高速 Ed25519 验签算法硬件架构的设计与实现

薛一鸣[1]，刘树荣[1]，郭书恒[1]，李岩[2]，胡彩娥[3]

（1. 中国农业大学信息与电气工程学院，北京 100083；2. 中国农业大学理学院，北京 100083；
3. 国网北京市电力公司，北京 100031）

摘 要：针对区块链等特定场景对验签速度有较高要求的特点，设计了一种高速 Ed25519 验签算法的硬件架构。提出了基于交错 NAF 的多点乘算法，通过预计算和查表的方式，有效减少了点加、倍点的次数；采用 Karatsuba 乘法和快速约简方法实现模乘运算，并设计了不需要模加、模减的点加、倍点操作步骤，有效提升了点加、倍点运算的性能。针对解压过程中耗时的模幂运算，设计了模逆和模乘并行的模幂计算方法，提高了解压运算的性能。整个设计充分考虑了资源的复用，在 Zynq-7020 平台上实现需要 13 695 个 Slices，在 81.61 MHz 的时钟频率下，每秒能够完成 8 347 次验签运算。

关键词：爱德华曲线；数字签名；多点乘；硬件实现

中图分类号：TN918

文献标志码：A

DOI: 10.11959/j.issn.1000-436x.2022061

High-speed hardware architecture design and implementation of Ed25519 signature verification algorithm

XUE Yiming[1], LIU Shurong[1], GUO Shuheng[1], LI Yan[2], HU Cai'e[3]

1. College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China
2. College of Science, China Agricultural University, Beijing 100083, China
3. State Grid Beijing Electric Power Company, Beijing 100031, China

Abstract: Aiming at the high performance requirements of signature verification for specific scenarios such as blockchain, a high-speed hardware architecture of Ed25519 was proposed. To reduce the number of calculations for point addition and point double, a multiple point multiplication algorithm based on interleaving NAF was conducted by using pre-computation and lookup tables. The modular multiplication operation was realized by using the Karatsuba multiplication and fast reduction method, and the point addition and point double operation was designed without modular addition and subtraction, which could effectively improve the performance of point addition and point double. Given that modular exponentiation was the most time-consuming operation in the decompression process, a new modular exponentiation approach was developed by parallelizing modular inverse and modular multiplication, and therefore the performance of the de-compression operation could be improved. The proposed architecture fully considers the use of resources and is implemented on the Zynq-7020 FPGA platform with 13 695 slices, achieving 8 347 verifications per second at 81.6 MHz.

Keywords: Edwards-curve, digital signature, multiple point multiplication, hardware implementation

0 引言

随着区块链、云计算、车联网等技术的快速发展，全球每天都有海量的数据产生，如何保证数据的安全性受到了学术界、产业界等的高度关注。数字签名可以验证数据源的真实性和数据内容的完整性，是保障数据安全的核心技术之一，目前使用最广泛的算法为基于椭圆曲线密码体制[1-2]（ECC,

收稿日期：2022−01−05；修回日期：2022−03−09

基金项目：国家自然科学基金资助项目（No.61872368）；国家重点研发计划基金资助项目（No.2021QY2312）

Foundation Items: The National Natural Science Foundation of China (No.61872368), The National Key Research and Development Program of China (No.2021QY2312)

elliptic curve cryptography）构建的数字签名算法[3]。2012 年，Bernstein 等[4-5]设计了爱德华曲线 Edwards25519，并根据 Schnorr 算法在该曲线上构建了 Ed25519 数字签名算法。与其他椭圆曲线构建的数字签名算法相比，Ed25519 设计结构简单、采用的参数完全公开，凭借着高性能、高安全性且安全性易于证明的特点[6]，该算法受到了广泛的研究。目前 Ed25519 数字签名算法已被大多数公有链的密码货币项目采用[7]，并被收录在传输层安全协议 TLS 1.3[8]中。

与 Ed25519 签名算法相比，其验签算法运算复杂度更高、需要的存储需求更大[9]，因此 Ed25519 验签算法的高性能实现逐渐成为研究重点。Faz-Hernández 等[10]使用矢量指令的方式在计算机上实现了高速 Ed25519 验签功能；文献[11-12]分别在 STM32F401 和 ESP32 平台上实现了曲线 Edwards25519 上的点乘运算；文献[13-15]通过对模乘算法和点乘算法进行优化在现场可编辑门阵列（FPGA, field programmable gate array）上设计了点乘运算加速结构；Turan 等[16]在 FPGA 上使用 DSP 单元实现了快速模乘运算，设计了能够实现 Ed25519 签名和验签功能的点乘运算结构，但是偏重于轻量级设计，更适合使用于对运算速度要求较低的嵌入式设备；于斌等[17]在 TSMC 的 55 nm 工艺下设计了一种高性能 Ed25519 算法的硬件实现架构，根据 Ed25519 模数的特点设计了快速模约简电路并使用 257 位的乘法器来实现模乘运算，设计了兼容签名和验签功能的点乘结构，但是整体面积较大；Bisheh-niasar 等[18]在 FPGA 上实现了高性能和轻量级的 Ed25519 验签算法硬件实现架构，使用 Shamir 方法来实现点乘运算，提高了验签的性能，但是使用 Shamir 方法存在窗口宽度不能过大的限制；文献[19-20]讨论了使用多基数表示方法实现素数域椭圆曲线点乘运算的复杂度和实现效率，但是并未完成具体的硬件实现。

本文提出了一种高速 Ed25519 验签算法的硬件架构。首先针对 Ed25519 验签算法中耗时的点乘运算进行优化，提出了基于交错非相邻形式（NAF, non-adjacent form）的多点乘算法；其次设计了可供解压运算、模 L 约简、坐标转换和多点乘运算复用的算术逻辑单元（ALU, arithmetic and logic unit），针对解压过程中耗时的模幂运算提出了模逆和模乘并行的计算方法，针对模 L 约简提出了运算周期固定的快速约简方法；最后基于 Zynq-7020 平台进行设计和实现，最高工作频率为 81.61 MHz，平均每秒完成 8 347 次验签运算，相比于文献[18]提高了 63.28%。

1 基础知识

Edwards25519 曲线的方程如式(1)所示，其中 $\mathrm{d}=\frac{121\ 665}{121\ 666}$。

$$-x^2+y^2=1-\mathrm{d}x^2y^2 \tag{1}$$

文献[5]给出了 Ed25519 算法的签名和验签流程，如算法 1 和算法 2 所示。其中 G 为曲线 Edwards25519 的基点，R' 和 A 为曲线上的动点，L 为 253 位的素数 $2^{252}+27\,742\,317\,777\,372\,353\,535\,851\,937\,790\,883\,648\,493$，$\mathrm{H}(x)$ 为整数 x 的 SHA-512 运算结果，R 和 pk 分别为点 R' 和点 A 的 256 位压缩结果，压缩过程为 $R=R'_y+(R'_x\ \&\ 1)$，$\mathrm{pk}=A_y+(A_x\ \&\ 1)$。

算法 1 Ed25519 数字签名算法的签名流程

输入 256 位的公钥 pk，任意长度的消息 M，256 位的私钥 sk

输出 512 位的签名结果(R,S)

1) 对 sk 做 SHA-512 运算，$\mathrm{H}(\mathrm{sk})=(h_{511},h_{510},\cdots,h_1,h_0)_2$

2) 取 H(sk) 的高 256 位，$h=(h_{511},h_{510},\cdots,h_{257},h_{256})_2$

3) $a=2^{254}+\sum_{i=3}^{253}2^i h_i$

4) $r=\mathrm{H}(h,M)\bmod L$

5) $R'=rG=(R'_x,R'_y)$，压缩点 R' 得到 $R=R'_y+(R'_x\ \&\ 1)$

6) $k=\mathrm{H}(R,\mathrm{pk},M)\bmod L$

7) $S=(r+ka)\bmod L$

8)返回签名结果(R,S)

算法 2 Ed25519 数字签名算法的验签流程

输入 256 位的公钥 pk，任意长度的消息 M，512 位的签名结果(R,S)

输出 验签的结果

1) 若 $R,S\notin[1,L-1]$，则验证失败，结束验证流程

2) 解压 R 得到点 R'

3) 解压 pk 得到点 A

4) $k=\mathrm{H}(R,\mathrm{pk},M)\bmod L$

5) 验证 $SG=R'+kA$，若等号成立，则验证成功

Ed25519 中的点加、倍点运算需要在拓展四元

齐次坐标下完成，操作步骤如算法 3 和算法 4 所示。其中仿射坐标 (x,y) 与四元齐次坐标 (X,Y,Z,T) 之间的转换关系为 $X=xZ$，$Y=yZ$，$T=xyZ$，Z 的初始值取 1。式(2)中 SG 和 $R'+kA$ 的运算结果比较需要在仿射坐标下完成，通过计算 $x=XZ^{-1}$、$y=YZ^{-1}$ 完成坐标的转换，其中计算 Z^{-1} 的过程需要一次模逆运算。验签过程中，通过对比式(2)中 SG 和 $R'+kA$ 的运算结果来判断签名的真实性。

$$SG=R'+kA \tag{2}$$

其中，SG 和 kA 被称为点乘运算或标量乘运算，通过调用连续的点加（PA, point addition）、倍点（PD, point double）运算实现。

算法3　Ed25519 数字签名算法中的点加操作步骤

输入　$P(X_1,Y_1,Z_1,T_1)$，$Q(X_2,Y_2,Z_2,T_2)$

输出　$Q(X_3,Y_3,Z_3,T_3)=P+Q$

1) $A=(Y_1-X_1)(Y_2-X_2)$
2) $B=(Y_1+X_1)(Y_2+X_2)$
3) $C=2\mathrm{d}T_1T_2$
4) $D=2Z_1Z_2$
5) $E=B-A$
6) $F=D-C$
7) $G=D+C$
8) $H=B+A$
9) $X_3=EF$
10) $Y_3=GH$
11) $T_3=EH$
12) $Z_3=FG$

算法4　Ed25519 数字签名算法中的倍点操作步骤

输入　$P(X_1,Y_1,Z_1,T_1)$

输出　$Q(X_2,Y_2,Z_2,T_2)=2P$

1) $A=X_1^2$
2) $B=Y_1^2$
3) $C=2Z_1^2$
4) $H=B+A$
5) $E=H-(X_1+Y_1)^2$
6) $G=A-B$
7) $F=C+G$
8) $X_3=EF$
9) $Y_3=GH$
10) $T_3=EH$
11) $Z_3=FG$

2　Ed25519 验签算法优化及流程优化

2.1　相关研究

点乘是 Ed25519 数字签名验签算法中的关键运算，其性能的优劣直接决定了验签算法的效率。式(2)需要先计算点乘 SG、kA，再通过对比 SG 和 $R'+kA$ 的结果判断验签是否成功，运算效率较低。

文献[17]针对该问题设计了式(3)所示的点乘运算单元

$$Q=mP_1+nP_2 \tag{3}$$

其中，P_1 和 P_2 为椭圆曲线上的点，m 为 255 位的标量，$n\in[0,1]$。n=0 时进行点乘 SG 的计算，n=1 时进行点乘 $R'+kA$ 的计算，通过对比两次点乘运算的结果即可判断验签是否成功，在点乘算法的选取上采用了窗口宽度 $w=2$ 的窗口方法，通过预计算和查表的方式降低了运算量。相比于二进制展开的点乘计算方法[21]，文献[21]使用的方法将点乘循环中的点加运算次数减少了 25%，增大窗口宽度可以进一步减少所需的点加运算次数，但是也会导致预计算的运算量和存储需求呈 2^w 的指数级增长。

2.2　基于交错 NAF 的多点乘算法

为进一步提高 Ed25519 验签性能，本文将式(2)进行变形，得到式(4)。基于式(4)，可以将点乘和点加运算采用多点乘的方式实现。

$$R'=SG-kA \tag{4}$$

相较于常规的点乘计算方法，使用多点乘可以减少点乘循环中 50%的倍点运算次数，通过选取合适的多点乘算法可以进一步减少点乘循环中点加的运算次数。常用的多点乘计算方法有 Shamir 方法和交错的 NAF 方法[21]，其中 Shamir 方法对标量 S 和 k 只能选用相同的窗口宽度，并且预计算过程中需要同时进行基点 G 和动点 A 的计算，随着窗口宽度的增大，预计算的运算量和存储需求呈 2^{2w} 的指数级增长。在 Ed25519 验签的过程中，基点 G 的预计算可以在系统初始化时完成，在存储资源满足设计要求的情况下可以选用较大的窗口宽度；而动点 A 的预计算会影响多点乘的性能，需要选用较小的窗口宽度。因此本文提出了一种基于交错 NAF 的多点乘实现算法，即对基点 G 和动点 A 的预计算选用不同的窗口宽度，具体算法流程如算法 5 所示。

其中，步骤 1)中基点 G 的预计算和 S 的 NAF 编码都是在多点乘运算之前完成的，动点 A 的预计算过程需要一个倍点和 $2^{w_2-2}-1$ 个点加运算，完成预计算后再对 k 进行 NAF 编码；步骤 6)的循环中需要计算 $Q=Q-|S_i|G$ 和 $Q=Q-k_iA$，预计算过程中只预先计算了 S_iG 和 k_iA，因此需要两次模减运算才能得到 $-|S_i|G$ 和 $-k_iA$ 的坐标。

算法 5 窗口宽度为 w_1, w_2 的交错 NAF 方法实现的多点乘算法

输入 253 位二进制数 $S=(S_{252},S_{251},\cdots,S_1,S_0)_2$，$k=(k_{252},k_{251},\cdots,k_1,k_0)_2$，基点 $G=(x_1,y_1)$，动点 $A=(x_2,y_2)$

输出 多点乘 $Q=SG-kA=(x_3,y_3)$

1) 构造基点 G 的预计算表 $\{3G,5G,\cdots,(2^{w_1-1}-3)G,(2^{w_1-1}-1)G\}$，对 S 进行 NAF 编码，得到 $S=(S_{253},S_{252},\cdots,S_1,S_0)_{2^{w_1-1}}$，其中 $S_i\in[-2^{w_1-1}+1,2^{w_1-1}-1]$

2) 构造动点 A 的预计算表 $\{3A,5A,\cdots,(2^{w_1-1}-3)A,(2^{w_1-1}-1)A\}$

3) 对 k 进行 NAF 编码，得到 $k=(k_{253},k_{252},\cdots,k_1,k_0)_{2^{w_2-1}}$，其中 $k_i\in[-2^{w_2-1}+1,2^{w_2-1}-1]$

4) $l=\max\_\text{length}\{S,k\}$，

若 $l=\text{length}\{S\}$，则 $k_i=0$，$i\in[\text{length}(k),l-1]$

否则 $S_i=0$，$i\in[\text{length}(S),l-1]$

5) 若 $S_{l-1}=0$，则 $Q=-k_{l-1}A$

否则，$Q=S_{l-1}G$

若 $k_{l-1}\neq 0$，则

若 $k_{l-1}>0$，则 $Q=Q-k_{l-1}A$

否则，$Q=Q+|k_{l-1}|A$

6) 对于 i 从 l–2 到 0，循环计算

$Q=2Q$

若 $S_i\neq 0$，则

若 $S_i>0$，则 $Q=Q+S_iG$

否则，$Q=Q-|S_i|G$

若 $k_i\neq 0$，则

若 $k_i>0$，则 $Q=Q-k_iA$

否则，$Q=Q+|k_i|A$

7) 返回 Q

2.3 验签流程优化

在多点乘算法研究的基础上，考虑到解压和 SHA-512 运算的性能也会影响 Ed25519 签名验证算法的效率，本文设计了如算法 6 所示的高速 Ed25519 验签算法流程；其中步骤 2 的解压、SHA-512、NAF 编码运算可以并行操作；步骤 4 的多点乘运算完成之后需要进行坐标转换，坐标转换过程中的模逆运算可以和解压过程中的乘法运算并行操作。

算法 6 高速 Ed25519 验签算法流程

输入 256 位的公钥 pk，任意长度的消息 M，512 位的签名数据(R, S)

输出 签名验证的结果

1) 若 $R,S\notin[1,L-1]$，则验证失败，结束验证流程

2) $k=\text{H}(R,\text{pk},M)\bmod L$，解压 pk 为点坐标 A，对 S 进行 NAF 编码

3) 对 k 进行 NAF 编码

4) 计算多点乘 $Q=SG-kA$，解压 R 为点坐标 R'

5) 验证 $Q=R'$ 是否成立，若成立，则验证成功

3 硬件架构设计及实现

根据算法 6 所提出的高速 Ed25519 验签算法流程，构建了如图 1 所示的高速 Ed25519 验签硬件架构。该架构由 Ed25519 验签状态机、控制单元、ALU 模块、SHA-512 哈希模块、寄存器、寄存器堆、模逆模块等组成，通过 AXI 总线实现与外部系统的交互。

Ed25519 验签状态机用于验签状态的控制，控制单元由多点乘、解压控制、模 L 约简、坐标转换 4 个子控制单元组成，ALU 模块包括乘法器、模 p 约简单元、加法器和减法器。控制单元只对数据的输入输出选择进行控制，最终的数据运算是通过调用 ALU 和模逆等运算单元实现的。寄存器堆用于存储 NAF 编码结果及预计算表；寄存器除了用于存放点加、倍点、坐标转换单元的输入、输出坐标值，还用于暂存 SHA-512 的运算结果和模 L 约简运算过程产生的计算中间值。

关于模逆和 SHA-512 的设计已经相对成熟，故不作为本文设计的重点。常用的素数域求逆方法有费马小定理和拓展欧几里得算法[21]，基于费马小定理实现的模逆算法需要的运算量较大[15,17]，且求逆的过程中无法释放乘法器和模 p 约简单元，本文采用拓展欧几里得算法来设计模逆模块。SHA-512 模块的设计采用了文献[18]的方法，一次运算需要 80 个时钟周期。

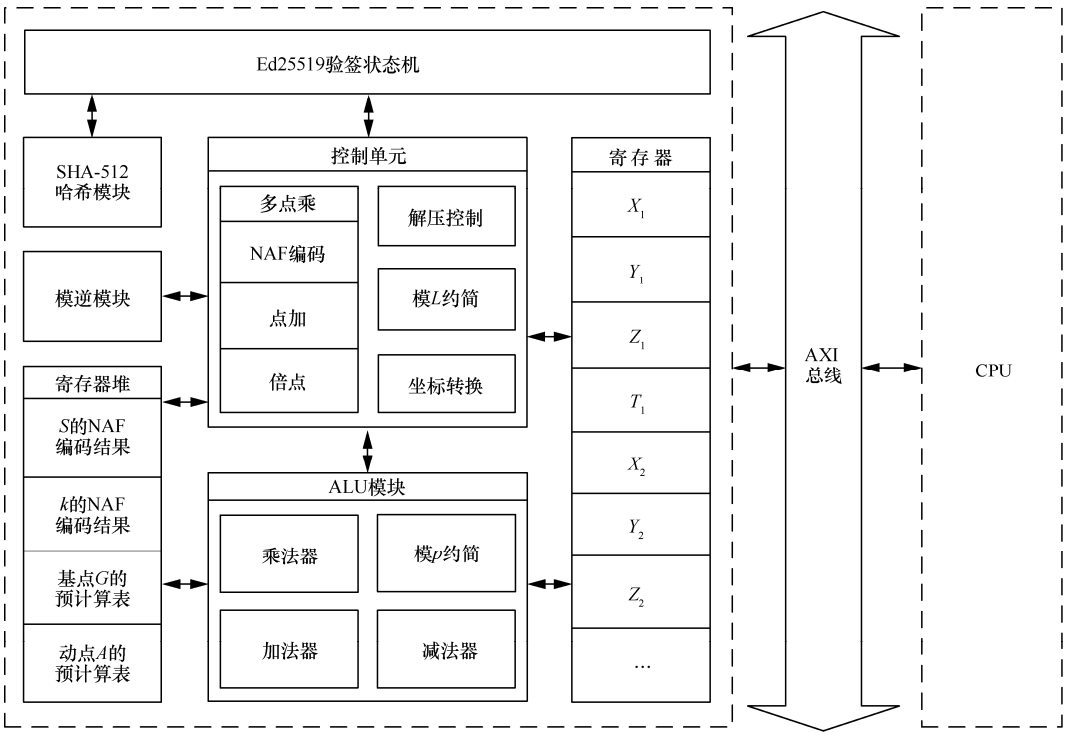

图 1　高速 Ed25519 验签硬件架构

3.1　验签状态机设计

高速 Ed25519 验签状态机的状态及跳转流程如图 2 所示。系统上电之后首先进行初始化（INIT），完成基点 G 的预计算，之后进入 IDLE 状态。接收到外部系统发出的验签请求后，跳转到 LOAD 状态，将接收到的公钥数据 pk、消息 M 等数据存入相应的寄存器单元，判断数据的合法性并跳转到 IDLE 状态，若签名数据不合法，则拒绝验签请求。开始验签流程后跳转到 SHADEC 状态，在该状态下 SHA-512 运算、pk 的解压运算、S 的 NAF 编码操作并行执行，若 pk 解压成功，则跳转到 MODL 状态。之后对 SHA-512 的输出结果进行模 L 约简并

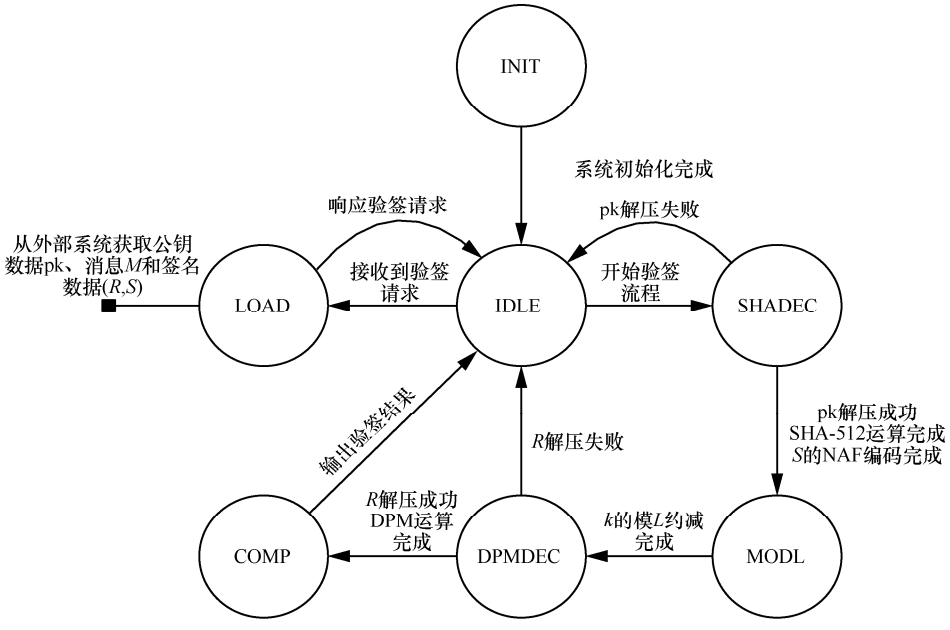

图 2　高速 Ed25519 验签状态机的状态及跳转过程

跳转到 DPMDEC 状态，在该状态下，先完成多点乘运算，再并行执行坐标转换和 R 的解压运算，若解压成功则跳转到 COMP 状态，对 R' 和多点乘的计算结果 Q 进行比较，若 2 个点的坐标相同，则验签成功，否则验签失败，最后跳转回 IDLE 状态。

3.2 算术逻辑单元设计

算术逻辑单元主要完成大数乘、模 p 约简、大数加减等素数域运算，是最基础的运算单元。解压运算、多点乘和坐标转换过程中的模乘运算通过调用乘法器和模 p 约简单元实现，模 L 约简运算的实现也需要使用乘法器。基于 FPGA 的大数乘法实现常用的方法有 Karatsuba 算法和 Toom-Cook 算法，其中 Karatsuba 算法的性能与实现面积的乘积要低于 Toom-Cook 算法[22]。若 $2m$ 位的乘法器使用 Karatsuba 算法实现，则表达式为

$$\begin{aligned} C = AB = (a_1\varphi + a_0)(b_1\varphi + b_0) = \\ a_1b_1\varphi^2 + a_0b_0 + ((a_1 + a_0)(b_1 + b_0) - \\ a_1b_1 - a_0b_0)\varphi = H_o\varphi^2 + L_o + (M_o - H_o - L_o)\varphi \end{aligned} \tag{5}$$

其中，$\varphi = 2^m$。该算法只需要 2 个 m 位乘法器和一个 m+1 位乘法器即可实现 $2m$ 位的乘法运算，减少了乘法器的资源开销。本文采用插入流水线的方式使用 81 个 DSP 单元设计了 3 个 130×130 的 Karatsuba 乘法器，再使用 130×130 的部分积构建出 258×258 的乘法计算结果，其中 130×130 的乘法运算需要 4 个时钟周期。

在 Ed25519 中，模数为 $p = 2^{255} - 19$，利用 $2^{258} (\mathrm{mod}\ p) \equiv 152 (\mathrm{mod}\ p)$、$2^{255} (\mathrm{mod}\ p) \equiv 19 (\mathrm{mod}\ p)$ 的特点，使用算法 7 对大数乘法的结果进行约简。快速约简的过程直接使用 130×130 的乘法结果作为输入，第 1 轮约简中 C_h 直接取乘法器的输出，通过计算 $152C_h + C_l$ 将 C 约简为一个 392 位的数；第 2 轮约简中取 T 的高 137 位作为 T_h，取低 255 位作为 T_l，计算 $19T_h + T_l$ 将 T 约简为一个 256 位的数；第 2 轮约简结果的取值范围为 $[0, 2^{256} - 1]$，最多只需要一次减 p 运算即可得到最终的约简结果。乘法器单元和模 p 约简单元的结构如图 3 所示。

算法 7 模 p 快速约简算法

输入 130×130 的乘法结果 H_o、L_o、M_o，模数 $p = 2^{255} - 19$

输出 $T'' = C (\mathrm{mod}\ p)$

根据 Karatsuba 算法，$C = 2^{258} H_o + 2^{129} (M_o - H_o - L_o) + L_o$

1) $C_h = H_o, C_l = 2^{129} (M_o - H_o - L_o) + L_o$

2) 第 1 轮约简：$T = 152C_h + C_l$

3) $T_h = (T_{391}, T_{390}, \cdots, T_{256}, T_{255})_2$，$T_l = (T_{254}, T_{253}, \cdots, T_1, T_0)_2$

4) 第 2 轮约简：$T' = 19T_h + T_l$

5) 第 3 轮约简：$T'' = T' - p$

6) 若 $T'' \leqslant 0$，则 $T'' = T'$

7) 返回 T''

根据算法 3 和算法 4 中的点加、倍点操作步骤，加法和减法运算的结果将作为乘法器的输入数据源，由于乘法器的位宽可以满足 258 位的数据输入，因此点加、倍点过程中加法单元、减法单元的运算

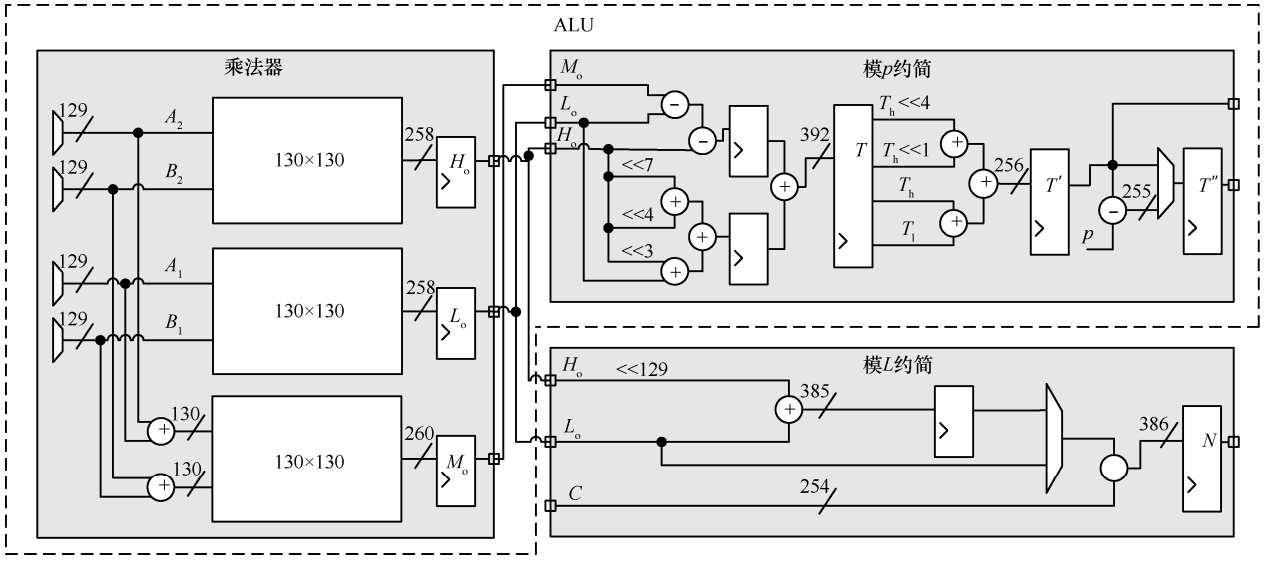

图 3 乘法器和模约简单元结构

结果不需要约简，可以直接作为乘法器的输入；模 p 约简单元的第 2 轮约简结果也可以直接作为加法单元、减法单元和乘法器的输入，第 3 轮约简结果只有在多点乘中的最后一次模乘运算才需要输出。此外，乘法器的第一级输入端并未插入寄存器，可以和加、减法单元共用一个时钟周期。乘法器和模 p 约简单元均采用了插入流水线的设计方式，因此连续的模乘运算完全流水只需要一个周期。

3.3　多点乘单元设计

对二进制数 S 和 k 进行 NAF 编码[21]可以减少多点乘循环的点加运算次数，但是窗口宽度的增大会导致预计算的运算量增加，如表 1 所示。不同窗口宽度下多点乘中倍点运算的次数均相同，因此可以将点加运算的次数作为衡量多点乘单元性能优劣的指标，在 w_1 固定的情况下，w_2 的增大可以减少多点乘循环中的点加运算次数，但是预计算过程中的点加次数会增加，总的点加运算次数呈增加的趋势；在 w_2 固定的情况下，点加运算的次数随窗口宽度 w_1 的增大而减少，但是减少的趋势逐渐减缓，并且 w_1 的增大会导致存储需求呈指数级增长，因此选用 $w_1=10$ 、 $w_2=5$ 作为多点乘单元的窗口宽度。

基于 ALU 结构设计了不需要模加、模减运算的点加和倍点的操作步骤，分别如表 2 和表 3 所示，完整的点加和倍点运算分别需要 24 个和 20 个时钟周期。表 2 中，输入 $P(X_1,Y_1,Z_1,T_1)$，$Q(X_2,Y_2,Z_2,T_2)$，输出 $Q(X_3,Y_3,Z_3,T_3)=P+Q$；表 3 中，输入 $P(X_1,Y_1,Z_1,T_1)$，输出 $Q(X_2,Y_2,Z_2,T_2)=2P$。多点乘循环的过程中上一个点加、倍点运算的输出需要作为下一个运算的输入，由于乘法器和模 p 约简单元采用了流水线设计，并且 DSP 单元只有在乘法运算的第一个周期使用到，连续的点加、倍点运算之间可以共用部分时钟周期，即点加−点加、点加−倍点、倍点−点加和倍点−倍点运算分别需要 46 个、42 个、41 个和 38 个时钟周期。

多点乘运算的结果需要从拓展四元齐次坐标转换为仿射坐标，通过调用一次模逆和两次模乘运算即可实现，坐标转换的结果被存入寄存器中。

3.4　高速解压单元设计

解压运算使用 x 坐标值的最低有效位和 y 坐标值来计算出完整的 x 坐标值，计算式为

$$x^2=\frac{y^2-1}{\mathrm{d}y^2+1}=\frac{u}{v}(\mathrm{mod}\ p) \tag{6}$$

文献[5]给出了 x 坐标的计算方法，其中模幂运算是该方法中的关键运算，计算式为

$$t=(uv^7)^{2^{252}-3}=a^{2^{252}-3} \tag{7}$$

文献[17]通过计算连续的 503 个模乘完成模幂运算，虽然只使用了模乘电路，但是运算量较大。

表 1　　**交错 NAF 方法实现的多点乘运算期望性能**

w_1	w_2	多点乘循环运算量/次		预计算运算量/次		总运算量/次	
		点加	倍点	点加	倍点	点加	倍点
7	4	82	253	3	1	85	254
7	5	73	253	7	1	80	254
7	6	67	253	15	1	82	254
8	5	70	253	7	1	77	254
8	6	64	253	15	1	79	254
9	5	67	253	7	1	74	254
9	6	61	253	15	1	76	254
9	7	56	253	31	1	87	254
10	5	65	253	7	1	72	254
10	6	59	253	15	1	74	254
10	7	54	253	31	1	85	254
10	8	51	253	63	1	114	254
11	5	63	253	7	1	70	254

表 2 **Ed25519 验签算法中点加算法重排布**

周期	大数乘				模 p 约简				加法	减法
	Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅰ	Ⅱ	Ⅲ	Ⅳ		
1	$t_1 = T_1\mathrm{d}$								$t_0 = T_2 + T_2$	$A_2 = Y_2 - X_2$
2	$A = A_1A_2$	t_1							$B_2 = Y_2 + X_2$	$A_1 = Y_1 - X_1$
3	$B = B_1B_2$	A	t_1						$B_1 = Y_1 + X_1$	
4	$t_2 = Z_1Z_2$	B	A	t_1						
…										
7				t_2	B	A	t_1			
8	$C = t_0t_1$				t_2	B	A	t_1		
9		C				t_2	B	A		
10	$T_3 = EH$		C				t_2	B	$H = B + A$	$E = B - A$
11		T_3		C				t_2		
12			T_3		C				$D = t_2 + t_2$	
⋮										
15	$X_3 = EF$					T_3		C	$G = D + C$	$F = D - C$
16	$Z_3 = FG$	X_3					T_3			
17	$Y_3 = GH$	Z_3	X_3					T_3		
⋮										
22						Y_3	Z_3	X_3		
23							Y_3	Z_3		
24								Y_3		

本文提出了模乘和模逆并行的模幂计算方法，设计了如图 4 所示的运算结构，只需要一次模逆和 254 次模乘即可完成模幂运算。由于模逆的实现不依赖于模乘，解压过程中模逆运算完成后模逆模块就可以被坐标转换单元调用。

3.5 模 L 约简单元设计

模 L 约简运算中的模数 L 可以整理为 $L = 2^{252} + L_0$ 的形式，其中 L_0 是一个 125 位的数。模数 L 与模数 p 具有相似的结构，可以使用算法 8 所示的快速约简方法。

第 1 轮约简中 C_h 为输入数据 C 的高 258 位，C_l 为低 254 位，根据 $2^{254} (\mathrm{mod}\, L) \equiv -2^2 L_0 (\mathrm{mod}\, L)$ 的特点，通过计算 $C_\mathrm{l} - 2^2 L_0 C_\mathrm{h}$ 将 C 约简为一个 386 位的数，需要一个 127×258 的乘法器；第 2 轮约简中，取 C 的高 134 位为 C_h，低 252 位为 C_l，由于 $2^{252} (\mathrm{mod}\, L) \equiv -L_0 (\mathrm{mod}\, L)$，通过计算 $C_\mathrm{l} - L_0 C_\mathrm{h}$ 将 C 约简为一个 260 位的数，需要一个 125×134 的乘法器；第 3 轮约简中取 C 的高 8 位为 C_h，低 252 位为 C_l，通过计算 $C_\mathrm{l} - L_0 C_\mathrm{h}$ 将 C 约简为一个 253 位的数，需要一个 125×8 的乘法器。这 3 种乘法运算都通过复用 ALU 中的乘法单元实现，运算结果需要与 C_l 进行减法运算，为了复用减法器和寄存器设计了如图 3 所示的模 L 约简结构；使用 130×130 的部分积 H、L 构建出 127×258、125×134 和 125×8 的乘法计算结果，减法运算过程中的参数 C 从寄存器中输入。前 2 轮约简各需要 7 个时钟周期，其中一个周期用于将约简结果写入寄存器，第 3 轮约简需要 5 个时钟周期，完整的模 L 约简需要 19 个时钟周期，约简的结果被存入寄存器中。

表 3　　**Ed25519 验签算法中倍点算法重排布**

周期	大数乘				模 p 约简				加法	减法
	Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅰ	Ⅱ	Ⅲ	Ⅳ		
1	$A=X_1X_1$									
2	$B=Y_1Y_1$	A							$t_2=X_1+Y_1$	
3	$t_1=Z_1Z_1$	B	A							
4	$t_2=t_2t_2$	t_1	B	A						
…										
7				t_2	t_1	B	A			
8					t_2	t_1	B	A		
9	$Y_2=GH$					t_2	t_1	B	$H=A+B$	$G=A-B$
10		Y_2					t_2	t_1	$C=t_1+t_1$	
11	$T_2=EH$		Y_2					t_2	$F=G+C$	$E=H-t_2$
12	$X_2=EF$	T_2		Y_2						
13	$Z_2=FG$	X_2	T_2		Y_2					
14		Z_2	X_2	T_2		Y_2				
15			Z_2	X_2	T_2		Y_2			
16				Z_2	X_2	T_2		Y_2		
17					Z_2	X_2	T_2			
18						Z_2	X_2	T_2		
19							Z_2	X_2		
20								Z_2		

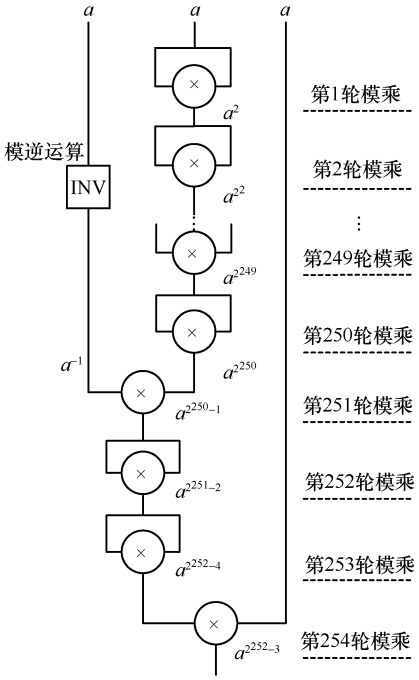

图 4　高速解压运算操作流程

算法 8　模 L 快速约简算法

输入　512 位的二进制数 $C=(C_{511},C_{510},\cdots,C_1,C_0)_2$，模数 $L=2^{252}+L_0$

输出　$C'=C(\mathrm{mod}\,L)$

1) $C_\mathrm{h}=(C_{511},C_{510},\cdots,C_{256},C_{254})_2$，$C_\mathrm{l}=(C_{253},C_{252},\cdots,C_1,C_0)_2$

2) 第 1 轮约简：$t=2^2L_0$，$C=-tC_\mathrm{h}+C_\mathrm{l}$

3) $C_\mathrm{h}=(C_{385},C_{384},\cdots,C_{253},C_{252})_2$，$C_\mathrm{l}=(C_{251},C_{250},\cdots,C_1,C_0)_2$

4) 第 2 轮约简：$t=L_0$，$C=-tC_\mathrm{h}+C_\mathrm{l}$

5) $C_\mathrm{h}=(C_{259},C_{258},\cdots,C_{253},C_{252})_2$，$C_\mathrm{l}=(C_{251},C_{250},\cdots,C_1,C_0)_2$

6) 第 3 轮约简：$t=L_0$，$C=-tC_\mathrm{h}+C_\mathrm{l}$

7) 返回 C

4　实现结果与性能分析

最终的高速 Ed25519 验签算法硬件结构采用了

Xilinx 的 Zynq-7020 进行实现，设计的实现和功能验证使用了 Vivado 2019.1。本文提出的高速 Ed25519 验签硬件架构共需要 22 436 个 LUT、12 631 个 FF 和 81 个 DSP 单元，未消耗块 RAM（BRAM, block RAM），在 81.61 MHz 的时钟频率下，每秒能够完成 8 347 次验签运算。

各模块的资源使用及运行周期数分别如表 4 和表 5 所示。其中模 p 约简单元只有在进行多点乘循环中的最后一次点加或倍点运算时才需要 4 个时钟周期，其余情况下只需要 3 个时钟周期；完整的点加和倍点运算分别需要 24 和 20 个时钟周期，多点乘循环过程中连续的点加、倍点运算之间可以共用部分周期，不需要占用完整的运算周期；模逆、坐标转换和多点乘单元的运算时间为多次验签测试结果的平均值，分别需要 507、516 和 6 174 个时钟周期。

表 4 高速 Ed25519 验签硬件架构中各模块的资源使用

模块	LUT/个	FF/个	DSP/个	BRAM/块
ALU 模块	12 881	5 338	81	0
SHA-512	2 923	2 203	0	0
控制单元	1 502	610	0	0
寄存器	136	2 318	0	0
寄存器堆	3 245	1 647	0	0
模逆模块	1 749	515	0	0

表 5 FPGA 硬件加速电路各模块的运行周期数

模块	周期数/个
乘法器	4
模 p 约简	4
加/减法器	1
模逆	507
SHA-512	80
解压控制	1 913
模 L 约简	19
坐标转换	516
NAF 编码	254
点加	24
倍点	20
多点乘	6 174[1]

4.1 功能验证

功能验证采用了 RFC8032 中推荐的密钥对，如表 6 所示。使用 SystemVerilog 语言和 Vivado 2019.1 工具构建了 Testbench，参考模型使用 Python 编写，用于产生签名和验签数据。验签过程中，选取任意长度的随机数作为待签名消息 M，并将其作为参考模型的输入得到对应的签名数据、验签的结果和多点乘的计算结果；再将公钥、消息和签名数据作为设计的输入得到验签的结果和多点乘的计算结果，通过对比设计和参考模型的输出判断验签功能的正确性；通过随机测试，验证了系统验签功能的正确性。待签名消息 M 为“123456789abcdef0”时的仿真结果如图 5 所示，其中（X_1, Y_1）为式(4)右端的多点乘运算 $SG-kA$ 的仿射坐标输出，（X_2, Y_2）为点 R' 的仿射坐标。

表 6 验签过程中使用的密钥对

数据类型	数据内容
公钥	c5aa8df43f9f837bedb7442f31dcb7b1 66d38535076f094b85ce3a2e0b4458f7
私钥	fc51cd8e6218a1a38da47ed00230f058 0816ed13ba3303ac5deb911548908025

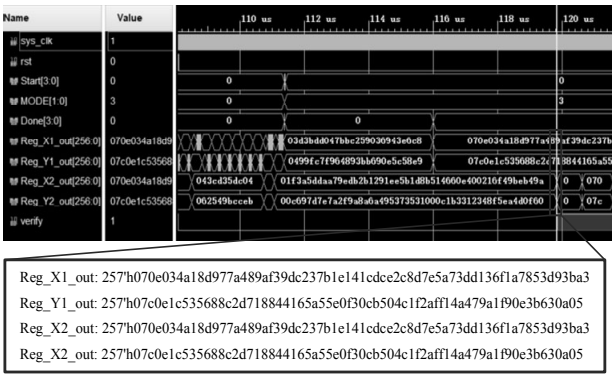

图 5 M 为“123456789abcdef0”的仿真结果

4.2 性能分析

为了客观评估设计的性能和面积，从点乘和验签 2 个层次对现有方案进行对比，分别如表 7 和表 8 所示。

表 7 对比了现有方案实现点乘所需的面积以及点乘运算的性能，ALU 模块是实现点乘的主要运算单元，因此使用 ALU 模块的面积来衡量实现点乘运算所需的硬件资源[18]。文献[16]使用了 15 个 DSP 单元来实现 256 位的模乘，完成一次运算平均需要 18.6 个时钟周期，虽然消耗的硬件资源较少，但是点乘运算所需的运算时间较长。文献[15]采用基为 8 的交错模乘算法，由于不需要消耗 DSP 单元，因此所需的资源最少，在点乘算法的选取上使用了 NAF 方法来减少点加倍点运算的次数，但是完成一次模乘运算需要 95 个时钟周期，点乘的整体性能有待

表 7　　点乘单元性能对比

方案	资源/个			性能			
	LUT	FF	DSP	频率/MHz	模 p 乘周期数/个	点乘周期数/个	点乘运算时间/μs
本文算法	12 881	5 338	81	81.61	7	6 174	75.66
文献[15]	8 680	3 472	0	137.00	95	86 025	627.92
文献[16]	7 911	1 041	15	82.00	18.6	120 260	608.00
文献[17]	52 512	9 342	225	13.87	2	3 895	280.80
文献[18]	9 864	3 885	81	73.00	5	9 181	126.00

表 8　　Ed25519 验签性能对比

方案	平台	资源/个				性能			
		Slices	LUT	FF	DSP	时钟频率/MHz	周期数/个	运算时间/μs	验签速度/(OP·s^{-1})
本文算法	Zynq-7020	13 695	22 561	12 632	81	81.61	9.8×10^3	119.8	8 347.0
文献[9]	MSP430F1611	—	—	—	—	8.00	$1.420\,67\times10^7$	1 775 837.5	0.5
文献[11]	STM32F401	—	—	—	—	84.00	1.331×10^6	15 845.2	63.0
文献[16]	Zynq SoC	2 176	2 707	962	15	82.00	3.01×10^5	3 676.5	272.0
文献[18]	XC7Z020	16 837	34 950	16 772	81	73.00	1.42×10^4	195.6	5 112.0

提升。文献[17]采用 257 位乘法器设计了模乘单元，并选用窗口方法实现点乘运算，虽然点乘所需的周期数较少，但是由于工作频率的限制导致点乘的性能较低，此外生成 257 位乘法器使用了 225 个 DSP 单元，造成了较大的资源开销。文献[18]基于 4 级展开的 Karatsuba 算法使用 81 个 DSP 单元设计了 255 位的模乘单元，所需的周期数较少，但是乘法器的位宽低于 256 位，在点加、倍点过程中加法和减法运算的结果需要先取模才能作为乘法器的输入，导致点加、倍点运算的效率较低；使用了 Shamir 方法来实现点乘运算，只适用于窗口宽度较小的情况，对于点加、倍点运算次数减少的幅度较小。与上述文献相比，本文首先充分利用 DSP 单元设计了 258 位的乘法器，结合模 p 快速约简算法实现了快速模乘运算；其次设计了不需要模加、模减的点加、倍点操作步骤，有效提高了点加、倍点运算的性能；最后使用多点乘替代了点乘，并采用交错 NAF 方法实现多点乘运算，对于 S 和 k 的 NAF 编码分别选用合适的窗口宽度，在系统初始化过程中就完成基点 G 的预计算，有效减少了点加、倍点的运算次数，显著提高了点乘单元的性能。

表 8 对比了现有 Ed25519 验签硬件实现方案的性能和面积。文献[9-10]分别在 MSP430 和 STM32F401 平台上实现了 Ed25519 验签功能，具有低功耗的特点，但是受限于处理器的计算能力，需要消耗的周期数较多，导致验签性能较低；文献[16]使用 Zynq SoC 实现了 Ed25519 的验签功能，没有采用多点乘算法因此需要计算两次点乘，并且模逆运算需要 10 786 个时钟周期导致坐标转换和解压操作较为耗时，平均每秒只能完成 272 次验签运算；文献[18]在 XC7Z020 平台上实现了 Ed25519 的验签功能，坐标转换和模 L 约简的性能与本文相当，但是点乘运算需要消耗的周期较多，每秒只能完成 5 112 次验签运算。点乘是 Ed25519 验签算法中最耗时的运算，相比于其他文献，本文使用了基于交错 NAF 的多点乘算法，在性能上具有显著的优势；此外本文设计了模逆和模乘并行的模幂运算步骤，减少了解压运算所需的周期，并且在验签过程中 SHA-512、NAF 编码、解压、坐标转换等运算可以并行操作，进一步提高了验签的性能。

5　结束语

本文对爱德华曲线数字签名算法 Ed25519 的验签流程进行了研究，针对基于标量乘方法性能较低的问题，提出了基于交错 NAF 方法的多点乘算法，使用 Karatsuba 算法和快速模 p 约简方法实现了快速模乘运算，并设计了不需要模加、模减运算的点加和倍点运算步骤，减少了多点乘运算所需的时钟周期。针对解压过程中耗时的模幂运算，设计了求逆和模乘并行的模幂运算步骤，减少了 50%的运算

量。根据模数 L 的特点设计了快速模 L 约简算法，约简过程中复用了乘法器，在保证运算性能的同时减少了硬件资源开销；设计了 SHA-512、解压和 NAF 编码运算可以并行操作的 Ed25519 验签的硬件架构。最终的设计实现了完整的验签功能，相比于其他设计，在运算速度上具有显著优势，每秒能够完成 8 347 次验签运算。

参考文献：

[1] KOBLITZ N. Elliptic curve cryptosystems[J]. Mathematics of Computation, 1987, 48(177): 203-209.

[2] MILLER V S. Use of elliptic curves in cryptography[C]//Lecture Notes in Computer Science. Berlin: Springer, 1986: 417-426.

[3] 王婧, 吴黎兵, 罗敏, 等. 安全高效的两方协同 ECDSA 签名方案[J]. 通信学报, 2021, 42(2): 12-25.
WANG J, WU L B, LUO M, et al. Secure and efficient two-party ECDSA signature scheme[J]. Journal on Communications, 2021, 42(2): 12-25.

[4] BERNSTEIN D J, DUIF N, LANGE T, et al. High-speed high-security signatures[J]. Journal of Cryptographic Engineering, 2012, 2(2): 77-89.

[5] JOSEFSSON S, LIUSVAARA I. Edwards-curve digital signature algorithm (EdDSA)[R]. 2017.

[6] GAYOSO M V, HERNÁNDEZ E L, MARTÍN M A, et al. Secure elliptic curves and their performance[J]. Logic Journal of the IGPL, 2018, 27(2): 277-238.

[7] 姚前, 张大伟. 区块链系统中身份管理技术研究综述[J]. 软件学报, 2021, 32(7): 2260-2286.
YAO Q, ZHANG D W. Survey on identity management in blockchain[J]. Journal of Software, 2021, 32(7): 2260-2286.

[8] RESCORLA E. The transport layer security (TLS) protocol version 1.3[R]. 2018.

[9] GROBSCHÄDL J, FRANCK C, LIU Z. Lightweight EdDSA signature verification for the ultra-low-power Internet of things[C]//Information Security Practice and Experience. Berlin: Springer, 2021: 263-282.

[10] FAZ-HERNÁNDEZ A, LÓPEZ J, DAHAB R. High-performance implementation of elliptic curve cryptography using vector instructions[J]. ACM Transactions on Mathematical Software, 2019, 45(3): 1-35.

[11] FUJII H, ARANHA D F. Curve25519 for the Cortex-M4 and beyond[C]//International Conference on Cryptology and Information Security in Latin America. Berlin: Springer, 2017: 109-127.

[12] SCOTT M. On the deployment of curve based cryptography for the Internet of things[J]. IACR Cryptol ePrint Arch, 2020, 2020: 514.

[13] 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.

[14] YANG H J, SHIN K W. A hardware implementation of point scalar multiplication on Edwards25519 curve[C]//Proceedings of 2021 International Conference on Electronics, Information, and Communication (ICEIC). Piscataway: IEEE Press, 2021: 1-3.

[15] MEHRABI M A, DOCHE C. Low-cost, low-power FPGA implementation of ED25519 and Curve25519 point multiplication[J]. Information, 2019, 10(9): 285.

[16] TURAN F, VERBAUWHEDE I. Compact and flexible FPGA implementation of Ed25519 and X25519[J]. ACM Transactions on Embedded Computing Systems, 2019, 18(3): 1-21.

[17] 于斌, 黄海, 刘志伟, 等. 高性能 Ed25519 算法硬件架构设计与实现[J]. 电子与信息学报, 2021, 43(7): 1821-1827.
YU B, HUANG H, LIU Z W, et al. High-performance hardware architecture design and implementation of Ed25519 algorithm[J]. Journal of Electronics & Information Technology, 2021, 43(7): 1821-1827.

[18] BISHEH-NIASAR M, AZARDERAKHSH R, MOZAFFARI-KERMANI M. Cryptographic accelerators for digital signature based on Ed25519[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021, 29(7): 1297-1305.

[19] 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.

[20] 尤文珠, 葛海波. 利用多基数系统的高效椭圆曲线多标量乘算法[J]. 计算机工程, 2021, 47(2): 182-187.
YOU W Z, GE H B. Efficient algorithm for multi-scalar multiplication of elliptic curves using multi-base number system[J]. Computer Engineering, 2021, 47(2): 182-187.

[21] HANKERSON D, VANSTONE S, MENEZES A J. Guide to elliptic curve cryptography[M]. Berlin: Springer Science & Business Media, 2006.

[22] 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.

[作者简介]

薛一鸣（1968− ），男，山西文水人，中国农业大学教授、硕士生导师，主要研究方向为信息安全、大规模集成电路设计。

刘树荣（1997− ），男，彝族，云南楚雄人，中国农业大学硕士生，主要研究方向为信息安全、集成电路设计。

郭书恒（1999− ），男，河南济源人，中国农业大学硕士生，主要研究方向为信息安全。

李岩（1982− ），男，内蒙古呼和浩特人，博士，中国农业大学副教授、硕士生导师，主要研究方向为数论、编码、密码。

胡彩娥（1971− ），女，山西文水人，博士，国网北京市电力公司高级工程师，主要研究方向为大数据分析与安全、电力系统自动化。