类 AES 分组密码统一框架及其 FPGA 实现

杨宏志 韩文报 董 博
(信息工程大学信息工程学院 郑州 450002)

摘 要 通过将 AES 算法模块化、运算一般化,给出了类 AES 算法的统一框架。在此框架下不仅可以同时实现 AES 的加密、解密,而且可以通过外部参数动态设定分组算法,使得密码算法的使用更加灵活、安全。给出了算法的 FPGA 实现。结果表明设计方案可行,速度较高。

关键词 AES,FPGA,分组密码
中图法分类号 TP309 **文献标识码** A

Unified Framework for Block Cipher Resembling AES Algorithm and its FPGA Implementation

YANG Hong-zhi HAN Wen-bao DONG Bo
(Institute of Information Engineering, Information Engineering University, Zhengzhou 450002, China)

Abstract AES algorithm was modularized and the operation of it was generalized. A unified framework was proposed for block cipher resembling AES algorithm. In this framework, not only AES encryption or decryption can be realized, but also the specific block cipher algorithm can be set through external parameters. The use of cryptographic algorithms is more flexible and security. FPGA realization for the unified framework was also given. The testing results show the scheme is feasible and has a higher speed.

Keywords AES, FPGA, Block cipher

1 引言

Kerckhoffs 假设秘密必须全寓于密钥中,算法的安全性不依赖于算法实现细节的保密[1]。但是在实际应用中,一方面诸多密码算法(DES, IDEA, Blowfish 等)或多或少存在弱密钥,另一方面算法实现细节公开,被片面理解为算法结构固定,而固定的算法结构不利于算法硬件实现芯片的更新换代,在实用中存在安全隐患。

近年来,基于可重构技术的密码算法硬件实现研究成为密码学界的热点问题之一,其研究内容大多集中在多个密码算法的整合实现[2]、密码算法实现效率[3-5]等方面。

本文借鉴可重构思想,将密码算法模块化,使得传统密钥扩展为算法密钥和数据密钥,有效拓展了 Kerckhoffs 假设的内容。文中基于 AES 算法,具体给出了类 AES 算法的统一框架。在不改变电路的前提下,通过算法密钥的设定,灵活切换密码算法,特别地可以在此框架下同时实现 AES 的加密和解密算法。算法可选择范围大,并且硬件实现吞吐率较高。

本设计与实现方式的好处在于:以牺牲一定的算法实现效率,换取算法应用的灵活性,提高了算法的硬件实现灵活性和应用安全性。对密码算法的设计与实现思路进行了新的有益尝试。

2 AES 算法

Rijndael 是一种迭代分组密码算法。该算法本身支持可变的分组长度和密钥长度,而 AES 只采用了 128 位分组长度和 128,192 或 256 位的密钥长度。不失一般性,本文讨论的算法分组长度和密钥长度设定为 128 位。

下面简单介绍一下 Rijndael 算法。图 1 为 AES 加密算法流程,解密运算逆序完成,相应的运算部件采用逆运算。

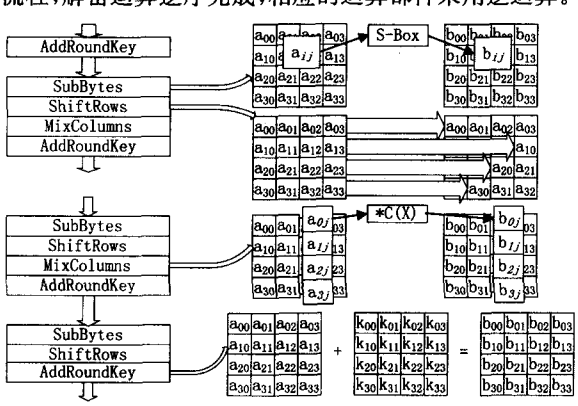

图 1 AES 加密算法

SubBytes:字节代换,用一个 S 盒函数完成分组中的字节代换,代数表达式为 $b_{ij}=S[a_{ij}]$,其中 $S[]$ 为一个变换函数。具体的 AES 加/解密对应的 $S[]$ 取值,限于篇幅,这里从略。

ShiftRows:行移位,一个简单的置换操作。即输入分组矩阵的第 1 行保持不变,第 2 行循环左移 1 个字节,第 3 行循环左移 2 个字节,第 4 行循环左(右)移 3 个字节。逆变换将

到稿日期:2009-05-19 返修日期:2009-07-22 本文受国家 863 计划(2006AA01Z425),国家自然科学基金重大研究计划(90104035)资助。
杨宏志(1978—),男,博士生,主要研究方向为密码理论,E-mail:hz_yang@sohu.com;韩文报(1963—),男,博士生导师,主要研究方向为密码理论、网络安全;董 博(1977—),男,硕士生,主要研究方向为网络安全。

上述循环左移改为循环右移即可。

MixColumns：列混淆，一个利用在域 $GF(2^8)$ 上的算术特性的代换。其代数式为

$$\begin{bmatrix}2&3&1&1\\1&2&3&1\\1&1&2&3\\3&1&1&2\end{bmatrix}\begin{bmatrix}a_{00}&a_{01}&a_{02}&a_{03}\\a_{10}&a_{11}&a_{12}&a_{13}\\a_{20}&a_{21}&a_{22}&a_{23}\\a_{30}&a_{31}&a_{32}&a_{33}\end{bmatrix}=\begin{bmatrix}b_{00}&b_{01}&b_{02}&b_{03}\\b_{10}&b_{11}&b_{12}&b_{13}\\b_{20}&b_{21}&b_{22}&b_{23}\\b_{30}&b_{31}&b_{32}&b_{33}\end{bmatrix},$$

逆变换将系数矩阵 $\begin{bmatrix}2&3&1&1\\1&2&3&1\\1&1&2&3\\3&1&1&2\end{bmatrix}$ 更改为 $\begin{bmatrix}e&b&d&9\\9&e&b&d\\d&9&e&b\\b&d&9&e\end{bmatrix}$ 即可。

AddRoundKey：轮密钥加，利用当前分组和扩展密钥的一部分进行按位异或操作。由于异或运算的特性，加、解密 AddRoundKey 没有区别。

密钥扩展：密钥扩展一共生成 44 个字，每一个字有 32 位，加、解密时密钥扩展过程没有区别，只是在具体运算时选取的密钥顺序正好相反。密钥扩展的具体过程与本文关系不大，在此不再赘述。

3 AES 算法框架

分组密码设计的安全性主要基于 Shannon 提出的混淆原则和扩散原则[6]，混淆原则使得密钥和明文以及密文之间的依赖关系复杂，扩散原则使得密钥及明文的每一个比特影响密文的多个比特。在 AES 算法中，SubBytes 变换实现混淆原则，ShiftRows 和 MixColumns 混合运算主要实现扩散原则。

我们知道，迭代足够多次后，即便是密码学性质很差的变换也可以达到安全的加密效果。基于此，给出类 AES 算法的统一框架，将 SubBytes，ShiftRows，MixColumns，AddRoundKey 变换模块化，密码变换每次迭代选用某一种变换，迭代若干次后，完成对明文的加密。解密只需要将该迭代次序反过来，对应的模块采用逆变换即可，如图 2 所示。

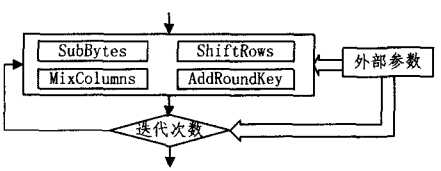

图 2 类 AES 算法统一框架

统一框架的具体设定通过密钥实现，传统的密钥这里扩展为算法密钥和数据密钥。其中算法密钥完成对每次密码算法的设定，达到一次一密的效果。在此方案中算法密钥包括轮密钥选取次序、每次迭代对应的运算模块及其迭代次数，以及 SubBytes，ShiftRows，MixColumns 模块的参数；数据密钥即为传统意义的密钥，经过密钥扩展后，参与密码运算。

需要说明的是，迭代次数由算法密钥指定，因此轮密钥扩展必须有足够的 AddRoundKey 模块迭代使用，具体扩展规则同标准 AES 算法。

接下来讨论 3 种运算模块的参数取值。

SubBytes：

SubBytes 变换是一个基于 S 盒的非线性置换，AES 算法基于有限域 $GF(2^8)$ 中的乘法逆实现。这里在保证变换可逆的前提下，只要给出对应的替换数值即可。参数取值占用 2048 比特。特别地，在设定解密算法时，需要给出一对互逆的 SubBytes 变换，其中一个用于扩展密钥运算，一个用于解密运算，此时参数取值占用 4096 比特，但仅有 2048 比特是有效作用。当然对于一类特殊的 S 盒构造，只要给出 S 盒的构造规则即可，比如乘法逆及仿射变换的系数，此时参数取值将大幅减少。

ShiftRows：

ShifRows 变换为简单的置换操作，AES 加密、解密对应的参数设定为 0,1,2,3 及 0,3,2,1，参数取值占用 2×4=8 比特。

MixColumns：

AES 的 MixColumns 变换设计相当巧妙，加密运算采用的系数矩阵很简单，每个系数最多只有 2 比特有效，而解密系数矩阵存在如下关系式[7]：$\begin{bmatrix}e&b&d&9\\9&e&b&d\\d&9&e&b\\b&d&9&e\end{bmatrix}=$

$\begin{bmatrix}2&3&1&1\\1&2&3&1\\1&1&2&3\\3&1&1&2\end{bmatrix}\begin{bmatrix}5&0&4&0\\0&5&0&4\\4&0&5&0\\0&4&0&5\end{bmatrix}$，进而可以简化电路规模。

MixColumns 的参数取值占用 4×4=16 比特。本文考虑最一般的 MixColumns 运算，给出相应的硬件实现效率。

运算模块的参数取值范围直接影响算法可选择范围。尽管密码学性质较好的运算模块可选择空间远远小于 $2^{2048+8+16}=2^{2072}$，但运算模块的参数设定及运算模块的组合顺序使得动态切换的密码算法可选择范围很大。

4 类 AES 框架的 FPGA 优化实现

4.1 流水设计的考虑

建立流水线是提高速度的常用办法。

由于迭代次数为不定值，且由外部参数设定，为方便讨论，不妨设定迭代次数为固定值 n，某密码运算模块在一个时钟周期内由一个流水站完成。当一组数据到达第 n 个流水站时，一个分组加/解密完成，在密码算法不以反馈方式工作的前提下，n 个流水线站可以同时处理 n 个分组。此设计方案可以在资源占用面积和算法实现速度间进行折中考虑，流水站数目不是必须为 n，只要保证为 n 的因子即可。

4.2 S 盒的设计

S 盒的设计在 AES 硬件实现过程中占据很重要的地位，目前对 AES 算法 S 盒硬件的实现主要基于 LUT 实现和基于逻辑电路实现。基于 LUT 采用查找表对每个状态字节进行变换，结构简单容易实现，处理速度也很快；基于逻辑电路则是利用 AES 算法 S 盒的数学性质，采用逻辑电路实现乘法求逆和仿射变换来实现。此方法结构比较复杂，处理速度较慢，但规模较小，适宜特定的硬件平台。

在本文中，由于 S 盒内容由外部参数指定，在最一般的情况下，其硬件实现采用 LUT 方式更为合理，而针对一类特殊的数学方法构造的 S 盒，在速度要求不高的情况下，采用逻辑电路方式也是可行的。

4.3 密钥扩展设计的考虑

密钥扩展设计，容易想到密钥扩展与加密运算并行处理

的方案。也就是在每一轮加密的过程中，同时进行下一轮密钥的扩展，这样设计的好处在于减少了存储器的需求。这种方案对于加密是最优的，但是对于解密运算不可行。解密的密钥扩展过程虽然与加密相同，但选取的次序正好相反，而密钥扩展过程是不可逆的，因此我们采用在算法设定阶段就一次性完成密钥扩展的设计方案，依据迭代中 AddRoundKey 的次数，一次性生成全部轮密钥，并保存起来。这种方式虽然占用比较多的存储器资源，但针对一个同时实现加密、解密两种功能的算法，也是唯一的设计方案。

4.4 迭代轮数的划分

在设计之初，考虑用 AddRoundKey 来划分迭代次数，也就是传统意义的轮数。在每一轮中根据外部参数来连线 SubBytes，ShiftRows，MixColumns 和 AddRoundKey 4 种不同的运算模块的输入输出，但由于参数的不确定性，4 个模块之间的连线剧增，直接导致最终实现的统一框架时钟频率不可行。综合考虑，最终的方案在一个时钟周期内完成一种运算模块。

对于标准 AES 加密，这里需要 42 个时钟周期完成一个分组的加、解密。

4.5 实验结果

以 Altera 公司的 Cyclone II 系列芯片 EP2C70F672C8 作为算法载体，通过 Verilog 编程实现了类 AES 算法统一框架，并利用 ModelSim 对其进行了仿真。

实验结果为 SubBytes 模块占用 1362 个逻辑单元，ShiftRows 模块占用 130 个逻辑单元，MixColumns 模块占用 391 个逻辑单元，AddRoundKey 占用 128 个逻辑单元。统一框架共实例化 20 个 SubBytes 模块(加密、解密运算部分并行实例化 16 个 SubBytes 模块，密钥扩展部分并行实例化 4 个 SubBytes 模块)、1 个 ShiftRows 模块、4 个 MixColumns 模块、1 个 AddRoundKey 模块，使用 59 个 I/O 引脚，共计 41007 个逻辑单元，时钟频率为 66MHz。以标准 AES 算法加密的迭代次数为例，加/解密使用 42 个时钟周期，吞吐率为 201Mbits/s。

5 安全性分析

我们给出的基于 AES 算法的统一框架模型，是采用 SP 结构，直接依据 Shannon 提出的混淆和扩散原则设计的。本文着重从工程实现的角度考虑，从最一般的角度实现 AES 算法组件评估，进而估算硬件实现效率。

算法的安全性由选取合适的算法组件及迭代次数保证，根据选用的算法组件密码学特性，通过密钥给出足够的迭代次数。算法安全性由算法组件的安全性分析驱动，通过外部密钥的设定，在不更改电路结构的情况下，灵活适应最新的密码学研究成果。

结束语 本文依托 Kerckhoffs 假设，创造性地将密钥扩展为算法密钥和数据密钥。通过将 AES 算法模块化，以牺牲一定资源为代价，在保证算法实现效率的前提下，兼顾了算法的灵活性，一定程度上提高了算法使用的安全性。

文中给出了类 AES 算法最一般的元素部件硬件资源评估。在实际应用中，如何快速、高效地找出密码学性质好的运算，进而利用该类运算特点优化硬件实现，提高算法效率，还需要进一步做工作。

参 考 文 献

[1] Schneier B. Applied Cryptography[M]. Second Edition: protocols, algorithms, and source code in C. New York: John Wiley & Sons Inc, 1996

[2] 高娜娜，王沁，李占才. 基于 AES 和 DES 算法的可重构 S 盒硬件实现[J]. 小型微型计算机系统，2006(3)：446-449

[3] Zhang X, Parhi K K. An efficient 21.56 Gbps AES implementation on FPGA[C]// Thirty-Eighth Asilomar Conference on Signals, Systems and Computers. Nov. 2004, 1: 465-470

[4] Tim G, Mohammed B. AES on FPGA from the fastest to the smallest [C]// Proceedings of CHES 2005. Springer, 2005: 427-441

[5] Gael R, Francois-Xavier S, Jean-Jacques Q, et al. Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications [C]// Proceedings of the International Conference on Information Technology: Coding and Computing. 2004, 2: 583-587

[6] 冯登国，吴文玲. 分组密码的设计与分析[M]. 北京：清华大学出版社，2000

[7] Daemen J, Rijmen V. The Design of Rijndael: AES- the Advanced Encryption Standard[S]. Springer-Verlag, 2002

(上接第 94 页)

[16] Liang Z Q, Shi W S. PET: A Personalized trust model with reputation and risk evaluation for P2P resource sharing[C]// the 38th Hawaii International Conference on System Science. 2005

[17] Lee S, Sherwood R, Bhattacharjee B. Cooperative peer groups in NICE[C]// IEEE Infocom. San Francisco, USA, 2003

[18] Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications[C]// SIGCOMM 2001. San Diego, California, USA, 2001

[19] Aberer K, Cudre-Mauroux P, Datta A, et al. P-Grid: A Self-organizing Structured P2P System[J]. SIGMOD Record, 2003, 32(3): 29-33

[20] Ratnasamy S, Handley M, Karp R, et al. Application-level multicast using content-addressable networks[C]// Proceedings of Third International Workshop on Networked Group Communication. 2001: 14-21

[21] 张蓉，张霞，文学志，等. Peer-to-Peer 环境下多粒度 Trust 模型构造[J]. 软件学报，2006, 17(1): 96-107

[22] Ratnasamy S, Shenker S, Stoica I. Routing algorithms for DHTs: Some open questions[C]// Druschel P, ed. Proc. of the 1st Int'l Workshop on P2P Systems. Berlin: Springer-Verlag, 2002: 45-52