密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research, 2019, 6(6): 704–712

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

分组密码专刊

Ballet：一个软件实现友好的分组密码算法*

崔婷婷[1]，王美琴[2,3]，樊燕红[2,3]，胡 凯[2,3]，付 勇[2,3]，黄鲁宁[2,3]

1. 杭州电子科技大学 网络空间安全学院, 杭州 310018
2. 山东大学 网络空间安全学院, 青岛 266237
3. 密码技术和信息安全教育部重点实验室, 青岛 266237

通信作者: 王美琴, E-mail: mqwang@sdu.edu.cn

摘 要: 本文提出了一个新的分组密码算法—Ballet 算法. 该算法共有三个版本: Ballet-128/128/46、Ballet-128/256/48 和 Ballet-256/256/74. 所有版本采用相同的轮函数, 无 S 盒和复杂线性层, 仅由模加、异或和循环移位操作组成, 即 ARX 结构算法. 因而本算法灵活性和延展性强, 并能够轻量化实现. 除此之外, Ballet 算法在 Lai-Massey 结构的基础上进行简化设计而成, 并采用 4 分支的近似对称 ARX 结构, 利于软件实现. 其在 32 位和 64 位平台环境下均有很好的表现, 即使在采用单路实现方式下依然具有很大的优势. 在安全性方面, Ballet 算法能够抵抗现有的差分分析和线性分析等已知攻击方法, 且因采用 ARX 结构, 无 S 盒的使用, 防护侧信道攻击的代价小.

关键词: 分组密码算法; ARX 结构; 简化 Lai-Massey 结构; 软件实现友好; 安全

中图分类号: TP309.7 **文献标识码:** J DOI: 10.13868/j.cnki.jcr.000335

中文引用格式: 崔婷婷, 王美琴, 樊燕红, 胡凯, 付勇, 黄鲁宁. Ballet: 一个软件实现友好的分组密码算法[J]. 密码学报, 2019, 6(6): 704–712.

英文引用格式: CUI T T, WANG M Q, FAN Y H, HU K, FU Y, HUANG L N. Ballet: A software-friendly block cipher [J]. Journal of Cryptologic Research, 2019, 6(6): 704–712.

Ballet: A Software-friendly Block Cipher

CUI Ting-Ting[1], WANG Mei-Qin[2,3], FAN Yan-Hong[2,3], HU Kai[2,3], FU Yong[2,3], HUANG Lu-Ning[2,3]

1. School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China
2. School of Cyber Science and Technology, Shandong University, Qingdao 266237, China
3. Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao 266237, China

Corresponding author: WANG Mei-Qin, E-mail: mqwang@sdu.edu.cn

* 基金项目: 国家自然科学基金 (61902100, 61572293, 61502276, 61692276); 国家密码发展基金 (MMJJ20170102); 山东省重大科技创新工程 (2017CXGC0704); 山东省自然科学基金 (ZR2016FM22)

Foundation: National Natural Science Foundation of China (61902100, 61572293, 61502276, 61692276); National Cryptography Development Fund (MMJJ20170102); Major Scientific and Technological Innovation Projects of Shandong Province, China (2017CXGC0704); Natural Science Foundation of Shandong Province, China (ZR2016FM22)

收稿日期: 2019-11-20 定稿日期: 2019-12-10

Abstract: A new block cipher family named Ballet was designed. Ballet has three versions: Ballet-128/128/46, Ballet-128/256/48, and Ballet-256/256/74. All versions have the same round function, which only includes modulo addition, XOR, and rotation operations (i.e. ARX construction), without any S-box and complex linear layer. This design enables Ballet cipher to have strong flexibility and scalability. In addition, Ballet cipher is inspired by simplifying the Lai-Massey construction, and adopts 4-branch almost symmetric ARX construction, so it has excellent performance in software implementation both on 32-bit and 64-bit platforms. Even though under one-way implementation, Ballet still has nice performance and some advantages. With respect to its security, Ballet can thwart against all known attacks, such as differential attack, linear attack, and so on. Since no S-box is used in Ballet, the cost to resist against side channel attack is small.

Key words: block cipher; ARX construction; simplified Lai-Massey construction; software-friendly; security

1 引言

分组密码算法法作为对称密码的一个重要分支, 由于其加密速度快、易于标准化和便于软硬件实现等特点, 成为保障信息机密性和完整性的重要手段, 且已在计算机通信和信息系统安全领域有着广泛应用. 现有的分组密码算法主要有三种设计结构: substitution-permutation network (SPN) 结构, 代表有高级加密标准 AES [1]、轻量级分组密码算法 SKINNY [2] 等; Feistel 结构, 如数据加密标准算法 DES [3]、NSA 设计的 Simon 算法 [4] 等; Lai-Massey 结构, 起源于 IDEA 算法 [5]. 三种结构各有其优势和劣势. SPN 结构混淆扩散速度快, 算法轮数一般较短, 算法实现时吞吐量较大, 但加解密不一致. Feistel 结构加解密一致, 但每轮只能处理一半的数据, 混淆扩散速度慢, 轮数一般较长. 而 Lai-Massey 结构, 主要为 IDEA 算法, 其混淆扩散速度很快, 加解密一致, 且软件实现速度快. 但是轮函数较为复杂, 基于轮的硬件实现时面积较大. 除此之外, 轮函数仅采用模加、异或和循环移位操作的 ARX 结构, 相对带 S 盒的传统算法而言, 软件实现效率具有明显的优势, 且防护侧信道攻击的代价小, 因而近年来也越来越得到青睐, 如 2013 年美国 NSA 设计的 SPECK 算法 [4]、CHES 2015 上提出的 SIMECK 算法 [6]、ASIACRYPT 2016 上提出的 SPARX 算法 [7] 等.

本文给出了一个软件实现友好的分组密码算法—Ballet 算法. Ballet 算法的设计理念是在保证算法安全性的前提下, 保持 Lai-Massey 结构混淆扩散快和 ARX 结构软件实现快的特点, 并且尽可能轻量化轮函数实现, 降低硬件实现代价.

1.1 主要贡献

Ballet 算法共三个版本: Ballet-128/128/46、Ballet-128/256/48 和 Ballet-256/256/74 (Ballet-$n/k/r$: n 为分组长度, k 为密钥长度, r 为总轮数). Ballet 算法的设计灵感来源于 Lai-Massey 结构, 结合 4 分支的 ARX 结构简化设计而成, 主要特色如下:

(1) **算法简洁、灵活、可延展性强.** Ballet 的轮函数中仅包含 2 个模加操作、3 个异或运算和 4 个循环移位操作, 每个均为 $n/4$ 比特操作. 轮函数简洁明了, 内存消耗小, 代码量少, 且所有版本均使用相同的轮函数, 算法灵活性和延展性强.

(2) **实现效率高.** 在软件实现方面, Ballet 算法的轮函数采用 4 分支的近似对称的 ARX 结构, 轮函数结构紧凑, 组件使用较少, 利于指令执行, 在 32 位平台和 64 位平台上均有高效的表现. 本文给出了 Ballet 算法在 64 位 Windows 环境下和 32 位 ARM 环境下的软件实现结果. 特别地, 由于 4 分支的实现方式, Ballet-128/128、Ballet-128/256 在 32 位平台和 Ballet-256/256 在 64 位平台上的资源契合度更好. 除此之外, Ballet 算法在采用单路 (1-way) 实现方式时依然具有较大的优势. 在硬件方面, Ballet 算法的轮函数消耗门电路的操作仅包含 2 个 $n/4$ 比特模加操作和 3 个 $n/4$ 比特异或运算, 密钥生成算法中仅包含简单的线性组件. 因此, Ballet 基于轮实现时, 硬件面积占用较小, 易于轻量化实现. 本文给出了 Ballet 算法的 Verilog 硬件仿真实现结果. 采用的

仿真工具为 ModelSim SE 10.1a, 综合工具为 Synopsys Design Vision(F-2011.09-SP1 Version), 以及工艺库为 HJTC 0.11 um.

(3) **安全.** 我们基于可满足性问题 (Boolean satisfiability problem, SAT) 的自动化搜索方法 [8] 和基于混合整数线性规划 (mixed integer linear programming, MILP) 的自动化搜索方法 [9,10] 对 Ballet 算法进行了差分分析 [11]、线性分析 [12]、不可能差分分析 [13,14]、零相关分析 [15]、积分分析 [16] 和相关密钥类的分析. 分析结果表明 Ballet 算法能够抵抗现有攻击方法, 并具有充分的安全冗余.

(4) **防护侧信道攻击代价小.** 现有的侧信道攻击大多针对算法中的 S 盒进行. Ballet 算法采用 ARX 结构, 非线性操作为模加操作而非 S 盒, 因此防护侧信道攻击的代价小.

1.2 本文框架

本文之后内容安排如下: 第 2 节给出 Ballet 算法描述; 第 3 节讨论 Ballet 的算法设计原理和优点; 第 4 节和第 5 节分别给出 Ballet 算法的安全性分析结果和软硬件实现结果; 最后, 第 6 节总结全文.

2 Ballet 算法描述

在描述算法之前, 如下定义本文档中使用的表示符号:

$n/k/r$:	分组长度/密钥长度/算法总轮数;
$\lll$ 和 $\ggg$:	左循环移位和右循环移位;
$\boxplus$ 和 $\boxminus$:	模加和模减操作;
$\oplus$:	异或操作;
$x = x_{n-1}x_{n-2}\ldots x_1x_0$:	x_{n-1} 为 x 的最高位, x_0 为 x 的最低位;
Ballet-n/k:	分组长度为 n, 密钥长度为 k 的 Ballet 算法版本;
Ballet-n:	分组长度为 n 的 Ballet 所有可能版本.

2.1 算法参数

本文档中给出了一个新的分组密码算法—Ballet 算法. 该算法共三个版本: Ballet-128/128、Ballet-128/256 和 Ballet-256/256, 每个版本的参数如表 1 所示.

表 1 Ballet 算法的三个版本的参数设定

Table 1 Three variants of Ballet block cipher

版本	分组长度 (n)	密钥长度 (k)	总轮数 (r)
Ballet-128/128	128	128	46
Ballet-128/256	128	256	48
Ballet-256/256	128	128	74

2.2 算法结构

Ballet 算法采用 ARX 结构, 轮函数中仅包含循环移位、模加和异或操作, 结构简单易实现. 其每个版本均使用相同的轮函数结构, 见图 1. 为了加解密的相似性, 最后一个轮函数运算时省略最后一个线性置换操作.

假设 Ballet 算法的第 i 个轮函数 ($i = 1, 2, \cdots, r-1$) 的输入为 $(X_0^i||X_1^i||X_2^i||X_3^i)$, 使用的子密钥为 $(\mathrm{sk}_i^L||\mathrm{sk}_i^R)$, 其中 $X_*^i, \mathrm{sk}_i^* \in \mathbb{F}_2^{n/4}$. 则加密过程和解密过程见算法 1 和算法 2.

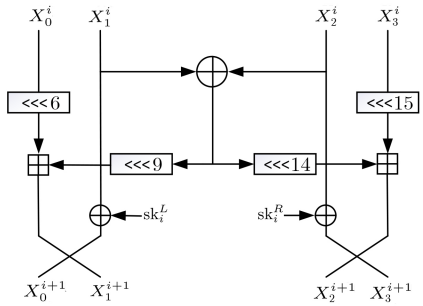

图 1 Ballet 算法的轮函数 (加密过程, 当作为算法的最后一轮时, 省略最后一个线性置换操作)
Figure 1 Round function of Ballet cipher (Encryption. The last permutation is ommited in the last round.)

算法 1 Ballet 算法的加密过程

for $0 \le i < r-1$ **do**

$X_0^{i+1} = X_1^i \oplus \mathrm{sk}_i^L$;
$X_1^{i+1} = (X_0^i \lll 6) \boxplus [(X_1^i \oplus X_2^i) \lll 9]$;
$X_2^{i+1} = (X_3^i \lll 15) \boxplus [(X_1^i \oplus X_2^i) \lll 14]$;
$X_3^{i+1} = X_2^i \oplus \mathrm{sk}_i^R$;

end

$X_0^r = (X_0^{r-1} \lll 6) \boxplus [(X_1^{r-1} \oplus X_2^{r-1}) \lll 9]$;
$X_1^r = X_1^{r-1} \oplus \mathrm{sk}_{r-1}^L$;
$X_2^r = X_2^{r-1} \oplus \mathrm{sk}_{r-1}^R$;
$X_3^r = (X_3^{r-1} \lll 15) \boxplus [(X_1^{r-1} \oplus X_2^{r-1}) \lll 14]$.

算法 2 Ballet 算法的解密过程

for $0 \le i < r-1$ **do**

$X_0^{i+1} = X_1^i \oplus \mathrm{sk}_i^L$;
$X_1^{i+1} = [X_0^i \boxminus (X_1^i \oplus X_2^i \oplus \mathrm{sk}_i^L \oplus \mathrm{sk}_i^R) \lll 9] \ggg 6$;
$X_2^{i+1} = [X_3^i \boxminus (X_1^i \oplus X_2^i \oplus \mathrm{sk}_i^L \oplus \mathrm{sk}_i^R) \lll 14] \ggg 15$;
$X_3^{i+1} = X_2^i \oplus \mathrm{sk}_i^R$;

end

$X_0^r = [X_0^{r-1} \boxminus (X_1^{r-1} \oplus X_2^{r-1} \oplus \mathrm{sk}_{r-1}^L \oplus \mathrm{sk}_{r-1}^R) \lll 9] \ggg 6$;
$X_1^r = X_1^{r-1} \oplus \mathrm{sk}_{r-1}^L$;
$X_2^r = X_2^{r-1} \oplus \mathrm{sk}_{r-1}^R$;
$X_3^r = [X_3^{r-1} \boxminus (X_1^{r-1} \oplus X_2^{r-1} \oplus \mathrm{sk}_{r-1}^L \oplus \mathrm{sk}_{r-1}^R) \lll 14] \ggg 15$.

2.3 密钥生成算法

Ballet 算法按照分组长度和密钥长度的关系可分为 Ballet-n/n 和 Ballet-$n/2n$ 两种. 针对 Ballet-n/n 版本, 密钥生成算法见算法 3. 针对 Ballet-$n/2n$ 版本, 密钥生成算法见算法 4.

算法 3 Ballet-n/n 版本的密钥生成算法

假设主密钥为 $K = k_0||k_1$;
for $0 \le i < r$ **do**

$\mathrm{sk}_i^L||\mathrm{sk}_i^R = k_0$;
输出 $\mathrm{sk}_i^L||\mathrm{sk}_i^R$;
$k_{\mathrm{temp}} = k_1$;
$k_1 = k_0 \oplus (k_1 \lll 3) \oplus (k_1 \lll 5) \oplus i$;
$k_0 = k_{\mathrm{temp}}$.

end

算法 4 Ballet-$n/2n$ 版本的密钥生成算法

假设主密钥为 $K = k_0||k_1||t_0||t_1$;
for $0 \le i < r$ **do**

$\mathrm{sk}_i^L||\mathrm{sk}_i^R = k_0$;
输出 $\mathrm{sk}_i^L||\mathrm{sk}_i^R$;
$t_{\mathrm{temp}} = t_1$, $k_{\mathrm{temp}} = k_1$;
$t_1 = t_0 \oplus (t_1 \lll 7) \oplus (t_1 \lll 17)$,
$k_1 = k_0 \oplus (k_1 \lll 3) \oplus (k_1 \lll 5)$;
$t_0 = t_{\mathrm{temp}}$, $k_0 = k_{\mathrm{temp}}$;
$k_1 = k_1 \oplus t_1 \oplus i$.

end

3 Ballet 设计原理

在本算法的设计时, 我们综合考虑算法的安全性、软件实现速度和硬件实现代价. 大致设计原理为: 提出简化版的 Lai-Massey 结构, 尽可能加快混淆扩散速度以保障算法的安全性; 采用模加操作代替 S 盒操作作为算法的非线性组件, 加快算法的软件实现速度, 并提高防御侧信道攻击的能力; 减少轮函数中的操作个数, 以模加操作配合循环移位操作的结构尽可能轻量化算法实现.

(1) **算法结构简化 Lai-Massey vs. Feistel, SPN**

Feistel、SPN 和 Lai-Massey 结构是算法设计中常用的三种成熟结构. 然而在实际的算法设计中, 大多采用前两种结构, 基于 Lai-Massey 结构设计的算法较少. 但是值得注意的是, Lai-Massey 结构具有良好的混淆扩散能力, 并且算法结构对称, 有利于组件的并行计算. 本算法结构在 Lai-Massey 结构的基础上进行简化, 去除 F 函数而增设模加操作, 在降低硬件实现代价的基础上依然保持良好的混淆扩散速度.

(2) **算法结构 4 分支 vs. 2 分支**

本算法结构将输入均分为 4 支进行运算, 具有两大优势. 第一, 可减小每个组件大小 (运算比特数为分组的 1/4), 增加组件个数, 从而增加算法设计的灵活性, 并有利于轮函数中组件的并行运算; 第二, 算法结构为 4 分支使得算法中每个组件的运算为分组大小的 1/4, 从而有利于 128 比特分组版本在 32 位平台的软件实现和 256 比特分组版本在 64 位平台下的软件实现.

(3) **模加 + 循环移位 vs. S 盒 + 线性矩阵**

模加操作的第 x 比特输出和所有的低 x 比特输入有关, 因此其混淆扩散程度并不均匀, 输入的低位比特混淆扩散较快, 高位比特混淆扩散较慢. 但佐以适当的循环移位操作, 选择适当的循环移位参数, 短轮数下即可实现快速的混淆扩散. 模加操作和循环移位操作配合的方式, 不仅有利于算法的混淆扩散速度, 也可以降低组件个数, 提升软件实现速度, 降低硬件实现代价.

4 Ballet 安全性分析

本设计文档给出 Ballet 算法的差分分析、线性分析、不可能差分分析、零相关分析和积分分析的结果. 鉴于算法结构本身特点, 其它分析方法 (如中间相遇攻击和代数攻击等) 对算法安全性的影响较小, 设计文档中不再给出分析结果.

4.1 差分分析

为了评估 Ballet 算法抵抗差分分析的能力, 我们基于 SAT 自动化搜索方法, 对短轮数下的最优差分路线进行搜索. Ballet-128 和 Ballet-256 的 9 轮差分路线概率上界均为 2^{-43}, 因此 Ballet-128 的有效差分路线 (概率 $> 2^{-128}$) 不超过 $3 \times 9 - 1 = 26$ 轮, Ballet-256 的有效差分路线 (概率 $> 2^{-256}$) 不超过 $6 \times 9 - 1 = 53$ 轮. 全轮 Ballet 算法能够抵抗差分攻击.

4.2 线性分析

为了评估 Ballet 算法抵抗线性分析的能力, 我们同样利用基于 SAT 的自动化搜索方法对短轮数下的 Ballet 算法进行分析.Ballet-128 的 8 轮和 9 轮的最优线性路线的相关度分别为 2^{-19} 和 2^{-23}, 因此 26 轮 (8 轮 +9 轮 +9 轮) 的线性路线的相关度不超过 $2^{-19} \times 2^{-23} \times 2^{-23} = 2^{-65} < 2^{-64}$, 从而均为无效线性路线. 换言之, Ballet-128 的有效线性路线不超过 25 轮.Ballet-256 的 8 轮最优线性路线的相关度为 2^{-19}, 9 轮的线性路线上界为 2^{-23}, 因此 52 轮的线性路线的相关度不超过 $2^{-130} < 2^{-128}$, 即 Ballet-256 的有效线性路线不超过 51 轮. 全轮 Ballet 算法能够抵抗线性分析.

4.3 不可能差分分析

不可能差分分析是利用概率为 0 的差分路线进行的分析方法. 我们利用基于 MILP 的自动化搜索方法对 Ballet 的不可能差分路线进行搜索. 鉴于 Ballet 算法为 ARX 结构, 我们仅搜索输入和输出差分汉明重量均为 1 的情况.

通过搜索发现, Ballet-128 在上述搜索空间中最长不可能差分路线为 7 轮路线, 其中一条路线如下:

$$0^6 10^{25} || 0^{32} || 0^{32} || 0^{32} \nrightarrow 0^{32} || 0^{32} || 10^{31} || 0^{32}$$

其中 0^* 表示 $*$ 比特 0. Ballet-256 在上述搜索空间中的最长不可能差分路线也为 7 轮, 其中一条路线如下:

$$0^6 10^{57}||0^{64}||0^{64}||0^{64} \nrightarrow 0^{64}||0^{64}||10^{63}||0^{64}$$

全轮 Ballet 算法能够抵抗不可能差分分析.

4.4 零相关线性分析

零相关线性分析是利用相关度为 0 的线性路线进行的分析方法. 我们利用基于 MILP 的自动化搜索方法对 Ballet 的零相关线性路线进行搜索. 由于 Ballet 算法为 ARX 结构, 我们仅搜索输入和输出掩码的汉明重量均为 1 的情况.

通过搜索发现, Ballet-128 在上述搜索空间中最长零相关线性路线为 6 轮路线, 其中一条路线如下:

$$0^{32}||10^{31}||0^{32}||0^{32} \nrightarrow 0^{31}1||0^{32}||0^{32}||0^{32}$$

Ballet-256 在上述搜索空间中的最长零相关线性路线也为 6 轮, 其中一条路线如下:

$$0^{64}||10^{63}||0^{64}||0^{64} \nrightarrow 0^{63}1||0^{64}||0^{64}||0^{64}$$

全轮 Ballet 算法能够抵抗零相关分析.

4.5 积分分析

积分分析是对分组密码算法有效的分析手段之一, 我们使用比特级的 Division Property 对 Ballet 算法抵抗积分分析的能力进行评估. 基于 SAT 的自动化积分路线搜索方法, 选用 2^{n-1} 个明文 (设置最高比特为常数, 遍历剩余 127 比特), 观察数据在密文上是否存在平衡比特 (2^{n-1} 个密文在该比特上的异或和为 0). 搜索结果发现 Ballet-128 和 Ballet-256 在该搜索范围内均不存在 7 轮及以上积分路线, 全轮 Ballet 算法能够抵抗积分分析.

4.6 相关密钥类分析

相关密钥类攻击中, 最有效的攻击手段有滑动攻击、相关密钥差分攻击和相关密钥不可能差分攻击等. 由于我们在密钥生成算法中加入了不同的轮常数, 因此能够有效的抵抗滑动攻击. 针对相关密钥差分攻击, 我们对短轮数下的相关密钥差分路线进行了搜索, 未找到高概率的迭代路线, 也未找到弱密钥的存在. 鉴于算法的足够安全冗余度和 ARX 算法分析的困难度, 我们认为 Ballet 算法能够抵抗相关密钥差分攻击. 对于相关密钥不可能差分攻击, 鉴于 7 轮的单密钥不可能差分路线和密钥生成算法的扩散速度, 全轮算法能够抵抗相关密钥不可能差分攻击.

5 Ballet 性能分析

5.1 软件性能分析

本节给出 Ballet 算法在 64 bit Windows 环境和 32 bit ARM 环境下的实现结果. 测试环境分别为: Intel(R) Core(TM) i7-6700T CPU 3.20 GHz 主频、64 位 Windows10 操作系统、8 GB 内存, X86-64 环境和基于 STM32 F103 的开发板 (主频 72 Mhz): stm32f1zet6 核心板 6、512 K 片内 Flash、64 K 片内 RAM.

在进行性能测试时, 采用 256 字节数据作为输入, 每次测试变换密钥, 取 10^5 次 CBC 模式运算测试结果的平均值为加/解密速率 (单位: Mbps) 的结果, 其中加/解密执行时间包含密钥扩展算法运算时间. 测试结果见表 2. 由于 CBC 模式下的解密过程可以并行实现, 我们在表中给出的解密速率是基于多路实现结果. 其中

$$\text{加密速率} = \text{加密数据大小}/\text{加密执行时间 (单位: Mbps)}$$
$$\text{解密速率} = \text{解密数据大小}/\text{解密执行时间 (单位: Mbps)}$$

表 **2** 软件实现结果
Table 2 Results of software implementation

软件实现类别	算法版本	加密速率 (Mbps)	解密速率 (Mbps)	ROM 占用
64bit Windows 环境下的算法速率优化 C 实现	Ballet-128/128	2038	7899	-
	Ballet-128/256	2289	7080	-
	Ballet-256/256	2246	4425	-
32bit ARM 环境下的算法速率优化 C 实现	Ballet-128/128	10.6	9.57	2.6 k
	Ballet-128/256	9.75	9.43	2.9 k
	Ballet-256/256	4.34	4.21	3.5 k

5.2 硬件性能分析

本节给出 Ballet 算法的 Verilog 硬件仿真实现结果. 采用的仿真工具为 ModelSim SE 10.1a, 综合工具为 Synopsys Design Vision (F-2011.09-SP1 Version), 以及工艺库为 HJTC 0.11 um. 算法 Verilog 硬件仿真实现采用 32 个分组数据进行加/解密. 算法实现的加/解密运算用时包含密钥扩展运算用时 (单位: 微秒). 算法电路面积规模为包含加密、解密、密钥扩展运算的算法整体实现的含互连线面积 (单位: 平方微米).

具体方法如下:

(1) 仿真验证

使用 ModelSim 对算法实现正确性进行仿真验证, 并记录 32 个分组数据运算 (含密钥扩展运算) 占用的时钟周期数.

(2) 性能自测试

在给定的时钟约束条件上进行前端综合, 得到并记录以下测试数据:

1) 关键路径时长

在 Synopsys Design Complier 中进行关键路径时长测试. 根据算法设计提供的自测数据, 设置相应的时钟约束条件, 通过综合软件得到关键路径时长.

2) 速率

在关键路径时长合理的条件下, 通过以下公式计算加解密速率:

$$\text{加密速率} = \text{加密数据长度}/(\text{时钟约束条件} \times \text{加密占用的时钟钟周期数})$$
$$\text{解密速率} = \text{解密数据长度}/(\text{时钟约束条件} \times \text{解密占用的时钟周期数})$$

3) 面积

在可满足的最小时钟约束条件下, 在 HJTC 0.11 um 工艺库基础上进行综合, 记录算法实现面积 (包含加密、解密、密钥扩展运算的算法整体实现的含互连线面积).

4) 吞面比

$$\text{加密吞面比} = \text{加密速率}/\text{算法电路面积}$$
$$\text{解密吞面比} = \text{解密速率}/\text{算法电路面积}$$

实现结果如表 3 所示.

6 总结

本文给出了一个新的分组密码算法—Ballet 算法. 该算法是在 Lai-Massey 结构的基础进行简化设计而成. 每个版本的轮函数相同, 均为 4 分支的近似对称 ARX 结构, 具有较强的灵活性和延展性. 轮函数中仅包含模加操作 (2 个)、异或操作 (3 个) 和循环移位操作 (4 个), 利于在 32 位和 64 位平台上的软件实

表 **3** Verilog 硬件仿真实现结果
Table 3 Results of Verilog harware simulation implementation

仿真结果	算法版本		
	Ballet-128/128	**Ballet-128/256**	**Ballet-256/256**
运行周期 (Cycles)	3054	3184	4874
时钟约束 (ns)	3.31	3.3	3.59
加密速率 (Mbps)	822.78	791.59	950.79
解密速率 (Mbps)	798.36	768.08	922.35
面积 (um^2)	45 967.11	64 610.57	96 026.89
面积 (GE)	4739.37	6661.57	9900.70
加密吞面比 ($Mbps/um^2$)	0.017 899	0.012 252	0.009 901
解密吞面比 ($Mbps/um^2$)	0.017 368	0.011 888	0.009 605

现, 尤其是 128 比特分组版本算法在 32 位平台上和 256 比特分组版本算法在 64 位平台上的实现. 在安全性方面, Ballet 算法能够抵抗差分分析和线性分析等已有攻击方法, 并具有充分的安全冗余, 同时防护侧信道攻击的代价小.

然而, 美中不足的是, 为了算法抗差分分析和线性分析的可证明安全性, Ballet 算法的轮数设定较长. 这是由 ARX 算法安全性评估困难导致的. 当前国内外密码算法分析和设计领域对 ARX 类算法安全性的合理评估是一个难点, 现多通过短轮数下的安全界叠加来评估长轮数下的安全性, 从而导致评估的安全界较松. 在此前提下, 为保证算法充分的抵抗现有攻击的能力, 我们将算法的轮数设定较高, 从而一定程度上影响了软硬件的表现. 在之后的研究工作中, 我们将进一步对 ARX 算法抗差分分析和线性分析的安全界进行研究, 提出更优的评估方法.

References

[1] DAEMEN J, RIJMEN V. AES proposal: Rijndael[OL]. 1999. https://cs.ru.nl/joan/papers/JDA_VRI_Rijndael_V2_1999.pdf

[2] BEIERLE C, JEAN J, KÖLBL S, et al. The SKINNY family of block ciphers and its low-latency variant MANTIS[C]. In: Advances in Cryptology—CRYPTO 2016, Part II. Springer Berlin Heidelberg, 2016: 123–153. [DOI: 10.1007/978-3-662-53008-5_5]

[3] Data encryption standard (DES)[S/OL]. Federal Information Processing Standards Publication 46-3. 1999. https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf

[4] BEAULIEU R, SHOR D, SMITH J, et al. The SIMON and SPECK lightweight block ciphers[C]. In: Proceedings of the 52nd Annual Design Automation Conference. ACM, 2015: Article No. 175. [DOI: 10.1145/2744769.2747946]

[5] LAI X J, MASSEY J L. A proposal for a new block encryption standard[C]. In: Advances in Cryptology—EUROCRYPT '90. Springer Berlin Heidelberg, 1991: 389–404. [DOI: 10.1007/3-540-46877-3_35]

[6] YANG G, ZHU B, SUNDER V, et al. The Simeck family of lightweight block ciphers[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2015. Springer Berlin Heidelberg, 2015: 307–329. [DOI: 10.1007/978-3-662-48324-4_16]

[7] DINU D, PERRIN L, UDOVENKO A, et al. Design strategies for ARX with provable bounds: Sparx and LAX[C]. In: Advances in Cryptology—ASIACRYPT 2016, Part I. Springer Berlin Heidelberg, 2016: 484–513. [DOI: 10.1007/978-3-662-53887-6_18]

[8] SONG L, HUANG Z, YANG Q. Automatic differential analysis of ARX block ciphers with application to SPECK and LEA[C]. In: Information Security and Privacy—ACISP 2016, Part II. Springer Cham, 2016: 379–394. [DOI: 10.1007/978-3-319-40367-0_24]

[9] XIANG Z J, ZHANG W T, BAO Z Z, et al. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers[C]. In: Advances in Cryptology—ASIACRYPT 2016, Part I. Springer Berlin Heidelberg, 2016: 648-678. [DOI: 10.1007/978-3-662-53887-6_24]

[10] CUI T T, CHEN S Y, FU K, et al. New automatic search tool for impossible differentials and zero-correlation linear approximations[J]. SCIENCE CHINA Information Sciences, to be published. [DOI: 10.1007/s11432-018-1506-4]

[11] BIHAM E, SHAMIR A. Differential cryptanalysis of DES-like cryptosystems[C]. In: Advances in Cryptology—CRYPTO '90. Springer Berlin Heidelberg, 1991: 2–21. [DOI: 10.1007/3-540-38424-3_1]
[12] MATSUI M. Linear cryptanalysis method for DES cipher[C]. In: Advances in Cryptology—EUROCRYPT '93. Springer Berlin Heidelberg, 1994: 386–397. [DOI: 10.1007/3-540-48285-7_33]
[13] BIHAM E, BIRYUKOV A, SHAMIR A. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials[C]. In: Advances in Cryptology—EUROCRYPT '99. Springer Berlin Heidelberg, 1999: 12–23. [DOI: 10.1007/3-540-48910-X_2]
[14] KNUDSEN L R. Deal: A 128-bit block cipher[R]. Technical Report 151, Department of Informatics, University of Bergen, Bergen, Norway, 1998.
[15] BOGDANOV A, RIJMEN V. Linear hulls with correlation zero and linear cryptanalysis of block ciphers[J]. Designs, Codes and Cryptography, 2014, 70(3): 369–383. [DOI: 10.1007/s10623-012-9697-z]
[16] KNUDSEN L R, WAGNER D A. Integral cryptanalysis[C]. In: Fast Software Encryption—FSE 2002. Springer Berlin Heidelberg, 2002: 112–127. [DOI: 10.1007/3-540-45661-9_9]

作者信息

崔婷婷 (1990–)，山东青岛人，讲师．主要研究领域为对称密码算法的设计和分析．
cuitingting@hdu.edu.cn

王美琴 (1974–)，宁夏银川人，教授．主要研究领域为对称密码算法的设计和分析．
mqwang@sdu.edu.cn

樊燕红 (1979–)，山东聊城人，博士在读．主要研究领域为对称密码算法的安全性分析和实现．
fanyh@mail.sdu.edu.cn

胡凯 (1992–)，山东临沂人，博士在读．主要研究领域为对称密码的分析．
hukai@mail.sdu.edu.cn

付勇 (1981–)，山东寿光人，副研究员．主要研究领域为密码工程．
yongfu0976@163.com

黄鲁宁 (1979–)，山东海阳人，助理研究员．主要研究领域为对称密码算法的安全性分析．
lnhuang@sdu.edu.cn