DOI:10.3969/j.issn.1003-5060.2024.11.009

Curve25519 点乘算法的高效 FPGA 实现

胡　越，肖　昊，赵延睿，刘笑帆
(合肥工业大学 微电子学院,安徽 合肥　230601)

摘　要:为了提高 X25519 密钥交换算法的运算效率,文章基于现场可编程门阵列(field programmable gate array,FPGA)提出一种高效的曲线 Curve25519 的点乘设计方案。首先在底层的有限域计算上,针对模约减计算次数多的问题,提出一种基于冗余数的模运算单元,减少了约减次数;同时,所提出的结构可以减少点乘中常系数乘法的运算周期,从而优化了点乘运算通路,提高了并行度,最终减少了运算时间。该文在 Xilinx XC7Z020 FPGA 上实现了该点乘设计方案,完成一次点乘需要 125 μs。研究结果表明,与现有的设计相比,所提出的方案具有较低的面积时间积,达到了面积和性能的平衡。

关键词:现场可编程门阵列(FPGA);椭圆曲线;曲线 Curve25519;点乘;模乘
中图分类号:TN918.1　　**文献标志码**:A　　**文章编号**:1003-5060(2024)11-1493-06

Efficient FPGA implementation of Curve25519 point multiplication algorithm

HU Yue,　XIAO Hao,　ZHAO Yanrui,　LIU Xiaofan
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: In order to improve the computational efficiency of X25519 key exchange algorithm, this paper proposes an efficient point multiplication design scheme of Curve25519 based on field programmable gate array(FPGA). Firstly, on the bottom level of the finite field calculation, a modular arithmetic unit based on redundant numbers is proposed to reduce the number of reduction calculation. At the same time, the proposed structure can reduce the operation cycle of constant coefficient multiplication in point multiplication, thus optimize the path of point multiplication, improve the parallelism, and finally reduce the operation time. The design scheme of point multiplication is implemented on Xilinx XC7Z020 FPGA, it takes 125 μs to complete a point multiplication. The results show that the proposed scheme has a lower area-time product than existing implementations, achieving a balance between area and performance.

Key words: field programmable gate array(FPGA); elliptic curve; Curve25519; point multiplication; modular multiplication

0　引　　言

椭圆曲线 Diffie-Hellman(elliptic curve Diffie-Hellman,ECDH)密钥交换[1]用于在不安全信道中建立共享密钥,是信息安全领域的常用方法。基于椭圆曲线[2-3]的 DH 密钥交换相比其他同类算法,具有密钥短、计算快等优势,应用十分广泛[4]。然而,目前 ECDH 使用的美国国家标准与技术局(National Institute of Standard and Technology,NIST)公布的 NIST-P 系列曲线,存在以

收稿日期:2022-12-30;**修回日期**:2023-03-16
基金项目:国家自然科学基金资助项目(61974039)
作者简介:胡　越(1998—),男,安徽六安人,合肥工业大学硕士生;
肖　昊(1982—),男,安徽合肥人,博士,合肥工业大学教授,博士生导师,通信作者,E-mail:xiaohao@hfut.edu.cn.

下问题:① NIST-P 曲线上的有限域运算十分复杂,不利于密码协议的快速实现;② “棱镜门”事件后,曲线的安全性受到质疑。因此,业界开始寻求一种快速、安全的新椭圆曲线。

文献[5]提出 Curve25519 曲线,公开参数设计,计算时间恒定,可以抵御计时攻击,具有更好的安全性;同时 Curve25519 曲线的运算性能也优于传统曲线,符合业界对于新曲线的要求,因此 Curve25519 曲线在各类密码库和互联网协议中得到广泛应用[6]。互联网工程任务组(Internet Engineering Task Force,IETF)以及新版传输层安全(transport layer security,TLS)协议推荐使用 X25519 密钥交换协议。X25519 算法由曲线 Curve25519 上的点乘来实现,因此对点乘的加速是快速实现 X25519 的关键。

虽然曲线原本是被设计在软件上实现,但目前学术界已有不少关于 Curve25519 点乘算法在不同计算平台上实现的研究成果,包括在图形处理器[7](graphics processing unit,GPU)和现场可编程门阵列(field programmable gate array,FPGA)上的实现。FPGA 能为密码算法设计加速电路,且具有较短的设计周期,符合网络协议更迭快的特点,因此本文讨论算法在 FPGA 上的实现。文献[8-10]利用 FPGA 中的数字信号处理单元(digital signal processing,DSP)资源搭建计算单元,并提供单核和多核的点乘实现,但点乘运算通路长,且模逆部分需要额外资源;文献[11-12]利用 karatsuba 算法[13]设计模乘器,设计了并行的点乘运算通路,模逆部分复用模乘器实现,但需要额外面积来实现模加和模减,且在点乘运算阶段中资源利用率不高;文献[14]设计统一的模运算单元,可以按需求计算模乘或者模加减,占用资源少,但每次模运算后的模约减过程长,需要较长的计算周期。

本文提出一种高效的 Curve25519 点乘硬件架构。在有限域运算上,设计计算模加减和模乘的统一计算单元,提高资源利用率,并采用冗余数的形式存储中间结果,减少模约减的次数。在点乘中,利用 Montgomery 阶梯计算过程的并行性,构建点乘运算通路,并压缩一个计算阶段,减少计算周期。最后在 Xilinx XC7Z020 FPGA 上实现该设计,在 125 μs 内完成一次点乘运算。

1 算法介绍

Curve25519 是一条定义在有限域上的椭圆曲线,其定义为:

$$y^2 = (x^3 + 486\ 662x^2 + x) \bmod (2^{255} - 19) \tag{1}$$

其中:x、y 为仿射坐标系上的坐标;$2^{255}-19$ 为有限域上的模 m。该椭圆曲线建立在有限域上,有限域上的运算是模运算,包括 255 位无符号数的模加、模减、模乘和模逆,即常规运算后还需要对模 m 取余,确保结果在[0, m)范围内;其中模逆为计算乘法逆元,运算过程较为复杂,算法中会尽量避免出现此类计算。模运算的复杂度与模 m 的选取有关,Curve25519 的模 $m=2^{255}-19$ 属于伪梅森素数,相比常规的模数,具有更快的模约减算法,下面给出算法 1。

输入:二进制数 $C=(c_{509},\cdots,c_0)_2$,
　　$m=2^{255}-19$
输出:约减的结果 $C'=C \bmod m$
C 的低位:$C_l=(c_{254},\cdots,c_0)_2$
C 的高位:$C_h=(c_{509},\cdots,c_{255})_2$
while $C_h \neq 0$ do
　$C=19C_h+C_l$
　$C_l=(c_{254},\cdots,c_0)_2$
　$C_h=(c_{509},\cdots,c_{255})_2$
end　while
$C'=C_l$
$X=C_l-m$
　if　$X \geqslant 0$　then
　$C'=X$
end if
return　C'

计算点乘时,Curve25519 使用 Montgomery 阶梯算法[15],只需要点的 x 坐标参与计算。由于在常用的仿射坐标下,点乘的每一步都涉及开销极大的模逆运算,因此不宜直接在仿射坐标下计算,而是将点的坐标转换为射影坐标表示,包括 X、Z 坐标,下面给出算法 2。

输入:标量 k,曲线 E 上点 P 的 x 坐标 x_P
输出:点乘 $k \cdot P$ 后的射影坐标 (X_{kP},Z_{kP})
$t=\lceil \mathrm{lb}(k+1) \rceil$
$X_1=x_P, X_2=1, Z_2=0, X_3=x_P, Z_3=1$
for $i=t-1$ to 0 do
　if $k_i=1$ then
　　$(X_3,Z_3,X_2,Z_2)=\mathrm{Ladderstep}(X_1,X_3,Z_3,X_2,Z_2)$
　else
　　$(X_3,Z_3,X_2,Z_2)=\mathrm{Ladderstep}(X_1,X_2,Z_2,X_3,Z_3)$
　end if

end for
return(X_2,Z_2)

算法 2 中,$k \cdot P$ 为椭圆曲线上的点乘,其中:P 为初始点;k 为表示倍数的标量值。$k \cdot P$ 表示的计算过程为:已知初始点 P 的坐标,求 k 倍 P 点的坐标。X_1、X_2、Z_2、X_3、Z_3 为参与计算的射影坐标;Ladderstep 为阶梯步骤,是被反复调用的主要计算部分,下面给出算法 3。

输入:X_1、X_2、Z_2、X_3、Z_3
输出:X_2、Z_2、X_3、Z_3
$T_1=X_2+Z_2$,　$T_2=X_2-Z_2$,
$T_3=X_3+Z_3$,　$T_4=X_3-Z_3$,
$T_7=T_2^2$,　$T_6=T_1^2$,
$T_5=T_6-T_7$,　$T_9=T_2T_3$,
$T_8=T_1T_4$,　$X_3=T_8+T_9$,
$Z_3=T_8-T_9$,　$X_3=X_3^2$,
$Z_3=Z_3^2$,　$Z_3=Z_3X_1$,
$X_2=T_6T_7$,　$Z_2=121\ 666T_5$,
$Z_2=Z_2+T_7$,　$Z_2=Z_2T_5$
return(X_2,Z_2,X_3,Z_3)

算法 3 中,$T_1 \sim T_9$ 为中间量。经过上述算法得到射影坐标(X_{kP},Z_{kP})后,最终需要转换为仿射坐标 $x_{kP}=X_{kP}/Z_{kP}=X_{kP}Z_{kP}^{-1}$,其中 Z_{kP}^{-1} 为模逆运算,该部分通过费马小定理 $Z_{kP}^{-1}=Z_{kP}^{m-2}$ 转换为计算模幂,下面给出算法 4。

输入:数 a,幂 b,模 m
输出:$a^b \bmod m$
$q_0=a$;$q_1=1$;$t=\lceil \mathrm{lb}(b+1) \rceil$
for $i=0$ to $t-1$ do
　if $k_i=1$ then
　　$q_1=q_0q_1 \bmod m$
　$q_0=q_0^2 \bmod m$
end for
return q_1

2　算法的硬件实现

2.1　总体结构

本文的 Curve25519 点乘系统结构如图 1 所示。图 1 中:标量 k、点坐标 x_P 为输入;点坐标 x_{kP} 为输出。模运算模块包括有限状态机(finite state machine,FSM)和模运算单元(modular arithmetic unit,MAU)。输入坐标经过初始化转换为射影坐标形式,用于计算 Montgomery 阶梯,每次计算的输入顺序根据标量 k 当前位的值决定;得到射影坐标后计算模逆,经过多次模乘后得到最终的仿射坐标输出。Montgomery 阶梯和模逆不会同时计算,可以复用同一个模运算模块。

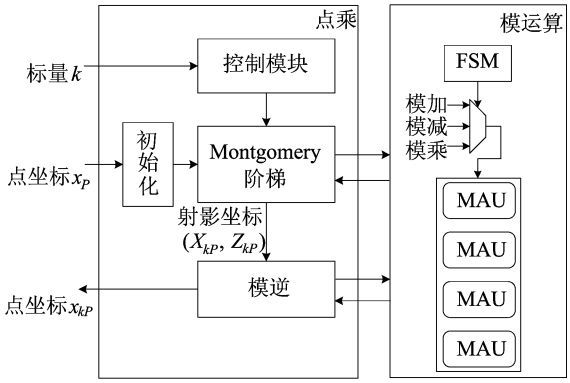

图 1　Curve25519 点乘系统结构

2.2　模运算

模运算包括模乘、模加和模减,由 MAU 来完成,并由 FSM 控制选择当前的模运算。

2.2.1　模乘

在模运算中,模乘是运算量最大、最耗时的底层运算。文献[14]将 255 位的输入数划分为 15 个17 位数,利用 FPGA 里的 DSP 资源计算 17×17位的部分积以及累加。然而,得到的 15 个部分积需要进位和模约减,在最差情况下进位需要传递 15 次,为了抵御计时攻击,算法要求在常数周期下实现,因此必须考虑最差的 15 次进位情况,需要花费大量的周期。

本文提出的算法和架构可以减少模约减的次数,下面给出算法 5。

输入:A(272 位) 和 B(272 位)
输出:$(P_{16},P_{15},\cdots,P_{00}) \leftarrow (A \times B) \bmod (2^{255}-19)$
定义:$\underline{X}=X \times 19$
$A=A_{00}+A_{01} \times 2^{1\times16}+A_{02} \times 2^{2\times16}+\cdots+A_{16} \times 2^{16\times16}$
$B=B_{00}+B_{01} \times 2^{1\times16}+B_{02} \times 2^{2\times16}+\cdots+B_{16} \times 2^{16\times16}$
步骤 1:相乘
$P_{00} \leftarrow A_{00}B_{00}+\underline{A_{16}}B_{01}+\underline{A_{15}}B_{02}+\cdots+\underline{A_{02}}B_{15}+\underline{A_{01}}B_{16}$
$P_{01} \leftarrow A_{01}B_{00}+A_{00}B_{01}+\underline{A_{16}}B_{02}+\cdots+\underline{A_{03}}B_{15}+\underline{A_{02}}B_{16}$
$P_{02} \leftarrow A_{02}B_{00}+A_{01}B_{01}+A_{00}B_{02}+\cdots+\underline{A_{04}}B_{15}+\underline{A_{03}}B_{16}$
⋮
$P_{15} \leftarrow A_{15}B_{00}+A_{14}B_{01}+A_{13}B_{02}+\cdots+A_{00}B_{15}+\underline{A_{16}}B_{16}$
$P_{16} \leftarrow A_{16}B_{00}+A_{15}B_{01}+A_{14}B_{02}+\cdots+A_{01}B_{15}+A_{00}B_{16}$

步骤 2:进位和模约减

$P_{00} \leftarrow P_{00} \bmod 2^{15} + \underline{P_{16}} \gg 15$

$P_{01} \leftarrow P_{01} \bmod 2^{15} + P_{00} \gg 15$

$\vdots$

$P_{16} \leftarrow P_{16} \bmod 2^{15} + P_{15} \gg 15$

算法 5 中:A 按位拆分为 $A_{00}, A_{01}, \cdots, A_{16}$;$B$ 按位拆分为 $B_{00}, B_{01}, \cdots, B_{16}$;$P_{00}, P_{01}, \cdots, P_{16}$ 为部分积的累加。模乘结构如图 2 所示。

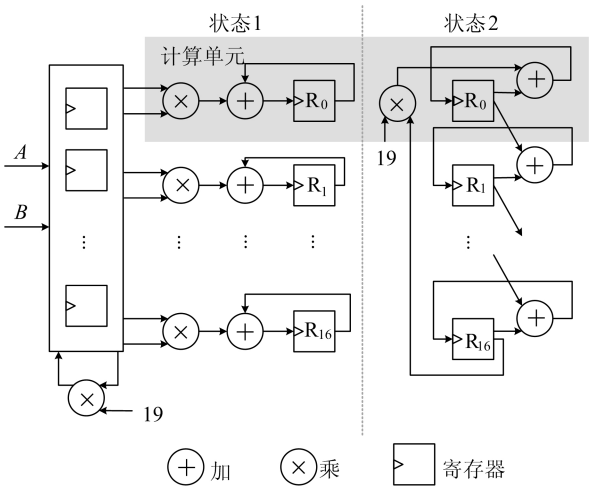

图 2 模乘结构

对于首次的 255 位输入,每隔 15 位添一个 0,扩为 272 位,这是为了和之后的输入输出保持一致。将 272 位的输入数划分为 17 个 16 位存入寄存器,首先按照步骤 1 计算部分积以及累加,通过状态 1 实现,每个计算单元包含一个 DSP,其中,R_0, $R_1, \cdots, R_{16}$ 为计算单元中寄存器的编号。$P_{00}, P_{01}, \cdots, P_{16}$ 的计算分别分配给每行计算单元并行完成,每个周期完成一列的计算和累加,一共需要 17 个周期。从第 2 列开始,部分乘数需要乘以 19,且之后不再用到原来的数。因此,使用一个单独的常系数乘法器,在存入输入后,每个周期计算一个数,并用结果替代原数。

执行步骤 2 时,状态 2 中每个计算单元将高于 15 位的数据进位到下一单元,与下一单元的低 15 位相加;并且最高位计算单元的进位数据在传输到最低位计算单元之前,还需要乘以 19,此乘法由最低位计算单元中的 DSP 完成。如果要确保每个计算单元中的数约减到 15 位,进位需要传递 17 次。为了减少计算次数,本文将约减的目标设定为 16 位,只需要 2 次进位就可以确保完成。加上数据存储,整个模乘需要 22 个周期。

得到的结果是 272 位的冗余数,即每第 16 位的数表示的是下一位的进位。虽然结果没有完全进位,但在此架构下,后续的模乘、模加和模减都可以正确进行,且每次模运算得到结果都能限制在 272 位。只需要在整个点乘算法的最后,再进行一次完整的进位。

2.2.2 模加减

模加减的流程与模乘类似。在步骤 1 中,各行计算单元只需要加法或减法,模加时,计算一次 $A+B$ 即可;模减时,为了确保不会在冗余数结果中引入负数,先预加一个模的倍数 $3m$,再相减,与模的倍数相加不会影响结果的正确性。模加减的步骤 2 同样为进位和模约减。最终,模加需要 4 个周期,模减需要 5 个周期。

2.3 Montgomery 阶梯

Montgomery 阶梯通过重复调用算法 3 所示的 Ladderstep 来实现,在硬件中的计算流程如图 3 所示。图 3 中,X_1、X_2、Z_2、X_3、Z_3 为参与计算的射影坐标。

Ladderstep 计算通路[11]分为 6 个阶段。该设计搭建了 2 个模加模块、2 个模减模块和 4 个模乘模块,在计算 Ladderstep 时,每个阶段内最多 4 个模块并行运算,同时得到结果。然而,在每个阶段中,至少有 4 个模块不在工作状态;在阶段 5,只有一个加法模块参与计算,资源利用率低。

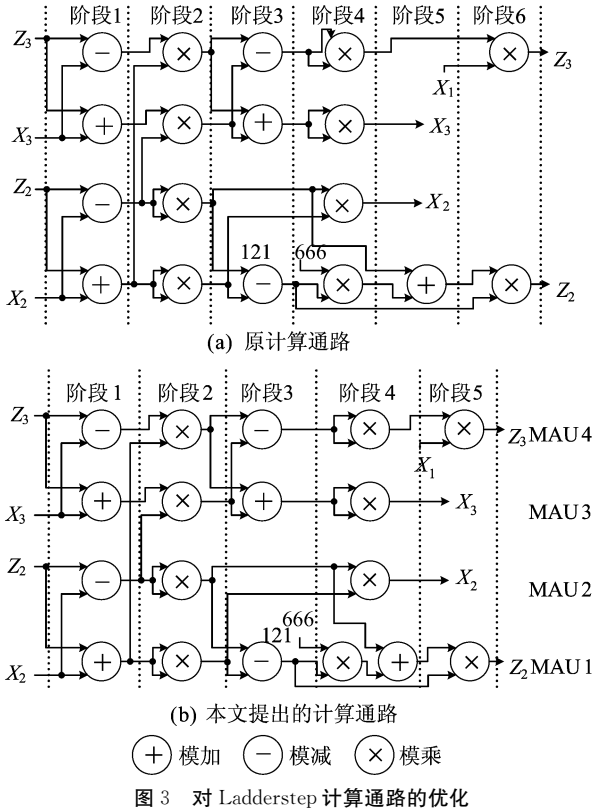

图 3 对 Ladderstep 计算通路的优化

本文优化了 Ladderstep 计算通路，将其分为 5 个阶段，通过状态机控制阶段的转换，每行的计算由一个 MAU 负责完成。每个阶段中最多 4 个 MAU 并行运算，且阶段中的最长运算为一次模加减或模乘，其中模加减阶段 5 个周期，模乘阶段 22 个周期。阶段 4 中，MAU1 先计算一次常系数乘法，由算法 5 的步骤 1 可知，只有前 2 个周期存在非零乘积，因此这次模乘可以用更少的周期完成，然后在同一阶段计算下一个模加，总时间不会超过一次完整模乘的时间。之后等待其他 MAU 完成模乘运算，进入下一阶段。

如算法 2 所述，每次 Montgomery 阶梯步骤完成后，会根据标量 k 当前位的值，选择下一次阶梯步骤输入的顺序，直到遍历过 k 的全部 255 位，便得到点乘后的射影坐标。

2.4　模逆运算

使用费马小定理计算模逆，主要是计算模幂的过程，且幂为固定值。文献[11-12]使用文献[5]给出的算法计算模幂，需要依次计算 265 个模乘。本文提出的设计有 4 个可以计算模乘的 MAU，因此采用算法 4，使用 2 个 MAU 并行计算，需要花费 255 次模乘的时间。

3　性能分析

为了验证所提出的方案，本文使用 Verilog 语言描述电路设计，通过 Vivado2020.2 在 Xilinx XC7Z020 FPGA 上进行综合实现。将本文所提出的 Curve25519 点乘设计方案与其他文献进行比较，在相同平台上的实现见表 1 所列。表 1 中，查找表(look-up table，LUT)和 DSP 是 FPGA 上的硬件资源，频率和时间反映了时序性能。为了更准确地比较综合性能，本文选取面积时间积(area-time product，ATP)指标，ATP 可以反映硬件设计在面积与性能之间的平衡，ATP1 代表 LUTs×时间，ATP2 代表 DSPs×时间。

表 1　本文与其他文献的资源与性能比较

项目	LUT 数量	DSP 数量	频率/MHz	时间/μs	ATP1/10^{-3}	ATP2/10^{-3}
文献[12]	17 939	175	115	92	1 650	16.1
文献[14]	2 707	15	105	714	1 933	10.7
文献[16]	5 478	40	170	352	1 928	14.1
本文方法	9 774	68	200	125	1 222	8.5

文献[12]提出一种短周期的流水线乘法器设计方案，相较于本文中使用的统一模运算单元，文献[12]中设计的模加减单元需要消耗大量的额外硬件资源，其 LUT、DSP 消耗量是本文设计的 2 倍以上。虽然文献[12]的时间仅需 92 μs，比本文的计算时间略短，但代价是花费了 2 倍以上的硬件资源，如使用单个计算单元的计算效率，即面积时间积来对比，则本文的设计比文献[12] ATP1、ATP2 分别降低了 25.9%、47.2%，具有更高的计算资源利用效率。

文献[14]构建一种紧凑的模运算单元。这种电路结构虽然节省了硬件资源，但代价是计算周期更长，在常数时间的要求下，其设计的执行时间将超过 1 400 μs；本文的设计虽然使用了 3 倍以上 LUT 资源和 4 倍的 DSP 资源，但运行速度达到了文献[14]的 11 倍以上。即使文献[14]与本文在 200 MHz 相同频率下运行，本文速度依旧是其 5 倍以上。本文与文献[14]相比，ATP1、ATP2 分别降低了 36.8%、20.6%。因此从 Curve25519 算法的运算效率角度考虑，本文的设计能适用于更广泛的应用场景。

文献[16]使用基于冗余数系统的模乘结构。本文的电路结构具有更短的计算路径，且每个模乘阶段最多可以并行进行 4 个模乘运算。相比之下，本文的设计在相近的运行频率下仅以 0.78 倍的额外 LUT 和 0.7 倍的额外 DSP 资源开销，就获得了 2.82 倍的计算速度提升。对于单位硬件资源的计算效率，本文的 ATP1、ATP2 仅为文献[16]的 63.4%、60.3%。

4　结　　论

本文提出一种低 ATP 的 Curve25519 点乘硬件设计方案，可在 FPGA 硬件平台上高效地实现。此设计方案中的模加减单元和模乘单元具有统一的计算形式，可以复用同一个计算电路。因此，设计中已有资源利用率大大提高。此外，本文采用冗余数的形式存储中间结果来减少模约减的次数，并利用 Montgomery 阶梯计算的并行性，缩短了计算周期，提高了计算速度。与其他同类型方案相比，本文的方案具有更低的 ATP 和更高的计算效率。

[参　考　文　献]

[1] DIFFIE W，HELLMAN M. New directions in cryptography [J]. IEEE Transactions on Information Theory，1976，22(6)：644-654.

(下转第 1504 页)

cient low-power NTT-Uncoupled architecture for NTT-based multiplication[J]. IEEE Transactions on Computers, 2020,69(4):520-533.

[5] BANERJEE U,UKYAB T S,CHANDRAKASAN A P. Sapphire:a configurable crypto-processor for post-quantum lattice-based protocols[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems,2019,2019(4):17-61.

[6] ZHANG N,YANG B H,CHEN C,et al. Highly efficient architecture of newhope-nist on FPGA using low-complexity NTT/INTT[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems,2020,2020(2):49-72.

[7] MERT A C,KARABULUT E,OZTURK E,et al. An extensive study of flexible design methods for the number theoretic transform[J]. IEEE Transactions on Computers, 2020,71(11):2829-2843.

[8] CHEN X R,YANG B H,YIN S Y,et al. CFNTT:scalable radix-2/4 NTT multiplication architecture with an efficient conflict-free memory mapping scheme[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022,2022(1):94-126.

[9] CHEND D, MENTENS N, VERCAUTEREN F, et al. High-speed polynomial multiplication architecture for Ring-LWE and SHE cryptosystems[J]. IEEE Transactions on Circuits and Systems Ⅰ,2015,62(1):157-166.

[10] POLLARD J M. The fast Fourier transform in a finite field[J]. Mathematics of Computation, 1971, 25(114):365-374.

[11] XIN G Z,HAN J,YIN T Y,et al. Vpqc:a domain-specific vector processor for post-quantum cryptography based on risc-v architecture[J]. IEEE Transactions on Circuits and Systems Ⅰ,2020,67(8):2672-2684.

[12] FRITZMANN T,SEPULVEDA J. Efficient and flexible low-power NTT for lattice-based cryptography[C]//2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). [S. l.]:IEEE,2019:141-150.

[13] 车文洁,高献伟. 基于FPGA的进位保留Barrett模乘法器设计与实现[J]. 电子设计工程,2016,24(4):7-9.

[14] KUO P C,CHEN Y W,HSU Y C,et al. High performance post-quantum key exchange on FPGAs[J]. Journal of Information Science and Engineering,2021,2021(5):37-53.

[15] 陈朝晖,马原,荆继武. 格密码关键运算模块的硬件实现优化与评估[J]. 北京大学学报(自然科学版),2021,57(4):595-604.

[16] JATI A,GUPTA N,CHATTOPADHYAY A,et al. SPQ-Cop:side-channel protected post-quantum cryptoprocessor [EB/OL]. (2019-06-30)[2022-11-20]. https://eprint.iacr.org/2019/765.pdf.

(责任编辑 张 镅)

(上接第1485页)

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

[3] MILLER V S. Use of elliptic curves in cryptography[C]// Advances in Cryptology-CRYPTO 85 Proceedings. Berlin: Springer-Verlag,1986:417-426.

[4] 侯惠芳,王云侠. 基于CPK和改进ECDH算法的可证安全的认证协议[J]. 计算机科学,2011,38(9):55-58.

[5] BERNSTEIN D J. Curve25519: new Diffie-Hellman speed records[C]//International Workshop on Public Key Cryptography. Berlin:Springer,2006:207-228.

[6] DIMOV V,KIRDAN E,PAHL M O. Resource tradeoffs for TLS secured MQTT-based IoTmanagement[C]//NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. [S. l.]:IEEE,2022:1-6.

[7] 成娟娟,郑昉昱,林璟锵,等. Curve25519椭圆曲线算法GPU高速实现[J]. 信息网络安全,2017(9):122-127.

[8] SASDRICH P,GÜNEYSU T. Efficient elliptic-curve cryptography using Curve25519 on reconfigurable devices[C]// International Symposium on Applied Reconfigurable Computing. Berlin:Springer,2014:25-36.

[9] SASDRICH P, GÜNEYSU T. Implementing Curve25519 for side-channel-protected elliptic curve cryptography[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS),2015,9(1):1-15.

[10] SASDRICH P, GÜNEYSU T. Exploring RFC 7748 for hardware implementation:Curve25519 and Curve448 with side-channel protection[J]. Journal of Hardware and Systems Security,2018,2(4):297-313.

[11] KOPPERMANN P, SANTIS F D, HEYSZL J, et al. X25519 hardware implementation for low-latency applications[C]//2016 Euromicro Conference on Digital System Design (DSD). [S. l.]:IEEE,2016:99-106.

[12] KOPPERMANN P,SANTIS F D,HEYSZL J,et al. Low-latency X25519 hardware implementation: breaking the 100 microseconds barrier[J]. Microprocessors and Microsystems,2017,52:491-497.

[13] KARATSUBA A,OFMAN Y. Multiplication of multidigit numbers on automata[J]. Soviet Physics Doklady,1962,7: 595-596.

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

[15] MONTGOMERY P L. Speeding the pollard and elliptic curve methods of factorization[J]. Mathematics of Computation,1987,48(177):243-264.

[16] ROY D B,MUKHOPADHYAY D. High-speed implementation of ECC scalar multiplication in GF(p) for generic montgomery curves[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019, 27(7): 1587-1600.

(责任编辑 张 镅)