,y_1)$的横坐标 x_1

输出　标量乘结果 $kP_1=(x_2,y_2)$的横坐标 x_2

1) $X_1=x_1$；$X_2=x_1$；$Z_2=0$；$X_3=x_1$；$Z_3=1$；swap=0

2) for i=254:0 do

swap=swap^$k[i]$

(X_2,X_3)=cswap(swap,X_2,X_3)

(Z_2,Z_3)=cswap(swap,Z_2,Z_3)

swap=$k[i]$

(X_2,Z_2,X_3,Z_3)=阶梯算法 Ladder step(X_1, X_2, Z_2,X_3,Z_3)

end for

3) (X_2,X_3)=cswap(swap,X_2,X_3)

(Z_2,Z_3)=cswap(swap,Z_2,Z_3)

4) 返回 $X_2=\dfrac{X_2}{Z_2}$

其中，cswap 操作是根据 swap 的值来交换 2 个输入的坐标值，算法不单独列出；阶梯算法 Ladder step 在 3.2 节中给出详细步骤。

3.2　高速点加和倍点运算设计

运算单元的功能是执行点加和倍点运算，按 2 条曲线划分，执行的计算式不同。对于椭圆曲线 secp256r1，采用的标量乘算法中点加操作相对较少，故采用倍点较快的雅可比坐标系，倍点计算式[16]为

$$X_3=T,Y_3=-8Y_1^4+M(S-T),\ Z_3=2Y_1Z_1 \tag{5}$$

其中，$S=4X_1Y_1^2,M=3X_1^2+aZ_1^4,\ T=-2S+M^2$。

点加计算式[16]为

$$\begin{aligned}X_3&=-H^3-2U_1H^2+r^2\\Y_3&=-S_1H^3+r(U_1H^2-X_3)\\Z_3&=Z_1Z_2H\end{aligned} \tag{6}$$

其中，$U_1=X_1Z_2^2,U_2=X_2Z_1^2,S_1=Y_1Z_2^3,S_2=Y_2Z_1^3$，$H=U_2-U_1,r=S_2-S_1$。

由于采用快速模约简进行模乘，模乘结果需要经过一个周期的乘法和一个周期的模约简才可以被继续使用，为减少模乘次数，并结合式(5)，将倍点运算的操作步骤排布如表 1 所示，共需 9 个周期

完成倍点运算$(X_3,Y_3,Z_3)=2(X_1,Y_1,Z_1)$。

表 1 倍点运算步骤

步骤	模乘	模加减
1	$t_1\leftarrow Z_1Z_1$	—
2	$t_2\leftarrow Y_1Y_1$	$t_3\leftarrow x_1+t_1$, $t_4\leftarrow X_1-t_1$
3	$t_1\leftarrow t_3t_4$	—
4	$t_3\leftarrow Y_1Z_1$	$t_4\leftarrow 8t_2$
5	$t_5\leftarrow X_1t_4$	$t_1\leftarrow 3t_1$
6	$t_3\leftarrow t_1t_1$	$Z_3\leftarrow t_3+t_3$
7	$t_2\leftarrow t_2t_4$	$X_3\leftarrow t_3-t_5$, $t_4\leftarrow 1.5t_5-t_3$
8	$t_1\leftarrow t_1t_4$	—
9	—	$Y_3\leftarrow t_1-t_2$

表 1 的步骤 5 中，t_5得到 $2S$，t_1得到 M，步骤 7 的 X_3即为 T 值，其余各值依次代入即可验证是否与式(5)一致。为了保证数据的有效性，模乘结果需间隔一周期才能作为输入，例如步骤 2 的乘法结果 t_2需在步骤 3 进行模约简，在步骤 4 得到模乘结果并继续参与计算，模约简过程并未列在算法中。模加减单元作了特殊设计，增加了 3 倍模加、被减数为 1.5 倍的模减，保证乘法器的连续使用并减少模乘的次数。同时考虑 256 位乘法的时间时延较大，而模约简的时间时延相对较小，所以在模约简之后可以安排一次模加减，进一步压缩操作步骤，如步骤 2 中 t_1的使用就是如此。表 1、表 2 和表 3 均采用同样的设计思路排布，模运算具体内容见 3.3 节。

正常情况下表 1 的倍点运算需要 8 个周期模乘加一个周期的约简等待，但倍点运算后立即会进行点加或者倍点运算，所以通过调整(X,Y,Z)的输出顺序，可以在下次乘法操作同时进行模约简，实际工作中需要 8 个周期可完成一次倍点运算。

对于点加公式，需要做分类处理以达到最优化，在 Shamir 算法中需构造预计算表，但是表内数据都是在计算开始后得到的，所以都是普通的三元坐标(X,Y,Z)，按式(6)排布操作步骤如表 2 所示，共需 17 个周期完成普通点加运算$(X_3,Y_3,Z_3)=(X_1,Y_1,Z_1)+(X_2,Y_2,Z_2)$。

表 2 中步骤 3 计算 U_2，步骤 4 计算 U_1，步骤 5 的 t_4计算 H，步骤 8 计算 S_2，步骤 9 计算 S_1，步骤 11 的 t_3计算 r，依次代入后，X_3可得

$$X_3=r^2-(U_1+U_2)H^2=r^2-(U_1+H+U_1)H^2=r^2-H^3-2U_1H^2$$

表 2 普通点加运算步骤

步骤	模乘	模加减
1	$t_1\leftarrow Z_1Z_1$	—
2	$t_2\leftarrow Z_2Z_2$	—
3	$X_3\leftarrow X_2t_1$	—
4	$t_3\leftarrow X_1t_2$	—
5	$t_1\leftarrow t_1Z_1$	$t_6\leftarrow X_3+t_3$, $t_4\leftarrow X_3-t_3$
6	$t_5\leftarrow t_4t_4$	—
7	$t_2\leftarrow t_2z_2$	—
8	$t_1\leftarrow Y_2t_1$	—
9	$t_2\leftarrow Y_1t_2$	—
10	$Y_3\leftarrow t_5t_6$	—
11	$t_6\leftarrow t_4t_5$	$t_3\leftarrow t_1-t_2$, $t_2\leftarrow t_1+t_2$
12	$t_1\leftarrow t_3t_3$	—
13	$t_5\leftarrow t_2t_6$	$X_3\leftarrow t_1-Y_3$
14	$Z_3\leftarrow Z_1Z_2$	$t_6\leftarrow Y_3-2X_3$
15	$Y_3\leftarrow t_3t_6$	—
16	$Z_3\leftarrow Z_3t_4$	$Y_3\leftarrow Y_3-t_5$
17	—	$Y_3\leftarrow 0.5Y_3$

步骤 16 中的 Y_3可得

$$\begin{aligned}&Y_3=r[(U_1+U_2)H^2-2X_3]-(S_1+S_2)H^3=\\&[r(U_1+U_2)-(S_1+S_2)H]H^2-2rX_3=\\&[2S_2U_1-2S_1U_2]H^2-2rX_3=\\&[2rU_1-2S_1H]H^2-2rX_3=\\&-2S_1H^3+2r(U_1H^2-X_3)\end{aligned}$$

步骤 17 中对 Y_3做 0.5 倍模加，Z_3代入即可，所得结果与式(6)一致。

在 Comb 算法中，由于每次点加的 α_iG 都是预先计算并存储的，其点坐标格式均为$(X,Y,1)$，Z=1 不需要存储，且可以将式(6)做进一步优化和整理，并按照时序排布操作步骤如表 4 所示，共需 13 个周期可完成一次特殊点加运算$(X_3,Y_3,Z_3)=(X_1,Y_1,Z_1)+(X_2,Y_2,1)$。

表 4 与表 2 相似，依次代入后即可验证是否与式(6)一致。点加运算结束后会立刻进行倍点运算，所以在 2 种点加运算的操作步骤上都对最后输出的坐标做了调整，以适应点加倍点的连续运算，最后一步的模加减均与下一运算的乘法同时运行，让乘法器使用率达到最高。

Curve25519 的点加和倍点是采用阶梯算法一起完成的，按 RFC7748 中的参考算法，将操作步骤

重新排布如表 3 所示，共需 12 个周期完成操作，可根据输入的 (X_1,X_2,Z_2,X_3,Z_3) 计算得到 (X_2,Z_2,X_3,Z_3)。

表 3　Ladder step 阶梯运算步骤

步骤	模乘	模加减
1	—	$t_1\leftarrow X_2+Z_2$
2	$t_6\leftarrow t_1t_1$	$t_2\leftarrow X_2-Z_2$
3	$t_7\leftarrow t_2t_2$	$t_3\leftarrow X_3-Z_3$
4	$t_8\leftarrow t_1t_3$	$t_4\leftarrow X_3+Z_3$, $t_5\leftarrow t_6-t_7$
5	$t_9\leftarrow t_2t_4$	—
6	$X_2\leftarrow t_6t_7$	$X_3\leftarrow t_8+t_9$,$Z_3=t_8-t_9$
7	$X_3\leftarrow X_3X_3$	—
8	$Z_2\leftarrow 121665t_5$	—
9	$Z_3\leftarrow Z_3Z_3$	$Z_2\leftarrow Z_2+t_6$
10	$Z_2\leftarrow Z_2t_5$	—
11	$Z_3\leftarrow Z_3X_1$	—
12	—	Z_3 约简

表 4　特殊点加运算步骤

步骤	模乘	模加减
1	$t_1\leftarrow Z_1Z_1$	—
2	$t_2\leftarrow Y_2Z_1$	—
3	$t_3\leftarrow X_2t_1$	—
4	$t_1\leftarrow t_1t_2$	$t_2\leftarrow t_3-X_1$, $t_3\leftarrow t_3+x_1$
5	$t_4\leftarrow t_2t_2$	$t_1\leftarrow t_1-Y_1$
6	$Z_3\leftarrow Z_1t_2$	—
7	$t_2\leftarrow t_2t_4$	—
8	$t_3\leftarrow t_3t_4$	—
9	$t_5\leftarrow t_1t_1$	—
10	$t_4\leftarrow X_1t_4$	$X_3\leftarrow t_5-t_3$
11	$t_2\leftarrow Y_1t_2$	$t_3\leftarrow t_4-X_3$
12	$t_1\leftarrow t_1t_3$	—
13	—	$Y_3\leftarrow t_1-t_2$

这样，根据曲线和算法来调用不同的点加控制和倍点控制，使用乘法器和模约简以及其他必需的模运算单元来完成对应操作。由于表 1～表 4 的运算步骤仅仅是控制数据的选择输入和输出，所有运算都由模运算完成，中间计算结果也存储在寄存器堆，所以运算本身占用硬件资源较少。

3.3　模运算设计

模运算部分主要是模加减、模乘和模逆的设计。在点加倍点的操作周期中需要正常模加和模减单元各一个，还出现了特殊模运算单元，例如对于素数域中 2 个数值 M 和 N，需要计算 $3M$、$8M$、$1.5M-N$、$M-2N$，以及模逆需要的 $0.5M$，此类运算算法简单，均设置独立单元用于运算。此外，由于乘法消耗时间较长，为配合点加和倍点的周期操作，在模约简的输出端直接接入一个模加单元和一个模减单元，使模约简的结果可以直接进行模加减操作，从而能在一个周期内安排更多的操作，整个模运算部分的结构如图 2 所示。

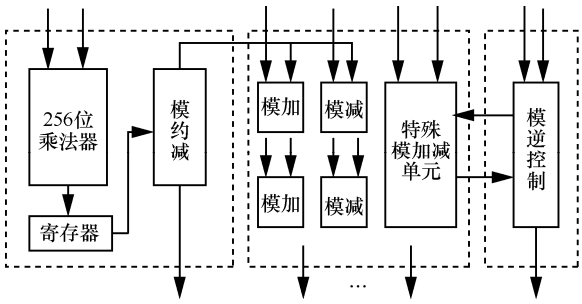


图 2　模运算单元结构

模乘采用模约简结构，即先进行乘法运算，再对乘法结果做模约简。由于 secp256r1 的数据位宽为 256 位，Curve25519 曲线的数据位宽为 255 位，故直接采用单元库中已有的 256 位乘法器，满足 2 条曲线的位宽需求，并在一个周期内完成乘法。模约简需单独设计，secp256r1 曲线的 $p=2^{256}-2^{224}+2^{192}+2^{96}-1$，乘法操作后得到一个 512 位的乘法结果 A，易知 $A<p^2$，用二进制表示后分为 16 个数据段，设 $A=A_{15}2^{224}+A_{14}2^{192}+\cdots+A_12^{32}+A_0$，其中 A_i 的宽度为 32 位，则约简公式如算法 4 所示[18]。

算法 4　secp256r1 的快速模约简公式

输入　$A=(A_{15}\|A_{14}\|A_{13}\|\cdots\|A_2\|A_1\|A_0)$

输出　$B=A \bmod p$

1) $T=(A_7\|A_6\|A_5\|A_4\|A_3\|A_2\|A_1\|A_0)$

2) $S_1=(A_{15}\|A_{14}\|A_{13}\|A_{12}\|A_{11}\|96'h0)$

3) $S_2=(32'h0\|A_{15}\|A_{14}\|A_{13}\|A_{12}\|96'h0)$

4) $S_3=(A_{15}\|A_{14}\|96'h0\|A_{10}\|A_9\|A_8)$

5) $S_4=(A_8\|A_{13}\|A_{15}\|A_{14}\|A_{13}\|A_{11}\|A_{10}\|A_9)$

6) $D_1=(A_{10}\|A_8\|96'h0\|A_{13}\|A_{12}\|A_{11})$

7) $D_2=(A_{11}\|A_9\|64'h0\|A_{15}\|A_{14}\|A_{13}\|A_{12})$

8) $D_3=(A_{12}\|32'h0\|A_{10}\|A_9\|A_8\|A_{15}\|A_{14}\|A_{13})$

9) $D_4=(A_{13}\|32'h0\|A_{11}\|A_{10}\|A_9\|32'h0\|A_{15}\|A_{14})$

10) $B=(T+2S_1+2S_2+S_3+S_4-D_1-D_2-D_3-D_4) \bmod p$

快速约简通过多个数据的加减操作来实现，仅

在最后一步取模运算。需要注意，由于 S_2 的高位为 0，因此最后得到的待约简数据至多有 5 个进位或 5 个借位，比较常见的方式是采用高位为 0 的形式化简以上计算式，进行再次约简，但即使再次模约简，也会出现超过模 p 的情况，依然需要进行判断并计算输出。由于设计包含 2 条曲线的模约简，需要考虑单元间的复用关系。

对于 Curve25519，$p=2^{255}-19=2^{255}-2^4-2^2+1$，位宽为 255 位，一次乘法操作的结果 A 在二进制下位宽最高位 510 位，根据 p 的形式特点，设 $A=A_{254}2^{508}+A_{253}2^{506}+A_{252}2^{504}+\cdots+A_1 2^2+A_0$，$A_i$ 的宽度为 2 位，利用此形式的 A 对 p 做模约简。

模约简过程分为两轮，第一轮处理高 254 位，首先要对 A 的最高项 $A_{254}2^{508}$ 做约简，需利用 $A_{254}2^{253}$ 乘以 p 值，得到多项式 $A_{254}2^{508}-A_{254}2^{257}-A_{254}2^{255}+A_{254}2^{253}$，消去最高项后得到剩余项 $A_{254}2^{257}+A_{254}2^{255}-A_{254}2^{253}$，以此类推，直至用 $A_{128}2^1$ 乘以 p 得到多项式 $A_{128}2^{256}-A_{128}2^5-A_{128}2^3+A_{128}2^1$，消去 $A_{128}2^{256}$ 项后得到剩余项 $A_{254}2^{257}+A_{254}2^{255}-A_{254}2^{253}$。第一轮运算得到的约简结果分为两部分，第一部分是 A 的低 256 位，即 $A_{127}2^{254}+\cdots+A_1 2^2+A_0$，此部分未操作，直接保留设为为 T；第二部分是所有剩余项的和，整理后为 $A_{254}2^{257}+(A_{253}+A_{254})2^{255}+(A_{252}+A_{253}-A_{254})2^{253}+(A_{251}+A_{252}-A_{253})2^{251}+\cdots+(A_{128}+A_{129}-A_{130})2^5+(A_{128}-A_{129})2^3+(-A_{128})2^1$，对第二部分的高两项进行第二轮约简，消去 $A_{254}2^{257}+(A_{253}+A_{254})2^{255}$，得到新增剩余项为 $A_{254}2^6+(A_{253}+2A_{254})2^4+A_{253}2^2-(A_{253}+A_{254})$。至此，两轮约简整理完毕，将各位置的数值按规律重新整理为和项和差项后，可得算法 5 所示的快速约简算法。

算法 5 Curve25519 曲线快速约简算法

输入 $A=(A_{254}\|A_{253}\|A_{252}\|\cdots\|A_2\|A_1\|A_0)$

输出 $B=A \bmod p$

1) $T=(A_{127}\|A_{126}\|A_{125}\|\cdots\|A_2\|A_1\|A_0)$

2) $S_1=(A_{252}\|A_{251}\|A_{250}\|\cdots\|A_{129}\|A_{128}\|5'\text{h}0)$

3) $S_2=(A_{253}\|A_{252}\|A_{251}\|\cdots\|A_{129}\|A_{128}\|3'\text{h}0)$

4) $S_3=(247'\text{h}0\|A_{254}\|A_{254}\|4'\text{h}0)$

5) $S_4=(249'\text{h}0\|A_{253}\|A_{253}\|2'\text{h}0)$

6) $S_5=(249'\text{h}0\|A_{254}\|4'\text{h}0)$

7) $D_1=(A_{254}\|A_{253}\|A_{252}\|\cdots\|A_{129}\|A_{128}\|1'\text{h}0)$

8) $D_2=(253'\text{h}0\|A_{254})$

9) $D_3=(253'\text{h}0\|A_{253})$

10) $B=(T+S_1+S_2+S_3+S_4+S_5-D_1-D_2-D_3) \bmod p$

考虑到高位为 0 的各部分，以及 T 所取的位数为 256 位，最后一步待约简数据会有至多 5 个进位或 2 个借位，与 secp256r1 曲线的情况基本类似，故采用加减法阵列计算出 2 种模约简所有最后可能的输出结果，通过判断加减法的标志位来选择输出正确结果，其结构如图 3 所示。

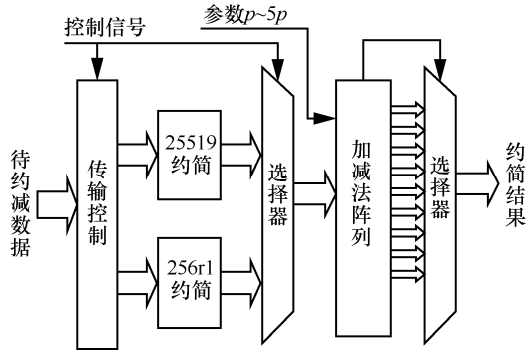


图 3 模约简结构

加减法阵列中需参数 $p\sim5p$ 参与运算，需要在寄存器堆中存放 2 条曲线的 $5p$ 参数，其中，参数 p 在标量乘中多次使用，设置为参数形式存储；$2p$ 在运算开始时通过 p 相加得到；$3p$ 通过 3 倍模加单元计算得到；$4p$ 通过模减单元 $5p-p$ 运算得到，这些值在首周期结束时固定在加减法阵列的输入端，并在次周期开始参与运算，与乘法前后衔接。

模逆运算使用二进制右移算法，其基本算法如算法 6 所示[20]。

算法 6 二进制右移算法求模逆

输入 素数 p 和 $a\in[1,p-1]$

输出 $a^{-1} \bmod p$

1) 令 $u=a$，$v=p$，$x=1$，$y=0$

2) 若 $u\neq1$ 且 $v\neq1$，则执行

① 若 u 为偶数，则 $u=\frac{u}{2}$，x 为偶数，则 $x=\frac{x}{2}$，x 为奇数，则 $x=\frac{x+p}{2}$

② 若 v 为偶数，则 $v=\frac{v}{2}$，y 为偶数，则 $y=\frac{y}{2}$，y 为奇数，则 $y=\frac{y+p}{2}$

③ 若 $u\geqslant v$，则 $u=u-v$，$x=x-$y；否则 $v=v-u$，$y=y-x$

3) 若 $u=1$，返回 $x \bmod p$，否则返回 $y \bmod p$

该算法理论上至多需要 512 个周期即可完成运

算，算法中每次向右移位一次，并需要做除 2 操作，但对于二进制计算很容易实现。如需更快的速度，则可以考虑每次右移 2 位，但需要额外做除 4 操作并需要更多的判断分支，故本文没有采用。考虑到使用标量乘进行椭圆曲线加密时也需要使用模逆运算，且与标量乘操作无时间上的冲突，所以将模逆模块单独引出，设立标志位以供标量乘内部使用或外部调用。

4　验证与实现

完整的标量乘工作流程如图 4 所示。图 4 仅表示了重要的操作步骤，点加 Z 和点加 1 分别使用表 2 和表 3 中的点加流程，临时寄存器在预计算和迭代计算部分都有使用，但图 4 中未全部标出，模约简也未在图中给出。运算时各环节所需要消耗的时钟周期如表 5 所示，其中包含了各工作状态之间转换消耗的周期数。

经综合后所得面积速度与文献[4-7,25-26]进行对比，得到表 6 所示的结果，表 6 中的 AT 是门数与单次运行时间的乘积。

表 5　不同运算消耗周期情况

运算	λG/周期	λP/周期	$\lambda P+\mu G$/周期	Curve25519/周期
预计算	0	24	168	0
迭代（平均）	1 233	3 684	3 684	3 572
迭代（最大）	1 285	4 228	4 228	3 572
模逆（平均）	384	384	384	384
模逆（最大）	512	512	512	510
后处理	7	7	7	3
合计（平均）	1 624	4 099	4 243	3 956
合计（最大）	1 804	4 771	4 915	4 082

最终设计采用 55 nmCMOS 工艺库综合，主频最高可到 625 MHz，此主频下的门数为 $1\ 022\times10^3$ 个，按表 5 中平均消耗周期计算，对于 secp256r1 曲线，计算基点的标量乘单次只需 2.60 μs，每秒可计算 38.5 万次；计算普通点的标量乘需要 6.56 μs，每秒可计算 15.3 万次；对于 Curve25519，计算一次标量乘需要 6.33 μs，每秒可计算 15.8 万次。

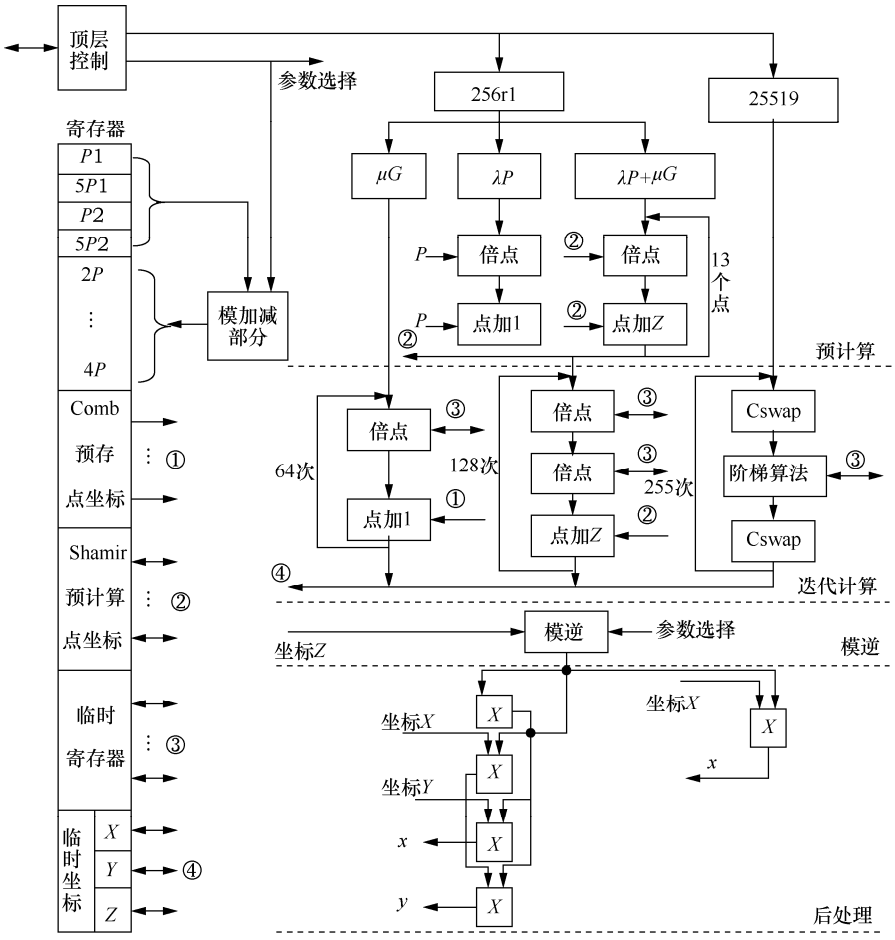


图 4　完整标量乘工作流程

表 6 标量乘对比结果

方案	曲线	工艺/nm	主频/MHz	门数/个	运算速度/（千次·秒$^{-1}$）	单次时间/μs	*AT*
本文工作	secp256r1 (λG)	55	625.00	1 022×10^3	384.8	2.60	2 657
	secp256r1 (λP)				152.5	6.56	6 704
	Curve25519				157.9	6.33	6 469
文献[4]	secp256r1	65	549.45	447×10^3	1.38	0.73	326 310
文献[5]	secp256r1	65	188.00	3.5×10^6	80	12.5	43 750
文献[6]	secp256r1	90	217.00	313×10^3	1.71	0.76	237 880
文献[7]	secp256r1	90	185.00	540×10^3	8.30	120	64 800
文献[25]	secp256r1	90	256.40	82.8×10^3	3.22	0.31	25 668
文献[26]	secp256r1	130	215.00	208×10^3	4.76	0.21	43 680

本文设计消耗的门数是文献[4]的 2.3 倍，最高主频提高了 70 MHz，由于文献[4]采用分段多次迭代的方式实现模乘，而本文设计采用了 256 位的乘法器配合模约简来实现模乘，频率受乘法器限制，故主频并未得到显著提升，但配合 3.2 节中的点加倍点操作，可以实现乘法和模约简的二级流水，标量乘计算过程中每个周期都可以得到模乘结果，所以实际每秒标量乘的运算次数远超文献[4]。其余文献也类似，差异在于模乘分段位数是 32 位、64 位还是 128 位，如文献[5]采用的是 128 位的分段模乘，每秒可计算标量乘 8 万次，是所有文献中每秒运算次数最高的，但消耗的门数是本文设计的 3.4 倍，同时本文设计的每秒运算量可达文献[5]的 1.9 倍。文献[6-7,25-26]的门数消耗小于本文设计，但本文设计的主频和门数均约为文献[6]的 3 倍，实际每秒运算次数远超文献[6]；文献[7]的门数为本文设计的 52%，但主频只能到本文设计的 30%，单次标量乘运行时间较短，但也是本文设计的 18 倍；文献[25]采用较低门数完成设计，且能达到相对较高的主频，*AT* 是所有文献中最小的，但也是本文设计的 3.8 倍；文献[26]也仅在门数一项上低于本文设计，运算速度和 *AT* 与本文设计相比均无优势。以上数据比较发生在普通点的计算上，本文设计在签名时还能提供基点 *G* 的标量乘 38.5 万次，适用于服务器端的高速运算。

5 结束语

本文面向椭圆加密的运算需求设计了标量乘单元，采用快速模约简的方式实现模乘运算，支持 secp256r1 和 Curve25519 曲线的标量乘，并能够在输入基点时得到更快的运算速度以应对椭圆曲线签名和公钥的生成需求，设计同时支持多标量乘以应对椭圆曲线验签使用，将模逆端口单独引出以应对签名和验签操作中的模逆需求，只需调用本单元即可完成加密算法中绝大部分操作。相较于其他设计，本文设计的运算速度有较大优势，能够提供 secp256r1 曲线上每秒 38.5 万次的基点标量乘和每秒 15.3 万次的普通点标量乘，可以实现 Curve25519 每秒 15.8 万次标量乘。

参考文献：

[1] 姜久兴, 厚娇, 黄海, 等. 低面积复杂度 AES 低熵掩码方案的研究[J]. 通信学报, 2019, 40(5): 201-210.
JIANG J X, HOU J, HUANG H, et al. Research on area-efficient low-entropy masking scheme for AES[J]. Journal on Communications, 2019, 40(5): 201-210.

[2] RESCORLA E, MOZILLA. The transport layer security (TLS) protocol version 1.3: RFC8446[S]. IETF, (2018-08) [2020-03-07].

[3] KUDITHI T, SAKTHIVEL R. High-performance ECC processor architecture design for IoT security applications[J]. Journal of Supercomputing, 2019, 75(1): 447-474.

[4] HOSSAIN M S, KONG Y, SAEEDI E, et al. High-performance elliptic curve cryptography processor over NIST prime fields[J]. IET Computers & Digital Techniques, 2017, 11(1): 33-42.

[5] LIU J W, GUAN Z Y, CHENG D X, et al. A high speed VLSI implementation of 256-bit scalar point multiplier for ECC over GF (p)[C]//2018 IEEE International Conference on Intelligence and Safety for Robotics. Piscataway: IEEE Press, 2018: 184-191.

[6] LEE J W, CHUNG S C, CHANG H C, et al. Efficient power-analysis-resistant dual-field elliptic curve cryptographic processor using heterogeneous dual-processing-element architecture[J]. IEEE Transactions on Very Large Scale Integration Systems, 2014, 22(1):49-61.

[7] 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]//IEEE International Symposium on Circuits and Systems. Piscataway: IEEE Press, 2012: 1456-1459.

[8] AL-SOMANI T F. High-performance generic-point parallel scalar multiplication[J]. Arabian Journal for Science and Engineering, 2017, 42(2): 507-512.

[9] JAVEED K, WANG X, SCOTT M. High performance hardware support for elliptic curve cryptography over general prime field[J]. Microprocessors & Microsystems, 2017, 51(6): 331-342.

[10] 王敏, 吴震. 抗 SPA 攻击的椭圆曲线 NAF 标量乘实现算法[J]. 通信学报, 2012, 33(Z1): 228-232.
WANG M, WU Z. Algorithm of NAF scalar multiplication on ECC against SPA[J]. Journal on Communications, 2012, 33(Z1): 228-232.

[11] MOHAMED N, HASHIM M, HUTTER M. Improved fixed-base comb method for fast scalar multiplication[C]//Proceedings of the 5th International Conference on Cryptology. Berlin: Springer, 2012: 342-359.

[12] ZHANG N, CHEN Z, XIAO G. Efficient elliptic curve scalar multiplication algorithms resistant to power analysis[J]. Information Sciences, 2007, 177(10): 2119-2129.

[13] 徐明, 史量. 基于伪四维投射坐标的多基链标量乘法[J]. 通信学报, 2018, 39(5): 74-84.
XU M, SHI L. Pseudo 4D projective coordinate-based multi-base scalar multiplication[J]. Journal on Communications, 2018, 39(5): 74-84.

[14] IEEE Standards Association. IEEE standard specifications for public-key cryptography: IEEE Std 1363-2000[S]. IEEE, (2000-07) [2020-05-06].

[15] LI W, ZENG X Y, FENG X. A high-throughput processor for dual-field elliptic curve cryptography with power analysis resistance[C]//2015 IEEE 15th International Conference on Scalable Computing and Communications and Its Associated Workshops. Piscataway: IEEE Press, 2015: 570-577.

[16] COHEN H, MIYAJI A, ONO T. Efficient elliptic curve exponentiation using mixed coordinates[C]//International Conference on the Theory & Applications of Cryptology & Information Security: Advances in Cryptology. Berlin: Springer, 1998: 51-65.

[17] 王潮, 时向勇, 牛志华. 基于蒙哥马利曲线改进 ECDSA 算法的研究[J]. 通信学报, 2010, 31(1): 9-13.
WANG C, SHI X Y, NIU Z H. The research of the promotion for ECDSA algorithm based on Montgomery-form ECC[J]. Journal on Communications, 2010, 31(1): 9-13.

[18] MARZOUQI H, AL-QUTAYRI M, SALAH K, et al. A 65nm ASIC based 256 NIST prime field ECC processor[C]//2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, 2016.

[19] MARZOUQI H, AL-QUTAYRI M, SALAH K. Review of elliptic curve cryptography processor designs[J]. Microprocessors and Microsystems, 2015, 39(2): 97-112.

[20] HANKERSON D, MENEZES A, VANSTONE S. Guide to elliptic curve cryptography[M]. Berlin: Springer, 2004.

[21] BERNSTEIN D. CURVE25519: new Diffie-Hellman speed records[C]// International Workshop on Public Key Cryptography. Berlin: Springer: 2006:207-228.

[22] LANGLEY A, GOOGLE, Hamburg M, et al. Elliptic curves for Security: RFC7748[S]. IETF, (2016-01) [2020-01-21].

[23] SALARIFARD R, BAYAT-SARMADI S. An efficient low-latency point-multiplication over Curve25519[J]. IEEE Transactions on Circuits and Systems-I: Regular Papers, 2019, 66(10): 3854 -3862.

[24] DÜELL M, HAASE B, HINTERWAELDER G, et al. High-speed curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers[J]. Designs Codes & Cryptography, 2015, 77(2-3): 493-514.

[25] CHEN Y L, LEE J W, LIU P C, et al. A dual-field elliptic curve cryptographic processor with a radix-4 unified division unit[C]// IEEE International Symposium of Circuits and Systems. Piscataway: IEEE, 2011: 713-716.

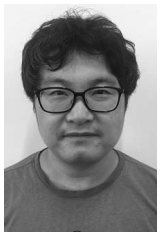
[26] RASHIDI B. A survey on hardware implementations of elliptic curve cryptosystems[J]. arXiv Preprint, arXiv: 1710.08336, 2017.

[作者简介]

于斌（1984− ），男，黑龙江饶河人，哈尔滨理工大学讲师，主要研究方向为密码算法、密码芯片设计、数字集成电路设计等。

黄海（1982− ），男，内蒙古巴彦淖尔人，博士，哈尔滨理工大学副教授、硕士生导师，主要研究方向为信息安全、可重构技术、集成电路设计等。

刘志伟（1987− ），男，黑龙江哈尔滨人，哈尔滨理工大学讲师、博士生，主要研究方向为可重构计算、高速密码算法、并行加密技术、密码芯片的安全设计等。



赵石磊（1979− ），男，黑龙江肇源人，博士，哈尔滨理工大学副教授、硕士生导师，主要研究方向为信息安全、高速密码算法、密码芯片的安全设计等。

那宁（1995− ），男，黑龙江牡丹江人，哈尔滨理工大学硕士生，主要研究方向为信息安全、集成电路设计等。