一种轻量级TWINE密码硬件优化实现研究

李 浪[1,2] 邹 祎[1] 贺位位[1] 李仁发[2] 刘波涛[1]
(衡阳师范学院计算机系 衡阳 421002)[1] (湖南大学信息科学与工程学院 长沙 410082)[2]

摘 要 随着物联网的广泛应用,如何有效实现轻量级密码算法成为近年的研究热点。对2011年提出的TWINE加密算法进行了硬件优化实现,相同的轮运算只实现一次,采用重复调用方式完成。TWINE算法总共有36轮运算,其中前35轮运算结构相同,可以重复调用实现,而第36轮相比前35轮在结构上少了块混淆,因此原始算法最多只能进行35轮重复调用。直接进行36轮循环调用运算,同时在36轮循环运算完成后构造一个块混淆逆运算,运算一次块混淆逆运算即可使输出密文正确。这样使TWINE最后的第36轮不必重新实现,而是直接复用前面可重复轮函数模块,只需增加一个比原始算法最后一轮运算相对简单的块混淆逆运算。下载到FPGA上的实验结果表明,优化后的TWINE密码算法在面积上减少了2204个Slices,系统速率提高了5倍。

关键词 TWINE,轻量级密码算法,优化,FPGA实现
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.2.027

Research on Hardware Optimization Implementation of TWINE

LI Lang[1,2] ZOU Yi[1] HE Wei-wei[1] LI Ren-fa[2] LIU Bo-tao[1]
(Department of Computer, Hengyang Normal University, Hengyang 421002, China)[1]
(College of Information Science and Engineering, Hunan University, Changsha 410082, China)[2]

Abstract With the development of IOT, effective implement of lightweight cryptographic algorithms becomes a hot topic in recent years. TWINE encryption algorithm was proposed in 2011. It is a lightweight block cipher algorithm. TWINE is suitable for the security of encryption in the environment of the Internet of Things. The same round operation is achieved only once and repeated calls. TWINE algorithm has 36 operations in which the first 35 operations can be called repeatedly for the same structure. The original algorithm can be repeated up to 35 repeat calls, because the 36th operation has less confusing compared to the first 35 operations. 36 cycle operation is called repeatedly. After that, a block is added. It makes the 36th operation of TWINE can not be re-implemented. A relatively simple algorithm of confusion inverse is added which makes the module highly multiplexed. FPGA experimental results show that area saves 2204 slices and speed increases 5 times than original TWINE. These can provide a reference to encryption researcher.

Keywords TWINE, Lightweight cryptographic algorithms, Optimization, FPGA implementation

1 引言

超轻量级密码算法TWINE于2011年在欧洲密码会议上被提出[1],它是日本NEC公司研发的一种新型快速、轻量化加密技术,用来保证以传感器为介质的信息通信安全。NEC开发的TWINE密码算法适用于资源受限的物联网加密环境,是一种快速、安全的新型加密算法,从硬件规模、内存使用量来看,都适合资源约束的智能卡加密应用[2]。

传统密码算法的优化论文较多,优化的方法包括循环展开、算术运算优化、字节序优化,特别在S盒的优化实现上的研究论文较多[2]。惠越超等就DES的S盒优化研究提出了一种轻量级密码算法LEA[3]。由于TWINE密码算法提出时间不长,目前对其相关研究不够深入,未见对其算法结构硬件优化的相关文献。TWINE加密算法是一种轻量级的对称分组密码算法,通过对其加密算法与结构的深入研究,发现可以采取一些方法对TWINE算法进行合理优化以有效节省实现资源,从而使其更适合在资源约束的传感器芯片上实现[4-7]。

将TWINE整个算法分为两个模块:轮函数模块(TWINERound)和主控制模块(TWINE);对TWINE密码算法的硬件结构进行重新组合实现,使其面积更少、加密运行速度更快[8]。

到稿日期:2014-03-12 返修日期:2014-06-24 本文受国家自然科学基金(61173036),湖南省博士后基金(897203005),衡阳师范学院产学研基金(12CXYZ01),湖南省科技厅科研项目(2013SK3178,2014FJ3061),湖南省十二五重点建设学科,湖南省大学生研究性学习和创新性实验计划项目湘教通2014[248]号(2014366),湖南省自然科学基金(2015JJ4011)资助。

李 浪(1971—),男,博士,教授,主要研究领域为嵌入式系统与信息安全,E-mail:lilang911@126.com;邹 祎(1983—),女,硕士,讲师,主要研究领域为嵌入式计算;贺位位(1989—),男,硕士生,主要研究领域为嵌入式计算;李仁发(1957—),男,教授,博士生导师,主要研究领域为嵌入式计算;刘波涛(1991—),男,主要研究领域为嵌入式计算。

2 TWINE 加密算法简介

TWINE 算法是一种轻量级的对称分组加密算法，分组长度为 64 位，密钥长度有 80 位和 128 位两种，分别记作 TWINE-80 和 TWINE-128，算法采用 Feistel 结构。TWINE 加密算法运算结构如图 1 所示。

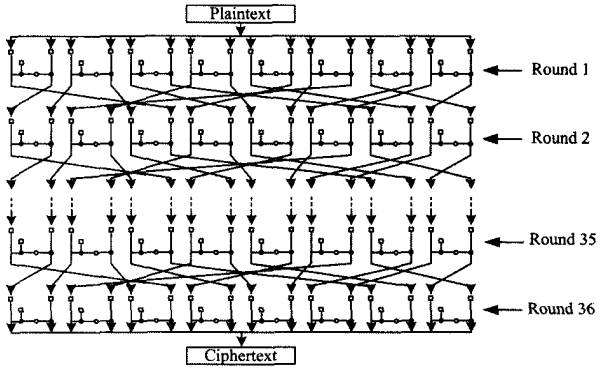

图 1 TWINE 加密算法结构图

S 盒为 4 位输入 4 位输出，共 36 轮运算，其中前 35 轮运算是相同的，第 36 轮计算对比前 35 轮运算少一个块混淆(shuffle)。其中块混淆运算结构如图 2 所示。

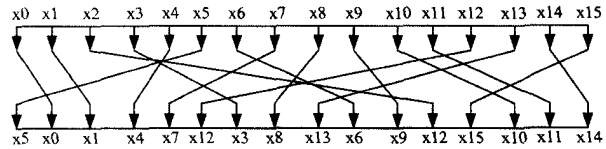

图 2 TWINE 块混淆

3 优化方法

在保证算法加密效率的前提下，通过改变算法结构和用计数器控制重复轮调用次数的方式，以更小的面积和资源占用来实现 TWINE 加密。

3.1 优化方案

1) 将整个算法划分为两个模块：轮函数 TWINERound 和主程序 TWINE。

2) TWINERound 模块包括 3 部分计算：轮运算加密、密钥更新、常数更新。

3) 如图 1 所示，TWINE 的第 36 轮计算对比前 35 轮运算少一个块混淆，也就是说原始算法只能重复 35 轮运算，第 36 轮要单独资源分配，进行独立运算。优化方法是直接重复 36 轮运算，但在运算最后添加一个块混淆逆运算(Reverse_SH)，其结构如图 3 所示，因为直接重复了 36 轮，也就多了一次块混淆，也就需要进行一次块混淆的逆运算，图 3 中灰色标记为添加的块混淆逆运算。

这样做的好处是块混淆逆运算相比原第 36 轮运算要简单得多，从而在资源占用和加密效率上都有优势。

4) 设计一个轮加密运算 TWINERound 模块，TWINE-Round模块包括 TWINE 中每轮的所有运算，从而使得重复运算的函数硬件上只需要实现一次，36 次重复运算均调用 TWINERound，通过常数寄存器 Con 控制次数，用时钟信号控制输入端信号更新。

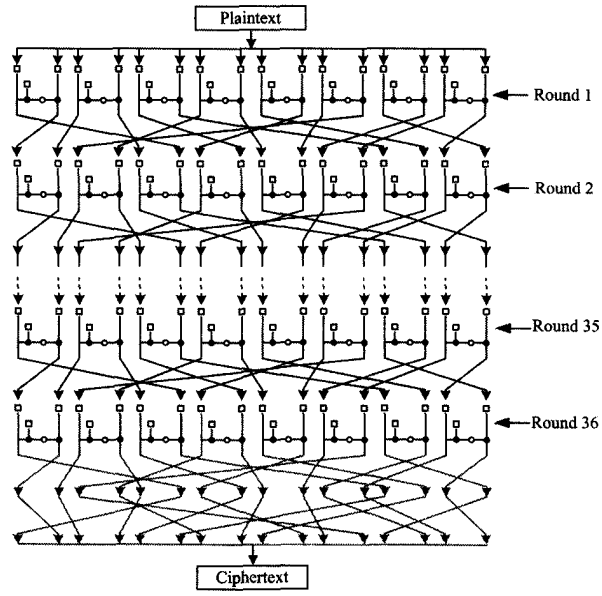

图 3 TWINE 优化加密运算结构图

3.2 优化方法

不失一般性，本文以 TWINE-80 加密算法为例给出详细的原理与具体的优化算法。

TWINE 加密算法每轮运算包括：轮运算加密、密钥更新、常数更新。图 4 所示为轮运算加密过程：首先将 state 分成 16 个 4bit 块，分别记做：$X_i(0\leqslant i<16)$，将密钥 key 分成 20 个 4bit 块，分别记做 $K_i(0\leqslant i<20)$，轮密钥 $RK=\{K_1,K_3,K_4,K_6,K_{13},K_{14},K_{15},K_{16}\}$。轮运算包括轮密钥加($RK\oplus\{X_0,X_2,\cdots,X_{14}\}$)、S 盒变换(对轮密钥加的结果进行 S 盒替换)、再将结果与$\{X_1,X_3,\cdots,X_{15}\}$异或。最后做一次块混淆运算。

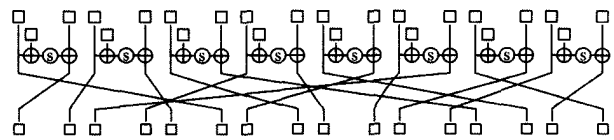

图 4 TWINERound 结构图

在块混淆运算中只需要按照 Shuffle 表中的 $\pi^{-1}[j]$列将 $X_i(0\leqslant i<16)$重新排列就可以实现。

密钥更新过程：将密钥 key 分成 20 个 4bit 块，分别记做 $K_i(0\leqslant i<20)$；对其中一部分 K_i 做如下运算：$K_1=K_1\oplus S(K_0)$，$K_4=K_4\oplus S(K_{16})$，$K_7=K_7\oplus Con[0:2]$，$K_{19}=K_{19}\oplus Con[3:5]$；最后做移位操作。

常数更新过程：$Con_1=1$；有如下递推关系：

$Con_i=(Con_{i-1}*2)\bmod P(x)$，其中 $P(x)$为不可约多项式：Z^6+Z+1。

TWINE-80 密码算法的加密过程主要包括 36 轮运算(TWINERound)。TWINERound 模块包括 3 个输入端口：加密中间结果(state:64bit)、加密密钥(key:80bit)、常数(Con:6 bit)，对应 3 个输出端口：轮加密结果(res:64 bit)，更新密钥(Up_key:80 bit)，更新常数(Up_con:6 bit)。在模块中，轮加密运算、密钥更新、常数更新都是并行执行的。

如图 5 所示，优化后的 TWINE 算法主要包括 36 轮运算和 1 次块混淆逆运算。

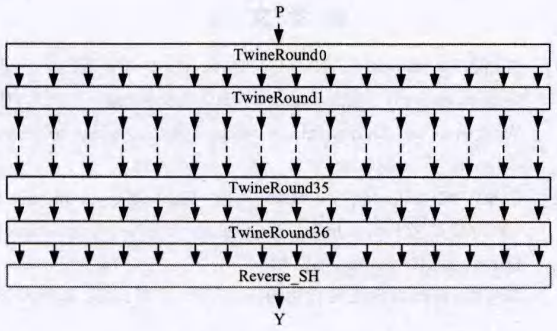

图 5　TWINE 优化加密运算简化结构图

图 6 给出了本文提出的块混淆逆运算 Reverse_SH 的运算结构。

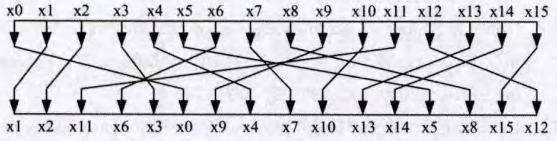

图 6　TWINE 块混淆逆运算

轮运算 TWINERound 模块中包括 4 个端口：1 个输出端口 result(64 位)和 3 个输入端口：密钥 key(80 位)、中间状态 state(64 位)、时钟信号 clk(1 位)。设置常数寄存器 Con、控制 TWINERound 调用的轮数、工作信号 ready，具体实现用硬件语言 Verilog HDL，描述如下。

```
        initial begin
            ready <=1;
            Con <=0;
          end
always @(posedge clk) begin
  Con<= (ready)? {(Con)?t_Con:1} :Con;
  result <=ready ? {(Con)?t_res:state} :result;
  Up_k <= ready ? {(Con)?t_k:key}:Up_k;
  ready <= (Con!=36 &&. ready)? 1:0;
                    end
TWINERound TR(t_res,t_k,t_Con,result,Up_k,Con);
assign res=(ready)?t_res:{t_res[20:23],t_res[0:3],t_res[4:7],t_
res[16:19],
  t_res[28:31],t_res[48:51],t_res[12:15],t_res[32:35],
  t_res[52:55],t_res[24:27],t_res[36:39],t_res[8:11],
  t_res[60:63],t_res[40:43],t_res[44:47],t_res[56:59]
                          };
```

图 7 给出了 TWINE 加密模块接口图，包括 3 个输入端：时钟信号 clk、明文输入 plaintext、密钥 key，1 个输出端：密文 Ciphertext。其中 clk 为 1 位位宽，plaintext，ciphertext 为 64 位位宽。

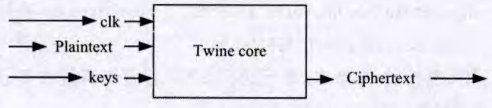

图 7　TWINE 模块接口

3.3　仿真实验

优化后的 TWINE-80 算法如下：

算法 1　TWINE 优化算法

输入：明文，密钥

输出：密文

步骤 1　初始化明文：

pt=64′h0123_4567_89AB_CDEF；

密钥：key=80′h0011223344_5566778899；

时钟信号：clk=1；跳到步骤(2)；

步骤 2　初始化计时器 Con=0，中间状态 state 等于 pt，跳到步骤(3)；

步骤 3　调用 TWINERound 模块，计数器加 Con=(Con * 2) mod (Z^6+Z+1)，跳到步骤(4)；

步骤 4　如果 Con 等于 36(36=8′h24)，跳到步骤(5)；否则跳到步骤(3)；

步骤 5　做一次 Reverse_SH 运算，跳到步骤(6)；

步骤 6　输出加密结果。

程序实现语言为 Verilog HDL，仿真软件为 Modelsim 6.1f，整个算法都是用连续赋值(assign)语句实现，用时钟信号控制计算器更新，完成加密只需要 36 个时钟周期。

仿真结果截图如图 8 所示。

图 8　优化后的 TWINE-80 正确性验证结果

从图 8 可以得到我们的优化算法正确。

4　FPGA 硬件实现结果分析

对优化前后的 TWINE 密码算法进行了 FPGA 实现，通过 ISE 13.2 综合下载进行性能分析，FPGA 型号为 xilinx virtex-5 LX50T。图 9 是未加优化的 TWINE 密码算法 FPGA 实现面积占用实验截图，图 10 是本文优化后的 TWINE 下载到 FPGA 上的实验数据截图。

Slice Logic Utilization:		
Number of Slice Registers:	**5292** out of 28800	18%
Number used as Flip Flops:	5276	
Number used as Latch-thrus:	16	
Number of Slice LUTs:	**7265** out of 28800	25%
Number used as logic:	7094 out of 28800	24%
Number using O6 output only:	6846	
Number using O5 output only:	75	
Number using O5 and O6:	173	
Number used as Memory:	155 out of 7680	2%

图 9　原始 TWINE 面积占用

Slice Logic Utilization:		
Number of Slice Registers:	**5479** out of 28800	19%
Number used as Flip Flops:	5463	
Number used as Latch-thrus:	16	
Number of Slice LUTs:	**4874** out of 28800	16%
Number used as logic:	4706 out of 28800	16%
Number using O6 output only:	4460	
Number using O5 output only:	75	
Number using O5 and O6:	171	
Number used as Memory:	155 out of 7680	2%

图 10　优化后的 TWINE 面积占用

从图 9 和图 10 的实验结果可以得到，优化前的 TWINE 面积占用为：5292+7265=12557 Slices；优化后的 TWINE 面积占用为：5479+4874=10353 Slices；其中 Slices 是 FPGA 中面积单元，由此可以得出优化后的 TWINE 密码算法在面积占用上少了 2204 个 Slices。

根据图 11 和图 12 的实验结果可以得到，优化前的 TWINE 系统所需加密速率为：Minimum period =45.421ns；优化后的 TWINE 系统所需加密速率为：Minimum period = 9.690ns；由此可以得出，优化后的 TWINE 密码算法在系统速率上提高了 5 倍左右。

Design statistics:
Minimum period: 45.421ns (Maximum frequency: 22.016MHz)
Maximum path delay from/to any node: 4.883ns
Maximum net delay: 0.838ns
Analysis completed Fri Feb 21 12:12:00 2014

图 11 原始 TWINE 加密速率实验数据

Design statistics:
Minimum period: 9.690ns (Maximum frequency: 103.199MHz)
Maximum path delay from/to any node: 4.455ns
Maximum net delay: 0.838ns
Analysis completed Fri Feb 21 11:22:34 2014

图 12 优化后的 TWINE 加密速率实验数据

结束语 随着物联网的应用越来越深入，特别是在目前的各种智能卡芯片越来越轻薄的情况下，如何更有效地减少加密算法在智能卡上的实现面积是一个重要课题。因此如何对 TWINE 密码算法进行硬件面积优化是必要的课题。本文对 TWINE 密码算法的结构进行优化，在面积减少的同时也保证了加密效率。

参 考 文 献

[1] Suzaki T, Minematsu K, Morioka S, et al. TWINE: A light weight, versatile block cipher[C]// Proceeding of ECRYPT Workshop on Lightweight Cryptography. Louvain-la-Neuve, Belgium, 2011: 146-169

[2] Çoban M, Karakoç F, Bozta ş Ö. Biclique cryptanalysis of TWINE[C]// Proceeding of Cryptology and Network Security. Darmstadt, Germany, 2012: 43-55

[3] 陈佳康. 密码学算法的优化与应用[D]. 北京：北京邮电大学，2013

[4] 惠越超，汪一鸣. 基于 S 盒优化的轻量级加密算法设计[J]. 通信技术，2010，43(5)：103-106

[5] 何德彪，胡进，陈建华. 基于 FPGA 的高速 AES 实现[J]. 华中科技大学学报：自然科学版，2010，38(2)：101-104

[6] Kundi D S, Aziz A, Ikram N. Resource efficient implementation of T-Boxes in AES on Virtex-5 FPGA[J]. Information Processing Letters, 2010, 110(10): 373-377

[7] Li C Y, Chien C F, Hong J H, et al. An efficient area-delay product design for mixcolumns/ Invmixcolumns in AES[C]// Proceeding of 2008 IEEE Computer Society Annual Symposium on VLSI. Montpellier, France, 2008: 503-506

[8] 李浪，李仁发，邹祎，等. PRESENT 密码硬件语言实现及其优化研究[J]. 小型微型计算机系统，2013，34(10)：2272-2275

(上接第 102 页)

参 考 文 献

[1] Gura N, Patel A, Wander A, et al. Comparing Elliptic Curve Cryptography and RSA on 8bit CPUs[J]. CHES, 2004(8): 11-13

[2] Hill J, Szewczyk R, Woo R, et al. System Architecture Directions for Networked Sensors[C]// Proceeding of Architectural Support for Programming Languages and Operating Systems. Quintana Roo, USA: IEEE Publisher, 2000: 93-104

[3] Malan D J, Welsh M, Smith M D. Implementing Public-Key Infrastructure for Sensor Networks [J]. ACM Transactions on Sensor Networks, 2008, 4(4): 1-23

[4] Levis P, Madden S, Polastre J, et al. TinyOS: An operating system for sensor networks [EB/OL]. 2004-6-18[2014-7-9]. http://www.cs.berkeley.edu/~culler/AIIT/papers/TinyOS/levis06tinyos.pdf

[5] Watro R, Kong D, Sue F C, et al. TinyPK: Securing Sensor Networks with Public Key Technology[C]// Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks. Miami, USA: IEEE Publisher, 2004: 59-64

[6] Benenson Z, Gedicke N, Raivio O. Realizing Robust User Authentication in Sensor Networks[C]// Proceedings of Workshop on Real-World Wireless Sensor Networks. LNSC4582, Palma de Mallorca, Spain: Springer-Verlag, 2005: 556-566

[7] Zhou Y, Zhang Y, Fang Y. Access Control in Wireless Sensor Networks [J]. Ad hoc Networks, 2007(5): 3-13

[8] Oliveira L B, Dahab R, Lopez J. Identity-Based Encryption for Sensor Networks[C]// Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications Workshops. San Francisco, USA: IEEE Publisher, 2012: 290-294

[9] Cheng Hai-bin, Yang Guan. An Authenticated Identity-based Key Establishment and Encryption Scheme for Wireless Sensor Networks[J]. The Journal of China Universities of Posts and Telecommunications, 2012, 13(2): 31-38

[10] Boneh D, Franklin M. Identity-based Encryption from the Weil Pairing[C]// Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. London, Britain: Springer-Verlag, 2001: 213-229

[11] 杨庚，程宏兵. 一种有效的无线传感器网络密钥协商方案[J]. 电子学报，2008，36(7)：1389-1395

[12] Zhang Yi-he, Liu Wei, Lou Wu Jun, et al. Location-based Compromise-tolerant Security Mechanisms for Wireless Sensor Networks[J]. IEEE Journal on Selected Areas in Communications, 2013, 24(2): 247-260

[13] Kampanakis P T. Identity-based Cryptography Feasibility & Applications in Next Generation Sensor Networks [EB/OL]. 2011-11-6[2014-7-9]. http://www.lib.ncsu.edu/theses/available/etd-08042007-125351/unrestricted/etd.pdf

[14] Dong Xiao-lei, Wei Li-fei, Zhu Hao-jin, et al. EP2DF: an efficient privacy-preserving date-forwarding scheme for service-oriented vehicular Ad hoc networks[J]. IEEE Transactions on Vehicular Technology, 2011, 60(2): 580-591

[15] 郭萍，傅德胜，朱节中，等. 无线 MESH 轻量级容侵方案[J]. 计算机科学，2013，40(12)：200-206

[16] Brown D R. SEC 2: Recommended Elliptic Curve Domain Parameters[EB/OL]. 2011-11-8[2014-7-10]. http://www.secg.org/download/aid-784/sec2-v2.pdf

[17] Burrows M, Abadi M, Needham R. Logic of Authentication [J]. ACM Transactions on Computer Systems, 1990(8): 18-36