AES中字节代换和列混合的硬件可逆设计

郑 东,王友仁,张 砦
(南京航空航天大学 自动化学院,江苏 南京 210016)

摘 要:针对AES硬件实现占用大量资源的缺点,对其两个核心计算部件(字节代换和列混合)进行了硬件可逆设计。该设计采用模块复用技术,使字节代换及其逆变换模块最大限度地共享GF(2^8)域中的模逆运算单元,而使列混合及其逆变换模块最大限度地共享$p(x)$乘运算单元,以较小的硬件代价实现了字节代换模块和列混合模块的硬件可逆设计。最后在Xilinx的FPGA VirtexE xcv2000e-6上进行了仿真验证,实验结果表明,与其他同类设计相比,新设计方案明显减少了硬件开销。

关键词:AES;字节代换;列混合;可逆S-box;复合域
中图分类号:TP309.7 **文献标识码**:A **文章编号**:1673-629X(2009)07-0191-04

Reversible Hardware Designs of ByteSub and MixColumn in AES

ZHENG Dong, WANG You-ren, ZHANG Zhai
(College of Automation and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: To solve the problem of high resource cost, which exits in implementation of AES, reversible hardware designs of two core parts (ByteSub and MixColumn) are presented in this paper. Utilizing the way of module reuse, pubic operational unit of module inverse in the GF(28) filed is shared by ByteSub and its inverse transformation, while pubic operational unit of multiplied $p(x)$ is shared by MixColumn and its inverse transformation furthest. Area efficient reversible hardware designs of ByteSub and MixColumn are implemented. Finally, the proposed architectures are implemented on the VirtexE xcv2000e-6 apparatus of Xilinx and the simulation results are provided. The results show its efficiency on saving hardware resource occupied comparing with other current designs.

Key words: AES; ByteSub/Inv ByteSub; MixColumn/Inv MixColumn; reversible S-box; composite field

0 引 言

随着电子信息技术和Internet的迅猛发展,计算机网络安全问题日益突出。网络安全的实质是信息安全,密码技术是信息安全的核心技术,现代密码学被认为是解决信息安全最有效的技术,因此,密码学的研究已成为当今国际上的一个研究热点。

美国高级加密标准(Advanced Encryption Standard,AES)[1]算法已经成为代替DES算法的新一代国际加密标准,占领了通信网络、银行、军队通讯等各个领域。但是AES算法的加/解密过程运算复杂,耗费大量的处理器时间。如果采用软件方案[2]实现,必将影响其性能;若采用硬件方案实现,将会大大地降低处理器的负担,同时能有效地提高加/解密效率和安全性。

密码算法的优化硬件设计实现是目前的研究热点[3]之一,业界人士在这方面做了较多的研究[4,5],但是普遍存在的问题是硬件开销很大,特别是用查找表[6]的方式来实现AES算法的字节代换模块。

针对AES硬件实现开销大的缺陷,文中在保证AES密码系统安全性的前提下,利用模块复用技术,对AES算法的2个主要计算部件字节代换(ByteSub)和列混合(MixColumn)进行了硬件可逆设计。

1 字节代换和列混合硬件可逆设计与实现

1.1 AES算法简介

AES是一个迭代型的分组密码,集安全性、高效性、灵活性于一身,但是在加密之前,需要对明文数据块做预处理。首先,把数据块写成字的形式,每个字包含4个字节,每个字节包含8比特信息;其次,把字记为列的形式。其加/解密过程主要由轮密钥加变换、字节代换/逆字节代换、行移位变换/逆行移位变换、列混

收稿日期:2008-11-13;**修回日期**:2009-02-28
基金项目:航空科学基金(2006ZD52044)
作者简介:郑 东(1983-),男,安徽马鞍山人,硕士研究生,研究方向为密码算法的可重构系统实现;王友仁,博士,教授,博士生导师,研究方向为演化硬件、可重构计算、电子设备在线测试与芯片级自修复。

合变换/逆列混合变换四部分组成。

1.2 字节代换(ByteSub)硬件可逆设计与实现

字节代换是 8 - bit 输入到 8 - bit 输出的非线性变换,独立地对状态的每个字节进行。这里的字节代换其实就是一个 8 * 8 的 S 盒。

传统的 AES 算法都使用了查找表(lookup table)的方法来实现字节代换。不仅消耗大量的存储资源,而且有较大的延时,文中通过将 GF(2^8)中元素变换到其复合域 GF((2^4)^2),则可用组合逻辑替代 RAM 查找表模块,并采用流水线设计方法加以实现。这样不仅节省了硬件存储单元,而且大大提高了 AES 算法模块的处理速度。更主要的是字节代换的逆变换与字节代换共用同一个硬件结构,而不用像查找表实现字节代换和其逆变换时要用两个查找表分别实现。这样更大地节省了硬件开销。

字节代换(ByteSub)由以下两个变换的合成得到:

(1)将字节看作 GF(2^8)上的元素,映射到自己的乘法逆元,'00'映射到自己;

(2)对字节做 (GF(2)上的,可逆的)仿射变换。

1.2.1 仿射变换及其逆变换的硬件实现

仿射变换用基本的异或门就可以实现了,具体变换可参考文献[2]。

1.2.2 GF(2^8)域中模逆运算的改进及其硬件实现

GF(2^8)域中模逆运算的硬件实现比较复杂,一种比较有效的实现方案将域 GF(2^8)分解为两个子域 GF(2^4)^2 的复合,通过在 GF(2^4)^2 域中求逆来代替 GF(2^8)域中求逆,域 GF(2^8)叫做扩展域,GF(2^4)域就是 GF(2^8)域的子域。如果用查找表实现域中求逆的变换,扩展域 GF(2^8)中求逆运算需要 256 字节空间的查找表,而扩展域的等价复合域 GF(2^4)^2 中实现求逆运算仅需要 8 字节空间的查找表。

由有限域的知识可知,复合域 GF(2^4)^2 中每一个元素都可表示为系数在 GF(2^4)上的一次多项式 $bx + c$,复合域上乘法定义为模一个二次不可约多项式 $x^2 + Ax + B$ 的多项式乘法(A,B 是域 GF(2^4) 中的常数,一般取 $A = 1, B = 9$)。设定义有限域 GF(2^4)^2 的乘法的二次不可约多项式为 $x^2 + Ax + B$,则易验证此时 GF(2^4)^2 中任一元素 $bx + c$ 的乘法逆元是:

$$(bx + c)^{-1} = b(b^2B + bcA + c^2)^{-1}x + (c + bA)(b^2B + bcA + c^2)^{-1} \quad (1)$$

其中$(b^2B + bcA + c^2)^{-1}$是$(b^2B + bcA + c^2)$在 GF(2^4)上的乘法逆元。

式(1) 求逆运算的结构示意图如图 1 所示。

图 1 中 $p = b(b^2B + bcA + c^2)^{-1}$, $q = (c + bA)(b^2B + bcA + c^2)^{-1}$。

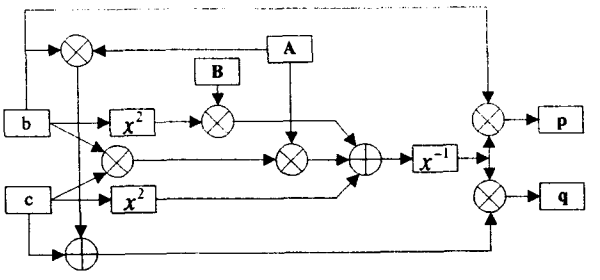


图 1 复合域 GF(2^4)^2 中 $bx + c$ 的求逆运算结构示意图

在图 1 被执行之前,首先要将 GF(2^8)域中的元素映射为以 GF(2^4)域中的数为系数的多项式,这种映射是可逆的。文中采用了 O'Driscoll 构造的一种更有效(矩阵中'1'的数量更少)构造转换矩阵的方法。能够找到一种转换矩阵包含 27 个'1'而和相应的逆矩阵包含 24 个'1',在这两个矩阵中总共包含 51 个'1'。O'Driscoll转换矩的等效方程及其逆映射方程见文献[7]。

1.2.3 字节代换可逆实现

将字节代换和它的逆变换统一起来就可以实现字节代换可逆变换,字节代换可逆实现结构如图 2 所示。

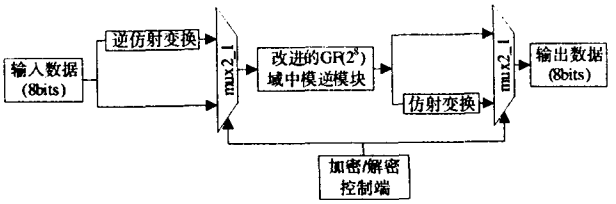


图 2 字节代换可逆实现的结构图

1.3 列混合(MixColumn)硬件可逆设计与实现

在列混合变换中,将状态阵列的每个列视为 GF(2^8) 上的多项式,再与一个固定的多项式 $p(x)$ 进行模 $x^4 + 1$ 乘法。当然要求 $p(x)$ 是模 $x^4 + 1$ 可逆的多项式,否则列混合变换就是不可逆的,会使不同的输入分组对应的输出分组可能相同。Rijindael 的设计者给出的 $p(x)$ 为(系数用十六进制数表示):

$$p(x) = \text{'03'}x^3 + \text{'01'}x^2 + \text{'01'}x + \text{'02'} \quad (2)$$

$p(x)$ 是与 $x^4 + 1$ 互素的,因此是模 $x^4 + 1$ 可逆的。列混合变换的表达式为 $b(x) = p(x) \otimes a(x)$。

而在逆列混合变换中,每一列通过乘以 $p^{-1}(x)$ 进行逆变换。其中

$$p^{-1}(x) = \text{'0}b\text{'}x^3 + \text{'0}d\text{'}x^2 + \text{'09'}x + \text{'0}e\text{'} \quad (3)$$

对比 $p(x)$ 和 $p^{-1}(x)$ 表达式,可以发现 $p^{-1}(x)$ 的系数比 $p(x)$ 的系数复杂多了,这就意味逆混合变换的硬件实现比混合变换硬件实现要复杂得多。为了降低逆变换硬件实现的复杂度,利用 $p(x)$ 对 $p^{-1}(x)$ 进行分解。一般有两种分解方式:并行分解方式和串行

分解方式。

并行分解方式如式(4) 所示:

$$p^{-1}(x) = p(x) + e(x) \tag{4}$$

其中 $e(x) = '08'(x^3 + x) + '0c'(x2 + 1)$。

串行分解方式如式(5) 所示:

$$p^{-1}(x) = p(x) \cdot d(x) \tag{5}$$

其中 $d(x) = p^{-2}(x) = '04'x^2 + '05'$。

由式(4) 和式(5) 可以看出,由于串行分解只有两项系数,而且系数都比较小,所以串行分解能够产生更有效的乘法实现,也更能节省硬件资源。

1.3.1　$x$time() 运算的实现

GF(2^8) 域上还定义了一个运算,称之为 x 乘法,记做 $x$time(),其定义为 $x\text{time}(a) = a(x) \cdot x \bmod m(x)$

其中

$a(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0, a(x) \in \text{GF}(2^8)$。

多项式 $x$time() 运算可以通过左移一位实现,但是如果左移后形成的新多项式级数大于 7,再将这个新多项式和多项式 $m(x) = x^8 + x^4 + x^3 + x + 1$ 做模 2 加运算,结果就是 $x$time() 运算的结果。

如果 $a(x) \in \text{GF}(2^8)$,令 $x\text{time}(a) = b(x)$;

$$x\text{time}(a) = a(x) \cdot x \bmod m(x)$$
$$= (a_7x^8 + a_6x^7 + a_5x^6 + a_4x^5 + a_3x^4 + a_2x^3 + a_1x^2 + a_0x) \bmod (a_7x^8 + a_7x^4 + a_7x^3 + a_7x + a_7)$$
$$= a_6x^7 + a_5x^6 + a_4x^5 + (a_3 \oplus a_7)x^4 + (a_2 \oplus a_7)x^3 + a_1x^2 + (a_0 \oplus a_7)x + a_7$$
$$= b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

1.3.2　列混合变换的硬件实现

根据列混合变换的表达式可以得到列混合变换的硬件实现结构,列混合硬件实现结构如图 3 所示,其中 $a = a_3a_2a_1a_0$ 为 32 位输入,这里每一个字节 a_i 是 8 位的。

1.3.3　列混合可逆实现

逆列混合变换的数学表达式为 $a(x) = b(x) \otimes p^{-1}(x)$,通过式(5),逆列混合变换的数学表达式可以变为:

$$a(x) = b(x) \otimes p^{-1}(x) = b(x) \otimes p(x) \otimes d(x) \tag{6}$$

(1) 乘 $d(x)$ 模块的实现。

设输入 $c = c_3c_2c_1c_0$,输出 $a = a_3a_2a_1a_0$。乘 $d(x)$ 模块的表达式为 $a(x) = d(x) \otimes c(x)$,由于 $d(x) = p^{-2}(x) = d(x) = p^{-2}(x) = '04'x^2 + '05'$。

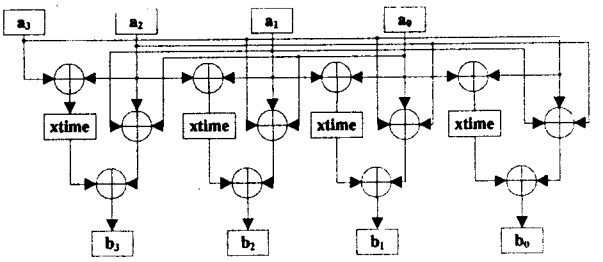


图 3　列混合硬件实现结构图

(2)列混合可逆实现。

将列混合变换和它的逆变换统一起来就可以实现列混合可逆变换,列混合可逆变换结构如图 4 所示。

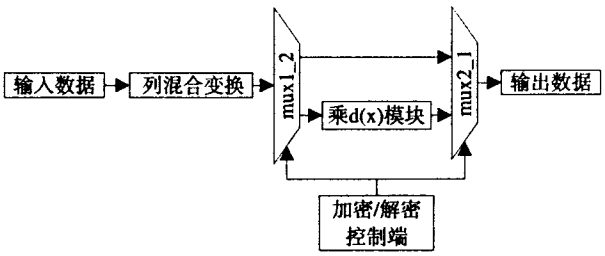


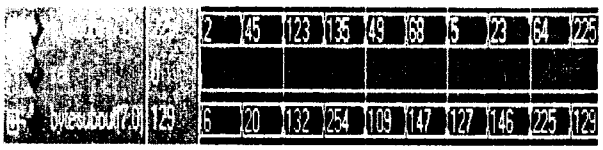
图 4　列混合可逆实现的结构图

2　实验结果与分析

文中用 VHDL 硬件描述语言分别对字节代换和列混合的硬件可逆实现建立了仿真模型,用 Xilinx6.2 的 XST 进行综合与布局布线,最后采用 ModelSim SE6.0 进行时序后仿真,所有仿真结果中的数据采用无符号数表示。通过实验来验证所设计电路的正确性,并与其他设计方案进行性能对比。

2.1　字节代换可逆模块仿真结果及性能分析

在字节代换可逆实现模块的仿真结果如图 5 所示,sel 信号为加/解密控制信号,bytesubin[7:0]为 8 位字节代换可逆实现模块的输入数据,bytesubout[7:0]为 8 位字节代换可逆实现模块的输出数据。当 sel 为'0',表示进行加密运算,此时,进行字节代换;当 sel 为'1',表示进行解密运算,此时,进行逆字节代换。由图 5 仿真结果可见,将字节代换后的输出数据作为逆字



(a) 字节代换结果仿真图

(b) 逆字节代换结果仿真图

图 5　字节代换可逆实现模块的结果仿真图

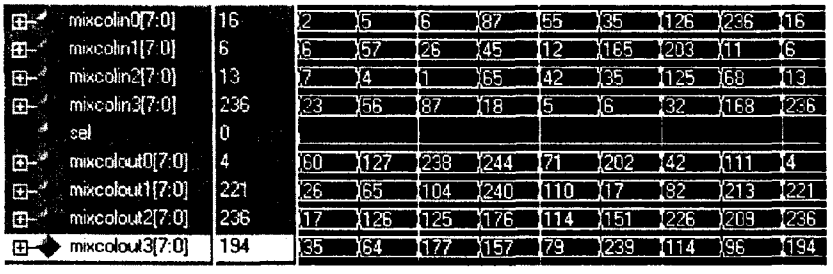
节代换的输入数据得到的输出数据正好和字节代换时的输入数据相同。由此，验证了字节代换可逆实现模块的正确性。与其他实现方案的比较如表 1 所示。通过表 1 可以看出文中所设计的字节代换可逆实现模块与其他实现方案相比确实大大减少了面积开销。

表 1 与其他字节代换实现方案的硬件开销比较

实现方案	Size (gates)
Table look－up(LUT)[8]	2109
PPRM[8]	2320
SOP[8]	1567
BDD[8]	2426
文中	648

2.2 列混合可逆模块仿真结果及性能分析

列混合可逆实现模块的仿真结果如图 6 所示，sel 信号为加/解密控制信号，mixcolin0[7:0]、mixcolin1[7:0]、mixcolin2[7:0]、mixcolin3[7:0]为一组列混合可逆实现模块的输入数据，mixcolout0[7:0]、mixcolout1[7:0]、mixcolout2[7:0]、mixcolout3[7:0]为一组可逆实现模块的输出数据。当 sel 为'0'，表示进行加密运算，此时，进行列混合变换；当 sel 为'1'，表示进行解密运算，此时，进行逆列混合变换。由图 6 仿真结果



(a) 列混合变换结果仿真图

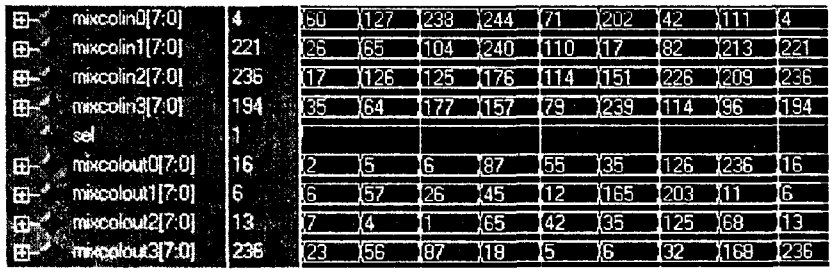


图 6 列混合可逆实现模块的结果仿真图

可见，将列混合变换后的输出数据作为逆列混合变换的输入数据得到的输出数据正好和列混合变换时的输入数据相同。由此，验证了列混合可逆实现模块功能的正确性。与文献[9]和文献[10]的方法比较如表 2 所示，表 2 中文献[9]和[10]只是列混合变换实现，要实现加/解密运算就要同时实现列混合变换和逆列混合变换，逆列混合变换的实现占用的资源比列混合变换还要多一点。所以，同时实现列混合变换和逆列混合变换，占用的资源是列混合变换的 2 倍多，通过表 2 可以看出文中所设计的列混合可逆实现模块与其他实现方案相比确实大大减少了面积开销。

表 2 与其他列混合实现方案的硬件开销比较

实现方案	Size (gates)
文献[9]中的列混合变换	552
文献[10]中的列混合变换	592
列混合可逆实现模块(文中)	816

3 结束语

文中在分析 AES 算法特点的基础上，利用模块复用技术，最大限度地提取了字节代换及其逆变换的公共运算单元和列混合及其逆变换的公共运算单元，给出了新的硬件可逆设计电路，实验证明了新设计方案与其他同类设计相比，能明显节省硬件资源。因此，新设计电路对 AES 算法硬件 IP 核的设计具有较好的参考价值。

参考文献：

[1] 肖国镇，白恩健，刘晓娟. AES 密码分析的若干新进展[J]. 电子学报，2003，31(10)：1549－1554.

[2] Samiah A，Aziz A，Ikram N. An Efficient Software Implementation of AES－CCM for IEEE 802. 11i Wireless Standard[C]//Proceedings of 31st Annual International Computer Software and Applications Conference. Beijing，China：[s. n.]，2007：689－694.

[3] 李 银，金晨辉. 适合 AES 算法硬件实现的新 S 盒[J]. 计算机应用，2007，27(4)：852－856.

[4] Huang Yujung，Lin Yangshih，Hung Kuangyu，et al. Efficient Implementation of AES IP[C]//Proceedings of IEEE Asia Pacific Conference on Circuits and Systems. Singapore：[s. n.]，2006：1418－1421.

[5] 赵 佳，曾晓洋，韩 军，等. 超低成本 AES 算法 VLSI 实现[J]. 小型微型计算机系统，2007，28(8)：1512－1515.

[6] Kuo H，Verbauwhede I. Architectural Optimization for a1. 82Gbits/sec VLSI Implementation of the AES Rijndael Algorithm[C]//Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems. Paris，

(下转第 198 页)

器交互控制器相同,但其交互控制关系模型与原子仪器模型交互控制关系模型有所不同,其交互控制关系模型层次为三层,即系统消息处理控制引擎层、复合仪器交互控制器层及其内部原子仪器交互控制器层。

其交互控制关系模型如图 4 所示。

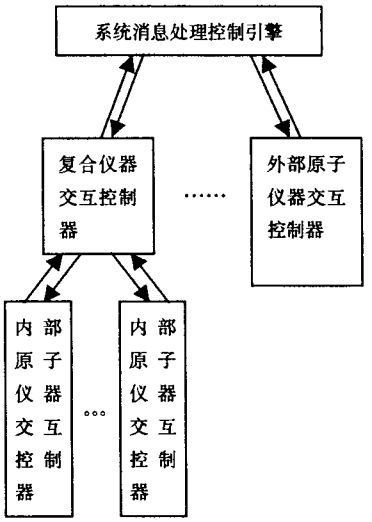


图 4 复合仪器交互控制关系模型

复合仪器交互控制关系模型工作过程分为六部分:外部消息生成、外部消息传递、外部消息接收、内部消息接收、内部消息传递及内部消息生成。

外部消息生成主要是利用仪器的复合仪器交互控制器将相关输出参数及接收仪器端口等信息合成需发送的信息,交由系统消息处理控制引擎处理。

外部消息传递主要是系统消息处理控制引擎根据仪器发送消息的相关接收仪器的端口信息及相关消息传递规则,将消息传递给接收仪器的交互控制器处理。

外部消息接收主要是接收仪器的复合仪器交互控制器接收到消息后,将其分解,把相关信息提交给接收设备的相应端口进行处理。

内部消息生成主要是复合仪器的接收端口将接收到的消息重新组合,目标接收仪器和端口为其内部与该端口相连的原子仪器端口,重新组合而成的消息发送给复合仪器交互控制器处理。

内部消息传递主要是复合仪器交互控制器根据内部原子仪器发送消息的相关内部接收原子仪器的端口信息及相关消息传递规则,将消息传递给内部原子接收仪器的交互控制器处理。

内部消息接收主要是内部原子接收仪器的交互控制器接收到消息后,将其分解,把相关信息提交给该原子接收设备的相应端口进行处理。

3 结束语

通过对网络虚拟实验室仪器模型的深入分析,归纳设计出一套较为通用可行的仪器模型设计方案以及仪器之间的交互控制方法,已经在实际程序设计过程中得到应用。目前的虚拟仪器模型的设计是基于电类实验给出的,对于其他虚拟实验系统设备模型的建立还有待进行通用性验证。

参考文献:

[1] 夏 晖. 基于面向对象的虚拟设备仿真模型的研究[D]. 武汉:华中科技大学,2003.

[2] Hoyer H, Jochheim A, Röhrig C, et al. A multiuser Virtual - Reality Enviroment for a Tele - Operated Laboratory[J]. IEEE Trans on Educ,2004,47(1):121 - 126.

[3] 张 毅,杨秀霞,周绍磊. 虚拟仪器技术分析与应用[M]. 北京:机械工业出版社,2004.

[4] 李仁发,周祖德,李方敏, 等. 虚拟实验室网络体系结构研究[J]. 系统仿真学报,2002,14:359 - 362.

[5] 刘 婧, 刘 丰, 朱俊林, 等. 虚拟网络实验室模型及关键技术研究[C]//第十届全国青年通信学术会议论文集. 北京:北京邮电大学出版社,2005.

[6] Valera A, Díez J L, Vallés M, et al. Virtual and Remote Control Laboratory Development[J]. IEEE Control Systems, 2005,12:35 - 39.

[7] 张 刚,罗小华,贺利芳. 构建网络虚拟实验室技术研究[J]. 实验室研究与探索,2008,27:55 - 58.

[8] 刘丹丹, 邓文生. 基于 Web 的化工协同虚拟现实系统的研究[J]. 计算机技术与发展,2006,16(12):220 - 223.

[9] Canessa E, Fonda C, Radicella S M. Virtual Labora - tory Strategies for DataSharing, Communications and Development [J]. Data Science Journal,2002,1:248 - 256.

(上接第 194 页)

France:[s. n.],2001:51 - 64.

[7] O'Driscoll C. Hardware Implementation Aspects of the Rijndael BlockCipher[D]. Belfield:National University of Ireland, 2001.

[8] Morioka S, Satoh A. A 10 - Gbps Full - AES Crypto Design With a Twisted BDD S - Box Architecture[J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems,2004, 12(7):686 - 691.

[9] Wolkerstorfer J. An ASIC implementation of the AES MixColumn operation[C]//Proceedings of Austrochip 2001. Vienna,Austria:[s. n.],2001:129 - 132.

[10] Noo - Intara P, Chantarawong S, Choomchuay S. Architectures for MixColumn Transform for the AES[C]//Proceedings of ICEP 2004. Phuket, Thailand:[s. n.],2004:152 - 156.